**СОДЕРЖАНИЕ**

[Введение 7](#_Toc197526409)

[1 Обзор литературы 8](#_Toc197526410)

[1.1 Умная теплица growUp 8](#_Toc197526411)

[1.2 Гидропоника – система умный сад 8](#_Toc197526412)

[1.3 Настольная садовая ферма L-Box 9](#_Toc197526413)

[1.4 Умный сад iGarden PRO 10](#_Toc197526414)

[2 Структурное проектирование 11](#_Toc197526415)

[2.1 Перечень блоков 11](#_Toc197526416)

[2.2 Связи между блоками 11](#_Toc197526417)

[3 Функциональное проектирование 12](#_Toc197526418)

[3.1 Состав устройства 12](#_Toc197526419)

[3.2 Микроконтроллер 12](#_Toc197526420)

[3.3 Датчик температуры и влажности воздуха 13](#_Toc197526421)

[3.4 Датчик уровня освещенности 15](#_Toc197526422)

[3.5 Датчик влажности почвы 15](#_Toc197526423)

[3.6 Датчик температуры почвы 15](#_Toc197526424)

[3.7 Мини-насос 16](#_Toc197526425)

[3.8 Фитолента 16](#_Toc197526426)

[3.9 Кулеры 16](#_Toc197526427)

[3.10 Нагревающий провод 16](#_Toc197526428)

[3.11 Реле 16](#_Toc197526429)

[3.12 Источники питания 17](#_Toc197526430)

[4 Разработка принципиальной электрической схемы устройства 18](#_Toc197526431)

[4.1 Расчет питания для устройств с напряжением 12 В 18](#_Toc197526432)

[4.2 Расчет питания для устройств с напряжением 5 В 18](#_Toc197526433)

[4.3 Расчет питания для нагревательного блока 19](#_Toc197526434)

[4.4 Расчет питания всего устройства 19](#_Toc197526435)

[5 Разработка программного обеспечения 20](#_Toc197526436)

[5.1 Требования к программе 20](#_Toc197526437)

[5.2 Схема программы 20](#_Toc197526438)

[5.3 Программа управления устройством 22](#_Toc197526439)

[5.4 Файл smartGrow.ino 22](#_Toc197526440)

[5.5 Файл devices.h 23](#_Toc197526441)

[5.6 Файл connectWiFi.h 24](#_Toc197526442)

[5.7 Файл tgCommunication.h 24](#_Toc197526443)

[5.8 Файл automaticMode.h 24](#_Toc197526444)

[5.9 Файл config.h 25](#_Toc197526445)

[6 Проектирование корпуса 26](#_Toc197526446)

[6.1 Разработка корпуса теплицы 26](#_Toc197526447)

[6.2 Разработка корпуса модуля с электроникой 26](#_Toc197526448)

[6.3 Общий вид проекта 27](#_Toc197526449)

[7 Проектирование платы 28](#_Toc197526450)

[7.1 Проектирование платы 28](#_Toc197526451)

[7.2 Производство платы 28](#_Toc197526452)

[8 Руководство пользователя 29](#_Toc197526453)

[8.1 Инструкция к физическим модулям 29](#_Toc197526454)

[8.2 Инструкция к telegram-боту 29](#_Toc197526455)

[Заключение 31](#_Toc197526456)

[Список использованных источников 32](#_Toc197526457)

[Приложение А 33](#_Toc197526458)

[Приложение Б 34](#_Toc197526459)

[Приложение В 35](#_Toc197526460)

[Приложение Г 36](#_Toc197526461)

[Приложение Д 37](#_Toc197526462)

[Приложение Е 51](#_Toc197526463)

[Приложение Ж 52](#_Toc197526464)

Введение

Современные технологии делают возможным создание умных решений для выращивания растений. Умная теплица – это система, предназначенная для создания оптимальных условий для роста растений в ограниченном пространстве.

Актуальность такого проекта обусловлена несколькими факторами. Во-первых, растущий интерес к здоровому образу жизни и экологически чистым продуктам питания стимулирует людей выращивать собственные овощи, фрукты и зелень. Во-вторых, климатические изменения и сезонные ограничения создают потребность в устройствах, способных обеспечить оптимальные условия для растений круглый год. В-третьих, развитие концепции "умного дома" создает спрос на интеллектуальные устройства, интегрируемые в общую систему автоматизации жилища.

Цель проекта заключается в создании компактной, эффективной и простой в использовании мини-теплицы с контролем микроклимата, способной автоматически поддерживать оптимальные условия для роста растений. Главные параметры микроклимата, которые необходимо контролировать в рамках теплицы, – это освещенность, температура воздуха, температура и влажность почвы.

Устройство, разрабатываемое в данном курсовом проекте, должно выполнять ряд задач:

– производить мониторинг температуры воздуха в теплице и автоматически переходить в режим проветривания в случае, если показатель выходит за установленные рамки;

– отслеживать уровень влажности воздуха, автоматически осуществлять полив и проветривание;

– отслеживать уровень влажности почвы, автоматически осуществлять и прекращать полив;

– определять интенсивность света в теплице, автоматически регулировать уровень искусственного освещения в зависимости от времени суток и погодных условий;

– обладать функцией дистанционного управления и мониторинга показателей датчиков.

Таким образом, проект умной теплицы является отличным решением вопроса оптимизации проращивания растений в домашних условиях, обеспечивая постоянство необходимых условий, показатели которых пользователь регулирует в соответствии с выращиваемым растением.

# 1 Обзор литературы

## 1.1 Умная теплица growUp

Умная офисная теплица growUp [1] (Рисунок 1.1) поможет обзавестись мини-садом и ухаживать за ним прямо на рабочем столе.



Рисунок 1.1 – Умная офисная теплица growUp

В теплице можно установить один из двух режимов освещения: более сильный подойдет светолюбивым растениям, а слабый – тенелюбивым и теневыносливым. Таймер поможет сымитировать природный солнечный ритм летнего времени: освещение автоматически включается и выключается на 12 часов. Встроенный вентилятор обеспечивает циркуляцию воздуха, что оказывает положительное влияние на рост растений. Рабочее напряжение и ток: 5 В, 1000 мА. Работает при питании от сети.

## 1.2 Гидропоника – система умный сад

Компактная домашняя гидропонная установка с электронной системой контроля уровня воды и лампой с таймером [2] (Рисунок 1.2) предназначена для выращивания растений без использования земли – это оптимальное решение для современных садоводов, стремящихся выращивать растения в условиях ограниченного пространства в любое время года.

Умный сад с фитолампой обеспечивает гармоничное освещение для роста растений. LED-лампа c тремя режимами освещения, а её ночной режим и плавный переход в яркость создают естественные условия для фотосинтеза.

Встроенный пульт управления, который соединен с кабелем, имеет индикаторы подключения насоса и лампы.

Насос после подключения работает в режиме 10 минут работа, 10 минут отдых. Помпа насоса снабжена индикатором уровня воды, при снижении уровня воды до её уровня загорается лампа зеленым светом, оповещая о критическом уровне воды в баке и необходимости её пополнения.



Рисунок 1.2 – Гидропонная система

## 1.3 Настольная садовая ферма L-Box

Продукт L-Box [3] представляет собой профессиональную модульную домашнюю ферму для выращивания различных видов растений, включая овощи, зелень и цветы. Эта система разработана для удобства пользователей и позволяет растить растения в условиях, не зависящих от внешней среды.

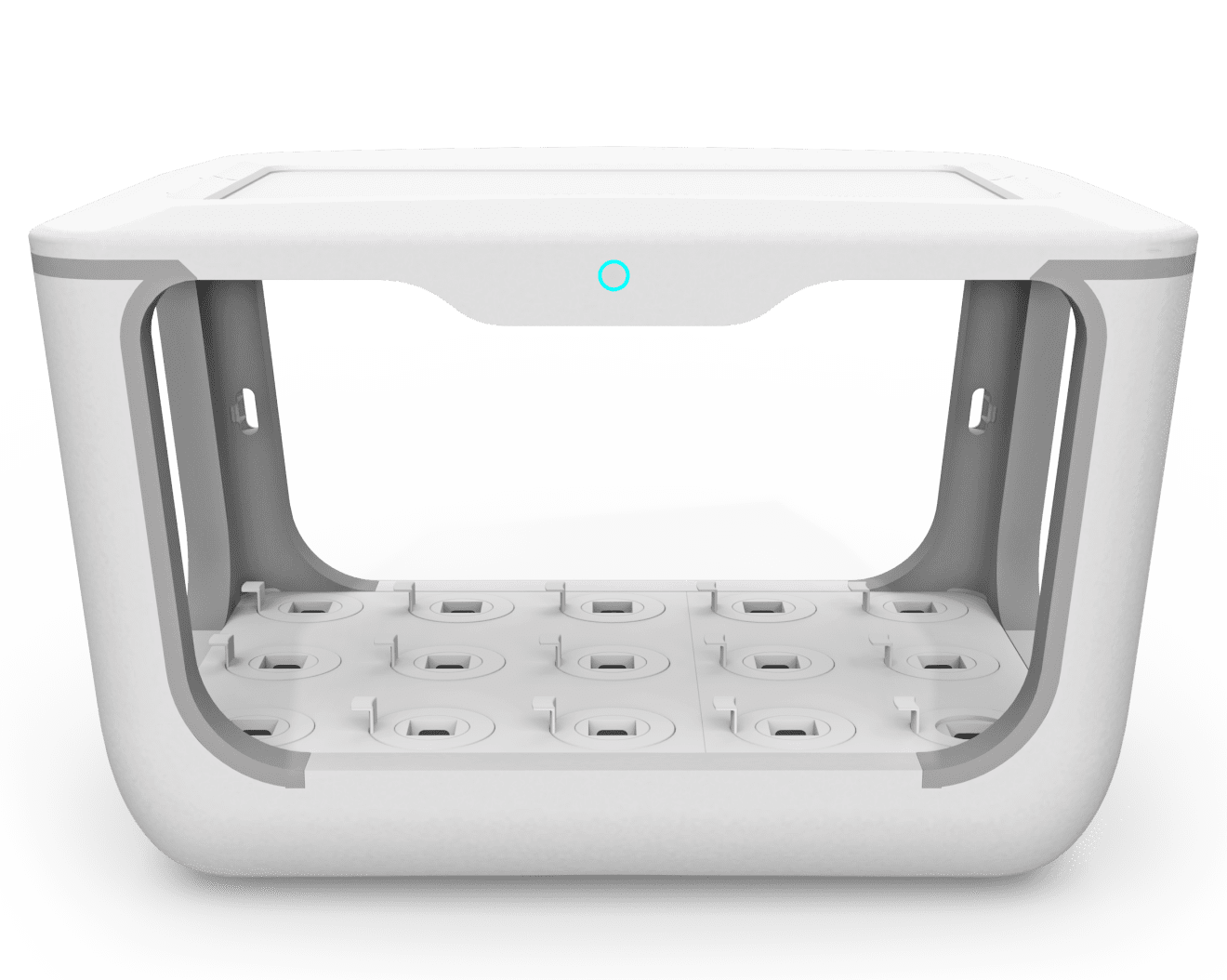


Рисунок 1.3 – Настольная садовая ферма L-Box

Все функции фермы контролируются через мобильное приложение 4d Box, что обеспечивает полный контроль над процессом выращивания. Пользователи могут следить за уровнем удобрений, температурой, уровнем питательного раствора, а также регулировать яркость освещения и время работы ламп и насосов.

Ферма оснащена датчиками, которые позволяют автоматически управлять поливом и освещением. Это делает процесс выращивания растений максимально простым и эффективным, устраняя необходимость постоянного контроля. Большая фитолампа мощностью 24 Вт обеспечивает необходимое освещение для всех стадий роста растений. Система также обеспечивает 100% циркуляцию воздуха, что способствует здоровому развитию растений.

## 1.4 Умный сад iGarden PRO

IGarden PRO [4] – это современное решение для выращивания растений в домашних условиях, которое использует гидропонику и автоматическую систему освещения (Рисунок 1.4). Этот компактный умный сад позволяет пользователю без особых усилий заниматься садоводством, даже если у него нет опыта или доступа к земле.



Рисунок 1.4 – Умный сад iGarden PRO

IGarden PRO использует специальный грунт (вермикулит) и систему гидропоники. Умный сад оснащен мощной LED-лампой с тремя режимами работы, которая обеспечивает оптимальные условия для роста растений, имитируя солнечный свет. Система контроля позволяет поддерживать необходимый уровень воды, что важно для здоровья растений.

Умный сад iGarden PRO предлагает инновационный подход к выращиванию растений, сочетая простоту использования с современными технологиями.

# 2 Структурное проектирование

Структурная схема устройства приведена в приложении А.

## 2.1 Перечень блоков

В данном устройстве можно выделить десять основных блоков:

– микроконтроллер;

– блок датчика температуры и влажности;

– блок датчика влажности почвы;

– блок датчика температуры почвы;

– блок датчика уровня освещенности;

– модуль проветривания;

– модуль освещения;

– модуль полива;

– модуль нагрева;

– модуль управления и отображения информации.

Блок датчиков включает в себя блок датчика температуры и влажности, блок датчика влажности почвы, блок датчика температуры почвы, блок датчика уровня освещенности и нужен для измерения показателей микроклимата в теплице.

Модули проветривания, освещения, полива и нагрева необходимы для изменения параметров микроклимата, а именно регулировка температуры, уровня освещенности и влажности воздуха и почвы.

Микроконтроллер необходим для взаимодействия и управления всеми блоками системы.

