3.2 Аппаратные блоки устройства

Комплекс для управления теплицей состоит из следующих аппаратных блоков:

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

Далее будет рассмотрена функциональная часть каждого блока.

3.2.1 Блок контроллера теплицы

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

Основными особенностями печатной платы являются:

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

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

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

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

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

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

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

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

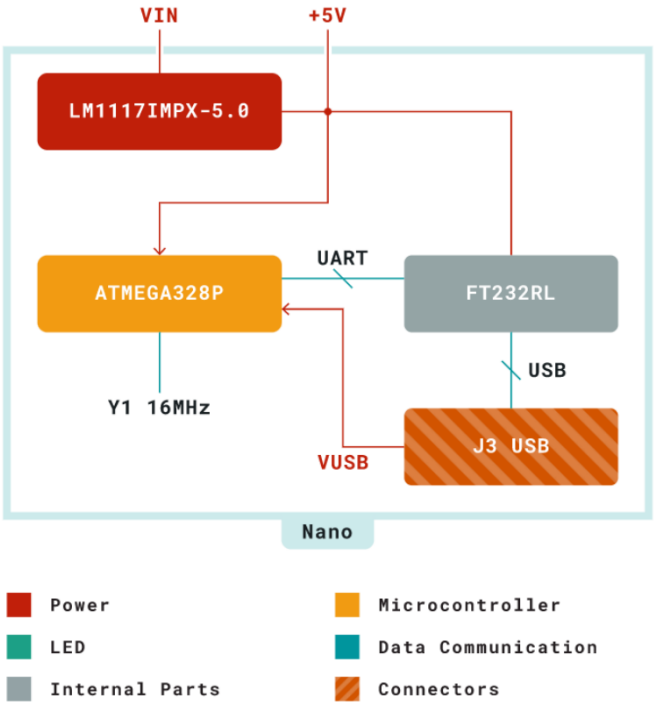


Рисунок 3.7 – Блок-схема микроконтроллера Arduino Nano

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

3.2.2 Блок беспроводной связи

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

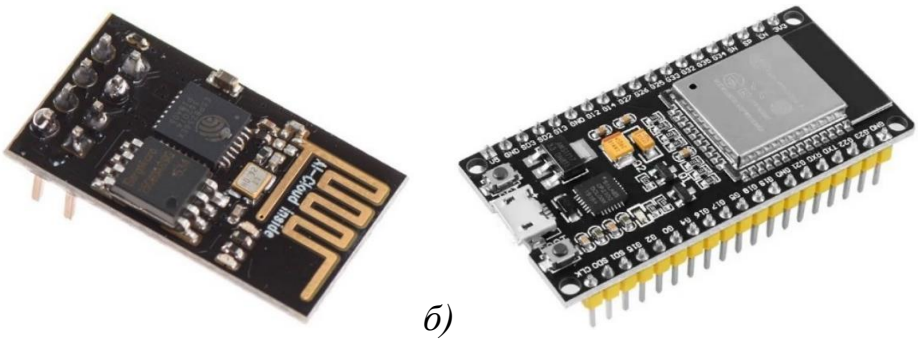
При выборе Wi-Fi модуля нужно обратить внимание на его характеристики, такие как дальность передачи, скорость передачи данных, надежность соединения и простота в настройке. Также при выборе модуля Wi-Fi для комплекса управления теплицей важно учитывать не только его характеристики, но и его совместимость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

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

Таблица 3.3 – Характеристики Wi-Fi модулей

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

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



а) *б)*

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

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

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

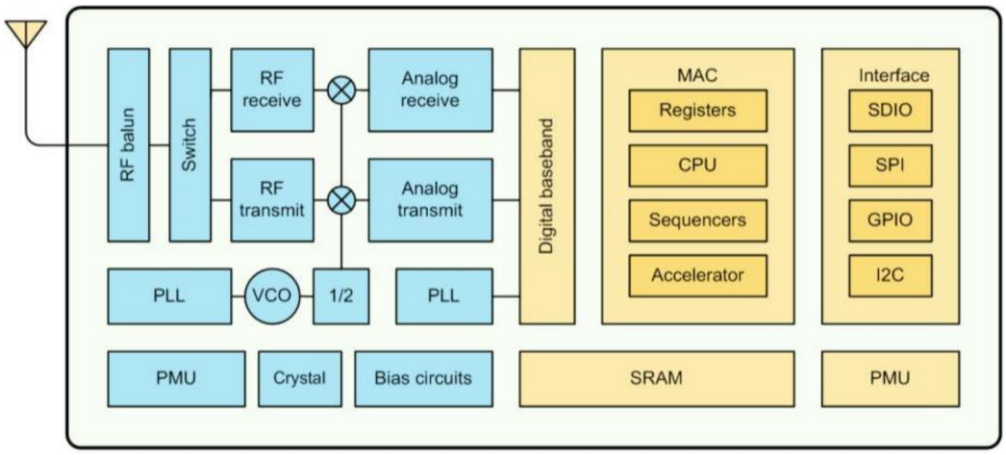


Рисунок 3.9 – Блок-схема ESP8266

Преимуществами ESP-01 являются:

– легкая интеграция с Arduino Nano через интерфейс UART;

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

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

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

3.2.3 Блок управления контроллером

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

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

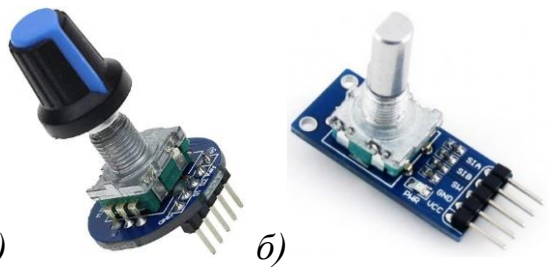
1. Управление режимами работы. Энкодер позволяет выбирать режимы работы системы, такие как автоматический режим и ручной режим. В автоматическом режиме система самостоятельно контролирует условия в теплице и принимает решения об управлении устройствами на основе заданных параметров. В ручном режиме пользователь может непосредственно управлять состоянием устройств, игнорируя автоматические настройки.
2. Отображение текущих значений. Энкодер также позволяет пользователю просматривать текущие значения различных параметров системы на дисплее. Это включает в себя информацию о температуре и влажности воздуха, уровне освещенности, влажности почвы, а также о состоянии других устройств, таких как дверь, заслонки и насос.
3. Управление ручными режимами. В ручном режиме пользователям предоставляется возможность непосредственного управления состоянием различных устройств, таких как заслонки и насос, вращая энкодер и выбирая соответствующие значения.

