

Министерство сельского хозяйства Российской Федерации Де-  
партамент научно-технологической политики и образования *Феде-  
ральное государственное бюджетное образовательное учреждение  
высшего образования*  
**«Красноярский государственный аграрный университет»**

**Амбросенко Н.Д.**

## **ТЕХНОЛОГИИ IoT В АГРОПРОМЫШЛЕННОМ КОМ- ПЛЕКСЕ**

**Методические указания к лабораторным работам**

*Электронное издание*

**ФГОС ВО 3++**

Направление подготовки 09.04.03 «Прикладная информатика»

Профиль: Цифровые технологии в АПК

Курс: 1

Семестр: 1, 2

Форма обучения: очная

Квалификация выпускника: магистр

Красноярск 2025

*Рецензент*

Л.В. Фомина, к.с.-х.н., доцент,  
доцент кафедры «Государственное, муниципальное управление  
и кадровая политика»

**Амбросенко Н.Д.**

**Технологии IoT в агропромышленном комплексе:** метод.  
указания к лабораторным работам [Электронный ресурс] /  
Н.Д. Амбросенко; Краснояр. гос. аграр. ун-т. – Красноярск, 2025. –  
50 с.

Содержит сведения о порядке приведения лабораторных работ,  
список необходимых модулей и деталей для эксперимента.

Предназначены для магистрантов 1-го курса, обучающихся по  
направлению 09.04.03 «Прикладная информатика».

.

Издается по решению редакционно-издательского совета Крас-  
ноярского государственного аграрного университета

© Амбросенко Н.Д., 2025  
© ФГБОУ ВО «Красноярский государственный  
аграрный университет», 2025

## СОДЕРЖАНИЕ

|                                                                      |    |
|----------------------------------------------------------------------|----|
| Введение .....                                                       | 4  |
| Лабораторная работа 1. Подключение устройства к Wi-Fi сети.<br>..... | 6  |
| Лабораторная работа 3: Система регистрации данных .....              | 28 |
| Лабораторная работа 4: Напоминальник .....                           | 32 |
| Лабораторная работа 5: Браузерный Dendy.....                         | 36 |
| Лабораторная работа 6. Умный дом.....                                | 41 |
| Лабораторная работа 7. Telegram-бот .....                            | 45 |
| Лабораторная работа 8. Блинк .....                                   | 48 |

## Введение

Интернет вещей (IoT) – это уже не просто тренд, а фундаментальная основа для построения информационно-управляющих систем будущего. Понимание принципов работы IoT требует комплексного подхода, охватывающего микроконтроллерные системы, схемотехнику, беспроводные сети, сервисы сбора и обработки данных. Только освоив все эти аспекты, можно создавать эффективные и надежные IoT-решения.

Данные методические указания предлагают вам погрузиться в мир IoT, изучив его ключевые компоненты и освоив практические навыки разработки IoT-устройств. Мы сосредоточимся на платформе Iskra UNO (Arduino) и комплекте для начинающих как на оптимальном инструменте для получения практического опыта.

В рамках данных методических указания обучающийся:

- Получит прочные знания в области микроконтроллерных систем и схемотехники.
- Научится настраивать и использовать беспроводные сети для IoT-устройств.
- Освоит принципы работы с облачными платформами для сбора, обработки и визуализации данных.
- Реализует ряд практических проектов, демонстрирующих возможности IoT.

Цель – подготовить обучающихся к решению сложных задач в области IoT и дать все необходимые инструменты для создания собственных инновационных решений. Iskra UNO (Arduino Uno) и комплект для начинающих, станут проводником в мир Интернета вещей. В результате выполнения лабораторных работ получим ответы на вопросы:

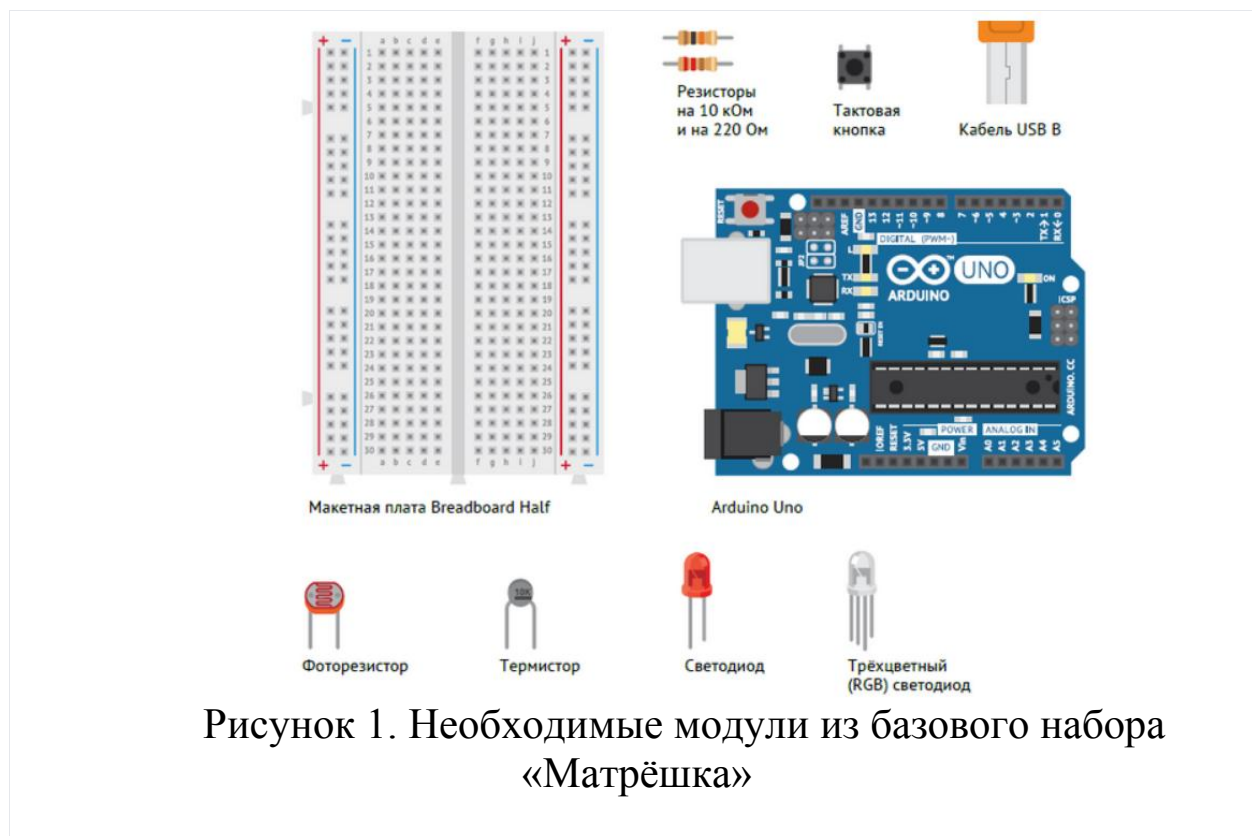
- Какие основные компоненты необходимы для создания проекта интернета вещей?
- Какие типы датчиков и исполнительных механизмов используются в интернете вещей?
- Какие облачные платформы лучше использовать для хранения и анализа данных интернета вещей?
- Какие преимущества использования платформы Arduino для разработки проектов интернета вещей?