В качестве модуля управления и отображения информации реализован телеграм-бот Smart Grow Bot. Данный блок необходим для управления микроконтроллером и получения информации о системе.

## 2.2 Связи между блоками

Модуль управления и отображения информации передает информацию микроконтроллеру о режиме работы (автоматический или ручной), запрашивает информацию о показаниях датчиков и в ручном режиме передает информацию об управлении внешними устройствами.

Блоки датчиков снимают показания и предают эту информацию блоку микроконтроллера.

Микроконтроллер обрабатывает информацию, принятую от блоков датчиков, и передает ее в модуль управления и отображения информации. Регистрируя отклонения значений в автоматическом режиме от заданных пользователем допустимых границ, блок микроконтроллера выполняет управление модулями внешних устройств. В ручном режиме выполняет управление модулями внешних устройств, информация о котором была получена от модуля управления и отображения информации.

# 3 Функциональное проектирование

Функциональная схема устройства приведена в приложении Б.

## 3.1 Состав устройства

В состав устройства системы контроля микроклимата теплицы входит:

– микроконтроллер;

– датчик температуры и влажности воздуха;

– датчик уровня освещенности;

– датчик влажности почвы;

– датчик температуры почвы;

– мини-насос;

– фитолента;

– кулеры;

– нагревающий провод;

– реле;

– источники питания.

## 3.2 Микроконтроллер

Среди микроконтроллеров были рассмотрены популярные платы торговых марок ESP [5], Raspberry [6], Nordic Semiconductor [7] и STM [8]. Их общие сравнительные характеристики представлены в таблице 1.1.

Таблица 1.1 – Характеристики микроконтроллеров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристика | ESP32 | Raspberry Pi Pico RP2040 | Nordic nRF52840 | STM32WB55 |
| Процессор | Dual-core Xtensa LX6 32-bit | Dual-core ARM Cortex-M0+ | ARM Cortex-M4F | ARM Cortex-M4F |
| Тактовая частота, МГц | До 240 | 133 | 64 | 64 |
| ОЗУ, КБ | 520 | 264 | 256 | 256 |
| Flash-память, МБ | 16 | 2 | 2 | 2 |
| Выходное напряжение, В | 3,3 | 3,3 | 1,7 – 3,6 | 1,7 – 3,6 |
| Количество входов/выходов | 34 | 26 | 48 | 72 |
| Ток потребления, мА | 80 | 20 | 5.3 | 5.2 |
| WiFi, ГГц | 2.4 | Нет | Нет | Нет |
| Bluetooth | BLE | Нет | BLE 5 | BLE 5 |

ESP32 выделяется высокой производительностью, большим объемом памяти и поддержкой технологии WiFi, что является важным критерием для выполнения данного проекта и реализации удаленного доступа и управления.

Линейка ESP32 предлагает широкий выбор моделей, которые в зависимости от своих характеристик применяются в различных областях. Их сравнительная характеристика приведена в таблице 1.2.

Таблица 1.2 – Сравнительные характеристики моделей ESP32

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Характеристика | ESP8266 | ESP32 | ESP32-S2 | ESP32-S3 | ESP32-C3 |
| Процессор | Tensilica L106 | Xtensa LX6 | Xtensa LX7 | Xtensa LX7 | RISC-V |
| Тактовая частота, МГц | 80-160 | До 240 | До 240 | До 240 | До 160 |
| ОЗУ, КБ | 160 | 520 | 320 | 512 | 400 |
| Flash память, МБ | 16 | 16 | 128 | 128 | 16 |
| Выходное напряжение, В | 3,3 | 3,3 | 3,3 | 3,3 | 3,3 |
| Количество входов/выходов | 17 | 34 | 43 | 45 | 22 |
| Ток потребления, мА | 80 | 80 | 70 | 75 | 60 |
| GPIO | 17 | 34 | 43 | 45 | 22 |
| WiFi, ГГц | 2,4 | 2,4 | 2,4 | 2,4 | 2,4 |
| Bluetooth | Нет | BT 4.2 & BLE | Нет | BLE 5.0 | BLE 5.0 |

В результате анализа сравнительных характеристик различных микроконтроллеров была выбрана ESP32-S3 в сборке ESP32-S3-DevKitC-1, оснащенная ESP32-S3-WROOM-1. Важным критерием выбора стало наличие встроенного Bluetooth и Wi-Fi, что делает данную плату отличным вариантом для реализации IoT-проектов. Также на выбор значительно повлиял объем встроенной памяти, который в сравнении с другими моделями является большим, а также количество входных и выходных пинов, что делает данный вариант оптимальным для реализации данного проекта.

## 3.3 Датчик температуры и влажности воздуха

На рынке распространены устройства, которые совмещают в себе функции измерения температуры и относительной влажности воздуха. Одним из таких решений являются датчики серии DHT [9]. Также на рынке существует ряд моделей датчиков BME [10], которые совмещают в себе помимо этого еще и функции измерения атмосферного давления. Сравнение данных датчиков приведено в таблице 1.3.

Таблица 1.3 – Сравнительные характеристики датчиков DHT и BME

|  |  |  |  |
| --- | --- | --- | --- |
| Датчик | DHT11 | DHT22 | BME280 |
| Измеряемые параметры | Температура, влажность | Температура, влажность | Температура, влажность, атмосферное давление |
| Диапазон измерения температуры, °C | 0 – +50 | -40 – +125 | -40 – +85 |
| Точность измерения температуры, °C | ±2 | ±0.5 | ±1 |
| Диапазон измерения относительной влажности воздуха, % | 20 – 80 | 0 – 100 | 0 – 100 |
| Точность измерения относительной влажности воздуха, % | ±5 | ±2 | ±3 |
| Интерфейс | Цифровой | Цифровой | I2C |
| Время отклика, с | 1 | 2 | < 1 |

В результате был выбран датчик BME280. Этот выбор обусловлен многими факторами. Данный датчик более функционален, так как помимо температуры и влажности воздуха измеряет атмосферное давление. Кроме того, BME280 отличается компактным размером, который также облегчает интеграцию в различные проекты. Получение значений с датчика происходит за меньшее время в сравнении с датчиками DHT, где опрос происходи один раз в секунду для DHT11 и один раз в две секунды для DHT22. Так же важным критерием является работа по интерфейсу I2C.

I2C – это последовательный интерфейс, который позволяет обмениваться данными между микроконтроллерами и различными периферийными устройствами, такими как датчики и дисплеи. I2C поддерживает двунаправленный обмен данными, что позволяет устройствам как отправлять, так и принимать данные. Каждое устройство на шине I2C имеет уникальный адрес, что позволяет одному мастер-устройству управлять несколькими подключенными устройствами. I2C поддерживает различные скорости передачи, включая стандартные 100 кбит/с и 400 кбит/с, а также высокоскоростные режимы до 3,4 Мбит/с, что позволяет повысить скорость опроса датчиков и соответственно адаптивность системы контроля микроклимата теплицы.

## 3.4 Датчик уровня освещенности

На рынке существуют различные аналоги датчиков, измеряющих уровень освещенности, которые отличаются в первую очередь способом измерения. Сравнение датчиков GY-302 [11], TSL2561 [12] и фоторезистора GL5528 [13] приведено в таблице 1.4.

Таблица 1.4 – Сравнительные характеристики датчиков освещенности

|  |  |  |  |
| --- | --- | --- | --- |
| Датчик | GY-302 | TSL2561 | Фоторезистор GL5528 |
| Диапазон измерения освещенности, lx | 1 – 65535 | 0.1 – 40000 | 1 – 1000 |
| Точность измерения, % | ± 20 | ± 10 | Зависит от условий |
| Напряжение питания, В | 3,3 – 5 | 3,3 – 5 | 3,3 – 5 |
| Интерфейс | I2C | I2C | Аналоговый |
| Время отклика, с | < 1 | < 1 | < 1 |

В результате был выбран датчик GY-302. Этот выбор обусловлен наличием I2C интерфейса, широким диапазоном измерения освещенности и относительной дешевизной.

## 3.5 Датчик влажности почвы

Емкостной датчик влажности почвы – это устройство, предназначенное для измерения уровня влаги в грунте. Он используется в системах автоматического полива, что позволяет обеспечить оптимальные условия для роста растений. Для данного проекта выбран Soil Moisture Sensor 1.2 [14] – емкостной аналоговый датчик, который работает в диапазоне от 3,3 до 5 В и потребляет ток до 6 мА. Аналоговый выходной сигнал варьируется от 0,5 до 3,3 В в зависимости от уровня влажности.

## 3.6 Датчик температуры почвы

Датчик температуры почвы необходим для более точного контроля нагрева теплицы и предотвращения выхода за оптимальные температурные показатели. С помощью датчика можно отслеживать температурные колебания и адаптировать работу системы нагрева. Это позволит избежать перегрева и обеспечить равномерный прогрев почвы, что важно для роста корней и общего состояния растений.

Важным критерием в выборе датчика для измерения температуры почвы является герметичность корпуса и устойчивость к коррозии вследствие полива. Решением данного вопроса является датчик, помещенный в гильзу.

Датчик температуры почвы DS18B20 [15] – это герметичный цифровой датчик, предназначенный для точного измерения температуры в различных условиях, включая почву. DS18B20 работает в диапазоне температур от -55 °C до +125 °C, что делает его подходящим для применения в различных климатических условиях. Высокая точность измерений в пределах заданного диапазона и возможность работы при низком напряжении также сыграли важную роль в выборе этого датчика для проекта.

## 3.7 Мини-насос

Для реализации функции увлажнения почвы в данном проекте был выбран водяной насос WP-370B. Данное устройство работает в диапазоне от 9,6 В до 14,4 В и потребляет ток до 100 мА. Насос способен перекачивать от 0,4 до 0,7 литра воды в минуту, что является достаточным в рамках мини-теплицы.

Благодаря своим характеристикам, насос эффективно справляется с задачами автоматического полива, способствуя поддержанию необходимого уровня влаги в почве.

## 3.8 Фитолента

Светодиодная лента для растений используется для создания оптимальных условий освещения, что особенно важно при выращивании растений в закрытых помещениях или в условиях недостатка света. Фитолента имеет рабочее напряжение 5 В и рабочий ток 400 мА на метр.

## 3.9 Кулеры

Для обеспечения циркуляции воздуха внутри теплицы были выбраны два кулера. Они обеспечивают вытяжку воздуха из теплицы и поступление свежего, что помогает поддерживать оптимальные климатические условия. Были выбраны кулеры с рабочим напряжением 12 В.

## 3.10 Нагревающий провод

В качестве нагревательного элемента был выбран углеродный кабель. Углеволокно гарантирует долговечность нагревательного провода и его полную изоляцию, что делает его безопасным для использования в теплице. Сопротивление данного провода 33 Ом/м, питание 220 В.

## 3.11 Реле

Для управления несколькими устройствами с разным рабочим напряжением был выбран электромагнитный релейный модуль на 4 канала с оптической развязкой и питанием 12 В. Максимальный протекающий ток составляет 50 мА.

Реле используется для управления работой таких устройств, как насос, кулеры и нагревательные элементы, обеспечивая их включение и выключение в зависимости от показаний датчиков в автоматическом режиме или при ручном управлении. Это позволяет автоматизировать процессы в теплице и эффективно управлять климатом внутри неё.

## 3.12 Источники питания

Для питания устройств с разным рабочим напряжением были выбраны модульные импульсные блоки питания на 5 В и 12 В. Блок питания на 5 В используется для питания светодиодной фитоленты, блоки питания на 12 В обеспечивает работу кулеров и реле.

В линейке готовых блоков питания в модульном исполнении на 12 В выбран блок питания B2L0ESB25 12 B 2A 25W.

В линейке готовых блоков питания в модульном исполнении на 5 В выбран блок питания Орбита ОТ-АРВ91 5В, 25W. IP20.

# **4**Разработка принципиальной электрической схемы устройства

Принципиальная схема устройства приведена в приложении В.

## 4.1 Расчет питания для устройств с напряжением 12 В

Для расчёта характеристик блока питания была составлена таблица 4.1

Таблица 4.1 – Характеристики устройств с напряжением 12 В

|  |  |  |  |
| --- | --- | --- | --- |
| Функциональное назначение устройства | Устройство | Потребление | Максимальная потребляемая мощность, Вт |
| Полив | Мини-насос водяной | 350 мА | 4,2 |
| Вентиляция вытяжная | Мини-кулер 12 В DC | 80 мА | 1 |
| Вентиляция приточная | Мини-кулер 12 В DC | 50 мА | 0,6 |
| Управление исполнительными устройствами | Релейный модуль на 4 канала с оптической развязкой. | 50 мА | 0,6 |

При выходном напряжении модуля питания 12 В максимальный потребляемый ток нагрузок составит около 0,53 А и 6,4 Вт соответственно. С целью повышения надежности и возможности масштабирования и модернизации схемы принимаем минимум двукратный запас мощности.

## 4.2 Расчет питания для устройств с напряжением 5 В

Для расчёта характеристик блока питания была составлена таблица 4.2.