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

Таблица 3.4 – Характеристики энкодеров

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

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



а) *б)*

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

Энкодер EC11 является предпочтительным выбором благодаря следующим преимуществам:

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

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

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

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

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

Здесь используются следующие обозначения:

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

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

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

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

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

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

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

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

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

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

3.2.4 Блок отображения информации

Блок отображения информации играет ключевую роль в представлении

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

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

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

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

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

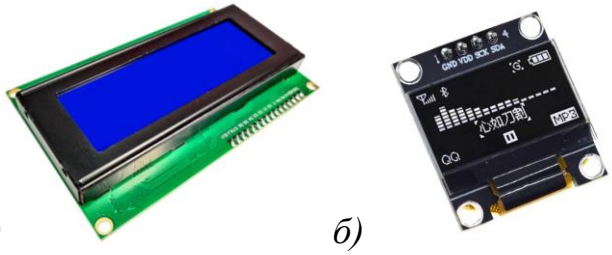
4. Подсветка. Подсветка экрана обеспечивает удобство использования в условиях недостаточного освещения. Желательно выбрать дисплей с настраиваемой подсветкой.

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

Таблица 3.5 – Сравнение характеристик дисплеев

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

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



а) *б)*

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

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

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

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

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

Взаимодействие с дисплеем в программе происходит в нескольких местах.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2.5 Блок управления вентиляцией

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

1. Угол поворота. Сервопривод должен обеспечивать достаточный угол

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

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

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

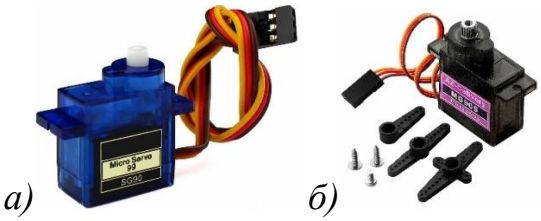
4. Совместимость с контроллером. Сервопривод должен быть совместим с контроллером, который будет управлять им. Это включает в себя соответствие электрическим характеристикам, протоколам связи и другим техническим требованиям.

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

Таблица 3.6 – Характеристики сервоприводов

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

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



а) *б)*

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

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

В коде программы инициализация и управление сервоприводами

происходит в нескольких местах:

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

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

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

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

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

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

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

3.2.6 Блок управления доступом в теплицу

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

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

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

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

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

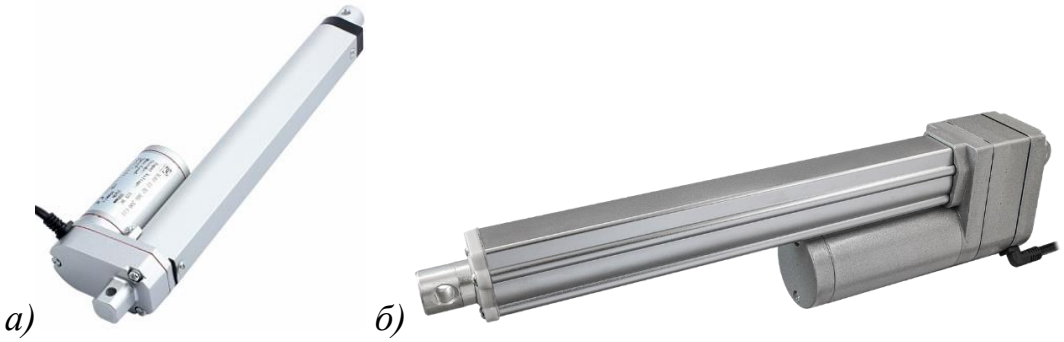
интегрировать его в вашу систему управления с минимальными сложностями.

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

Таблица 3.7 – Характеристики электроприводов

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

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



а) *б)*

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

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

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

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

Таблица 3.8 – Характеристики драйвера TB6612FNG

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

На рисунке 3.13 представлен данный драйвер.

Рисунок 3.13 – Драйвер TB6612FNG

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

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

В данном коде:

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

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

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

– `DRV_PWM` – пин для управления скоростью движения.

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

3.2.7 Блок реле

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

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

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

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

На рисунке 3.14 представлен данный модуль реле [44].

Рисунок 3.14 – Модуль реле

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

Таблица 3.9 – Характеристики модуля реле

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

Продолжение таблицы 3.9

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

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

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

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

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

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

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

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

`flap1.state` и `flap2.state`.

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

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

3.2.8 Блок системы орошения

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

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

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

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

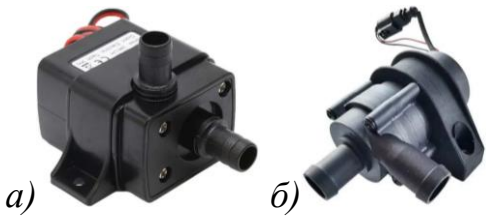


Рисунок 3.15 – Водяные насосы: *а* – DC30E, *б* – LF Bros

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

Таблица 3.10 – Характеристики водяных насосов

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

Продолжение таблицы 3.10

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

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

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

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

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

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

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

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

Перед включением или выключением насоса происходит установка

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

3.2.9 Блок искусственного освещения

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

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

Таблица 3.11 – Характеристики LED ленты

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

На рисунке 3.16 изображена данная лента [47].

Рисунок 3.16 – LED лента SMD 2835

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

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

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

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

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

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

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

3.2.10 Блок определения уровня воды

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

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

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

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

и быть легко интегрируемым в общую систему управления теплицей.

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

Таблица 3.12 – Характеристики датчика уровня воды

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

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

Рисунок 3.17 – Датчик уровня воды

3.2.11 Блок определения интенсивности освещения

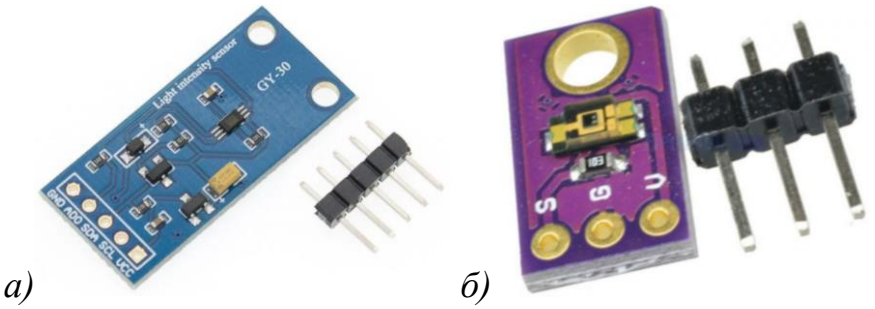
Для блока определения интенсивности освещения предлагается выбрать датчик освещенности, такой как фотодатчик или фоторезистор. Этот датчик будет использоваться для измерения уровня освещенности внутри теплицы. Это важно для определения необходимости включения и выключения искусственного освещения в зависимости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

