**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc183742605)

[1 ОБЗОР ЛИТЕРАТУРЫ 7](#_Toc183742606)

[1.1 Умная теплица growUp 7](#_Toc183742607)

[1.2 Гидропоника – система умный сад 7](#_Toc183742608)

[1.3 Настольная садовая ферма L-Box 8](#_Toc183742609)

[1.4 Умный сад iGarden PRO 9](#_Toc183742610)

[2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА 10](#_Toc183742611)

[2.1 Перечень блоков 10](#_Toc183742612)

[2.2 Связи между блоками 10](#_Toc183742613)

[3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА 11](#_Toc183742614)

[3.1 Состав устройства 11](#_Toc183742615)

[3.2 Микроконтроллер 11](#_Toc183742616)

[3.3 Датчик температуры и влажности воздуха 12](#_Toc183742617)

[3.4 Датчик уровня освещенности 14](#_Toc183742618)

[3.5 Датчик влажности почвы 14](#_Toc183742619)

[3.2 Мини-насос 14](#_Toc183742620)

[3.3 Фитолента 15](#_Toc183742621)

[3.4 Кулеры 15](#_Toc183742622)

[3.5 Нагревающий провод 15](#_Toc183742623)

[3.5 Реле 15](#_Toc183742624)

[3.5 Источники питания 15](#_Toc183742625)

[4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА 16](#_Toc183742626)

[4.1 Расчет питания для устройств с напряжением 12 В 16](#_Toc183742627)

[4.2 Расчет питания для устройств с напряжением 5 В 16](#_Toc183742628)

[4.3 Расчет питания для нагревательного блока 17](#_Toc183742629)

[4.4 Расчет питания всего устройства 17](#_Toc183742630)

[5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 18](#_Toc183742631)

[5.1 Требования к программе 18](#_Toc183742632)

[5.2 Схема программы 18](#_Toc183742633)

[5.3 Программа управления устройством 19](#_Toc183742634)

[5.4 Файл smartGrow.ino 20](#_Toc183742635)

[5.5 Файл devices.h 20](#_Toc183742636)

[5.6 Файл connectWiFi.h 21](#_Toc183742637)

[5.7 Файл tgCommunication.h 21](#_Toc183742638)

[ЗАКЛЮЧЕНИЕ 22](#_Toc183742639)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc183742640)

[ПРИЛОЖЕНИЕ А 24](#_Toc183742641)

[ПРИЛОЖЕНИЕ Б 25](#_Toc183742642)

[ПРИЛОЖЕНИЕ В 26](#_Toc183742643)

[ПРИЛОЖЕНИЕ Г 27](#_Toc183742644)

[ПРИЛОЖЕНИЕ Д 28](#_Toc183742645)

[ПРИЛОЖЕНИЕ Е 37](#_Toc183742646)

[ПРИЛОЖЕНИЕ Ж 38](#_Toc183742647)