Таблица 4.2 – Характеристики устройств с напряжением 5 В

|  |  |  |  |
| --- | --- | --- | --- |
| Функциональное назначение устройства | Устройство | Потребление | Макс. потреб. мощность, Вт |
| Освещение | Фитолента 60 LED/ метр | 750 мА/м | 9 |
| Логическое управление исполнительными устройствами | ESP-32-S3-DevKitC-1 | 300 мА | 3 |

Продолжение таблицы 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Измерение влажности почвы | Soil Moisture Sensor 1.2 | 30 мА | 0,36 |
| Измерение температуры почвы | DS18B20 | 2 мА | 0,01 |
| Измерение температуры и влажности воздуха | BME280 | 3,5 мА | 0,03 |
| Измерение уровня освещенности | GY-302 | 1,2 мА | 0,014 |

При выходном напряжении модуля питания 5 В максимальный ток составит около 1,1 А и 12,4 Вт. С целью повышения надежности и возможности масштабирования и модернизации схемы принимаем минимум двукратный запас мощности.

## 4.3 Расчет питания для нагревательного блока

Кабель Греющий углеродный карбоновый, 33 Ом/м. При длине 12 м мощность изготовленного нагревателя составит 122 Вт при температуре поверхности 48 C при подключении с промышленной сети 220 В (паспортные данные).

## 4.4 Расчет питания всего устройства

Исходя из взятых блоков питания произведен расчет мощности всего устройства умной теплицы (Таблица 4.3).

Таблица 4.3 – Расчет питания всего устройства

|  |  |  |  |
| --- | --- | --- | --- |
| Потребитель | Выходная мощность, Вт | Усредненный КПД БП, % | Потребляемая мощность, Вт |
| Блок питания 220В/12В DС | 25 | 90 | 28 |
| Блок питания 220В/5В DС | 25 | 90 | 28 |
| Элемент нагревательный 220В | 122 | - | 122 |
| ИТОГО |  |  | 178 |

# 5 Разработка программного обеспечения

Схема программы приведена в приложении Г.

## 5.1 Требования к программе

Программа, управляющая системой теплицы, должна реализовывать следующий функционал:

– считывание показаний с датчиков;

– управление внешними устройствами;

– подключение к заданной точке доступа;

– обработка входящих сообщений;

– отправка информации о показаниях датчиков;

– отправка информации о состоянии устройств;

– индикация состояния подключения и работы нагревательного элемента;

– автоматический режим управления параметрами теплицы на основе заданных пороговых значений.

Исполняемый файл программы не должен превышать объём флеш-памяти микроконтроллера, равной 16 МБ, и не должен использовать объём оперативной памяти больший, чем 160 КБ.

## 5.2 Схема программы

Описание блоков:

1. Начало. Инициализация всех необходимых переменных и настроек для работы устройства.

2. Инициализация шины I2C. Выполнить вызов Wire.begin() для инициализации I2C, что необходимо для связи с датчиками.

3. Настройка датчиков и устройств. Выполнить настройку и инициализацию всех подключенных датчиков с помощью соответствующих библиотек. Определить пины для управления устройствами.

4. Подключение к WiFi. Составить список всех сетей, поиск в списке заданной сети, использовать WiFi.begin(ssid, password) для подключения к заданной сети WiFi.

5. Настройка подключения к телеграм-боту. Инициализировать библиотеку Telegram с токеном бота, чтобы установить связь с вашим ботом.

6. Обновление статусов подключения датчиков. Проверить и обновить статусы всех подключенных датчиков, убедившись, что они работают корректно.

7. Проверка состояния подключения к WiFi с помощью WiFi.status(). Если подключение успешно, перейти к следующему шагу, иначе выполнить шаг 8.

8. Подключение к WiFi. Если WiFi не подключен, выполнить повторный вызов WiFi.begin(ssid, password).

9. Обработка сообщений бота. Вызвать метод bot.getUpdates() для получения новых сообщений от пользователей, чтобы обработать их.

10. Сообщение /sensors\_stats? Проверить, содержит ли полученное сообщение команду /sensors\_stats. Если да, перейти к шагу 11, иначе к шагу 12.

11. Отправка значений датчиков. Собрать данные с датчиков и отправить их пользователю через бот, используя bot.sendMessage(chat\_id, sensor\_data).

12. Сообщение /help? Проверить, содержит ли полученное сообщение команду /sensors\_stats. Если да, перейти к шагу 13, иначе к шагу 14.

13. Отправка пользователю всего списка команд с их описанием.

14. Сообщение /heater\_on? Проверить, содержит ли полученное сообщение команду /heater\_on. Если да, перейти к шагу 15, иначе к шагу 16.

15. Включение нагревателя. Включить нагреватель, установив соответствующий выходной пин в HIGH.

16. Сообщение /heater\_off? Проверить, содержит ли полученное сообщение команду /heater\_off. Если да, перейти к шагу 17, иначе к шагу 18.

17. Выключение нагревателя. Выключить нагреватель, установив соответствующий выходной пин в LOW.

18. Сообщение /cooler\_on? Проверить, содержит ли полученное сообщение команду /cooler\_on. Если да, перейти к шагу 19, иначе к шагу 20.

19. Включение кулеров. Включить кулеры, установив соответствующий выходной пин в HIGH.

20. Сообщение /cooler\_off? Проверить, содержит ли полученное сообщение команду /cooler\_off. Если да, перейти к шагу 21, иначе к шагу 22.

21. Выключение кулеров. Выключить кулеры, установив соответствующий выходной пин в LOW.

22. Сообщение /lighting\_on? Проверить, содержит ли полученное сообщение команду /lighting\_on. Если да, перейти к шагу 23, иначе к шагу 24.

23. Включение фитоленты. Включить фитоленту, установив соответствующий выходной пин в HIGH.

24. Сообщение /lighting\_off? Проверить, содержит ли полученное сообщение команду /lighting\_off. Если да, перейти к шагу 25, иначе к шагу 26.

25. Выключение фитоленты. Выключить фитоленту, установив соответствующий выходной пин в LOW.

26. Сообщение /pomp\_on? Проверить, содержит ли полученное сообщение команду /pomp\_on. Если да, перейти к шагу 27, иначе к шагу 28.

27. Включение мини-насоса. Включить мини-насос, установив соответствующий выходной пин в HIGH.

28. Сообщение /pomp\_off? Проверить, содержит ли полученное сообщение команду /pomp\_off. Если да, перейти к шагу 29, иначе к шагу 30.

29. Выключение мини-насоса. Выключить мини-насос, установив соответствующий выходной пин в LOW.

30. Сообщение /automatic\_mode\_on? Проверить, содержит ли полученное сообщение команду /automatic\_mode\_on. Если да, перейти к шагу 31, иначе к шагу 32.

31. Включение автоматического режима. Установка флага automaticModeState в true и отправка подтверждения.

32. Сообщение /automatic\_mode\_off? Проверить, содержит ли полученное сообщение команду /automatic\_mode\_off. Если да, перейти к шагу шагу 33, иначе к шагу 6.

33. Выключение автоматического режима. Установка флага automaticModeState в false и отправка подтверждения.

## 5.3 Программа управления устройством

Листинг кода программы приведен в приложении Д.

Проект SmartGrow представляет собой систему, предназначенную для контроля условий в теплице. Основной функционал реализован с использованием микроконтроллера и включает в себя управление различными устройствами и мониторинг параметров через Telegram-бота.

Код использует библиотеку UniversalTelegramBot для взаимодействия с Telegram API, что позволяет пользователю отправлять команды и получать данные о состоянии датчиков. Поддерживаемые команды включают /sensors\_stats для получения текущих показаний (освещённость, температура и влажность воздуха, температура и влажность почвы), /devices\_status для проверки состояния устройств, а также команды для управления нагревателем, охладителем, фитолентой и насосом.

Система поддерживает автоматический режим, в котором параметры теплицы регулируются на основе заданных пороговых значений и гистерезиса для предотвращения частых переключений. При потере WiFi-соединения система автоматически пытается переподключиться, обеспечивая стабильную работу. Индикация состояния подключения и работы нагревателя осуществляется через светодиоды.

## 5.4 Файл smartGrow.ino

Строки 1 – 5: Подключение необходимых заголовочных файлов для работы с устройствами, Telegram и WiFi.

Строка 6: Инициализация последовательного порта для отладки с помощью Serial.begin(115200).

Строка 7: Инициализация шины I2C с помощью Wire.begin(), что необходимо для работы с датчиками.

Строки 9 – 12: Настройка датчиков (освещённости, температуры, влажности, почвы) с помощью функций setupLightmeter(), setupBme(), setupSoilMoistureSensor(), setupDs18b20().

Строки 9 – 12: Настройка датчиков (освещённости, температуры, влажности, почвы) с помощью функций setupLightmeter(), setupBme(), setupSoilMoistureSensor(), setupDs18b20().

Строка 14: Вызов setupLeds() для инициализации светодиодов.

Строка 16: Вызов setupDevices() для настройки пинов управления устройствами.

Строка 18: Вызов setupWiFi() для подключения к WiFi.

Строка 19: Вызов setupTelegram() для инициализации Telegram-бота.

Строка 20: Вызов initSensorData() для инициализации структуры хранения данных датчиков.

Строка 22: Начало функции loop().

Строка 23: Проверка подключения датчиков через checkConnectsSensors().

Строка 24: Вывод текущих показаний датчиков с помощью printSensorsValues().

Строки 25 – 26: Выполнение автоматического режима, при automaticModeState == true.

Строки 27 – 29: Проверка состояния WiFi. При отсутствии соединения вызов setupWiFi().

Строка 31: Обработка сообщений Telegram-бота через loopTelegram().

Строка 32: Задержка 500 мс для предотвращения перегрузки цикла.

## 5.5 Файл devices.h

Строки 1 – 6: Подключение библиотек для работы с I2C, датчиками BH1750, BME280, DS18B20 и конфигурацией.

Строки 8 – 11: Создание объектов bme, lightMeter, oneWire, sensors для датчиков.

Строки 13 – 23: Функция setupLeds() для инициализации пинов светодиодов и их начального состояния (LOW).

Строки 25 – 32: Функция flash\_led() для управления миганием светодиодов с заданным количеством вспышек и длительностью.

Строки 34 – 75: Функции для настройки и считывания данных с датчиков: setupLightmeter(), setupBme(), setupSoilMoistureSensor(), setupDs18b20(), getLightLevel(), getTemperature(), getPressure(), getHumidity(), getSoilMoisturePercent(), getSoilTemperature().

Строки 77 – 89: Функции printLightLevel(), printBmeValues(), printSoilMoisturePercent(), printSoilTemperature() для вывода показаний датчиков в последовательный порт.

Строки 91 – 111: Функция printSensorsValues() для вывода всех данных датчиков.

Строки 113 – 134: Функция checkConnectsSensors() для проверки подключения датчиков и обновления флагов.

Строки 136 – 143: Функция highlightHeaterWork() для индикации состояния нагревателя через жёлтый светодиод.

Строки 145 – 156: Функция setupDevices() для инициализации пинов управления устройствами (насос, нагреватель, охладитель, фитолента).

## 5.6 Файл connectWiFi.h

Строки 1 – 4: Подключение библиотек для работы с WiFi.

Строки 6 – 10: Функция printWifiStatus() для вывода информации о текущем состоянии WiFi.

Строки 12 – 29: Функция initWiFi(), которая устанавливает соединение с заданной сетью и управляет состоянием красного светодиода во время подключения.

Строки 31 – 54: Функция scanWiFi(), которая сканирует доступные сети и ищет заданный SSID.

Строки 56 – 66: Функция setupWiFi(), которая отключает WiFi и запускает сканирование для подключения.

Строки 68 – 76: Функция isConnectWiFi(), которая проверяет текущее состояние WiFi и обновляет состояние светодиодов.

## 5.7 Файл tgCommunication.h

Строки 1 – 5: Подключение необходимых библиотек для работы с Telegram API и WiFi.

Строки 7 – 9: Создание объекта UniversalTelegramBot с использованием токена бота.

Строки 11 – 30: Функция sendStats(), которая формирует и отправляет статистику о состоянии датчиков в указанный чат.

Строки 32 – 86: Функция handleNewMessages(), которая обрабатывает входящие сообщения и выполняет соответствующие команды, такие как управление устройствами и запрос статистики.

Строки 88 – 90: Функция setupTelegram(), которая устанавливает сертификат для защищённого соединения.

Строки 92 – 103: Функция loopTelegram(), которая проверяет наличие новых сообщений и обрабатывает их.

## 5.8 Файл automaticMode.h

Строки 1 – 4: Подключение заголовочных файлов для работы с устройствами, Telegram и настройками.

Строка 6: Определение константы MAX\_MEASUREMENTSдля количества измерений в периоде усреднения.

Строки 8–16: Определение структуры SensorDataдля хранения измерений датчиков.

Строка 18: Создание глобальной структуры sensorData.

Строки 20–21: Переменные для отслеживания времени последнего нагрева и полива.

Строки 23–34: Функция initSensorData() для инициализации структуры sensorData.

Строки 36–53: Функция collectSensorData() для периодического сбора данных с датчиков.

Строки 55–73: Функция calculateAverages()для вычисления средних значений данных датчиков.

Строки 75–90: Функция controlLightLevel()для управления фитолентой на основе уровня освещённости.

Строки 92–106: Функция controlTemperature()для управления охладителем на основе температуры воздуха.

Строки 108–129: Функция controlSoilTemperature() для управления нагревателем с учётом паузы и длительности нагрева.

Строки 131–145: Функция controlHumidity() для управления охладителем на основе влажности воздуха.

Строки 147–184: Функция controlSoilMoisture()для импульсного полива и управления охладителем при высокой влажности почвы.

Строки 186–204: Функция automaticMode()для выполнения автоматического режима с вызовом функций управления.

## 5.9 Файл config.h

Строки 3–10: Определение пинов для датчиков и устройств (влажность почвы, температура почвы, фитолента, насос, охладитель, нагреватель, светодиоды).