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

Таблица 3.13 – Характеристики датчиков интенсивности освещения

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

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



а) *б)*

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

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

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

3.2.12 Блок определения влажности почвы

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

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

Таблица 3.14 – Характеристики датчика влажности почвы

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

Продолжение таблицы 3.14

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

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

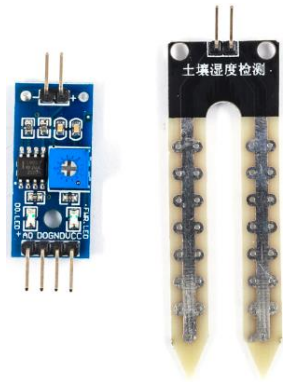


Рисунок 3.19 – Датчик влажности почвы YL-69

3.2.13 Блок определения температуры и влажности воздуха

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

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

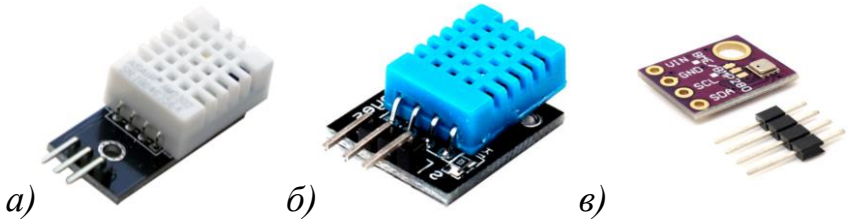
Таблица 3.15 – Характеристики датчиков температуры и влажности

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

Продолжение таблицы 3.15

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

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



а) *б)* *в)*

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

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

3.2.14 Блок идентификации дождя

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

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

Таблица 3.16 – Характеристики датчика дождя

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

На рисунке 3.21 изображен данный датчик [55].

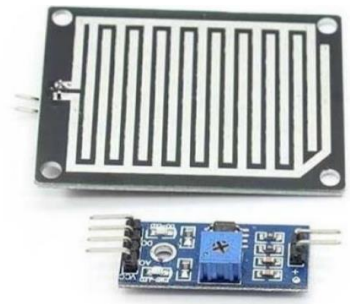


Рисунок 3.21 – Датчик дождя MH-RD

3.2.15 Блок питания

Для обеспечения питания системы требуется три блока питания:

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

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

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

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

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

Таблица 3.17 – Характеристики блока питания 5 В

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

На рисунке 3.22 изображен данный блок питания [56].

Рисунок 3.22 – Блок питания 5 В 2 А

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

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

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

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

Таблица 3.18 – Характеристики блока питания 12 В

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

Продолжение таблицы 3.18

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

На рисунке 3.23 изображен данный блок питания [57].

Рисунок 3.23 – Блок питания 12 В 5 А

3.3 Программные блоки устройства

Комплекс для управления теплицей состоит из следующих аппаратных блоков:

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

Далее будет рассмотрена функциональная часть каждого блока.

3.3.1 Блок API и обработки команд

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

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

состоянии датчиков;

– функция-обработчик: `getSensors();`

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

2. Получение данных о состоянии устройства:

– URI: `/states`;

– метод: GET;

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

– функция-обработчик: `getStates();`

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

3. Установка ручного режима работы:

– URI: `/manual`;

– метод: POST;

– описание: позволяет установить ручной режим управления выходами устройства;

– функция-обработчик: `setManual();`

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

4. Получение текущего ручного режима:

– URI: `/manual`;

– метод: GET;

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

– функция-обработчик: `getManual();`

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

5. Установка настроек устройства:

– URI: `/settings`;

– метод: POST;

– описание: позволяет установить настройки устройства;

– функция-обработчик: `setSettings();`

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

6. Получение текущих настроек устройства:

– URI: `/settings`;

– метод: GET;

– описание: Запрос для получения текущих настроек устройства;

– функция-обработчик: `getSettings();`

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

7. Сброс настроек до заводских:

– URI: `/factoryReset`;

– метод: POST;

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

– функция-обработчик: `factoryReset();`

Общий принцип работы:

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

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

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

3.3.2 Блок взаимодействия с веб-интерфейсом

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

Функциональное взаимодействие с веб-интерфейсом:

1. Отправка данных о состоянии датчиков и устройства:

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

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

2. Обработка команд от веб-интерфейса:

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

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

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

4. Обработка HTTP-запросов от клиента:

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

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

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

Общий принцип работы:

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

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

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

4. Полученные команды обрабатываются устройством и выполняются соответствующие действия.

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

3.3.3 Блок считывания данных с сенсоров

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

Функционал блока считывания данных с сенсоров:

1. Инициализация сенсоров:

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

2. Чтение значений сенсоров:

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

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

3. Обновление данных и отправка на обработку:

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

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

Общий принцип работы:

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

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

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

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

3.3.4 Блок обновления выходов по данным сенсоров

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

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

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

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

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

3.3.5 Блок обработки ошибок и логирования

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

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

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

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

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

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

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

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

3.3.6 Блок управления выходами

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

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

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

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

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

3.3.7 Блок работы с временем и часами реального времени

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

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

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

3.3.8 Блок вспомогательных функций

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

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

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

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

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

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

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

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

3.3.9 Блок настроек и инициализации

Блок настроек и инициализации включает в себя все необходимые

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

Функционал блока настроек и инициализации:

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

Поля:

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

– `driveSpeed` – скорость привода;

– `sensorPeriod` – период считывания данных с сенсоров;

– `flapsTemperatureThreshold` – пороговая температура для открытия заслонок;

– `doorTemperatureThreshold` – пороговая температура для открытия двери;

– `ventelationPeriod` - период вентиляции;

– `ventelationTime` – время вентиляции;

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

– `wateringMoistureThreshold` – порог влажности почвы для полива;

– `wateringTime` – время полива;

– `wateringTimeout` – таймаут после полива;

– `wateringPeriod` – период полива;

– `lightLevelThreshold` – пороговый уровень освещенности.

Файлы: `settings.h`.

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

Файлы: `settings.h`.

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

Файлы: `sensors.cpp`, `web.cpp`.

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

Файлы: `outputs.cpp`.

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

Файлы: `sensors.cpp`.

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

Файлы: `web.cpp`.

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

Файлы: `esp_api.ino`.

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

Файлы: `esp_api.ino`.