- Какие методы анализа данных, такие как data mining и machine learning, могут быть использованы в проектах интернета вещей?

## Лабораторная работа 1. Подключение устройства к Wi-Fi сети.

Список модулей и деталей для эксперимента:

1. Набор для экспериментов IoT (Интернет вещей) — дополнение к набору «Матрёшка Z»
2. Необходимые модули из базового набора «Матрёшка Z»:



- Iskra UNO
- Макетная площадка Breadboard Half
- Переменный резистор (потенциометр)
- Кнопка тактовая
- Светодиод
- Трёхцветный светодиод
- Резисторы на 220 Ом
- Резисторы на 10 кОм
- Фоторезистор
- Термистор

3. Библиотеки для Arduino:

Все необходимые библиотеки находятся в одном архиве.

1. Скачайте .zip-архив со всеми библиотеками.

2. Разархивируйте его. Оставьте каждую библиотеку в своём архиве.

3. В меню Arduino IDE выберите пункт Эскиз → Импорт библиотек... → Добавить библиотеку....

4. Добавьте таким же образом остальные библиотеки.

Файлы на флеш-карте

На флешке в составе набора уже есть файлы. Они понадобятся для проектов, не изменяйте и не удаляйте их. Но если всё-таки что-то пошло не так — всегда можно скачать архив с файлами.

- Скачайте .zip архив с необходимыми для экспериментов файлами.

- Разархивируйте файлы и скопируйте их на microSD-карту.

*Внимание! Iskra UNO может работать только с картами, отформатированными в FAT32.*

### **Эксперимент: Подключение устройства к Wi-Fi сети**

Wi-Fi (Тройка-модуль): распиновка, схема подключения и программирование

На рисунке 2 приведен Wi-Fi (Тройка-модуль) на модуле ESP-32 с чипом ESP8266EX, который позволит передать данные по Wi-Fi сети .

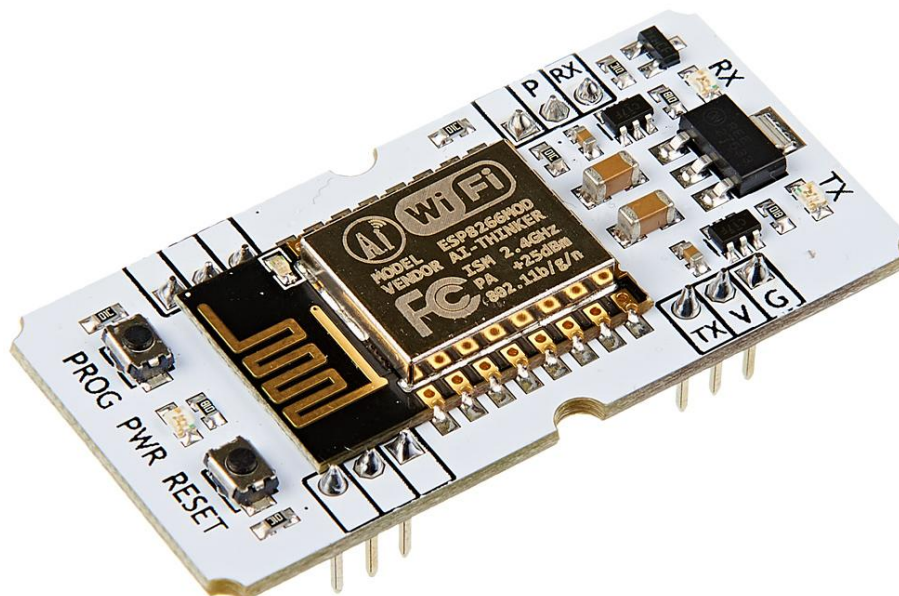


Рисунок 2. Wi-Fi (Тройка-модуль) на модуле ESP-32 с чипом ESP8266EX

Видеообзор: Подключение модулей на базе esp8266 и esp32 к облачной платформе : <https://www.youtube.com/watch?v=7XzaUGr3-BA>

Общие сведения:

По умолчанию модуль настроен на работу через «АТ-команды». Управляющая плата посылает команды — Wi-Fi модуль выполняет соответствующую операцию. Под металлической крышкой прячется целый микроконтроллер, который является самодостаточным устройством. Прошивать модуль можно на разных языках программирования.

Подключение и настройка:

В стандартной прошивке Troyka Wi-Fi общается с управляющей платой через «АТ-команды» по протоколу UART. Дополнительный сигнальный пин **Р** служит для перевода модуля в режим сна или пониженного энергопотребления.

На всех платах Iskra и Arduino присутствует хотя бы один аппаратный UART — HardwareSerial. Если же по каким-то причинам он занят другим устройством, можно воспользоваться программным UART — SoftwareSerial.

HardwareSerial

На управляющей плате Iskra JS и платах Arduino с микроконтроллером ATmega32U4 / ATSAMD21G18 данные по USB и общение через пины 0 и 1 осуществляется через два отдельных UART. Это даёт возможность подключить Wi-Fi модуль к аппаратному UART на пинах 0 и 1.

Список поддерживаемых плат:

- Iskra JS
- Iskra Neo
- Arduino Leonardo
- Arduino Leonardo ETH
- Arduino Leonardo ETH PoE
- Arduino Yún
- Arduino Tian

При подключении удобно использовать Troyka Shield ( см. рисунок 3).



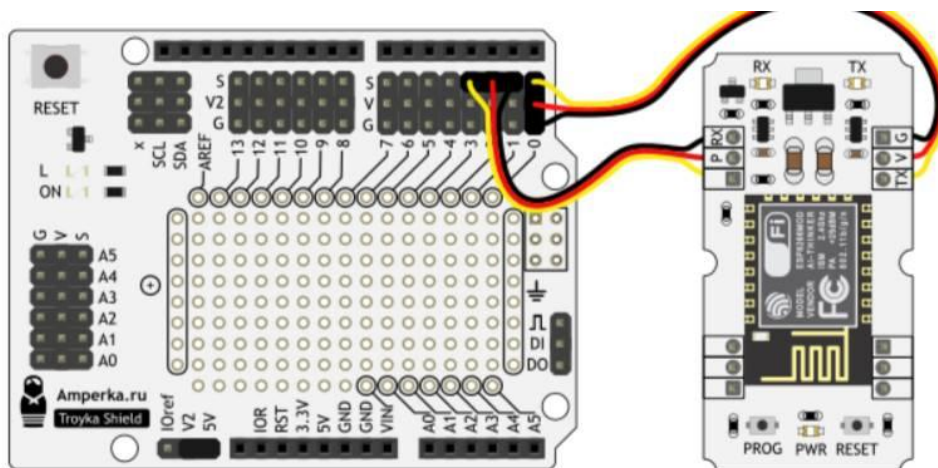


Рисунок 3: Схема подключения Troyka Shield и Wi-Fi модуля

С Troyka Slot Shield можно обойтись без лишних проводов (см. рисунок 4).