Строки 12–13: Определение SSID и пароля WiFi.

Строки 15–16: Определение токена Telegram-бота и ID чата.

Строки 18–21: Флаги для отслеживания подключения датчиков.

Строки 23–27: Флаги для отслеживания состояния устройств и автоматического режима.

Строки 29–31: Флаги для состояния светодиодов.

Строки 33–34: Константы для калибровки датчика влажности почвы AirValue, WaterValue.

# 6 Проектирование корпуса

Проект умной мини-теплицы предполагает реализацию нескольких корпусных модулей: корпус теплицы, корпус модуля с основной электроникой и резервуар для воды. Все модули располагаются на общей подставке.

## 6.1 Разработка корпуса теплицы

В качестве материала для корпуса теплицы было выбрано оргстекло. Корпус спроектирован в виде домика с откидывающейся крышей для простоты и удобства доступа к контейнерам с землей. К коньку крыши внутри теплицы прикреплена стойками платформа с фитолентой. На передней и задней стенках выполнены отверстия для вентиляции, к которым прикреплены кулеры на ввод и вывод воздуха. Также на передней панели с внешней стороны крепится мини-насос и вводятся трубки для полива. Всю нижнюю часть корпуса занимает нагревающая панель, выполненная из плитки и нагревающего провода. Корпус теплицы представлен на рисунке 6.1.

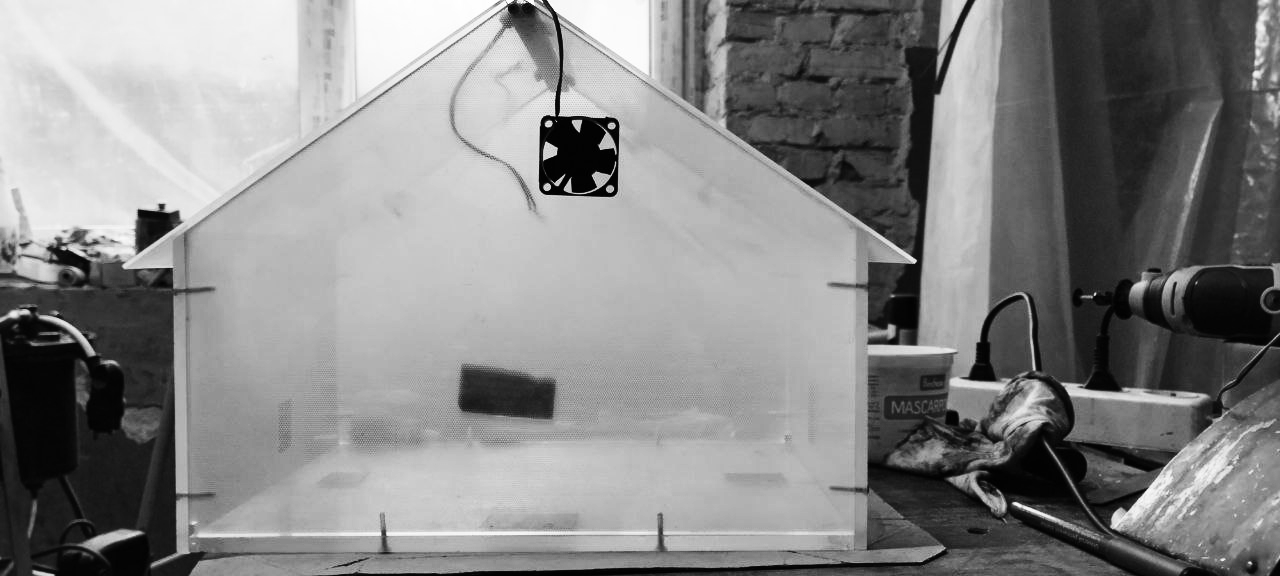


Рисунок 6.1 – Корпус теплицы

## 6.2 Разработка корпуса модуля с электроникой

Модуль с электроникой должен защищать платы от попадания воды и загрязнений при эксплуатации теплицы. Также он должен быть относительно прозрачным для реализации функции индикации подключения к сети и работы нагревающей панели. В качестве материала для выполнения данного корпуса было выбрано оргстекло,к которому металлическими стяжками или стойками прикреплены все компоненты. Корпус данного модуля представлен на рисунке 6.2.

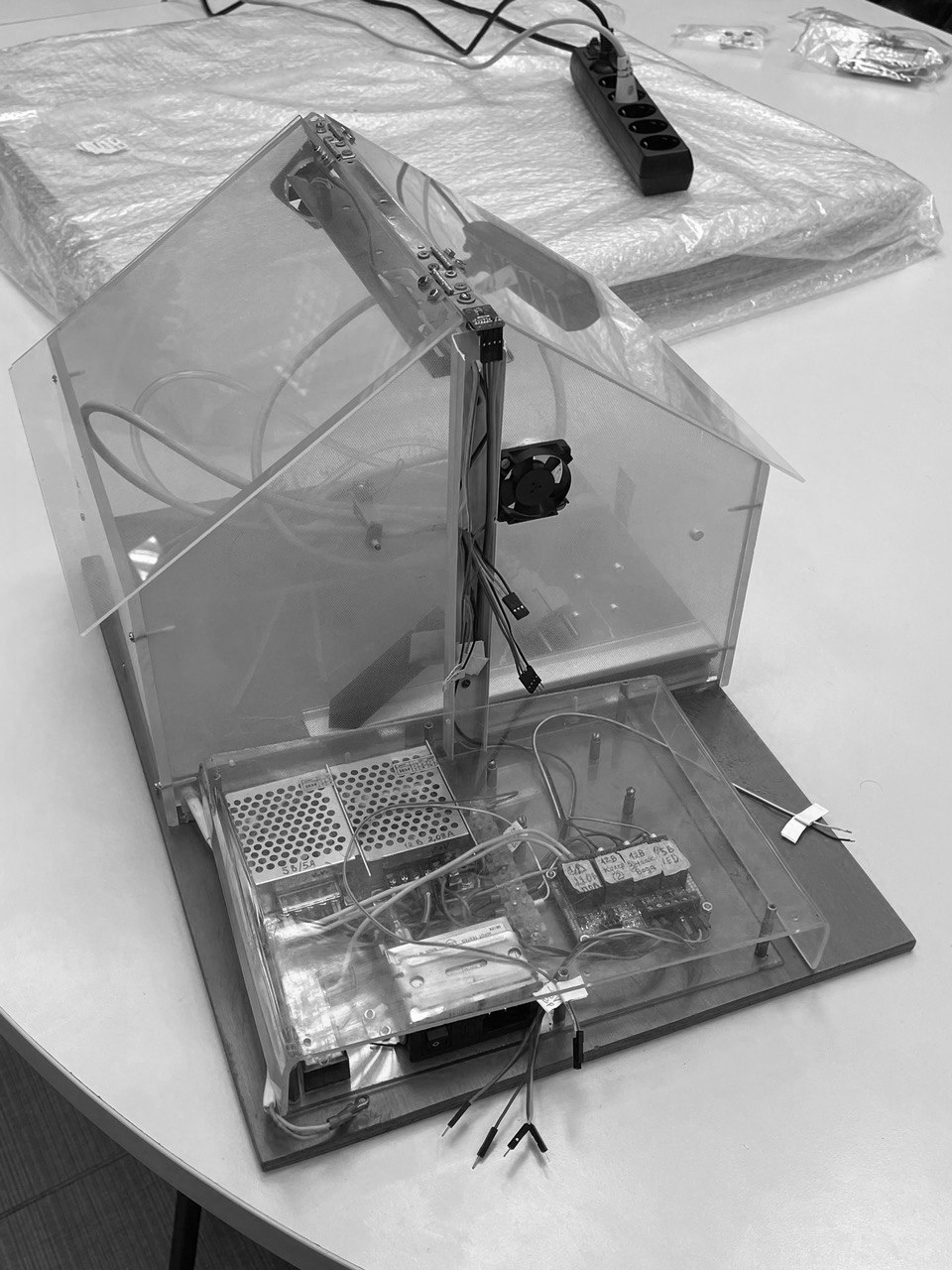


Рисунок 6.2 – Корпус модуля с электроникой

## 6.3 Общий вид проекта

Итог сборки всех модулей проекта представлен на рисунке 6.3.

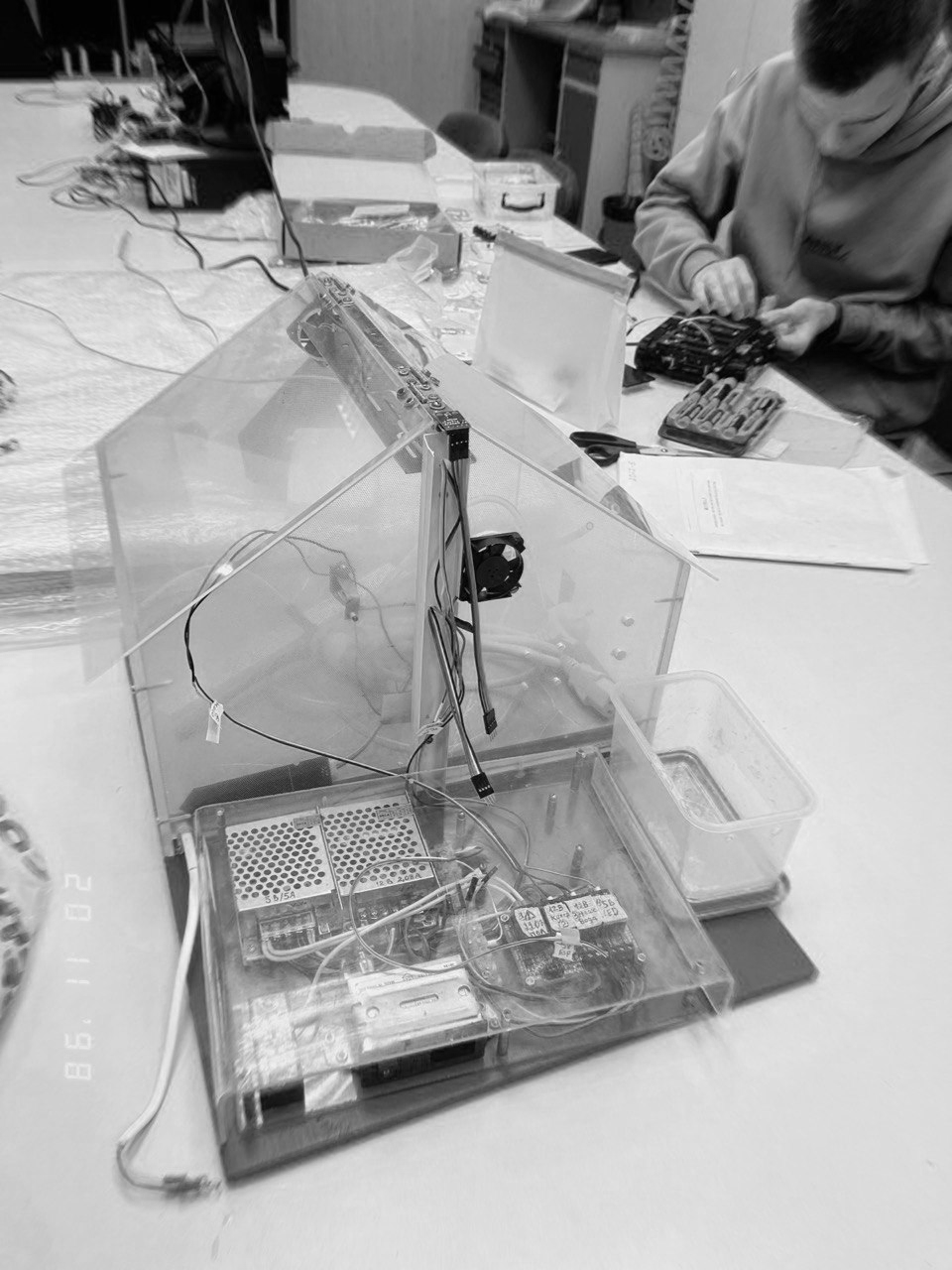


Рисунок 6.3 – Общий вид проекта

# 7 Проектирование платы

## 7.1 Проектирование платы

В процессе проектирования платы была использована платформа EasyEDA [16]. Результат проектирования представлен на рисунке 7.1.