4 ПРИНЦИПИАЛЬНОЕ ПРОЕКТИРОВАНИЕ

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

4.1 Блок контроллера теплицы

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

Таблица 4.1 – Описание контактов модуля Nano V3.0 ATmega328P

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

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

Таблица 4.2 – Описание контактов модуля DS3231

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

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

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

4.2 Блок беспроводной связи

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

Таблица 4.3 – Описание контактов модуля ESP-01

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

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

Таблица 4.4 – Описание контактов модуля ESP-01

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

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

– GND – общая земля;

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

4.3 Блок управления контроллером

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

Таблица 4.5 – Описание контактов модуля энкодера EC11

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

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

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

4.4 Блок отображения информации

В состав данного блока входит дисплей LCD2004, который представлен на схеме микросхемой DD7. Для подключения LCD2004 (DD7) к плате микроконтроллера (DD3) по I2C используется модуль расширения портов HW-061 (микросхема DD5). Описание контактов LCD2004 приведено в таблице 4.6.

Таблица 4.6 – Описание контактов дисплея LCD2004

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

Описание контактов модуля HW-061 приведено в таблице 4.7.

Таблица 4.7 – Описание контактов модуля HW-061

Выход	Название	Описание
1 – 16	–	Соответствующие контакты дисплея (DD7)
17	SDA	Линия данных I2C
18	SCL	Линия тактирования I2C
19	VCC	Напряжение питания +5 В
20	GND	Земля питания

Дисплей LCD2004 (DD7) подключается к соответствующим контактам модуля HW-061 (DD5). Исходя из информации в таблицах 4.6 и 4.7, соединение модуля HW-061 (DD5) с платой микроконтроллера (DD3) выполняется следующим образом:

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

4.5 Блок управления вентиляцией

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

Таблица 4.8 – Описание контактов сервопривода SG-90

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

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

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

– VCC – шина +5 В устройства;

– PWM – выход A0 платы микроконтроллера (DD3);

– GND – общая земля.

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

– VCC – шина +5 В устройства;

– PWM – выход D13 платы микроконтроллера (DD3);

– GND – общая земля.

4.6 Блок управления доступом в теплицу

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

Таблица 4.9 – Описание контактов линейного привода 750N

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

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

Таблица 4.10 – Описание контактов драйвера TB6612FNG Ultra L298N

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

Продолжение таблицы 4.10

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

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

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

4.7 Блок реле

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

Таблица 4.11 – Описание контактов модуля реле

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

Продолжение таблицы 4.11

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

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

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

4.8 Блок системы орошения

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

Таблица 4.12 – Описание контактов водяной помпы DC30E

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

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

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

4.9 Блок искусственного освещения

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

Таблица 4.13 – Описание контактов LED ленты SMD2835

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

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

– VCC – выход NC2 модуля реле (KM1);

– GND – общая земля.

4.10 Блок определения уровня воды

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

Таблица 4.14 – Описание контактов модуля датчика уровня воды

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

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

– VCC – шина +5 В устройства;

– Signal – выход A6 платы микроконтроллера (DD3)

– GND – общая земля.

4.11 Блок определения интенсивности освещения

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

Таблица 4.15 – Описание контактов модуля GY-30

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

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

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

4.12 Блок определения влажности почвы

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

Таблица 4.16 – Описание контактов модуля YL-38

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

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

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

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

4.13 Блок определения температуры и влажности воздуха

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

Таблица 4.17 – Описание контактов модуля DHT22

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

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

Таблица 4.18 – Описание контактов модуля BME280

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

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

– VCC – шина +5 В устройства;

– OUT – выход A2 платы микроконтроллера (DD3);

– GND – общая земля.

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

– VCC – шина +5 В устройства;

– GND – общая земля;

– SCL – выход A5 платы микроконтроллера (DD3);

– SDA – выход A4 платы микроконтроллера (DD3).

4.14 Блок идентификации дождя

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

Таблица 4.19 – Описание контактов модуля YL-38

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

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

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

– VCC – шина +5 В устройства;

– GND – общая земля;

– DO – выход A3 платы микроконтроллера (DD3);

– AO – не подключается;

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

4.15 Блок питания

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

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

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

Блок питания XP3 обеспечивает питание следующим устройствам:
– плата микроконтроллера Nano V3.0 (DD3);
– стабилизатор напряжения AMS1117 (DA1);
– дисплей LCD2004 (DD7);
– модуль BME280 (B4);
– модуль DS3231 (DD4);
– энкодер EC11 (DD6).

4.16 Разводка печатной платы

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

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

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

5.1 Общие сведения

Тестирование проводилось в следующих условиях:

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

5.2 Тестирование API

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

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

5.2.1 Тестирование работы с датчиками

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

Методика:

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

влажности и других параметров.

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

Полученный результат:

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

Описание каждого ключа в полученном результате:

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

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

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

– `lightLevel` – уровень освещенности в теплице люксах;

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

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

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

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

5.2.2 Тестирование управления состояниями

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

Методика:

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

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

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

Полученный результат:

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

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

Описание каждого ключа в полученном результате:

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

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

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

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

Полученные данные с датчиков соответствуют ожидаемым. Тест пройден успешно.

5.2.3 Тестирование получения настроек системы

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

Методика:

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

2. Ожидать получение текущих настроек системы.

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

Полученный результат:

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

Описание каждого ключа в полученном результате:

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

– `sensorPeriod` – периодичность считывания данных с датчиков,

выраженная в секундах;

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

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

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

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

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

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

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

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

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

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

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

5.2.4 Тестирование установки настроек системы

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

Методика:

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

2. Ожидать успешного ответа от сервера.

Пример запроса:

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

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

Полученный результат:

– код ответа: 200;

– настройки системы обновлены согласно запросу.

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

5.2.5 Тестирование сброса настроек к заводским

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

Методика:

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

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

Полученный результат:

– код ответа: 200;

– настройки системы сброшены.

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

5.3 Тестирование локального управления

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

5.3.1 Проверка инициализации ЖК-дисплея

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

Методика:

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

Ожидаемый результат: дисплей должен инициализироваться и отобразить начальные данные.

Полученный результат: дисплей инициализировался и первую страницу меню.

5.3.2 Проверка обновления данных на ЖК-дисплее

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

Методика:

1. Систематически изменять значения датчиков (например,

температуру, влажность).

2. Наблюдать за обновлением данных на LCD-дисплее.

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

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

На рисунке 5.1 представлено отображение данных на ЖК-дисплее.

Рисунок 5.1 – Отображение данных на ЖК-дисплее

5.3.3 Проверка управления энкодером

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

Методика:

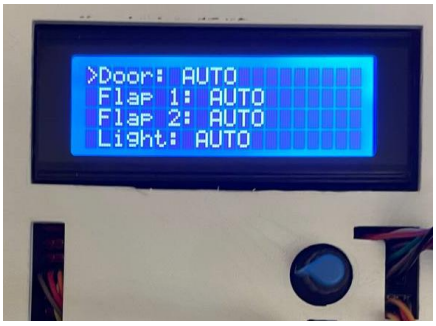
1. Вращать энкодер для перемещения по меню.