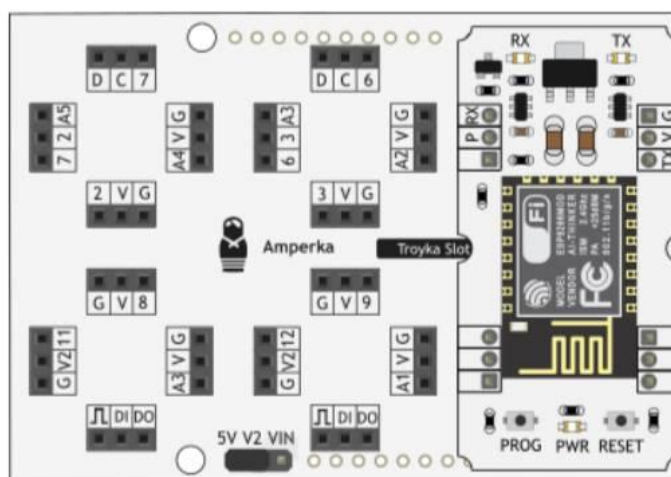


Рисунок 4: Схема подключения Troyka Slot Shield и Wi-Fi модуля

Прошейте управляющую платформу кодом ниже.

HardwareSerialAT115200.ino

// serial-порт к которому подключён Wi-Fi модуль

#define WIFI\_SERIAL Serial1

void setup()

{

// открываем последовательный порт для мониторинга действий в программе

// и передаём скорость 9600 бод

```

Serial.begin(9600);
while (!Serial) {
  // ждём, пока не откроется монитор последовательного порта
  // для того, чтобы отследить все события в программе
}
Serial.print("Serial init OK\r\n");
// открываем Serial-соединение с Wi-Fi модулем на скорости 115200
бод
WIFI_SERIAL.begin(115200);
}

void loop()
{
  // если приходят данные из Wi-Fi модуля - отправим их в порт компь-
  ютера
  if (WIFI_SERIAL.available()) {
    Serial.write(WIFI_SERIAL.read());
  }
  // если приходят данные из компьютера - отправим их в Wi-Fi модуль
  if (Serial.available()) {
    WIFI_SERIAL.write(Serial.read());
  }
}

```

### SoftwareSerial

Некоторые платы Arduino, например Uno, прошиваются через пины 0 и 1. Это означает невозможность использовать одновременно прошивку/отладку по USB и общение с Wi-Fi модулем. Решение проблемы — программный UART. Подключите пины TX и RX Troyka Wi-Fi к другим контактам управляющей платы и используйте библиотеку SoftwareSerial.

Для примера подключим управляющие пины Wi-Fi модуля TX и RX — на 8 и 9 контакты управляющей платы.

При подключении удобно использовать Troyka Shield ( см. рисунок 5).

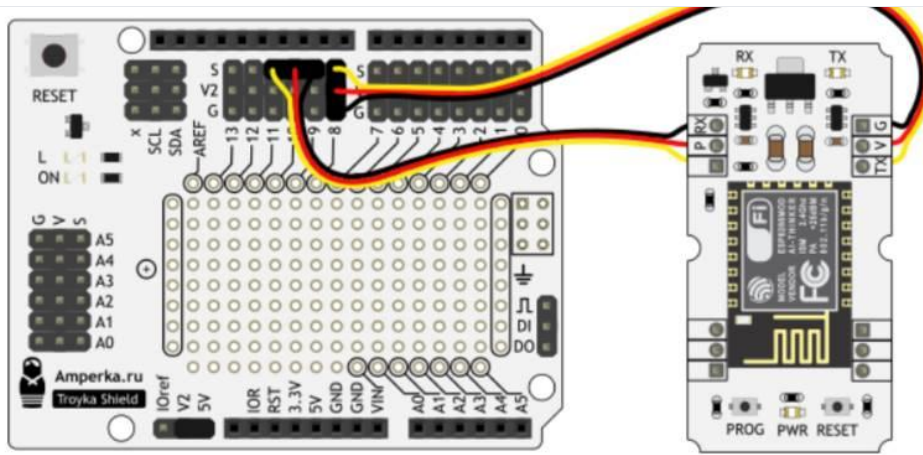


Рисунок 5: Схема подключения Troika Shield и Wi-Fi модуль

Прошейте управляющую платформу кодом ниже.

Код прошивки

SoftwareSerialAT115200.ino

// библиотека для работы программного Serial

#include <SoftwareSerial.h>

// создаём объект для работы с программным Serial

// и передаём ему пины TX и RX

SoftwareSerial mySerial(8, 9);

// serial-порт к которому подключён Wi-Fi модуль

#define WIFI\_SERIAL mySerial

void setup()

{

// открываем последовательный порт для мониторинга действий в программе

// и передаём скорость 9600 бод

Serial.begin(9600);

while (!Serial) {

// ждём, пока не откроется монитор последовательного порта

// для того, чтобы отследить все события в программе

}

Serial.print("Serial init OK\r\n");

// открываем Serial-соединение с Wi-Fi модулем на скорости 115200

бод

WIFI\_SERIAL.begin(115200);

}

void loop()

```

{
  // если приходят данные из Wi-Fi модуля - отправим их в порт компь-
  ютера
  if (WIFI_SERIAL.available()) {
    Serial.write(WIFI_SERIAL.read());
  }
  // если приходят данные из компьютера - отправим их в Wi-Fi модуль
  if (Serial.available()) {
    WIFI_SERIAL.write(Serial.read());
  }
}

```

### *Примеры работы AT-команд*

Рассмотрим несколько примеров по работе с «АТ-командами» при подключении к Uno.

Тестовая команда «АТ»

Откройте монитор порта (рисунок 6). Настройте скорость соединения — 9600 бод. Конец строки — NL & CR. Введите команду АТ и нажмите «Отправить». Это — базовая команда для проверки работы Wi-Fi модуля.

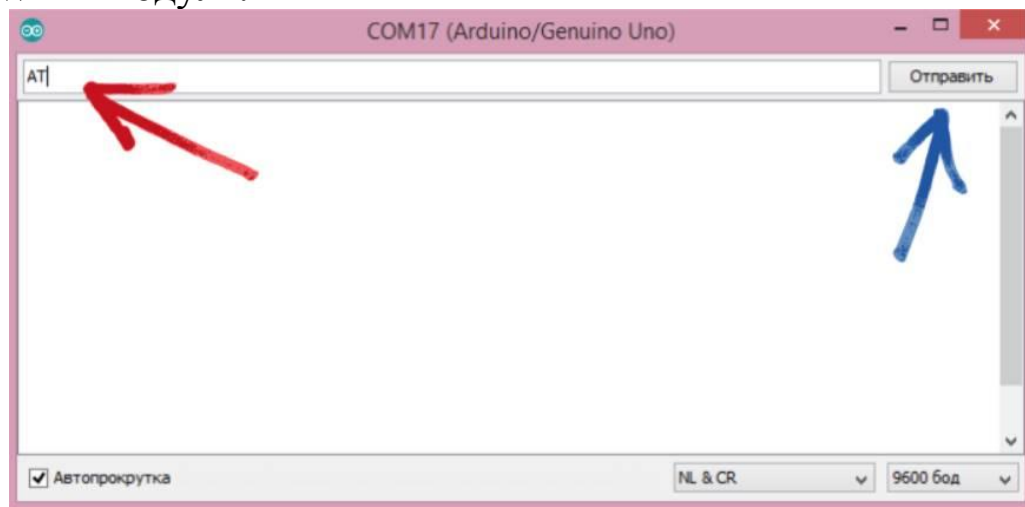


Рисунок 6: Монитор порта

В ответ получим «ОК» рисунок 7

:

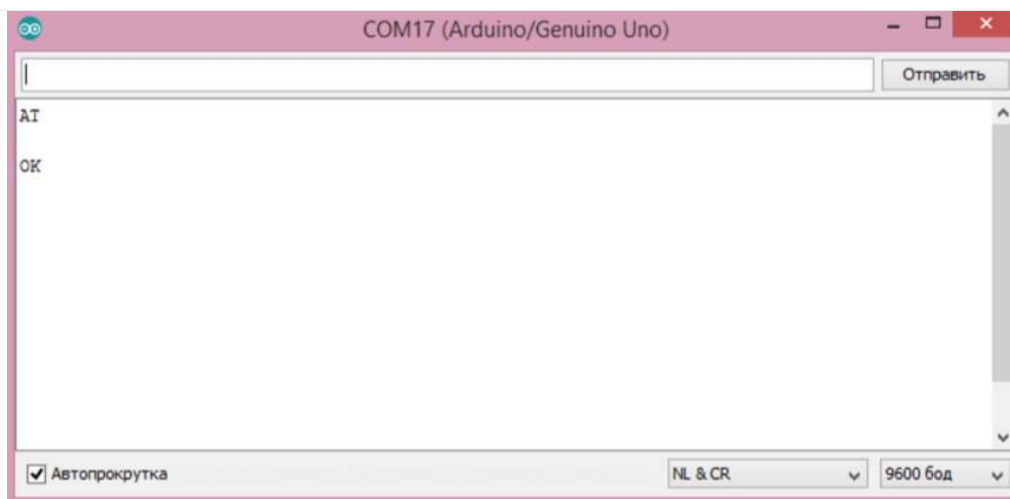


Рисунок 7. Окно выполнения команды

Если ответа нет или появляются непонятные символы — проверьте правильность подключения и настройки скорости обмена данными.

### *Настройка режима работы*

Wi-Fi модуль умеет работать в трёх режимах:

1. Режим точки доступа
2. Режим клиента
3. Смешанный режим

Переведём чип в смешанный режим командой ( см. рисунок 8: `AT+CWMODE_DEF=3`

После установки модуль должен ответить «OK»:

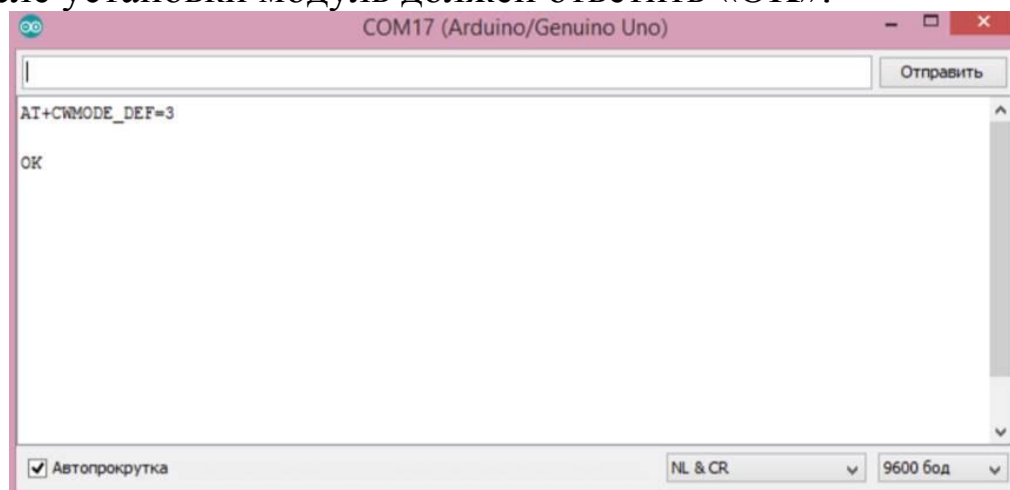


Рисунок 8 Экранное окно перевода чипа в смешанный режим

*В отличие от аппаратного UART (HardwareSerial), за работу программного UART (SoftwareSerial) отвечает микроконтроллер, который назначает другие пины в режим работы RX и TX, соот-*

ответственно и данные которые приходят от Wi-Fi модуля обрабатывает сам микроконтроллер во время программы. По умолчанию скорость общения Тройка Wi-Fi равна 115200, что значительно выше чем позволяет библиотека SoftwareSerial. В итоге часть информации которая приходит с Wi-Fi модуля будет утеряна. Если вы используете плату с HardwareSerial подключением модуля можете пропустить пункт настройки скорости и сразу перейти к дальнейшей работе с модулем.

#### *AT установка скорости общения*

Для корректной работы с большими объемами необходимо понизить скорость соединения модуля и микроконтроллера. Для этого используйте «AT-команду»:

AT+UART\_DEF=9600,8,1,0,0

После проделанной операции, измените скорость программного UART в скетче программы и прошейте плату.

SoftwareSerialAT9600.ino

// библиотека для работы программного Serial

#include <SoftwareSerial.h>

// создаём объект для работы с программным Serial

// и передаём ему пины TX и RX

SoftwareSerial mySerial(8, 9);

// serial-порт к которому подключён Wi-Fi модуль

#define WIFI\_SERIAL mySerial

void setup()

{

// открываем последовательный порт для мониторинга действий в программе

// и передаём скорость 9600 бод

Serial.begin(9600);

while (!Serial) {

// ждём, пока не откроется монитор последовательного порта

// для того, чтобы отследить все события в программе

}

Serial.print("Serial init OK\r\n");

// открываем Serial-соединение с Wi-Fi модулем на скорости 9600 бод

WIFI\_SERIAL.begin(9600);

}

```

void loop()
{
  // если приходят данные из Wi-Fi модуля - отправим их в порт компь-
  ютера
  if (WIFI_SERIAL.available()) {
    Serial.write(WIFI_SERIAL.read());
  }
  // если приходят данные из компьютера - отправим их в Wi-Fi модуль
  if (Serial.available()) {
    WIFI_SERIAL.write(Serial.read());
  }
}

```

По итогу программный UART успеет обработать каждый пришедший байт с Wi-Fi модуля.

### *AT сканирование WI-FI сетей*

Откройте Serial-порт и отправьте на модуль «АТ-команду» для сканирования всех доступных Wi-Fi сетей:

Команда: AT+CWLAP

При наличии доступных WI-FI сетей в ответ получим сообщение:

```

AT+CWLAP
+CWLAP:(0,"AI-THINKER_03842B",-65,"62:01:94:03:84:2b",1,1,-40)
+CWLAP:(3,"PiRExternal",-90,"f0:29:29:26:b8:31",1,-12,0)
+CWLAP:(3,"PiRGroup",-84,"44:ad:d9:87:c8:f0",1,-4,0)
+CWLAP:(4,"corp.Catherine.ru",-82,"f8:1a:67:4c:bf:59",2,10,0)
+CWLAP:(3,"PiRGroup",-87,"f0:29:29:26:b8:30",1,-11,0)
+CWLAP:(3,"Amperka",-44,"6c:3b:6b:ff:0f:4d",6,8,0)
+CWLAP:(3,"SEC_LinkShare_ddd43e",-79,"d0:66:7b:08:93:dd",6,23,0)
+CWLAP:(3,"DIT_AMPP",-82,"1c:b9:c4:25:13:68",1,32767,0)
+CWLAP:(3,"MZPK",-89,"24:a2:e1:eb:5d:08",11,-16,0)
+CWLAP:(0,"Arduino-Yun-B4218AF05F28",-
62,"b4:21:8a:f0:5f:28",11,25,0)
+CWLAP:(3,"PiRExternal",-85,"44:ad:d9:87:c8:f1",1,-4,0)