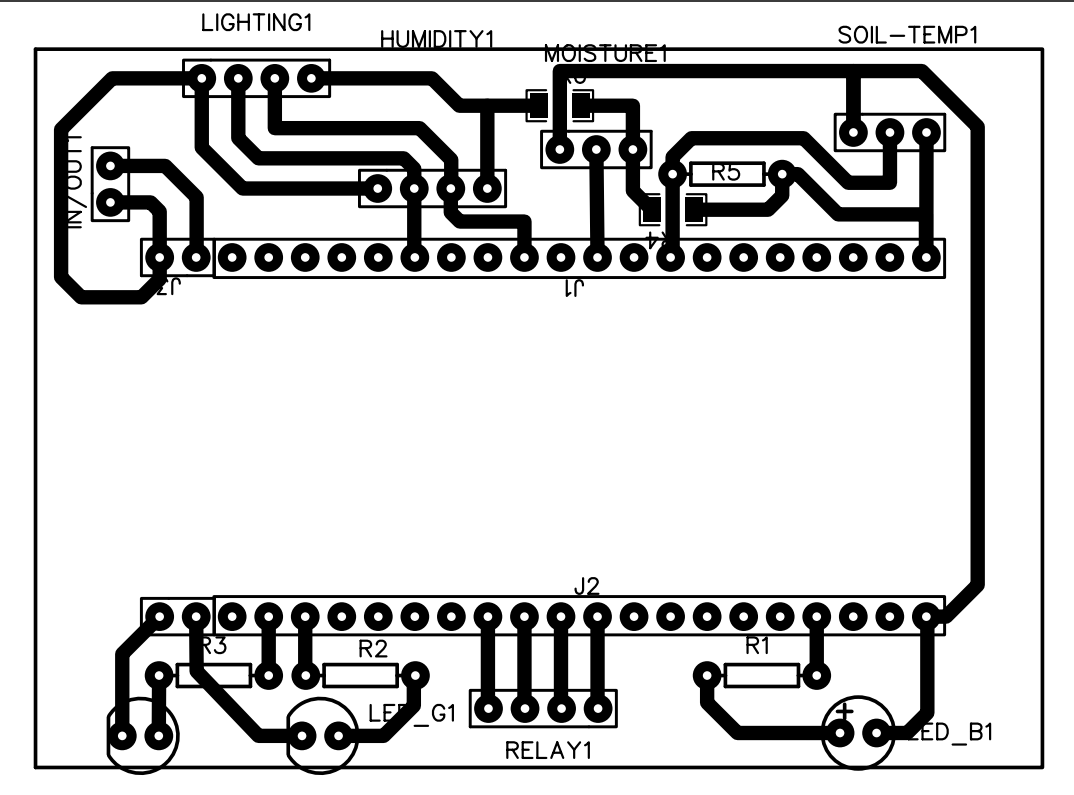


Рисунок 7.1 – Результат проектирования платы

## 7.2 Производство платы

Плата изготавливалась вручную лутом. Результат представлен на рисунке 7.2.

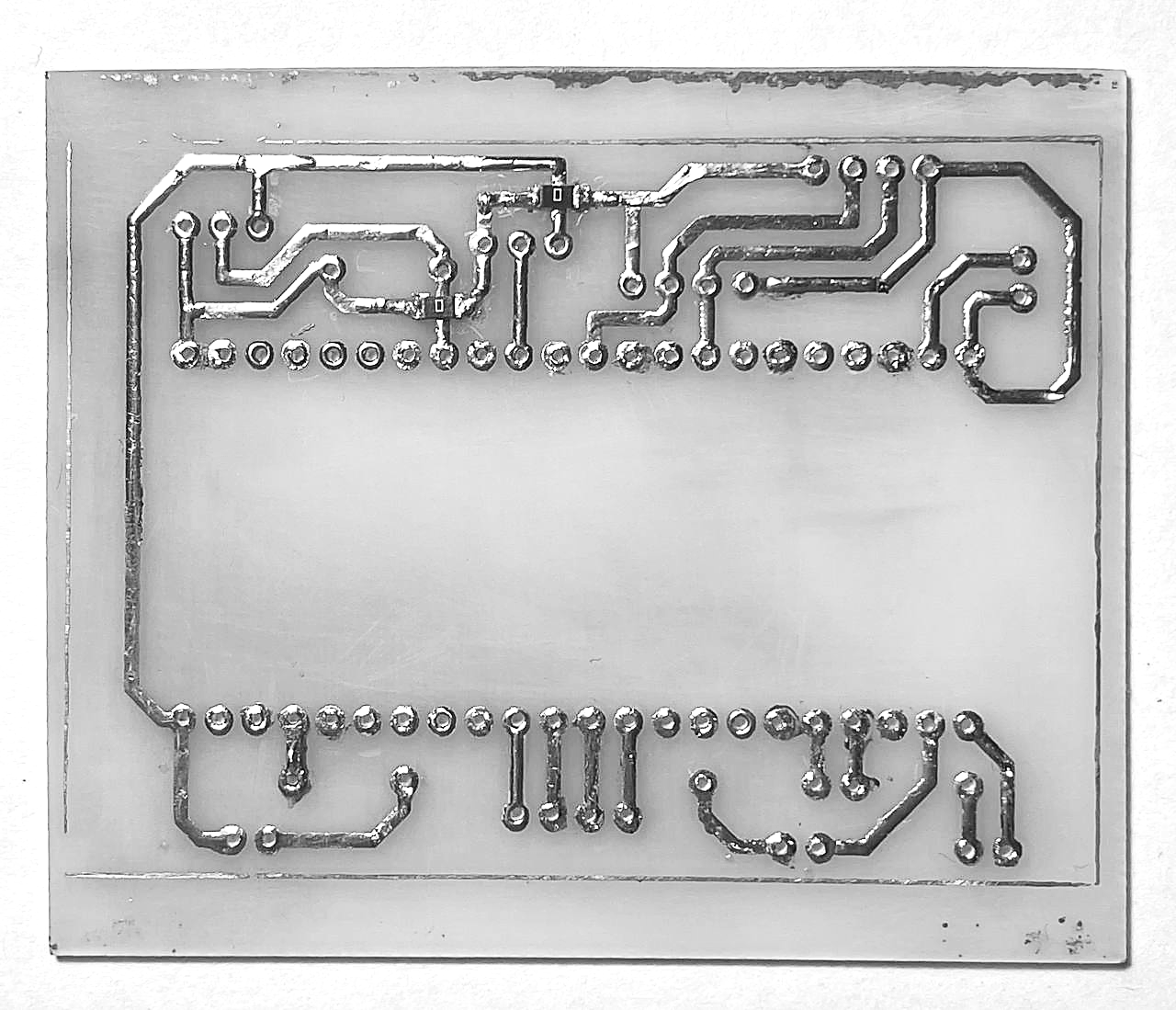


Рисунок 7.2 – Результат изготовления платы

# 8 Руководство пользователя

## 8.1 Инструкция к физическим модулям

Для включения умной теплицы необходимо вставить вилку провода питания в розетку 220 В, после чего переведите тумблер сетевого фильтра в положение «I». Сетевой фильтр с тумблером отображен на рисунке 8.1.

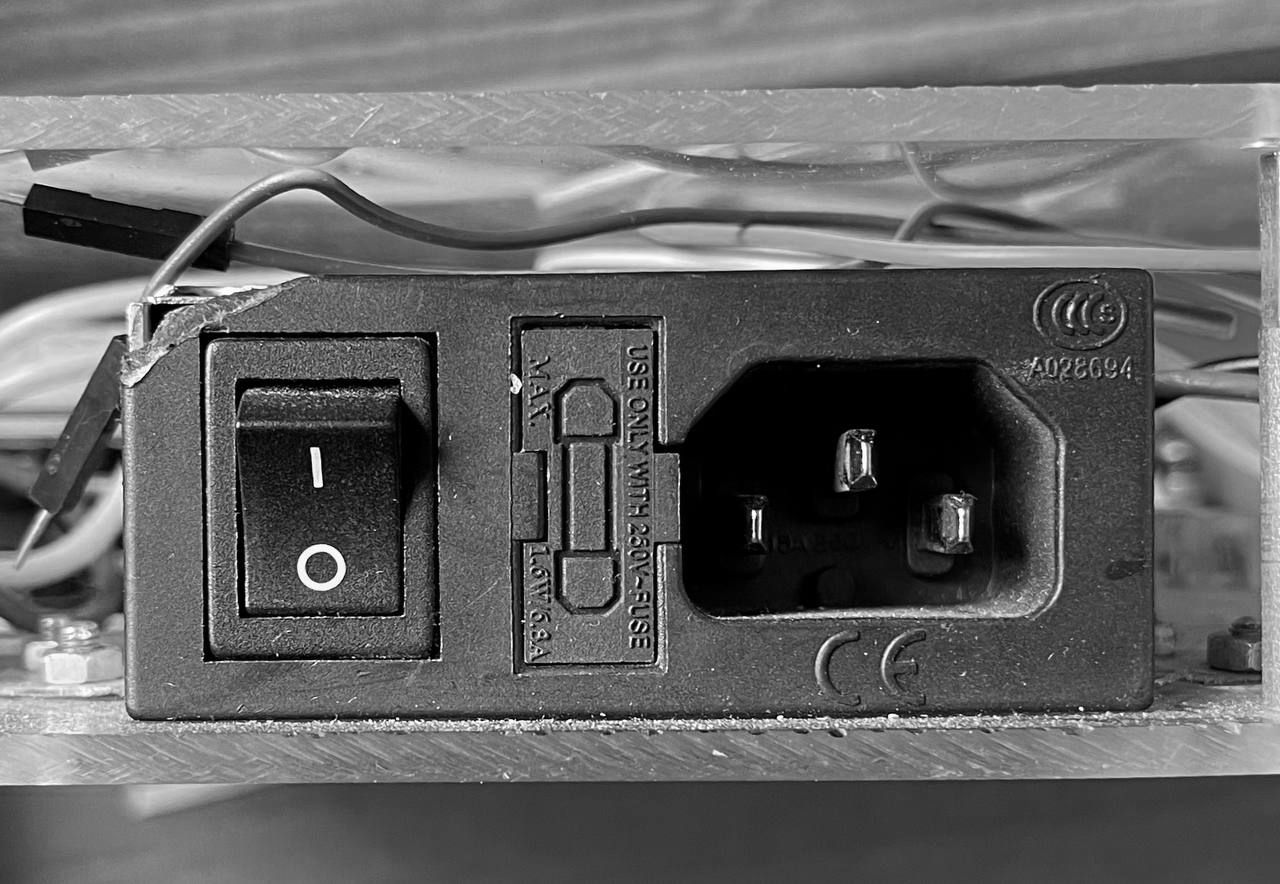


Рисунок 8.1 – Сетевой фильтр устройства

Далее на плате с микроконтроллером начнет мигать красный светодиод, который отображает подключение к сети. Медленное мигание отображает процесс поиска сети, быстрое мигание отображает процесс подключения к найденной сети. В случае успеха красный светодиод прекратит индикацию, загорится зеленый светодиод.

Желтый светодиод на плате отображает работу нагревательного элемента. В случае, когда нагревательный элемент активен, светодиод горит.

Для выключения устройства умной теплицы переведите тумблер сетевого фильтра в положение «0». Достаньте вилку из розетки.

## 8.2 Инструкция к telegram-боту

Управление системой осуществляется через telegram-бот.

В качестве приветствия бот присылает сообщение с описанием всех команд для управления теплицей. Скриншот сообщения представлен на рисунке 8.2. Также это сообщение присылается после команды /help.

Запрос текущих показаний датчиков осуществляется командой /sensors\_stats. Пример ответа представлен на рисунке 8.3. В случае, если датчик выдает некорректные показатели или отсутствует, будет выведена соответствующая информация.

Команда /devices\_status используется для отображения состояния всех устройств. Для управления состоянием устройств используются команды /heater\_on и /heater\_off для нагревательного элемента, /cooler\_on и /cooler\_off для кулеров, /lighting\_on и /lighting\_off для фитоленты, /pump\_on и /pump\_off для насоса.

Автоматический режим теплицы включается командой /automatic\_mode\_on. В автоматическом режиме система самостоятельно регулирует параметры теплицы на основе настроек из файла settings.h. Фитолента включается, если уровень ниже 50 люкс, и выключается при превышении порога с гистерезисом 10 люкс. Кулеры включаются при превышении 35 °C и выключаются с гистерезисом 3 °C. Нагреватель включается при температуре ниже 18 °C на время до 5 минут с паузой 1 минута. Кулеры включаются при влажности выше 80 % и выключаются с гистерезисом 5 %. Импульсный полив (до 5 импульсов по 4 секунды с паузой 10 секунд) запускается при влажности ниже 20 %. Охладитель включается при влажности выше 80 %.

Бот отправляет уведомления о действиях в автоматическом режиме (например, включение/выключение устройств).