# 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.

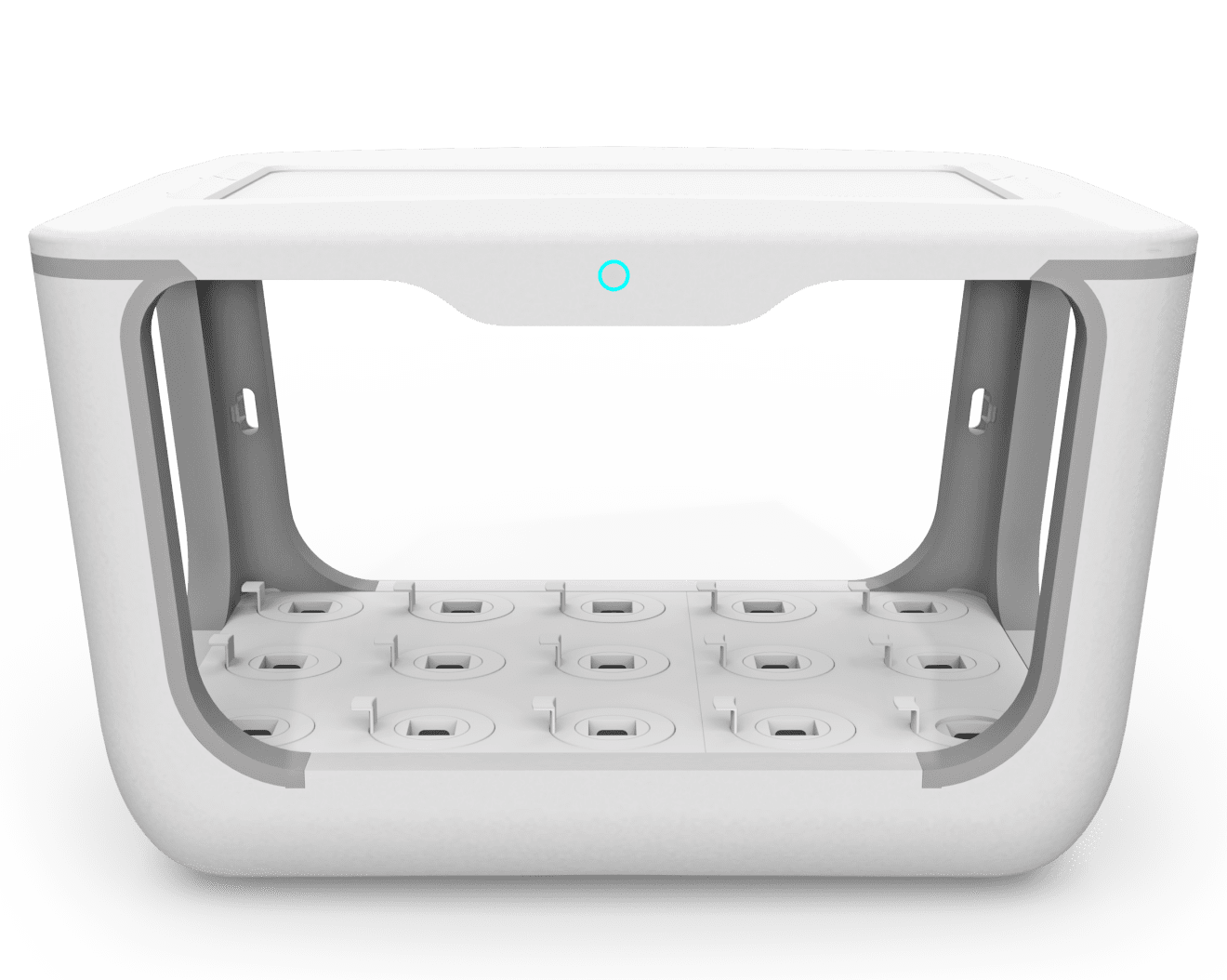


Рисунок 1.3 – Настольная садовая ферма L-Box.

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 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА

Структурная схема устройства приведена в приложении А.

## 2.1 Перечень блоков

В данном устройстве можно выделить девять основных блоков:

– блок датчика температуры и влажности;

– блок датчика влажности почвы;

– блок датчика уровня освещенности;

– микроконтроллер;

– 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 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 и модулями системы.

В качестве модуля управления и отображения информации реализован телеграм-бот Smart Grow Bot. Данный блок необходим для управления микроконтроллером и получение 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 отклонение считанных датчиками значений от заданных пользователем допустимых границах в автоматическом режиме, 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 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

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;

– мини-насос;

– фитолента;

– кулеры;

– нагревающий провод;

– реле;

– источники питания;

– светодиоды.

## 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 был выбран ESP32-S3 в сборке ESP32-S3-DevKitC-1. Основными критериями выбора стали объем памяти и 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. 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.

## 3.2 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.

## 3.3 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.

## 3.4 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.

## 3.5 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.

## 3.5 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.

## 3.5 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 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.

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 6.

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

## 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 для работы с Telegram API, что позволяет пользователю управлять системой и получать данные о состоянии датчиков через сообщения. Пользователь может отправлять команды, такие как /sensors\_stats, чтобы запросить текущие значения с датчиков (температура и влажность воздуха, влажность почвы, уровень освещенности). Система может включать или выключать устройства, такие как нагреватели, кулеры и освещение, в ответ на команды от пользователя. Код включает функции для подключения к WiFi, что необходимо для работы с Telegram-ботом и отправки данных на сервер. При потере соединения система автоматически пытается переподключиться, что обеспечивает стабильную работу.