```

OK

На рисунке 9 приведена экранная форма доступных Wi-Fi сетей



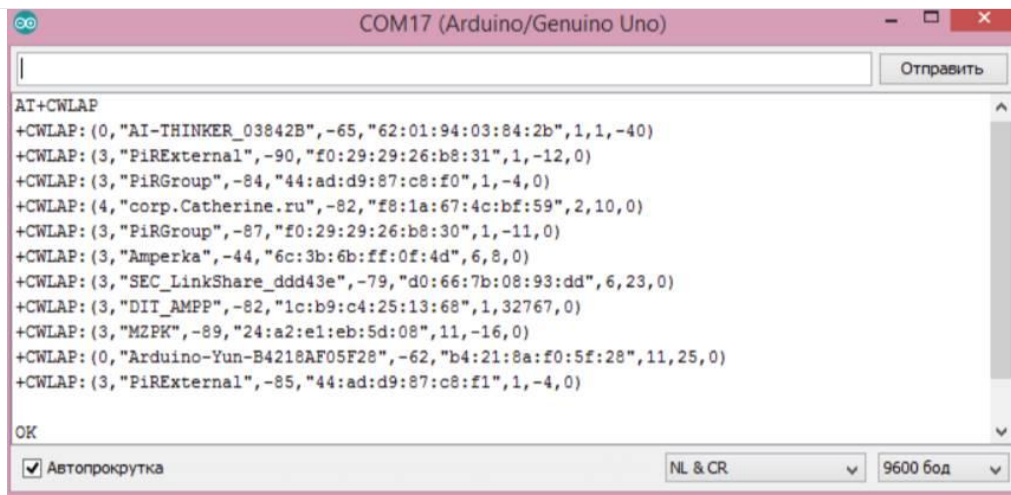


Рисунок 9. Окно доступных Wi-Fi сетей

Для продолжение работы используйте перечень «АТ-команд».

### *Wi-Fi модуль как самостоятельный контроллер*

Wi-Fi (Тройка-модуль) — очень умный модуль. Под металлической крышкой прячется целый микроконтроллер, который можно программировать на языке C++ через Arduino IDE и JavaScript через Espruino Web IDE.

#### *1. Настройка железа*

Ввиду отсутствия у платформы Тройка WiFi собственного USB-порта, подключите её к компьютеру, используя один из перечисленных способов:

- через платформу Arduino Uno (рисунок 10)

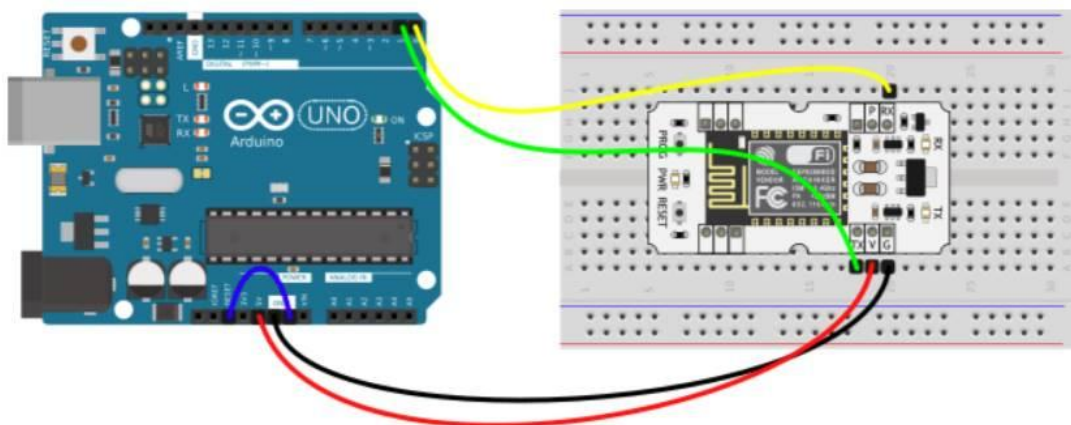


Рисунок 10. Схема подключенная Тройка WiFi через платформу Arduino Uno

- через USB-Serial адаптер (рисунок 11)



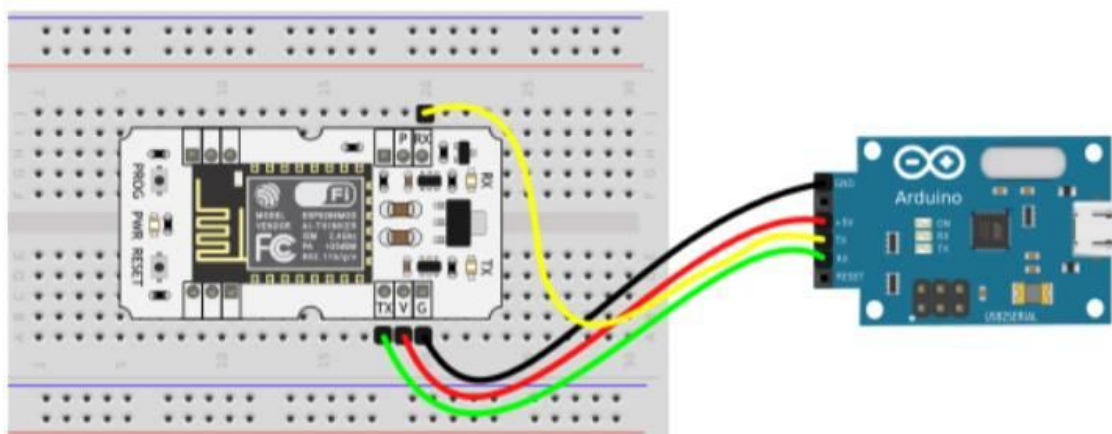


Рисунок 11. Схема подключенная Тройка WiFi через USB-Serial адаптер

- через USB-UART преобразователь (рисунок 12)

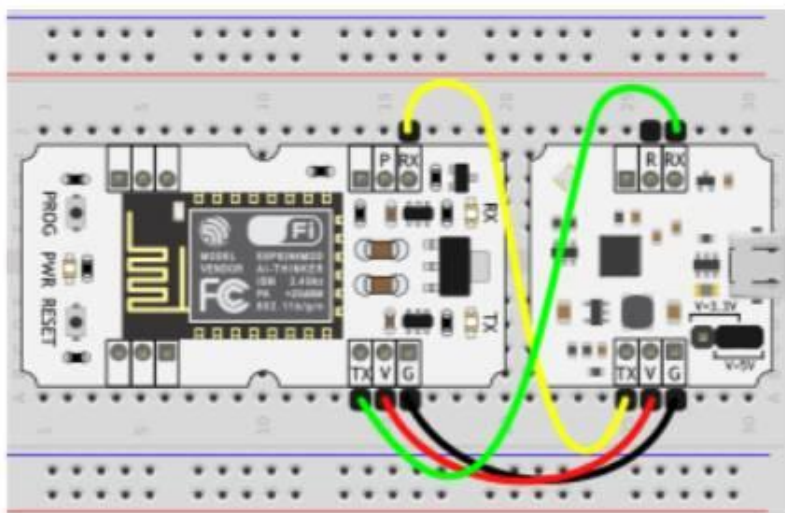


Рисунок 12. Схема подключенная Тройка WiFi через USB-UART преобразователь

2. *Переведите модуль в режим программирования:  
Необходимо каждый раз выполнять перед прошивкой модуля.*
  - Зажмите кнопку PROG;
  - Нажмите и отпустите кнопку RESET;
  - Отпустите кнопку PROG
3. *Железо готово, приступайте к программной части.*