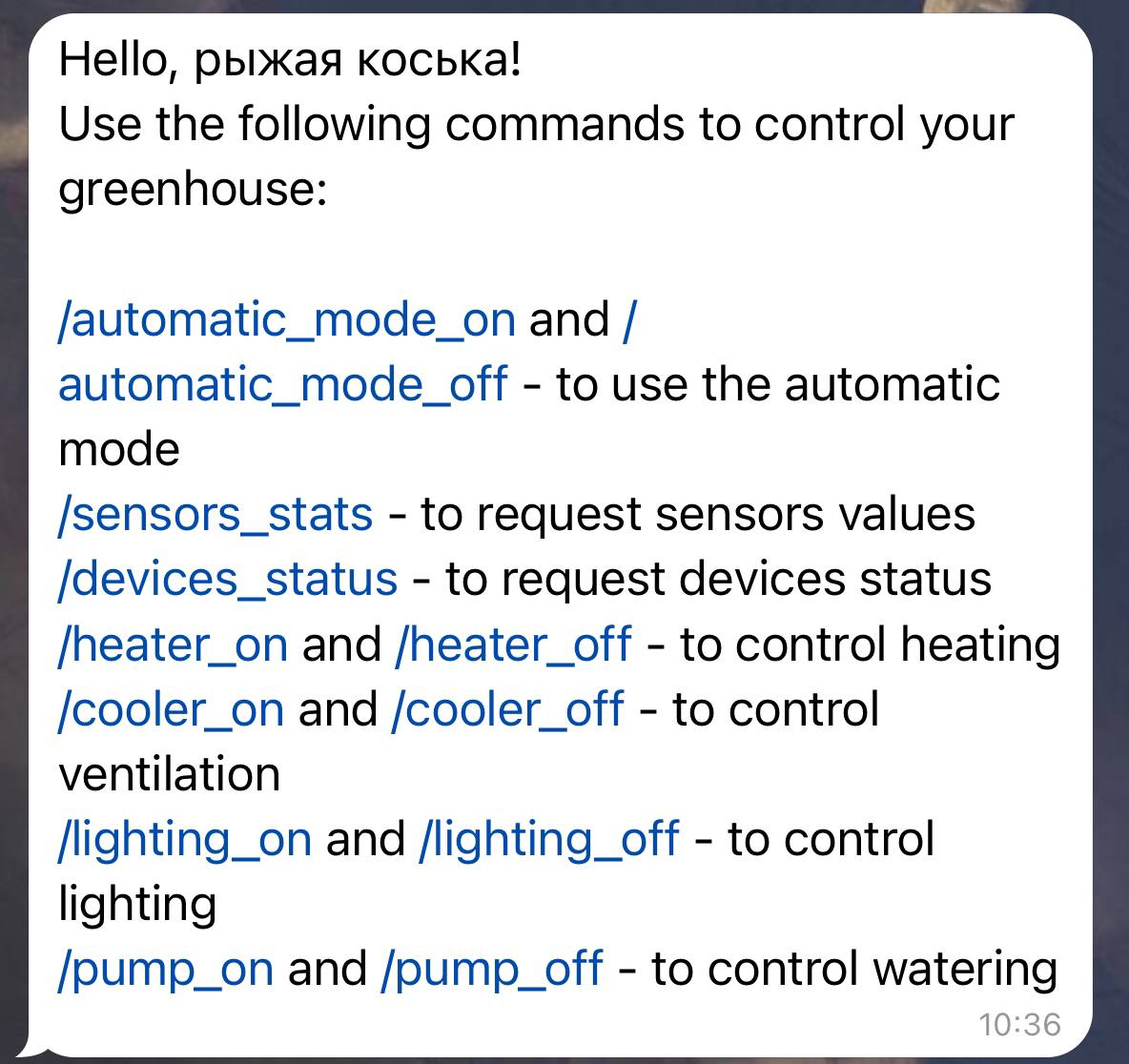


Рисунок 8.2 – Приветственное сообщение

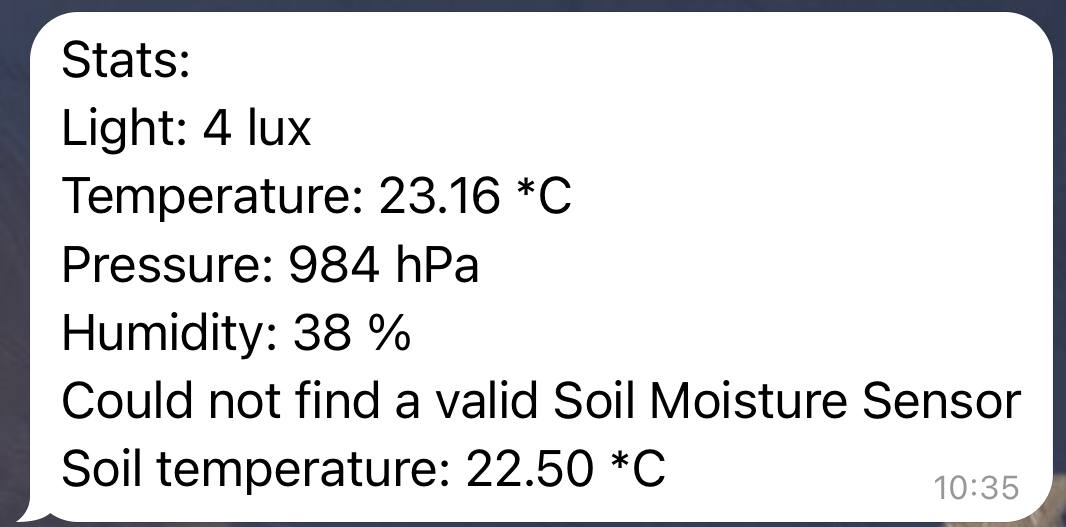


Рисунок 8.3 – Вывод показателей датчиков

Заключение

Проект умной теплицы представляет собой решение для создания оптимальных условий для роста растений, обеспечивая высокий уровень автоматизации и контроля. Реализованная система эффективно мониторит и управляет ключевыми параметрами микроклимата, такими как температура, влажность и освещенность, что значительно упрощает уход за растениями и повышает их урожайность.

В рамках текущей реализации был разработан Telegram-бот, который позволяет пользователям управлять поливом, нагревом, проветриванием и регулировать освещение с помощью фитоленты. Кроме того, бот предоставляет доступ к данным с датчиков, что позволяет оперативно отслеживать состояние теплицы.

Уже достигнутые результаты демонстрируют, что система работает корректно и надежно, что позволяет пользователям легко контролировать условия в теплице и вносить необходимые изменения. Это решение делает процесс ухода за растениями более доступным и эффективным, что особенно важно для тех, кто хочет заниматься садоводством в домашних условиях.

В перспективе проект может быть значительно расширен и улучшен. Одним из направлений развития является добавление дополнительных датчиков, которые позволят еще более точно контролировать условия в теплице.

Также можно внедрить системы автоматического регулирования полива на основе данных о влажности почвы и температуре воздуха. Автоматическая система удобрения, работающая по аналогичному принципу, позволит экономить ресурсы и поддерживать здоровье растений на оптимальном уровне.

Разработка мобильного приложения для управления теплицей также является перспективным направлением. Это позволит пользователям получать уведомления о состоянии растений, изменениях в микроклимате и необходимости вмешательства. Интеграция с системами "умного дома" еще больше упростит управление теплицей, позволяя пользователям контролировать её состояние через единый интерфейс.

Таким образом, умная теплица имеет большой потенциал для дальнейшего развития и модернизации. Расширение функциональности системы и интеграция новых технологий открывает новые возможности для повышения эффективности и удобства в выращивании растений.

Список использованных источников

[1] Умная теплица GrowUp [Электронный ресурс]. – Режим доступа: https://vikowest.by/product/umnaya-teplitsa-growup-belaya/?srsltid=AfmBOoqzWcFGmiKnjfHIV1rCh2KPXXwEkQo3aU8Zp2rM5hyfnKYM\_X76&sku=162620.

[2] Гидропоника - система Умный сад [Электронный ресурс]. – Режим доступа: https://wmctools.by/gidroponika-sistema-umnyy-sad/.

[3] Профессиональная домашняя садовая ферма L-Box [Электронный ресурс]. – Режим доступа: https://vegebox.ru/l-box-1.

[4] Умный сад с подсветкой iGarden PRO [Электронный ресурс]. – Режим доступа: https://igarden.by/product/umnyjj-sad-s-podsvetkojj-igarden-pro/.

[5] Espressif Systems [Электронный ресурс]. – Режим доступа: https://www.espressif.com/.

[6] Raspberry Pi [Электронный ресурс]. – Режим доступа: https://www.raspberrypi.org/.

[7] Nordic Semiconductor [Электронный ресурс]. – Режим доступа: https://www.nordicsemi.com/.

[8] STMicroelectronics [Электронный ресурс]. – Режим доступа: https://www.st.com/.

[9] Сравнение датчиков DHT11, DHT22 и DHT21 [Электронный ресурс]. – Режим доступа: https://voltiq.ru/dht11-dht22-and-dht21/.

[10] Датчик влажности BME280 [Электронный ресурс]. – Режим доступа: https://www.bosch-sensortec.com/products/environmental-sensors /humidity-sensors-bme280/.

[11] Датчик интенсивности света GY-302 [Электронный ресурс]. – Режим доступа: https://3d-diy.ru/blog/datchik-intensivnosti-sveta-gy-302-bh1750/.

[12] Датчик интенсивности света TSL2561 [Электронный ресурс]. – Режим доступа: https://3d-diy.ru/product/tsifrovoy-datchik-osveshchennosti-tsl2561-gy-2561.

[13] Фоторезистор GL5528 [Электронный ресурс]. – Режим доступа: https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/SEN-09088.pdf.

[14] Емкостный датчик влажности почвы [Электронный ресурс]. – Режим доступа: https://www.robotics.org.za/CAP-SW-12.

[15] Датчик температуры почвы DS18B20 [Электронный ресурс]. – Режим доступа: https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf.

[16] Платформа EasyEDA [Электронный ресурс]. – Режим доступа: https://easyeda.com/.

Приложение А

(*обязательное*)

**Схема электрическая структурная**

Приложение Б

(*обязательное*)

**Схема электрическая функциональная**

Приложение В

(*обязательное*)

**Схема электрическая принципиальная**

Приложение Г

(*обязательное*)

**Схема программы**

Приложение Д

(*обязательное*)

**Листинг кода**

Файл automaticMode.h:

001 #pragma once

002 #include "devices.h"

003 #include "tgCommunication.h"

004 #include "settings.h"

005 // Параметры для усреднения

006 #define MAX\_MEASUREMENTS (AVERAGE\_PERIOD / MEASUREMENT\_INTERVAL) // Количество измерений в периоде

007 // Структура для хранения измерений

008 struct SensorData

009 {

010 float lightLevel[MAX\_MEASUREMENTS];

011 float temperature[MAX\_MEASUREMENTS];

012 float soilTemperature[MAX\_MEASUREMENTS];

013 float humidity[MAX\_MEASUREMENTS];

014 float soilMoisture[MAX\_MEASUREMENTS];

015 int count;

016 unsigned long lastMeasurementTime;

017 };

018 // Глобальная структура для данных

019 SensorData sensorData = {{0}, {0}, {0}, {0}, {0}, 0, 0};

020 // Переменные для отслеживания времени последнего действия

021 unsigned long lastHeatingTime = 0;

022 unsigned long lastWateringTime = 0;

023 // Инициализация структуры

024 void initSensorData()

025 {

026 sensorData.count = 0;

027 sensorData.lastMeasurementTime = millis();

028 for (int i = 0; i < MAX\_MEASUREMENTS; i++)

029 {

030 sensorData.lightLevel[i] = 0;

031 sensorData.temperature[i] = 0;

032 sensorData.soilTemperature[i] = 0;

033 sensorData.humidity[i] = 0;

034 sensorData.soilMoisture[i] = 0;

035 }

036 }

037 // Сбор данных с датчиков

038 void collectSensorData()

039 {

040 if (millis() - sensorData.lastMeasurementTime >= MEASUREMENT\_INTERVAL)

041 {

042 int index = sensorData.count % MAX\_MEASUREMENTS;

043 if (isConnectedBh1750)

044 sensorData.lightLevel[index] = getLightLevel();

045 if (isConnectedBme280)

046 {

047 sensorData.temperature[index] = getTemperature();

048 sensorData.humidity[index] = getHumidity();

049 }

050 if (isConnectedDs18b20)

051 sensorData.soilTemperature[index] = getSoilTemperature();

052 if (isConnectedMoistureSensor)

053 sensorData.soilMoisture[index] = getSoilMoisturePercent();

054 sensorData.count++;

055 sensorData.lastMeasurementTime = millis();

056 }

057 }

058 // Вычисление средних значений

059 void calculateAverages(float &avgLight, float &avgTemp, float &avgSoilTemp, float &avgHum, float &avgSoil)

060 {

061 avgLight = 0;

062 avgTemp = 0;

063 avgSoilTemp = 0;

064 avgHum = 0;

065 avgSoil = 0;

066 int count = min(sensorData.count, MAX\_MEASUREMENTS);

067 if (count == 0)

068 return;

069 for (int i = 0; i < count; i++)

070 {

071 avgLight += sensorData.lightLevel[i];

072 avgTemp += sensorData.temperature[i];

073 avgSoilTemp += sensorData.soilTemperature[i];

074 avgHum += sensorData.humidity[i];

075 avgSoil += sensorData.soilMoisture[i];

076 }

077 avgLight /= count;

078 avgTemp /= count;

079 avgSoilTemp /= count;

080 avgHum /= count;

081 avgSoil /= count;

082 }

083 // Проверка освещенности (без ограничений по длительности)

084 void controlLightLevel(float avgLight)

085 {

086 static bool lightOn = false;

087 if (avgLight < minLightLevel - lightHysteresis && !lightOn)

088 {

089 lightingState = HIGH;

090 digitalWrite(LED\_STRIP\_PIN, lightingState);

091 lightOn = true;

092 bot.sendMessage(CHAT\_ID, "Average light level too low (" + String(avgLight) + " lux) - lighting ON", "");

093 }

094 else if (avgLight > minLightLevel + lightHysteresis && lightOn)

095 {

096 lightingState = LOW;

097 digitalWrite(LED\_STRIP\_PIN, lightingState);

098 lightOn = false;

099 bot.sendMessage(CHAT\_ID, "Average light level restored (" + String(avgLight) + " lux) - lighting OFF", "");

100 }

101 }

102 // Проверка температуры воздуха

103 void controlTemperature(float avgTemp)

104 {

105 static bool coolerOn = false;

106 if (avgTemp > maxTemperature + tempHysteresis && !coolerOn)

107 {

108 coolerState = HIGH;

109 digitalWrite(COOLER\_PIN, coolerState);

110 coolerOn = true;

111 bot.sendMessage(CHAT\_ID, "Average air temperature too high (" + String(avgTemp) + " \*C) - cooler ON", "");

112 }

113 else if (avgTemp < maxTemperature - tempHysteresis && coolerOn)

114 {

115 coolerState = LOW;

116 digitalWrite(COOLER\_PIN, coolerState);

117 coolerOn = false;