## 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 и устройств.

Строка 14: Вызов функции setupDevices() для инициализации других подключенных устройств.

Строка 16: Вызов функции setupWiFi() для подключения к WiFi.

Строка 17: Вызов функции setupTelegram() для инициализации связи с Telegram-ботом.

Строка 19: Начало функцияи loop().

Строка 20: Проверка подключения датчиков с помощью checkConnectsSensors().

Строки 21 – 24: Проверка состояния WiFi. Если WiFi не подключен, повторная настройка с помощью setupWiFi().

Строка 25: Вызов функции loopTelegram(), которая обрабатывает сообщения от Telegram-бота.

Строка 26: Задержка в 500 миллисекунд для предотвращения перегрузки цикла. Конец функции loop().

## 5.5 Файл devices.h

Строки 1 – 4: Подключение необходимых библиотек для работы с I2C и датчиками.

Строки 6 – 7: Создание объектов для работы с датчиками BME280 и BH1750.

Строки 9 – 19: Функция setupLeds() и flash\_led(), которая отвечает за инициализацию светодиодов и их мигание.

Строки 23 – 47: Функции setupLightmeter(), setupBme(), и setupSoilMostureSensor() для инициализации датчика освещенности, датчика температуры и влажности воздуха и датчика влажности почвы.

Строки 49 – 73: Функции getTemperature(), getPressure(), getHumidity(), и getSoilMoisturePercent() для считывания данных с соответствующих датчиков.

Строки 75 – 88: Функция checkConnectsSensors(), которая проверяет подключение всех сенсоров и выводит соответствующие сообщения.

Строки 90 – 110: Функция setupDevices() для инициализации выходов для управления устройствами (насос, обогреватель, охладитель).

## 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.

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.

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 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

010 setupLightmeter();

011 setupBme();

012 setupSoilMostureSensor();

013 setupLeds();

014

015 setupDevices();

016

017 setupWiFi();

018 setupTelegram();

019 }

020

021 void loop()

022 {

023 checkConnectsSensors();

024 if (!isConnectWiFi())

025 {

026 setupWiFi();

027 }

028 loopTelegram();

029 delay(500);

030 }

Файл settings.h

001 const int minLightLevel = 50;

002 const int maxLightLevel = 150;

003 const int minTemperature = 20;

004 const int maxTemperature = 35;

005 const int minHumidity = 30;

006 const int maxHumidity = 80;

007 const int minSoilMoisture = 20;

008 const int maxSoilMoisture = 80;

009 const int duringWatering = 10;

010 const int maxTimeLighting = 10;

011 const int maxTimeWatering = 10;

012 const int maxTimeHeating = 10;

013 const int maxTimeCooling = 10;

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 "config.h"

007

008 #define SEALEVELPRESSURE\_HPA (1013.25)

009

010 Adafruit\_BME280 bme;

011 BH1750 lightMeter;

012 void setupLeds()

013 {

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

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

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

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

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

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

020 }

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

022 {

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

024 {

025 digitalWrite(LED\_PIN, HIGH);

026 delay(duration);

027 digitalWrite(LED\_PIN, LOW);

028 delay(duration);

029 }

030 }

031 void setupLightmeter()

032 {

033 bool status = lightMeter.begin();

034 if (!status)

035 {

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

037 isConnectedBh1750 = false;

038 }

039 else

040 isConnectedBh1750 = true;

041 }

042 uint16\_t getLightLevel()

043 {

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

045 return lux;

046 }

047 void printLightLevel()

048 {

049 uint16\_t lux = getLightLevel();

050 Serial.print("Light: ");

051 Serial.print(lux);

052 Serial.println(" lx");

053 }

054 void setupBme()

055 {

056 bool status;

057 status = bme.begin(0x76);

058 if (!status)

059 {

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

061 isConnectedBme280 = false;

062 }

063 else

064 isConnectedBme280 = true;

065 }

066 float getTemperature()

067 {

068 return bme.readTemperature();

069 }

070 int getPressure()

071 {

072 return bme.readPressure() / 100.0F;

073 }

074 int getHumidity()

075 {

076 return bme.readHumidity();

077 }

078 void printBmeValues()