2. Нажимать на энкодер для выбора пунктов меню.

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

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

На рисунке 5.2 представлено изменение энкодером состояния подключенных устройств.

а) 

б) 

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

5.3.3 Проверка кнопки для управления линейным приводом

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

Методика:

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

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

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

5.3.4 Проверка отображения критических уведомлений

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

Методика:

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

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

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

5.4 Подведение итогов

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

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

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

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

6.1 Установка системы

Подключение компонентов:

– соединить все датчики к соответствующим портам на Arduino;

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

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

6.2 Подключение ESP-01

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

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

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

Рисунок 6.1 – UBS адаптер для ESP-01

Следует выполнить следующий порядок действий:

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

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

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

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

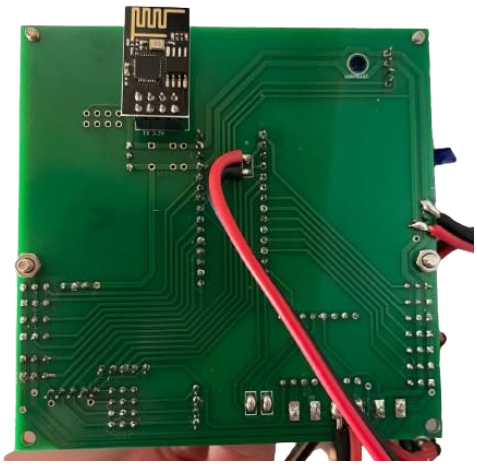


Рисунок 6.2 – Подключение ESP-01 к печатной плате

6.3 Подача питания

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

Рисунок 6.3 – Порты для подключения питания

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

6.3 Управление через веб-интерфейс

Доступ к веб-интерфейсу:

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

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

Просмотр данных с датчиков:

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

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

Управление устройствами:

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

состояниями исполнительных механизмов (насос, двери, свет);

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

Обновление настроек:

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

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

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

Управление в ручном режиме:

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

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

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

Сброс к заводским настройкам:

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

– подтвердить операцию и проверить, что система вернулась к первоначальным настройкам.

6.3 Локальное управление

Использование энкодера:

– вращать энкодер для изменения параметров и навигации по меню на ЖК-дисплее;

– нажимать на энкодер для подтверждения выбора или входа в подменю.

На рисунке 6.4 представлена навигация по меню.

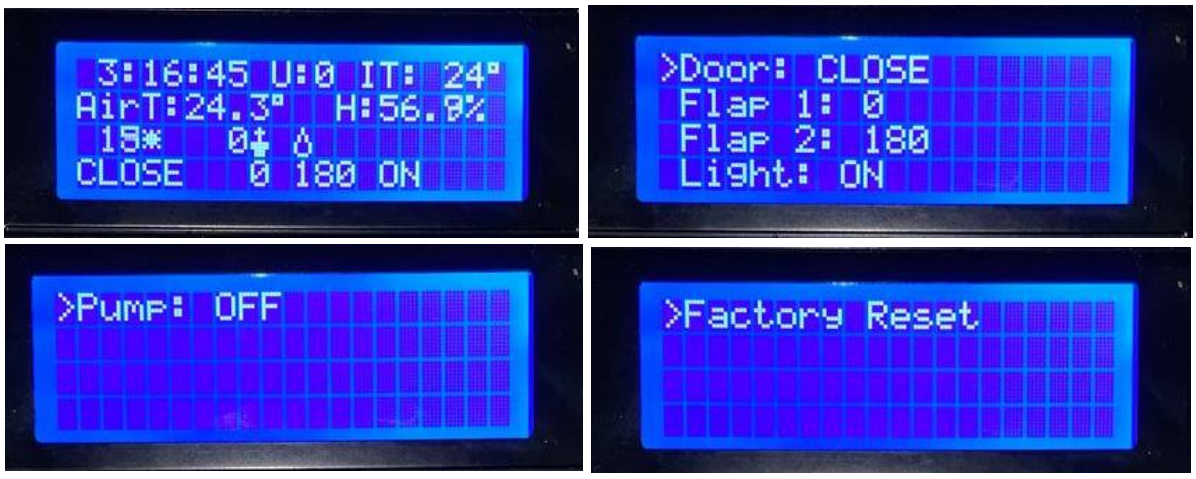


Рисунок 6.4 – Навигация по меню

Просмотр данных на ЖК-дисплее:

– данные с датчиков и состояния исполнительных механизмов

отображаются на LCD-дисплее;

– обновление данных происходит автоматически, позволяя следить за текущими параметрами в режиме реального времени.

Проверка кнопки для управления линейным приводом:

– нажимать кнопку для открытия или закрытия дверей с помощью линейного привода;

– убедиться, что линейный привод корректно реагирует на нажатие, открывая или закрывая двери.

6.4 Подведение итогов

Руководство пользователя предоставляет детальные инструкции по установке, настройке и эксплуатации комплекса для управления теплицей. Следуя инструкциям, пользователи смогут эффективно установить и настроить систему, управлять ее функциями как удаленно через веб-интерфейс, так и локально, обеспечивая оптимальные условия в теплице.

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА ДЛЯ УПРАВЛЕНИЯ ТЕПЛИЦЕЙ НА ОСНОВЕ БЕСПРОВОДНОЙ ТЕХНОЛОГИИ

7.1 Характеристика программно-аппаратного комплекса

Разрабатываемый в данном проекте программно-аппаратный комплекс для управления теплицей на основе беспроводной технологии представляет собой комплексное техническое решение, включающее в себя аппаратную составляющую и программное обеспечение. Он предназначен для автоматизации и управления процессами в теплице с использованием беспроводной технологий Wi-Fi. Такой комплекс эффективно решает задачи по контролю и регулированию микроклимата, полива, освещения и других процессов в тепличном хозяйстве.

Основные преимущества разрабатываемого программно-аппаратного комплекса заключаются в его гибкости и масштабируемости. Пользователь имеет возможность настройки параметров управления в зависимости от специфики своего хозяйства, а также расширения функционала системы по мере необходимости. Это позволяет адаптировать комплекс под конкретные потребности пользователей и оптимизировать его работу в соответствии с изменяющимися условиями.

Дополнительно, данный программно-аппаратный комплекс обладает преимуществами в области экологической устойчивости. Использование беспроводной технологии управления теплицей снижает необходимость в проведении кабельной разводки, что сокращает использование материальных ресурсов и уменьшает воздействие на окружающую среду.