118 bot.sendMessage(CHAT\_ID, "Average air temperature restored (" + String(avgTemp) + " \*C) - cooler OFF", "");

119 }

120 }

121 // Проверка температуры почвы (нагрев интервалами, не дольше 5 минут)

122 void controlSoilTemperature(float avgSoilTemp)

123 {

124 static bool heaterOn = false;

125 if (millis() - lastHeatingTime < heatingPause && heaterOn)

126 return; // Пауза между циклами нагрева

127 int heatingDuration = heatingFactor \* (minSoilTemperature - avgSoilTemp); // Длительность пропорциональна отклонению

128 heatingDuration = constrain(heatingDuration, 0, maxHeatingDuration);

129 if (avgSoilTemp < minSoilTemperature - soilTempHysteresis && !heaterOn)

130 {

131 heaterState = HIGH;

132 digitalWrite(HEATER\_PIN, heaterState);

133 highlightHeaterWork();

134 heaterOn = true;

135 lastHeatingTime = millis();

136 bot.sendMessage(CHAT\_ID, "Average soil temperature too low (" + String(avgSoilTemp) + " \*C) - heater ON for " + String(heatingDuration / 1000) + " sec", "");

137 delay(heatingDuration);

138 heaterState = LOW;

139 digitalWrite(HEATER\_PIN, heaterState);

140 highlightHeaterWork();

141 bot.sendMessage(CHAT\_ID, "Heater OFF after heating", "");

142 }

143 else if (avgSoilTemp > minSoilTemperature + soilTempHysteresis && heaterOn)

144 {

145 heaterState = LOW;

146 digitalWrite(HEATER\_PIN, heaterState);

147 highlightHeaterWork();

148 heaterOn = false;

149 bot.sendMessage(CHAT\_ID, "Average soil temperature restored (" + String(avgSoilTemp) + " \*C) - heater OFF", "");

150 }

151 }

152 // Проверка влажности воздуха

153 void controlHumidity(float avgHum)

154 {

155 static bool coolerOn = false;

156 if (avgHum > maxHumidity + humHysteresis && !coolerOn)

157 {

158 coolerState = HIGH;

159 digitalWrite(COOLER\_PIN, coolerState);

160 coolerOn = true;

161 bot.sendMessage(CHAT\_ID, "Average humidity too high (" + String(avgHum) + " %) - cooler ON", "");

162 }

163 else if (avgHum < maxHumidity - humHysteresis && coolerOn)

164 {

165 coolerState = LOW;

166 digitalWrite(COOLER\_PIN, coolerState);

167 coolerOn = false;

168 bot.sendMessage(CHAT\_ID, "Average humidity restored (" + String(avgHum) + " %) - cooler OFF", "");

169 }

170 }

171 // Проверка влажности почвы (импульсный полив)

172 void controlSoilMoisture(float avgSoil)

173 {

174 static bool watering = false;

175 if (millis() - lastWateringTime < wateringPause && watering)

176 return; // Пауза между циклами полива

177 int numPulses = wateringFactor \* (minSoilMoisture - avgSoil); // Количество импульсов пропорционально отклонению

178 numPulses = constrain(numPulses, 0, maxWateringPulses);

179 if (avgSoil < minSoilMoisture - soilHysteresis && !watering)

180 {

181 watering = true;

182 lastWateringTime = millis();

183 bot.sendMessage(CHAT\_ID, "Average soil Moisture too low (" + String(avgSoil) + " %) - starting " + String(numPulses) + " watering pulses", "");

184 for (int i = 0; i < numPulses; i++)

185 {

186 pumpState = HIGH;

187 digitalWrite(PUMP\_PIN, pumpState);

188 bot.sendMessage(CHAT\_ID, "Pump ON for pulse " + String(i + 1), "");

189 delay(pulseWateringDuration);

190 pumpState = LOW;

191 digitalWrite(PUMP\_PIN, pumpState);

192 bot.sendMessage(CHAT\_ID, "Pump OFF after pulse " + String(i + 1), "");

193 if (i < numPulses - 1)

194 delay(wateringPause); // Пауза между импульсами

195 }

196 bot.sendMessage(CHAT\_ID, "Finished watering", "");

197 }

198 else if (avgSoil > minSoilMoisture + soilHysteresis && watering)

199 {

200 watering = false;

201 pumpState = LOW;

202 digitalWrite(PUMP\_PIN, pumpState);

203 bot.sendMessage(CHAT\_ID, "Average soil Moisture restored (" + String(avgSoil) + " %) - watering OFF", "");

204 }

205 static bool coolerOn = false;

206 if (avgSoil > maxSoilMoisture + soilHysteresis && !coolerOn)

207 {

208 coolerState = HIGH;

209 digitalWrite(COOLER\_PIN, coolerState);

210 coolerOn = true;

211 bot.sendMessage(CHAT\_ID, "Average soil Moisture too high (" + String(avgSoil) + " %) - cooler ON", "");

212 }

213 else if (avgSoil < maxSoilMoisture - soilHysteresis && coolerOn)

214 {

215 coolerState = LOW;

216 digitalWrite(COOLER\_PIN, coolerState);

217 coolerOn = false;

218 bot.sendMessage(CHAT\_ID, "Average soil Moisture restored (" + String(avgSoil) + " %) - cooler OFF", "");

219 }

220 }

221 // Основная функция автоматического режима

222 void automaticMode()

223 {

224 if (!automaticModeState)

225 return;

226 collectSensorData();

227 if (millis() - sensorData.lastMeasurementTime >= AVERAGE\_PERIOD)

228 {

229 float avgLight, avgTemp, avgSoilTemp, avgHum, avgSoil;

230 calculateAverages(avgLight, avgTemp, avgSoilTemp, avgHum, avgSoil);

231 if (isConnectedBh1750)

232 controlLightLevel(avgLight);

233 if (isConnectedBme280)

234 {

235 controlTemperature(avgTemp);

236 controlHumidity(avgHum);

237 }

238 if (isConnectedDs18b20)

239 controlSoilTemperature(avgSoilTemp);

240 if (isConnectedMoistureSensor)

241 controlSoilMoisture(avgSoil);

242 // Сброс счетчика после обработки

243 sensorData.count = 0;

244 }

245 }

Файл config.h:

001 #pragma once

002 #define MOISTURE\_SENSOR\_PIN 17

003 #define SOIL\_TEMP\_SENSOR\_PIN 15

004 #define LED\_STRIP\_PIN 35

005 #define PUMP\_PIN 36

006 #define COOLER\_PIN 37

007 #define HEATER\_PIN 38

008 #define YELLOW\_LED\_PIN 1

009 #define RED\_LED\_PIN 20

010 #define GREEN\_LED\_PIN 21

011 // GPIO 8 (SDA)

012 // GPIO 9 (SCL)

013 char \*ssid = "iPhone Ann";

014 char \*password = "12345678910";

015 #define BOT\_TOKEN "8127877503:AAGpV7Wk6JFMBG-SFpz3vonvdET0n0ZKUTk"

016 #define CHAT\_ID "922443025"

017 bool isConnectedMoistureSensor = false;

018 bool isConnectedBme280 = false;

019 bool isConnectedBh1750 = false;

020 bool isConnectedDs18b20 = false;

021 bool heaterState = LOW;

022 bool coolerState = LOW;

023 bool lightingState = LOW;

024 bool pumpState = LOW;

025 bool automaticModeState = false;

026 bool redLedState = LOW;

027 bool greenLedState = LOW;

028 bool yellowLedState = LOW;

029 const int AirValue = 590;

030 const int WaterValue = 360;

Файл connectWiFi.h:

001 #pragma once

002 #include <WiFi.h>

003 #include "config.h"

004 #include "devices.h"

005 void printWifiStatus()

006 {

007 Serial.print("SSID: ");

008 Serial.println(WiFi.SSID());

009 IPAddress ip = WiFi.localIP();

010 Serial.print("IP Address: ");

011 Serial.println(ip);

012 long rssi = WiFi.RSSI();

013 Serial.print("signal strength (RSSI):");

014 Serial.print(rssi);

015 Serial.println(" dBm");

016 }

017 void initWiFi()

018 {

019 redLedState = HIGH;

020 digitalWrite(RED\_LED\_PIN, redLedState);

021 WiFi.mode(WIFI\_STA);

022 WiFi.begin(ssid, password);

023 Serial.print("Connecting to WiFi ");

024 Serial.println(ssid);

025 while (WiFi.status() != WL\_CONNECTED)

026 {

027 flash\_led(RED\_LED\_PIN, 1, 300);

028 Serial.print("Status: ");

029 Serial.println(WiFi.status());

030 redLedState = HIGH;

031 digitalWrite(RED\_LED\_PIN, redLedState);

032 }

033 redLedState = LOW;

034 digitalWrite(RED\_LED\_PIN, redLedState);

035 greenLedState = HIGH;

036 digitalWrite(GREEN\_LED\_PIN, greenLedState);

037 Serial.println("Successful connection to WiFi!");

038 printWifiStatus();

039 }

040 bool scanWiFi(char \*ssid)

041 {

042 bool isFoundSsid = false;

043 Serial.println("scan start");

044 int n = WiFi.scanNetworks();

045 Serial.println("scan done");

046 if (n == 0)

047 {

048 Serial.println("no networks found");

049 }

050 else

051 {

052 Serial.print(n);

053 Serial.println("networks found");

054 for (int i = 0; i < n; ++i)

055 {

056 Serial.print(i + 1);

057 Serial.print(": ");

058 Serial.print(WiFi.SSID(i));

059 Serial.print(" (");

060 Serial.print(WiFi.RSSI(i));

061 Serial.print(")");

062 Serial.println((WiFi.encryptionType(i) == WIFI\_AUTH\_OPEN) ? " " : "\*");

063 delay(10);

064 if (!isFoundSsid)

065 isFoundSsid = ((WiFi.SSID(i)) == ssid);

066 }

067 }

068 Serial.println("");

069 return isFoundSsid;

070 }

071 void setupWiFi()

072 {

073 WiFi.mode(WIFI\_STA);

074 WiFi.disconnect();

075 while (!scanWiFi(ssid))

076 flash\_led(RED\_LED\_PIN, 5, 100);

077 initWiFi();

078 }

079 bool isConnectWiFi()

080 {

081 if (WiFi.status() != WL\_CONNECTED)

082 {

083 redLedState = HIGH;

084 digitalWrite(RED\_LED\_PIN, redLedState);

085 greenLedState = LOW;

086 digitalWrite(GREEN\_LED\_PIN, greenLedState);

087 }

088 return WiFi.status() == WL\_CONNECTED;

089 }

Файл devices.h:

001 #pragma once

002 #include <Wire.h>

003 #include <BH1750.h>

004 #include <Adafruit\_Sensor.h>

005 #include <Adafruit\_BME280.h>

006 #include <OneWire.h>

007 #include <DallasTemperature.h>

008 #include "config.h"

009 #define SEALEVELPRESSURE\_HPA (1013.25)

010 Adafruit\_BME280 bme;

011 BH1750 lightMeter;

012 OneWire oneWire(SOIL\_TEMP\_SENSOR\_PIN);

013 DallasTemperature sensors(&oneWire);

014 void setupLeds()

015 {

016 pinMode(YELLOW\_LED\_PIN, OUTPUT);

017 digitalWrite(YELLOW\_LED\_PIN, LOW);

018 pinMode(RED\_LED\_PIN, OUTPUT);

019 digitalWrite(RED\_LED\_PIN, LOW);

020 pinMode(GREEN\_LED\_PIN, OUTPUT);

021 digitalWrite(GREEN\_LED\_PIN, LOW);

022 }

023 void flash\_led(int LED\_PIN, int flash\_number, int duration)

024 {

025 for (int i = 0; i < flash\_number; i++)

026 {

027 digitalWrite(LED\_PIN, HIGH);

028 delay(duration);

029 digitalWrite(LED\_PIN, LOW);

030 delay(duration);

031 }

032 }

033 // GY-302 or BH1750

034 void setupLightmeter()

035 {

036 bool status = lightMeter.begin();

037 if (!status)

038 {

039 Serial.println("Could not find a valid BH1750 sensor");

040 isConnectedBh1750 = false;

041 }

042 else

043 isConnectedBh1750 = true;

044 }

045 uint16\_t getLightLevel()

046 {

047 uint16\_t lux = lightMeter.readLightLevel();

048 return lux;

049 }

050 void printLightLevel()

051 {

052 uint16\_t lux = getLightLevel();

053 Serial.print("Light: ");

054 Serial.print(lux);

055 Serial.println(" lx");

056 }

057 // BME280

058 void setupBme()

059 {

060 bool status;

061 status = bme.begin(0x76);

062 if (!status)

063 {

064 Serial.println("Could not find a valid BME280 sensor");

065 isConnectedBme280 = false;

066 }

067 else

068 isConnectedBme280 = true;

069 }

070 float getTemperature()

071 {

072 return bme.readTemperature();

073 }

074 int getPressure()

075 {

076 return bme.readPressure() / 100.0F;

077 }

078 int getHumidity()

079 {

080 return bme.readHumidity();

081 }

082 void printBmeValues()

083 {

084 Serial.print("Temperature = ");

085 Serial.print(getTemperature());

086 Serial.println(" \*C");

087 Serial.print("Pressure = ");

088 Serial.print(getPressure());

089 Serial.println(" hPa");

090 Serial.print("Humidity = ");

091 Serial.print(getHumidity());

092 Serial.println(" %");

093 }

094 // Soil Moisture Sensor v1.2

095 void setupSoilMoistureSensor()