079 {

080 Serial.print("Temperature = ");

081 Serial.print(getTemperature());

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

083

084 Serial.print("Pressure = ");

085 Serial.print(getPressure());

086 Serial.println(" hPa");

087

088 Serial.print("Humidity = ");

089 Serial.print(getHumidity());

090 Serial.println(" %");

091 }

092 void setupSoilMostureSensor()

093 {

094 if (analogRead(MOSTURE\_SENSOR\_PIN) > 100)

095 isConnectedMostureSensor = true;

096 else

097 Serial.println("Could not find a valid Soil Mosture Sensor");

098 }

099 int getSoilMoisturePercent()

100 {

101 float soilMoistureValue = analogRead(MOSTURE\_SENSOR\_PIN);

102 float soilmoisturepercent = 100 - (float)(soilMoistureValue - WaterValue) / (AirValue - WaterValue) \* 100;

103 return soilmoisturepercent;

104 }

105 void printSoilMoisturePercent()

106 {

107 Serial.print("Soil mosture value = ");

108 float soilmoisturepercent = getSoilMoisturePercent();

109 if (soilmoisturepercent >= 100)

110 {

111 Serial.println("100 %");

112 }

113 else if (soilmoisturepercent <= 0)

114 {

115 Serial.println("0 %");

116 }

117 else if (soilmoisturepercent > 0 && soilmoisturepercent < 100)

118 {

119 Serial.print(soilmoisturepercent);

120 Serial.println("%");

121 }

122 }

123 void checkConnectsSensors()

124 {

125 if (analogRead(MOSTURE\_SENSOR\_PIN) < 100)

126 {

127 isConnectedMostureSensor = false;

128 Serial.println("Could not find a valid Soil Mosture Sensor");

129 }

130 else

131 isConnectedMostureSensor = true;

132

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

134 {

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

136 isConnectedBme280 = false;

137 }

138 else

139 isConnectedBme280 = true;

140

141 if (!lightMeter.begin())

142 {

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

144 isConnectedBh1750 = false;

145 }

146 else

147 isConnectedBh1750 = true;

148 }

149 void highlightHeaterWork()

150 {

151 if (heaterState == LOW)

152 {

153 yellowLedState = LOW;

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

155 }

156 else

157 {

158 yellowLedState = HIGH;

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

160 }

161 }

162 void setupDevices()

163 {

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

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

166

167 pinMode(POMP\_PIN, OUTPUT);

168 digitalWrite(POMP\_PIN, LOW);

169

170 pinMode(HEATER\_PIN, OUTPUT);

171 digitalWrite(HEATER\_PIN, LOW);

172

173 pinMode(COOLER\_PIN, OUTPUT);

174 digitalWrite(COOLER\_PIN, LOW);

175 }

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

010 IPAddress ip = WiFi.localIP();

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

012 Serial.println(ip);

013

014 long rssi = WiFi.RSSI();

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

016 Serial.print(rssi);

017 Serial.println(" dBm");

018 }

019 void initWiFi()

020 {

021 redLedState = HIGH;

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

023 WiFi.mode(WIFI\_STA);

024 WiFi.begin(ssid, password);

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

026 Serial.println(ssid);

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

028 {

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

030 Serial.print("Status: ");

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

032 redLedState = HIGH;

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

034 }

035 redLedState = LOW;

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

037 greenLedState = HIGH;

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

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

040 printWifiStatus();

041 }

042 bool scanWiFi(char \*ssid)

043 {

044 bool isFoundSsid = false;

045 Serial.println("scan start");

046 int n = WiFi.scanNetworks();

047 Serial.println("scan done");

048 if (n == 0)

049 {

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

051 }

052 else

053 {

054 Serial.print(n);

055 Serial.println(" networks found");

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

057 {

058 Serial.print(i + 1);

059 Serial.print(": ");

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

061 Serial.print(" (");

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

063 Serial.print(")");

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

065 delay(10);

066 if (!isFoundSsid)

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

068 }

069 }

070 Serial.println("");

071 return isFoundSsid;

072 }

073 void setupWiFi()

074 {

075 WiFi.mode(WIFI\_STA);

076 WiFi.disconnect();

077 while (!scanWiFi(ssid))