Еще одним значимым преимуществом является экономическая выгода от использования беспроводных технологий. По сравнению с проводными системами, беспроводные технологии обладают более низкой стоимостью развертывания и обслуживания, а также обеспечивают более гибкую архитектуру системы. Это снижает затраты на инфраструктуру и упрощает процесс монтажа, что в свою очередь уменьшает общий бюджет внедрения системы управления теплицей на основе данного комплекса.

Таким образом, аппаратно-программный комплекс для управления теплицей на основе беспроводной технологии представляет собой перспективное решение, обеспечивающее эффективное управление процессами в тепличном хозяйстве и экономическую выгоду от его использования.

7.2 Расчет экономического эффекта от производства программно-аппаратного комплекса

Для определения результата от вложения инвестиций в проектирование и производство программно-аппаратного комплекса необходимо определить

отпускную цену программно-аппаратного комплекса на основе расчета затрат на производство аппаратной части и разработку программной части.

Расчет прямых затрат на материалы и комплектующие изделия для производства аппаратной части комплекса осуществляется в соответствии с представленной в конструкторской документации дипломного проекта номенклатурой, нормой расхода материалов, количеством комплектующих на изделие и рыночным ценам. Для производства данного комплекса были использованы материалы, представленные в таблице 7.1, и комплектующие, представленные в таблице 7.2.

Расчет затрат по статье «Основные и вспомогательные материалы», в которую включается стоимость необходимых для изготовления изделия основных и вспомогательных материалов, осуществляется по формуле (7.1):

$$Р_м = К_{тр} \cdot \sum_{i=1}^{n} Н_{рi} \cdot Ц_{отпi}, \quad (7.1)$$

где $К_{тр}$ – коэффициент транспортных расходов;

n – номенклатура применяемых материалов;

$Н_{рi}$ – норма расхода материала *i*-го вида на единицу изделия, нат. ед./шт.;

$Ц_{отпi}$ – цена за единицу материала *i*-го вида, р.

Таблица 7.1 – Расчет затрат на основные и вспомогательные материалы

Наименование материала	Единица измерения	Норма расхода материала	Цена за единицу материала, р.	Сумма, р.
1	2	3	4	5
1 Припой ПОС 61, 100 г	г	0,65	22,74	14,79
2 Флюс NC-559-ASM, 10 мл	мл	0,72	8,76	6,31
3 Кабель акустический 2х1,0 кв.мм., 1м	м	1,8	1,5	2,70
4 Трубка термоусаживаемая, 1 м	м	0,1	5,04	0,51
5 Пластиковая нить Creality CR-ABS 1,75мм 1 кг	г	0,9	57,00	51,30
6 Вилка штыревая 2,54 мм, 40 шт	шт	0,675	0,64	0,44
7 Клеммник винтовой DG306-5.0 2-контактный, 10 шт.	шт	0,4	10,37	4,15
8 Гнездо на плату PBD-8 2,54мм 2х4pin, 10 шт.	шт	0,1	14,90	1,49

Продолжение таблицы 7.1

1	2	3	4	5
9 Текстолит двусторонний 100х100 мм, 5 шт.	шт	0,2	6,54	1,31
10 Гнездо питания на панель DS-026A (DJK-04B), 2,5х5,5мм, 1 шт.	шт	3	3,30	9,90
11 Стойка пластмассовая 10мм, 1 шт.	шт	4	0,39	1,56
Итого				94,46
Всего с учетом транспортных расходов (1,1)($Р_м$)				103,91

Расчет затрат по статье «Покупные комплектующие изделия, полуфабрикаты», осуществляется по формуле (7.2):

$$Р_к = К_{тр} \cdot \sum_{i=1}^{m} N_i \cdot Ц_{отпi}, \quad (7.2)$$

где $К_{тр}$ – коэффициент транспортных расходов;

m – номенклатура применяемых комплектующих;

N_i – количество комплектующих *i*-го вида на единицу изделия, нат. ед./шт.;

$Ц_{отпi}$ – цена за единицу комплектующего *i*-го вида, р.

Таблица 7.2 – Расчет затрат на комплектующие изделия

Наименование комплектующего	Количество на изделие, шт.	Цена за единицу, р.	Сумма, р.
1	2	3	4
1 Микроконтроллер Nano V3.0 ATmega328P CH340G 16 МГц	1	26,00	26,00
2 Модуль RTC часов реального времени DS3231 (AT24C32) 3,3В/5В	1	9,00	9,00
3 Модуль датчика дождя MH-RD	1	5,00	5,00
4 Дисплей LCD2004 IIC/I2C	1	19,50	19,50
5 Модуль Wi-Fi ESP8266 (ESP-01)	1	23,80	23,80
6 Датчик температуры и влажности DHT22	1	20,40	20,40
7 Драйвер двигателей TB6612FNG Ultra L298N	1	16,00	16,00
8 Датчик влажности почвы	1	8,50	8,50
9 Модуль энкодера EC11	1	11,66	11,66
10 Линейный электропривод 12В, 35 см	1	128,15	128,15

Продолжение таблицы 7.2

1	2	3	4
11 Сервопривод SG90	2	17,00	24,00
12 Модуль стабилизатора напряжения AMS1117 3,3В	1	5,8	5,8
13 Модуль датчика уровня воды	1	1,18	1,18
14 Модуль датчика температуры, давления и влажности BME280	1	20,40	20,40
15 Кнопка без фиксации PBS-10B-2, OFF-(ON) (1A 250VAC)	1	1,90	1,90
16 Насос центробежный погружной DC30E 12В	1	18,86	18,86
17 LED лента 12В 9,6Вт	1	14,41	14,41
18 Блок питания 12В 5А	1	25,25	25,25
19 Блок питания 5В 2А	2	10,25	20,50
20 Оптический датчик интенсивности освещенности GY-30 на базе BH1750	1	12,79	12,79
21 Модуль реле 2 канала 5В	1	7,50	7,50
Итого			419,89
Всего с учетом транспортных расходов (1,1)($Р_м$)			432,97

Расчет общей суммы прямых затрат на производство аппаратной части представлен в таблице 7.3.

Таблица 7.3 – Расчет общей суммы прямых затрат на производство аппаратной части

Показатель	Сумма, р.
1 Сырье и материалы	103,91
2 Покупные комплектующие изделия	432,97
Всего прямые затраты на производство аппаратной части ($З_р^{а.ч}$)	536,88

Расчет затрат на основную заработную плату разработчиков программной части комплекса представлен в таблице 7.4.

При расчете заработной платы используется среднемесячная заработная плата в Республике Беларусь для сотрудников ИТ-отрасли. Премия не начисляется.

Основная зарплата определяется по формуле (7.3):

$$З_о = К_{пр} \cdot \sum_{i=1}^{n} З_{чi} \cdot t_i, \quad (7.3)$$

где $К_{пр}$ – коэффициент премий и иных стимулирующих выплат;

n – категории исполнителей, занятых разработкой;

$З_{чi}$ – часовой оклад плата исполнителя *i*-й категории, р.;

t_i – трудоемкость работ, выполняемых исполнителем *i*-й категории, ч.