096 {

097 Serial.println(analogRead(MOISTURE\_SENSOR\_PIN));

098 if (analogRead(MOISTURE\_SENSOR\_PIN) > 100)

099 isConnectedMoistureSensor = true;

100 else

101 Serial.println("Could not find a valid Soil Moisture Sensor");

102 }

103 int getSoilMoisturePercent()

104 {

105 float soilMoistureValue = analogRead(MOISTURE\_SENSOR\_PIN);

106 float soilMoisturepercent = 100 - (float)(soilMoistureValue - WaterValue) / (AirValue - WaterValue) \* 100;

107 return soilMoisturepercent;

108 }

109 void printSoilMoisturePercent()

110 {

111 Serial.print("Soil moisture value = ");

112 float soilMoisturepercent = getSoilMoisturePercent();

113 if (soilMoisturepercent >= 100)

114 {

115 Serial.println("100 %");

116 }

117 else if (soilMoisturepercent <= 0)

118 {

119 Serial.println("0 %");

120 }

121 else if (soilMoisturepercent > 0 && soilMoisturepercent < 100)

122 {

123 Serial.print(soilMoisturepercent);

124 Serial.println("%");

125 }

126 }

127 // DS18B20

128 void setupDs18b20()

129 {

130 sensors.begin();

131 sensors.requestTemperatures();

132 if (sensors.getTempCByIndex(0) == -127)

133 {

134 Serial.println("Could not find a valid DS18B20 sensor");

135 isConnectedDs18b20 = false;

136 }

137 else

138 isConnectedDs18b20 = true;

139 }

140 float getSoilTemperature()

141 {

142 sensors.requestTemperatures();

143 float temp = sensors.getTempCByIndex(0);

144 return temp;

145 }

146 void printSoilTemperature()

147 {

148 sensors.requestTemperatures();

149 float temperatureC = sensors.getTempCByIndex(0);

150 float temperatureF = sensors.getTempFByIndex(0);

151 Serial.print(temperatureC);

152 Serial.println("ºC");

153 Serial.print(temperatureF);

154 Serial.println("ºF");

155 }

156 void printSensorsValues()

157 {

158 printLightLevel();

159 printBmeValues();

160 printSoilMoisturePercent();

161 printSoilTemperature();

162 }

163 void checkConnectsSensors()

164 {

165 if (analogRead(MOISTURE\_SENSOR\_PIN) < 100 || getSoilMoisturePercent() < 0)

166 {

167 isConnectedMoistureSensor = false;

168 Serial.println("Could not find a valid Soil Moisture Sensor");

169 }

170 else

171 isConnectedMoistureSensor = true;

172 if (!bme.begin(0x76))

173 {

174 Serial.println("Could not find a valid BME280 sensor");

175 isConnectedBme280 = false;

176 }

177 else

178 isConnectedBme280 = true;

179 if (!lightMeter.begin())

180 {

181 Serial.println("Could not find a valid BH1750 sensor");

182 isConnectedBh1750 = false;

183 }

184 else

185 isConnectedBh1750 = true;

186 sensors.requestTemperatures();

187 if (sensors.getTempCByIndex(0) == -127)

188 {

189 Serial.println("Could not find a valid DS18B20 sensor");

190 isConnectedDs18b20 = false;

191 }

192 else

193 isConnectedDs18b20 = true;

194 }

195 void highlightHeaterWork()

196 {

197 if (heaterState == LOW)

198 {

199 yellowLedState = LOW;

200 digitalWrite(YELLOW\_LED\_PIN, yellowLedState);

201 }

202 else

203 {

204 yellowLedState = HIGH;

205 digitalWrite(YELLOW\_LED\_PIN, yellowLedState);

206 }

207 }

208 // setup output pins

209 void setupDevices()

210 {

211 pinMode(LED\_STRIP\_PIN, OUTPUT);

212 digitalWrite(LED\_STRIP\_PIN, LOW);

213 pinMode(PUMP\_PIN, OUTPUT);

214 digitalWrite(PUMP\_PIN, LOW);

215 pinMode(HEATER\_PIN, OUTPUT);

216 digitalWrite(HEATER\_PIN, LOW);

217 pinMode(COOLER\_PIN, OUTPUT);

218 digitalWrite(COOLER\_PIN, LOW);

219 }

Файл settings.h:

001 #pragma once

002 // Пороговые значения

003 const int minLightLevel = 50; // Минимальный уровень освещенности (люкс)

004 const int maxLightLevel = 1500; // Максимальный уровень освещенности (люкс)

005 const int minTemperature = 20; // Минимальная температура воздуха (°C)

006 const int maxTemperature = 35; // Максимальная температура воздуха (°C)

007 const int minSoilTemperature = 18; // Минимальная температура почвы (°C)

008 const int maxSoilTemperature = 30; // Максимальная температура почвы (°C)

009 const int minHumidity = 30; // Минимальная влажность воздуха (%)

010 const int maxHumidity = 80; // Максимальная влажность воздуха (%)

011 const int minSoilMoisture = 20; // Минимальная влажность почвы (%)

012 const int maxSoilMoisture = 80; // Максимальная влажность почвы (%)

013 // Гистерезис для предотвращения скачков

014 const int lightHysteresis = 10; // Гистерезис для освещенности (люкс)

015 const int tempHysteresis = 3; // Гистерезис для температуры воздуха (°C)

016 const int soilTempHysteresis = 3; // Гистерезис для температуры почвы (°C)

017 const int humHysteresis = 5; // Гистерезис для влажности воздуха (%)

018 const int soilHysteresis = 10; // Гистерезис для влажности почвы (%)

019 // Параметры для нагрева

020 const int maxHeatingDuration = 300000; // Максимальная длительность одного цикла нагрева (5 минут = 300000 мс)

021 const int heatingPause = 60000; // Пауза между циклами нагрева (1 минута = 60000 мс)

022 const float heatingFactor = 60000; // Коэффициент для нагрева (мс на °C отклонения)

023 // Параметры для полива

024 const int pulseWateringDuration = 4000; // Длительность одного импульса полива (4 секунды = 4000 мс)

025 const int wateringPause = 10000; // Пауза между импульсами полива (10 секунд = 10000 мс)

026 const int maxWateringPulses = 5; // Максимальное количество импульсов за один цикл полива

027 const float wateringFactor = 0.5; // Коэффициент для количества импульсов (импульсов на % отклонения)

028 // Параметры для усреднения данных

029 const int MEASUREMENT\_INTERVAL = 10000; // Интервал между измерениями (10 секунд)

030 const int AVERAGE\_PERIOD = 60000; // Период усреднения (1 минута)

Файл SmartGrow.ino:

001 #include "devices.h"

002 #include "tgCommunication.h"

003 #include "automaticMode.h"

004 #include "connectWiFi.h"

005 void setup()

006 {

007 Serial.begin(115200);

008 Wire.begin();

009 setupLightmeter();

010 setupBme();

011 setupSoilMoistureSensor();

012 setupDs18b20();

013 setupLeds();

014 setupDevices();

015 setupWiFi();

016 setupTelegram();

017 initSensorData();

018 }

019 void loop()

020 {

021 checkConnectsSensors();

022 printSensorsValues();

023 if (automaticModeState)

024 automaticMode();

025 if (!isConnectWiFi())

026 {

027 setupWiFi();

028 }

029 loopTelegram();

030 delay(500);

031 }

Файл tgCommunication.h:

001 #pragma once

002 #include <WiFiClientSecure.h>

003 #include <ArduinoJson.h>

004 #include <UniversalTelegramBot.h>

005 #include "config.h"

006 #include "devices.h"

007 #include <WiFi.h>

008 WiFiClientSecure client;

009 UniversalTelegramBot bot(BOT\_TOKEN, client);

010 int botRequestDelay = 500;

011 unsigned long lastTimeBotRan;

012 void sendStats(String chat\_id)

013 {

014 String stats = "Stats:\n";

015 stats += isConnectedBh1750 ? ("Light: " + String(getLightLevel()) + " lux \n") : "Could not find a valid BH1750 sensor\n";

016 stats += isConnectedBme280 ? ("Temperature: " + String(getTemperature()) + " \*C\nPressure: " + String(getPressure()) + " hPa\nHumidity: " + String(getHumidity()) + " %\n") : "Could not find a valid BME280 sensor\n";

017 stats += isConnectedMoistureSensor ? ("Soil moisture: " + String(getSoilMoisturePercent()) + " %\n") : "Could not find a valid Soil Moisture Sensor\n";

018 stats += isConnectedDs18b20 ? ("Soil temperature: " + String(getSoilTemperature()) + " \*C\n") : "Could not find a valid DS18B20 Sensor\n";

019 bot.sendMessage(chat\_id, stats, "");

020 }

021 void handleNewMessages(int numNewMessages)

022 {

023 Serial.println("handle New Messages ");

024 Serial.println(String(numNewMessages));

025 for (int i = 0; i < numNewMessages; i++)

026 {

027 String chat\_id = String(bot.messages[i].chat\_id);

028 if (chat\_id != CHAT\_ID)

029 {

030 bot.sendMessage(chat\_id, "Unauthorized user", "");

031 continue;

032 }

033 String text = bot.messages[i].text;

034 Serial.println(text);

035 String from\_name = bot.messages[i].from\_name;

036 if (text == "/automatic\_mode\_on")

037 {

038 bot.sendMessage(chat\_id, "automatic mode set to ON ", "");

039 automaticModeState = true;

040 }

041 if (text == "/automatic\_mode\_off")

042 {

043 bot.sendMessage(chat\_id, "automatic mode set to OFF ", "");

044 automaticModeState = false;

045 }

046 if (text == "/sensors\_stats")

047 {

048 sendStats(chat\_id);

049 }

050 if (text == "/devices\_status")

051 {

052 String status = "Devices status :\n";

053 status += "Heater: " + String(heaterState == LOW ? "OFF" : "ON") + "\n";

054 status += "Cooler: " + String(coolerState == LOW ? "OFF" : "ON") + "\n";

055 status += "Lighting: " + String(lightingState == LOW ? "OFF" : "ON") + "\n";

056 status += "Pump: " + String(pumpState == LOW ? "OFF" : "ON") + "\n";

057 bot.sendMessage(chat\_id, status, "");

058 }

059 if (text == "/heater\_on")

060 {

061 bot.sendMessage(chat\_id, "heater state set to ON", "");

062 heaterState = HIGH;

063 digitalWrite(HEATER\_PIN, heaterState);

064 highlightHeaterWork();

065 }

066 if (text == "/heater\_off")

067 {

068 bot.sendMessage(chat\_id, "heater state set to OFF", "");

069 heaterState = LOW;

070 digitalWrite(HEATER\_PIN, heaterState);

071 highlightHeaterWork();

072 }

073 if (text == "/cooler\_on")

074 {

075 bot.sendMessage(chat\_id, "cooler state set to ON", "");

076 coolerState = HIGH;

077 digitalWrite(COOLER\_PIN, coolerState);

078 }

079 if (text == "/cooler\_off")

080 {

081 bot.sendMessage(chat\_id, "cooler state set to OFF", "");

082 coolerState = LOW;

083 digitalWrite(COOLER\_PIN, coolerState);

084 }

085 if (text == "/lighting\_on")

086 {

087 bot.sendMessage(chat\_id, "lighting state set to ON", "");

088 lightingState = HIGH;

089 digitalWrite(LED\_STRIP\_PIN, lightingState);

090 }

091 if (text == "/lighting\_off")

092 {

093 bot.sendMessage(chat\_id, "lighting state set to OFF", "");

094 lightingState = LOW;

095 digitalWrite(LED\_STRIP\_PIN, lightingState);

096 }

097 if (text == "/pump\_on")

098 {

099 bot.sendMessage(chat\_id, "pump state set to ON", "");

100 pumpState = HIGH;

101 digitalWrite(PUMP\_PIN, pumpState);

102 }

103 if (text == "/pump\_off")

104 {

105 bot.sendMessage(chat\_id, "pump state set to OFF", "");

106 pumpState = LOW;

107 digitalWrite(PUMP\_PIN, pumpState);

108 }

109 if (text == "/help")

110 {

111 String help = "Hello, " + from\_name + "!\n";

112 help += "Use the following commands to control your greenhouse:\n\n";

113 help += "/automatic\_mode\_on and /automatic\_mode\_off - to use the automatic mode \n";

114 help += "/sensors\_stats - to request sensors values \n";

115 help += "/devices\_status - to request devices status\n";

116 help += "/heater\_on and /heater\_off - to control heating\n";

117 help += "/cooler\_on and /cooler\_off - to control ventilation\n";

118 help += "/lighting\_on and /lighting\_off - to control lighting\n";

119 help += "/pump\_on and /pump\_off - to control watering\n";

120 bot.sendMessage(chat\_id, help, "");

121 }

122 }

123 }

124 void setupTelegram()

125 {

126 client.setCACert(TELEGRAM\_CERTIFICATE\_ROOT);

127 }

128 void loopTelegram()

129 {

130 if (millis() > lastTimeBotRan + botRequestDelay)

131 {

132 int numNewMessages = bot.getUpdates(bot.last\_message\_received + 1);

133 while (numNewMessages)

134 {

135 Serial.println("got response");

136 handleNewMessages(numNewMessages);

137 numNewMessages = bot.getUpdates(bot.last\_message\_received + 1);

138 }

139 lastTimeBotRan = millis();

140 }

141 }

Приложение Е

(*обязательное*)

**Перечень элементов**

Приложение Ж

(*обязательное*)

**Ведомость документов**