## Программирование на C++



Рисунок 13. Набор компонентов

1. Для начала работы с платформой Troika Wi-Fi на языке C++, скачайте и установите на компьютер интегрированную среду разработки Arduino IDE.
2. По умолчанию среда программирования настроена только на AVR-платы. Для платформы Troika Wi-Fi добавьте в менеджере плат поддержку платформ на модуле ESP8266.
3. В пункте меню Инструменты Плата выбирайте Generic ESP8266 Module. (hbseyjr 14)

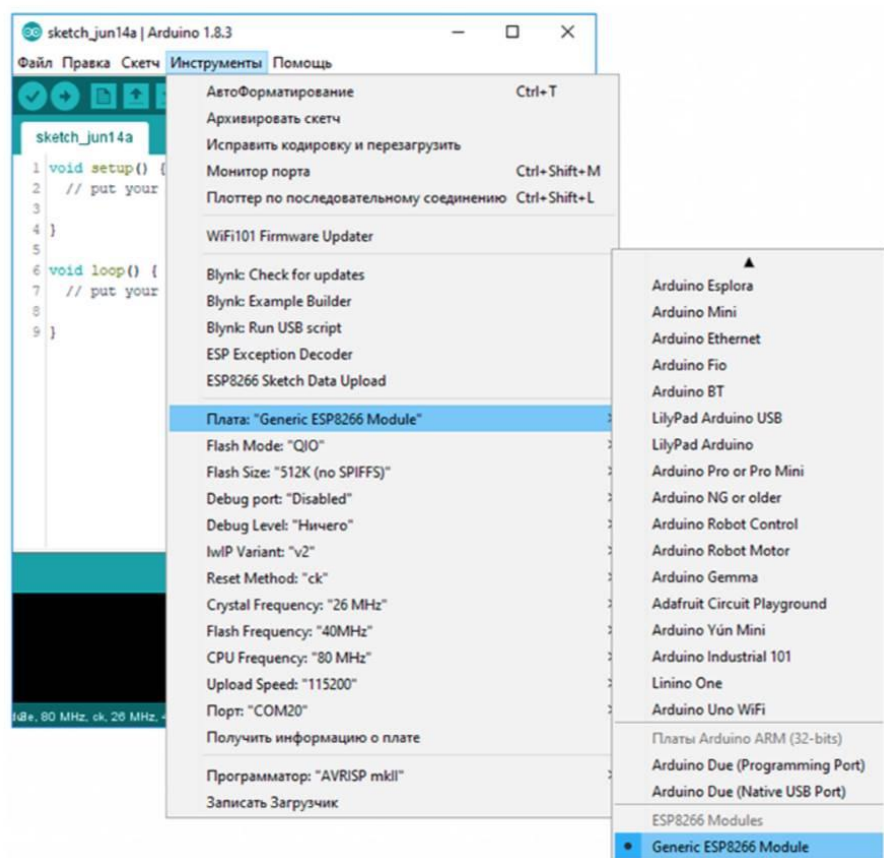


Рисунок 14. Настройка Arduino IDE

После выполненных действий платформа Troyka Wi-Fi готова к программированию через Arduino IDE.

Подробности о функциях и методах работы Troyka Wi-Fi (ESP8266) на языке C++ читайте на ESP8266 Arduino.

### *Восстановление стандартной AT-прошивки*

После программирования платформы в режиме самостоятельного контроллера может понадобиться восстановить на модуле стандартную AT-прошивку. Для этого необходимо воспользоваться утилитой Flash Download Tool.

Элементы платы показаны на рисунке 15.



Рисунок 15. Элементы модуля ESP 12F

### *Чип ESP8266EX*

Чип ESP8266 — выполнен по технологии SoC (англ. System-on-a-Chip — система на кристалле). В основе кристалла входит процессор семейства Xtensa — 32-х битный Tensilica L106 с частотой 80 МГц с ультранизким энергопотреблением, радиочастотный трансивер с физическим уровнем WiFi IEEE 802.11 b/g/ и блоки памяти SRAM. Мощности процессорного ядра хватает для работы сложных пользовательских приложений и цифровой сигнальной обработки.

Программное приложение пользователя должно храниться на внешней микросхеме Flash-памяти и загружаться в ESP8266EX через один из доступных интерфейсов (SPI, UART, SDIO и др.) каждый раз в момент включения питания системы.

Чип ESP8266 не содержит в себе Flash-память и многих других компонентов для пользовательского старта. Микросхема является основой на базе которой выпускаются модули с необходимой периферией, например ESP-12.

### *Модуль ESP-12*

ESP-12 — плата-модуль WiFi на базе популярного чипсета ESP8266EX. Под металлическим кожухом находится микросхема Flash-памяти объёмом 2 МБ, чип ESP8266EX и кварцевый резонатор. Flash-память необходима для хранения программного обеспечения. При каждом включении питания, ПО автоматически загружается в чип ESP8266EX.

Рядом с кожухом расположен индикаторный светодиод и миниатюрная антенна из дорожки на верхнем слое печатной платы в виде змейки. Металлический кожух экранирует компоненты модуля и тем самым улучшает электромагнитные свойства.

По периметру платы расположены 22 пина через которые модуль ESP-12 взаимодействует с внешним миром. В нашем случае коммуникация с компонентами и модулями на платформе Tioyka Wi-Fi.

### *Tioyka контакты*

На рисунке 16 приведены контакты платы.

### *Первая группа*

Сигнальный (TX) — цифровой выход Wi-Fi модуля. Используется для передачи данных в микроконтроллер. Подключите к пину RX микроконтроллера.

Питание (V) — соедините с рабочим напряжением микроконтроллера.

Земля (G) — соедините с пином GND микроконтроллера.

### *Вторая группа*

Сигнальный (P) — используйте для перевода модуля в режим сна. Подключите к любому цифровому пину микроконтроллера.

Сигнальный (RX) — цифровой вход Wi-Fi модуля. Используется для приёма данных из микроконтроллера. Подключите к пину TX микроконтроллера.

### *Дополнительные контакты*

Через Tioyka-контакты вы можете получить доступ к 4, 5, 12, 13 и 14 пину модуля ESP-12. Для этого достаточно капнуть припой на контактные площадки с обратной стороны устройства.



Рисунок 16. Тройка контакты

### *Светодиодная индикация*

| Имя светодиода | Назначение                                                               |
|----------------|--------------------------------------------------------------------------|
| RX и TX        | Мигают при обмене данными между Wi-Fi модулем и управляющим устройством. |
| POWER          | Горит, если на модуль подано питание                                     |

### *Кнопка RESET*

Кнопка предназначена для ручного сброса прошивки — аналог кнопки RESET обычного компьютера.