Часовая заработная плата каждого исполнителя определяется путем деления его месячной заработной платы (оклад плюс надбавки) на количество рабочих часов в месяце (расчетная норма рабочего времени на 2024 г. для 5-дневной недели составляет 168 часов по данным Министерства труда и социальной защиты населения на момент проведения расчетов).

Таблица 7.4 – Расчет затрат на основную заработную плату команды разработчиков

Категория разработчика	Месячный оклад, р.	Часовой оклад, р.	Трудоемкость работ, ч	Итого, р.
1 Инженер-программист	2250,00	13,40	30	402,00
2 Инженер-системотехник	1720,00	10,23	38	388,74
3 Инженер-отладчик	1470,00	8,75	30	262,50
Итого				1053,24
Премия и стимулирующие выплаты				0,00
Всего затраты на основную заработную плату разработчиков				1053,24

Дополнительная заработная плата определяется по формуле (7.4):

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (7.4)$$

где $Н_д$ – норматив дополнительной зарплаты, 15%.

Отчисления в фонд социальной защиты населения и обязательное страхование БелГосстрах ($З_{сз}$) определяется в соответствии с действующим законодательством по формуле (7.5):

$$Р_{соц} = \frac{(З_о + З_д) \cdot Н_{соц}}{100}, \quad (7.5)$$

где $Н_{соц}$ – норматив отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством по состоянию на апрель 2024 г. – 35%).

Расчет общей суммы затрат на разработку программной части программно-управляемого комплекса представлен в таблице 7.5.

Таблица 7.5 – Расчет затрат на разработку программного средства

Наименование статьи затрат	Формула/таблица для расчета	Сумма, р.
1	2	3
1 Основная заработная плата разработчиков	Табл. 7.4	1053,24
2 Дополнительная заработная плата разработчиков	$З_д = \frac{1053,24 \cdot 15}{100}$	157,99

Продолжение таблицы 7.5

1	2	3
3 Отчисления на социальные нужды	$Р_{соц}= \frac{(1053,24+157,99)\cdot 35}{100}$	423,93
Затраты на разработку программной части ($З_р^{п.ч}$)		1635,16

Формирование отпускной цены программно-аппаратного комплекса осуществляется в соответствии с методикой, представленной в таблице 7.6.

Таблица 7.6 – Методика формирования отпускной цены программно-аппаратного комплекса

Показатель	Формула/таблица для расчета
1 Затраты на производство аппаратной части ($З_р^{а.ч}$)	Табл. 7.3
2 Затраты на разработку программной части ($З_р^{п.ч}$)	Табл. 7.5
3 Сумма затрат на производство программно-аппаратного комплекса	$З_{пр} = З_р^{а.ч.} + З_р^{п.ч.}$ (7.6)
4 Накладные расходы	$Р_{накл} = \frac{З_{пр} \cdot Н_{накл}}{100}$, (7.7) где $Н_{накл}$ – норматив накладных расходов, 65%
5 Расходы на реализацию	$Р_{реал} = \frac{З_{пр} \cdot Н_{реал}}{100}$, (7.8) где $Н_{реал}$ – норматив расходов на реализацию, 2%
6 Полная себестоимость	$С_п = З_{пр} + Р_{накл} + Р_{реал}$ (7.9)
7 Плановая прибыль, включаемая в цену	$П_{ед} = \frac{С_п \cdot Р_{пр}}{100}$, (7.10) где $Р_{пр}$ – рентабельность продукции, 30%
8 Отпускная цена	$Ц_{отп} = С_п + П_{ед}$ (7.11)

Расчет отпускной цены осуществляется в таблице 7.7.

Таблица 7.7 – Формирование отпускной цены программно-аппаратного комплекса

Показатель	Формула/таблица для расчета	Сумма, р.
1	2	3
1 Затраты на производство аппаратной части ($З_р^{а.ч}$)	Табл. 7.3	536,88
2 Затраты на разработку программной части ($З_р^{п.ч}$)	Табл 7.5	1635,16

Продолжение таблицы 7.7

1	2	3
3 Сумма затрат на производство программно-аппаратного комплекса	$З_{пр}$ = 536,88 + 1635,16	2172,04
4 Накладные расходы	$Р_{накл}$ = 2172,04 · 0,65	1411,83
5 Расходы на реализацию	$Р_{реал}$ = 2172,04·0,02	43,44
6 Полная себестоимость	$С_{п}$ = 2172,04 + 1411,83 + 43,44	3627,31
7 Плановая прибыль, включаемая в цену	$П_{ед}$ = 3627,31 · 0,3	1088,20
8 Отпускная цена	$Ц_{отп}$ = 3627,31 + 1088,20	4715,51

Результатом производства аппаратно-программного комплекса является прирост чистой прибыли, полученный от их реализации и рассчитываемый по формуле (7.12):

$$\Delta П_{ч} = П_{ед} \cdot N_{п} \left(1 - \frac{Н_{п}}{100}\right), \quad (7.12)$$

где $N_{п}$ – прогнозируемый годовой объем производства и реализации, шт.;

$П_{ед}$ – прибыль, включаемая в цену, р.;

$Н_{п}$ – ставка налога на прибыль согласно действующему законодательству, (по состоянию на апрель 2024 г. – 20 %).

$$\Delta П_{ч} = 1088,20 \cdot 200 \cdot (1 - 0,20) = 174112,00 \text{ р.}$$

7.3 Расчет инвестиций в проектирование и производство программно-аппаратного комплекса

Инвестиции в производство аппаратно-программного комплекса включают в общем случае:

– инвестиции на его разработку;

– инвестиции в прирост собственного оборотного капитала (затраты на приобретение необходимых для производства нового изделия материалов, комплектующих, начатой, но незавершенной продукции и т.п.);

– инвестиции в прирост основного капитала (затраты на приобретение необходимого для производства нового изделия оборудования, станков и т.п.).

7.3.1 Инвестиции в разработку нового изделия

Инвестиции ($И_{р}$) рассчитываются по затратам на разработку нового изделия. Основная заработная плата разработчиков рассчитывается по формуле (7.3). Исходя из расчетов, полученных в таблице 7.5, размер инвестиций для разработки системы составляет 1635,16 р.

7.3.2 Расчет инвестиций в прирост собственного оборотного капитала

Расчет инвестиций в прирост собственного оборотного капитала осуществляется следующим образом:

1. Определяется годовая потребность в материалах по формуле (7.13).

$$П_м = Р_м \cdot N_п , \quad (7.13)$$

где $N_п$ – прогнозируемый годовой объем производства и реализации, шт.;

$Р_м$ – затраты на материалы на единицу изделия, р.