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

079 initWiFi();

080 }

081 bool isConnectWiFi()

082 {

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

084 {

085 redLedState = HIGH;

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

087 greenLedState = LOW;

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

089 }

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

091 }

Файл config.h

001 #pragma once

002 #define MOSTURE\_SENSOR\_PIN 17

003 #define LED\_STRIP\_PIN 10

004 #define POMP\_PIN 11

005 #define COOLER\_PIN 12

006 #define HEATER\_PIN 13

007 #define YELLOW\_LED\_PIN 4

008 #define RED\_LED\_PIN 5

009 #define GREEN\_LED\_PIN 6

010 char \*ssid = "iPhone Ann";

011 char \*password = "12345678910";

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

013 #define CHAT\_ID "922443025"

014 bool isConnectedMostureSensor = false;

015 bool isConnectedBme280 = false;

016 bool isConnectedBh1750 = false;

017 bool heaterState = LOW;

018 bool coolerState = LOW;

019 bool lightingState = LOW;

020 bool pompState = LOW;

021 bool automaticModeState = false;

022 bool redLedState = LOW;

023 bool greenLedState = LOW;

024 bool yellowLedState = LOW;

025 const int AirValue = 590;

026 const int WaterValue = 360;

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

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

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

018 stats += isConnectedMostureSensor ? ("Soil mosture: " + String(getSoilMoisturePercent()) + " %\n") : "Could not find a valid Soil Mosture Sensor\n";

019

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

021 }

022 void handleNewMessages(int numNewMessages)

023 {

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

025 Serial.println(String(numNewMessages));

026

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

028 {

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

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

031 {

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

033 continue;

034 }

035

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

037 Serial.println(text);

038

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

040

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

042 {

043 bot.sendMessage(chat\_id, "automatic mode set to ON (sorry, but nothing's going to happen)", "");

044 automaticModeState = true;

045 }

046

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

048 {

049 bot.sendMessage(chat\_id, "automatic mode set to OFF (sorry, but nothing's going to happen)", "");

050 automaticModeState = false;

051 }

052

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

054 {

055 sendStats(chat\_id);

056 }

057

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

059 {

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

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

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

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

064 status += "Pomp: " + String(pompState == LOW ? "OFF" : "ON") + "\n";

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

066 }

067

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

069 {

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

071 heaterState = HIGH;

072 digitalWrite(HEATER\_PIN, heaterState);

073 highlightHeaterWork();

074 }

075

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

077 {

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

079 heaterState = LOW;

080 digitalWrite(HEATER\_PIN, heaterState);

081 highlightHeaterWork();

082 }

083

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

085 {

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

087 coolerState = HIGH;

088 digitalWrite(COOLER\_PIN, coolerState);

089 }

090

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

092 {

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

094 coolerState = LOW;

095 digitalWrite(COOLER\_PIN, coolerState);

096 }

097

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

099 {

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

101 lightingState = HIGH;

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

103 }

104

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

106 {

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

108 lightingState = LOW;

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

110 }

111

112 if (text == "/pomp\_on")

113 {

114 bot.sendMessage(chat\_id, "pomp state set to ON", "");

115 pompState = HIGH;

116 digitalWrite(POMP\_PIN, pompState);

117 }

118

119 if (text == "/pomp\_off")

120 {

121 bot.sendMessage(chat\_id, "pomp state set to OFF", "");

122 pompState = LOW;

123 digitalWrite(POMP\_PIN, pompState);

124 }

125

126 if (text == "/help")

127 {

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

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

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

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

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

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

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

135 help += "/pomp\_on and /pomp\_off - to control watering\n";

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

137 }

138 }

139 }

140 void setupTelegram()

141 {

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

143 }

144 void loopTelegram()

145 {

146 if (millis() > lastTimeBotRan + botRequestDelay)

147 {

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

149

150 while (numNewMessages)

151 {

152 Serial.println("got response");

153 handleNewMessages(numNewMessages);

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

155 }

156 lastTimeBotRan = millis();

157 }

158 }

ПРИЛОЖЕНИЕ Е

(*обязательное*)

Перечень элементов

ПРИЛОЖЕНИЕ Ж

(*обязательное*)

Ведомость документов