### *Кнопка PROG*

Кнопка служит для перевода модуля в режим прошивки:

- Зажмите кнопку PROG;
- Нажмите и отпустите кнопку RESET;
- Отпустите кнопку PROG

### *Стабилизатор напряжения*

Стабилизатор MC33275ST-3.3T3G с выходом 3,3 вольта, обеспечивает питание модуля ESP-12. Максимальный выходной ток составляет 300 мА.

На плате так же присутствует необходимая обвязка для сопряжения устройств с разными питающими напряжениями.

В нашем случае это может быть управляющая плата Iskra UNO с 5 вольтовой логикой и Wi-Fi модуль с 3,3 вольтовой логикой.



## Ответы на вопросы:

1. *Какой модуль используется для подключения Arduino к Wi-Fi? Приведите основные характеристики модуля.*

Ответ: Для подключения Arduino к Wi-Fi часто используется модуль ESP8266, а наиболее популярной его версией является ESP-01/12.

Основные характеристики ESP-01:

Работа в режимах: клиент (STA), точка доступа (AP) и режим комбинированного использования (STA+AP).

Поддерживаемые стандарты Wi-Fi: 802.11 b/g/n.

Интерфейс: Общение с Arduino осуществляется через UART с использованием AT-команд.

Напряжение питания: 1,7-3,6 В.

2. *Что нужно сделать перед загрузкой кода для ESP8266 в Arduino IDE? (Указать последовательность шагов выполнения)*

Ответ: Перед загрузкой кода для ESP8266 в Arduino IDE необходимо выполнить несколько шагов:

1. Установка Arduino IDE: Убедитесь, что у вас установлена последняя версия Arduino IDE. Если нет, скачайте и установите её с официального сайта Arduino.

2. Добавление поддержки плат ESP8266:

- Откройте Arduino IDE и перейдите в меню Файл -> Настройки.

- В поле Дополнительные ссылки для Менеджера плат добавьте следующую строку:

- [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

- Нажмите "ОК"<sup>24</sup>.

3. Установка пакета ESP8266:

- Перейдите в меню Инструменты -> Плата -> Менеджер плат....

- В открывшемся окне найдите esp8266 by ESP8266 Community и установите его<sup>12</sup>.

- Установка драйверов: Если вы используете плату, такую как NodeMCU или ESP-01, убедитесь, что у вас установлены необходимые драйверы для USB-UART преобразователя (например, CP2102 или CH340). Эти драйверы можно скачать с официальных сайтов производителей<sup>12</sup>.

- Подключение платы к компьютеру: Подключите вашу плату ESP8266 к компьютеру с помощью USB-кабеля. Убедитесь, что кабель исправен и поддерживает передачу данных.

4. Выбор платы и порта:

В меню Инструменты выберите соответствующую плату (например, NodeMCU 1.0 или Generic ESP8266 Module).

5. Выберите COM-порт, к которому подключена ваша плата<sup>34</sup>.

6. Настройка скорости загрузки: Установите скорость загрузки (Upload Speed) на 115200 бит/с для большинства плат ESP8266.

3. *Что нужно сделать, если Wi-Fi сигнал нестабильный?*

Ответ: Если ваше устройство будет работать в условиях с нестабильным сигналом Wi-Fi, рассмотрите возможность использования внешней антенны для улучшения качества сигнала.

4. *Какой код используется для подключения к Wi-Fi с помощью ESP8266?*

Ответ: `#include <SoftwareSerial.h>`

Подключение библиотеки: `#include <ESP8266WiFi.h>` - подключает библиотеку для работы с Wi-Fi.

## Лабораторная работа 2: Удалённый термометр

**Цель:** собрать устройство для наблюдения за температурой через Интернет, используя сервис dweet.io для визуализации данных.

**Рисунок:** (Здесь должна быть схема подключения датчика температуры к Arduino и ESP8266).

1. Собрана схема и разработана программа, которая каждую секунду отправляет данные о температуре на dweet.io. Параметры SSID и PASSWORD заменены на актуальные для вашей сети Wi-Fi. Вместо «твой\_ключ» необходимо указать свой уникальный ключ для dweet.io.

Листинг 2.1 – Листинг программы (Исходный код):

```
arduino
#include "ESP8266.h"
#include <SoftwareSerial.h>
#include <math.h>

#define SSID "имя_твоего_Wi-Fi"
#define PASSWORD "пароль_твоего_Wi-Fi"
#define TEMP_PIN A0

SoftwareSerial mySerial(4, 5);
ESP8266 wifi(mySerial);
String name = "твой_ключ";

void setup(void) {
  Serial.begin(9600);
  if (wifi.joinAP(SSID, PASSWORD)) {
    Serial.println("https://dweet.io/follow/" + name);
  } else {
    Serial.println("Wi-Fi connection error");
  }
}

void loop(void) {
  float v_temp = 1023.0 / analogRead(TEMP_PIN) - 1.0;
  float temp = 1.0 / ( -log(v_temp) / 3977.0
    + 1.0 / 295.0 ) - 273.0;
  if (wifi.createTCP("www.dweet.io", 80)) {
```



```

String data = "GET /dweet/for/" + name + "?";
data += "temp=" + String(temp) + " HTTP/1.1\r\n";
data += "Host: dweet.io\r\n\r\n";
wifi.send(data.c_str(), data.length());
wifi.releaseTCP();
} else {
  Serial.println("create TCP error");
}
delay(1000);
}

```

2. Код примера изменен таким образом, чтобы отправка данных на dweet.io происходила только в случае изменения температуры более чем на 0,5 градуса по Цельсию.

Листинг 2.2 — Листинг программы (модифицированный код):

arduino

```

#include "ESP8266.h"
#include <SoftwareSerial.h>
#include <math.h>

```

```

#define SSID "имя_твоего_Wi-Fi"
#define PASSWORD "пароль_твоего_Wi-Fi"
#define TEMP_PIN A0

```

```

SoftwareSerial mySerial(4, 5);
ESP8266 wifi(mySerial);
String name = "твой_ключ";
float lastTemp = 0; // Переменная для хранения последнего значения
температуры

```

```

void setup(void) {
  Serial.begin(9600);
  if (wifi.joinAP(SSID, PASSWORD)) {
    Serial.println("https://dweet.io/follow/" + name);
  } else {
    Serial.println("Wi-Fi connection error");
  }
}

```

```

void loop(void) {
  float v_temp = 1023.0 / analogRead(TEMP_PIN) - 1.0;
  float temp = 1.0 / ( -log(v_temp) / 3977.0
    + 1.0 / 295.0 ) - 273.0;
}

```

```

// Проверяем, изменилась ли температура на 0.5 градуса
if (abs(temp - lastTemp) > 0.5) {
    if (wifi.createTCP("www.dweet.io", 80)) {
        String data = "GET /dweet/for/" + name + "?";
        data += "temp=" + String(temp) + " HTTP/1.1\r\n";
        data += "Host: dweet.io\r\n\r\n";
        wifi.send(data.c_str(), data.length());
        wifi.releaseTCP();
    } else {
        Serial.println("create TCP error");
    }
    lastTemp = temp; // Обновляем последнее значение температуры
}
delay(1000);
}

```

Ответы на вопросы:

1. Что такое SoftwareSerial и почему её нужно использовать?

Ответ: SoftwareSerial — это библиотека, которая позволяет реализовать последовательный порт на цифровых выводах Arduino. Она необходима, так как стандартный последовательный порт (выводы 0 и 1) может использоваться для отладки и загрузки скетчей.