2. Определяется годовая потребность в комплектующих изделиях по формуле (7.14):

$$П_к = Р_к \cdot N_п , \quad (7.14)$$

где $N_п$ – прогнозируемый годовой объем производства и реализации, шт.;

$Р_м$ – затраты на комплектующие изделия на единицу продукции, р.

3. Вычисляются инвестиции в прирост собственного оборотного капитала в процентах от годовой потребности в материалах и комплектующих изделиях – β (исходя из среднего уровня по экономике: 25 %) по формуле (7.15):

$$И_{с.о.к} = \beta \cdot (П_м + П_к). \quad (7.15)$$

Расчет инвестиций в прирост собственного оборотного капитала представлен в таблице 7.9.

Таблица 7.9 – Расчет инвестиций в прирост собственного оборотного капитала

Наименование статьи затрат	Формула/таблица для расчета	Сумма, р.
1 Годовая потребность в материалах	$П_м = 103,91 \cdot 200$	20782,00
2 Годовая потребность в комплектующих изделиях	$П_м = 432,97 \cdot 200$	86594,00
Инвестиции в прирост собственного капитала	$И_{с.о.к} = 0,25 \cdot (20782,00 + 86594,00)$	26844,00

Исходя из расчетов, полученных в таблице 7.9, размер инвестиций в прирост собственного оборотного капитала составляет 26844,00 р.

7.3.3 Расчет инвестиций в прирост основного капитала

Инвестиции в прирост основного капитала не требуются, так как

производство нового изделия планируется осуществлять на действующем оборудовании в связи с наличием на предприятии-производителе свободных производственных мощностей.

7.4 Расчет показателей экономической эффективности инвестиций в проектирование и производство программно-аппаратного комплекса

Оценка экономической эффективности разработки и производства нового изделия зависит от результата сравнения инвестиций в производство нового изделия (инвестиции в разработку и прирост собственных оборотных средств) и полученного годового прироста чистой прибыли.

Сумма инвестиций меньше суммы годового экономического эффекта, то есть инвестиции окупятся менее чем за год, оценка экономической эффективности инвестиций в производство нового изделия осуществляется на основе расчета рентабельность инвестиций (затрат) по формуле (7.16):

$$ROI = \frac{\Delta П_ч - (И_р + И_{с.о.к})}{И_р + И_{с.о.к}} \cdot 100\%, \quad (7.16)$$

где $\Delta П_ч$ – прирост чистой прибыли от производства и реализации новых изделий, р.;

$И_р$, $И_{с.о.к}$ – инвестиции в разработку нового изделия и прирост собственного оборотного капитала, р.

$$ROI = \frac{174112{,}00 - (1635{,}16 + 26844{,}00)}{1635{,}16 + 26844{,}00} \cdot 100\% = 511{,}37\%$$

7.5 Вывод об экономической эффективности

В результате технико-экономического обоснования был произведен расчет инвестиций в разработку программно-аппаратного комплекса, расчет результата от разработки и использования программного средства и расчет показателей экономической эффективность. Общая сумма затрат на разработку программно-аппаратного комплекса составила 2172,04 рублей, отпускная цена – 4715,51 рублей. Экономическая эффективность инвестиций составляет 511,37 %.

Сравнивая инвестиции в разработку изделия и прирост собственных оборотных средств с приростом годовой чистой прибыли, можно сделать вывод, что инвестиции окупаются в течение года.

Рентабельность инвестиций в разработку и производство данного комплекса превышает ставки по долгосрочным депозитам в белорусских рублях на момент расчетов, что свидетельствует об экономической эффективности инвестиций. Следовательно, производство программно-

аппаратного комплекса для управления теплицей на основе беспроводной технологии является экономически эффективным, инвестиции в его разработку целесообразны.

Полученные результаты технико-экономического обоснования позволяют сделать вывод о высокой эффективности инвестиций в разработку программно-аппаратного комплекса для управления теплицей. Этот комплекс обладает перспективой стать востребованным на рынке, благодаря своей компактности, функциональности и экономической выгодности. Внедрение данного продукта позволит повысить эффективность управления тепличным хозяйством, обеспечивая оптимальные условия для роста растений.

Кроме того, разработанный программно-аппаратный комплекс представляет собой инновационное решение, способное повысить конкурентоспособность агропромышленных предприятий. Его использование позволит автоматизировать процессы управления теплицей, сократить затраты на энергию и ресурсы, а также повысить качество и количество выращенной продукции.

ЗАКЛЮЧЕНИЕ

В результате работы над данным дипломным проектом был разработан и протестирован программно-аппаратный комплекс для управления теплицей, включающий в себя микроконтроллер, модули датчиков, конечные устройства и систему удаленного мониторинга и управления через разработанноеAPI.

На основе принципиальной схемы которой была разведена и изготовлена печатная плата. Данный этап включал выбор компонентов, проектирование дорожек и подключение всех элементов системы.

Основу системы составляет плата микроконтроллера Arduino Nano V3.0, которая управляет всеми аспектами тепличного контроля. Для реализации связи и передачи данных использовались модули Wi-Fi (ESP-01) и датчики окружающей среды. Программная часть комплекса написана на языке C с использованием среды разработки Arduino IDE, что обеспечило гибкость и удобство работы с различными модулями и датчиками.

Программно-аппаратный комплекс реализует следующие функциональные возможности:

– сбор данных с различных датчиков окружающей среды;

– управление конечными устройствами для поддержания оптимальных условий в теплице (включение/выключение освещения, системы полива и вентиляции);

– обработка и хранение данных, а также предоставление их пользователю через удобный веб-интерфейс;

– возможность удаленного мониторинга и управления системой через интернет.

Проектирование и реализация системы управления теплицей также включало написание и тестирование программных фрагментов для взаимодействия с аппаратными модулями. В процессе разработки были выполнены все необходимые расчеты для обеспечения надежной и стабильной работы системы.

Экономическая эффективность проекта была тщательно проанализирована. Общая сумма затрат на разработку программно-аппаратного комплекса составила 2172,04 рублей, а отпускная цена – 4715,51 рублей. Экономическая эффективность инвестиций составляет 511,37 %. Эти расчеты показывают, что инвестиции в разработку и производство данного комплекса окупаются в течение года, делая проект экономически целесообразным.

Проект продемонстрировал высокую надежность и функциональность системы, что подтверждается результатами тестирования в реальных условиях. В будущем возможна дальнейшая доработка и расширение функциональности комплекса, включая добавление новых датчиков, улучшение алгоритмов управления и интеграция с другими системами автоматизации.