2. Что означают параметры SoftwareSerial mySerial(4, 5)?

Ответ: Это означает, что RX (приёмник) последовательного порта подключен к цифровому контакту 4, а TX (передатчик) — к цифровому контакту 5.

3. Какие параметры необходимо указывать при подключении к Wi-Fi сети?

Ответ: Необходимо указать SSID (имя Wi-Fi сети) и PASSWORD (пароль от сети).

4. Что такое HTTP-запрос GET и как он используется в данном проекте?

Ответ: HTTP-запрос GET используется для получения данных с сервера. В данном проекте он используется для отправки данных о температуре на сервер dweet.io.

5. Что такое «твой\_ключ» в коде программы?

Ответ: «твой\_ключ» — это уникальный ключ, который необходимо указать для идентификации вашего устройства на dweet.io. Вы можете выбрать любое имя, которое будет использоваться для доступа к вашим данным на сервисе.

6. Почему используется математическая формула для преобразования показаний с датчика температуры?

Ответ: Потому что датчик температуры выдаёт аналоговое значение, которое нужно преобразовать в градусы Цельсия (или другую единицу измерения температуры).

Вывод: я изучил процесс отправки данных с датчика температуры на облачный сервис dweet.io с помощью платы Arduino и модуля ESP8266. Освоил работу с HTTP-запросами и библиотекой SoftwareSerial.

### Лабораторная работа 3: Система регистрации данных

Цель: снимать показания с датчиков температуры и освещённости и записывать их в файл на карте microSD в формате .csv.

Рисунок: (Здесь должна быть схема подключения датчиков температуры, освещённости и SD-карты к Arduino).

1. Собрана схема и разработана программа, которая записывает показания датчиков температуры и освещённости в файл «log.csv» на SD-карте, разделяя данные точкой с запятой.

Листинг 3.1 – Листинг программы (Исходный код):

```
arduino
#include <SPI.h>
#include <SD.h>
#include <math.h>

#define LIGHT_PIN A3
#define TEMP_PIN A2
#define CS 8

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

  if (!SD.begin(CS)) {
    Serial.println("initialization failed!");
    return;
  }
}

void loop() {
  float r_light = 10.0
    / (1023.0 / analogRead(LIGHT_PIN) - 1.0);
  float light = 10.0 * pow(14.0 / r_light, 1.6);
  float v_temp = 1023.0 / analogRead(TEMP_PIN) - 1.0;
  float temp = 1.0 / ( -log(v_temp) / 3977.0
    + 1.0 / 295.0 ) - 273.0;
  String data = String(millis() / 1000) + ";"
    + String(light) + ";" + String(temp);
  data.replace(".", ",");
  Serial.println(data);

  File logFile = SD.open("log.csv", FILE_WRITE);
```

```
logFile.println(data);  
logFile.close();
```

```
delay(1000);  
}
```

2. Код примера изменен таким образом, чтобы в файл «log.csv» также записывались дата и время каждой записи. Для этого необходимо подключить модуль RTC (часы реального времени). (Предполагается, что модуль RTC уже подключен и настроен).

Листинг 3.2 — Листинг программы (модифицированный код):  
arduino

```
#include <SPI.h>  
#include <SD.h>  
#include <math.h>  
#include <RTCLib.h> // Библиотека для работы с RTC модулем
```

```
#define LIGHT_PIN A3  
#define TEMP_PIN A2  
#define CS 8
```

```
RTC_DS3231 rtc; // Создаем объект RTC
```

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

```
  if (!SD.begin(CS)) {  
    Serial.println("initialization failed!");  
    return;  
  }
```

```
  if (!rtc.begin()) {  
    Serial.println("Couldn't find RTC");  
    Serial.flush();  
    abort();  
  }  
}
```

```
void loop() {  
  DateTime now = rtc.now(); // Получаем текущую дату и время с RTC  
  float r_light = 10.0 / (1023.0 / analogRead(LIGHT_PIN) - 1.0);  
  float light = 10.0 * pow(14.0 / r_light, 1.6);  
  float v_temp = 1023.0 / analogRead(TEMP_PIN) - 1.0;
```

```
float temp = 1.0 / (-log(v_temp) / 3977.0 + 1.0 / 295.0) - 273.0;
```

```
String data = String(now.year(), DEC) + "/" + String(now.month(), DEC)  
+ "/" + String(now.day(), DEC) + " " +  
String(now.hour(), DEC) + ":" + String(now.minute(), DEC) +  
":" + String(now.second(), DEC) + ";" +  
String(light) + ";" + String(temp);
```

```
data.replace(".", ",");  
Serial.println(data);
```

```
File logFile = SD.open("log.csv", FILE_WRITE);  
logFile.println(data);  
logFile.close();
```

```
delay(1000);  
}
```

Ответы на вопросы:

1. Что такое SPI и для чего он используется при работе с SD-картой?

Ответ: SPI (последовательный периферийный интерфейс) — это последовательный интерфейс, используемый для связи микроконтроллера с SD-картой.

2. Для чего нужен контакт CS (Chip Select) при работе с SD-картой?

Ответ: Пин CS (Chip Select) используется для выбора конкретной SD-карты, если к микроконтроллеру подключено несколько SD-карт.

3. Что означает SD.begin(CS)?

Ответ: Функция SD.begin(CS) инициализирует работу с SD-картой, используя указанный пин CS.

4. Что такое RTC и для чего он нужен в данном проекте?

Ответ: RTC (Real Time Clock) — это модуль реального времени, который позволяет отслеживать текущую дату и время. Он нужен для записи времени каждой записи в файл «log.csv».

5. Какие существуют альтернативные способы хранения данных, полученных с датчиков, помимо SD-карты?

Ответ: Данные можно хранить в EEPROM микроконтроллера, отправлять в облачный сервис (например, ThingSpeak, Adafruit IO) или передавать на компьютер по последовательному порту.

6. Как можно организовать хранение данных в нескольких файлах, чтобы избежать переполнения одного файла «log.csv»?

Ответ: Можно создавать новые файлы каждый день или каждую неделю, используя дату и время из RTC в качестве имени файла.

Вывод: я научился записывать данные с датчиков на SD-карту в формате .csv, а также интегрировал модуль RTC для добавления информации о времени каждой записи.



## Лабораторная работа 4: Напоминальник

**Цель:** создать систему напоминаний, отправляющую электронное письмо при нажатии кнопки, используя Webhooks и электронную почту.

**Рисунок:** (Здесь должна быть схема подключения кнопки и ESP8266 к Arduino).

1. Собрана схема и разработана программа, отправляющая электронное письмо при нажатии кнопки с помощью IFTTT Webhooks. Параметры SSID и PASSWORD заменены на актуальные для вашей сети Wi-Fi. ID компонента Webhooks необходимо указать свой.

Листинг 4.1 – Листинг программы (Исходный код):

```
arduino
#include "ESP8266.h"
#include <SoftwareSerial.h>

#define SSID      "имя_твоего_Wi-Fi"
#define PASSWORD  "пароль_твоего_Wi-Fi"
#define BTN_PIN   12

SoftwareSerial mySerial(4, 5);
ESP8266 wifi(mySerial);
boolean buttonWasUp = true;
String maker_ID = "id_компонента_webhooks";

void setup(void) {
    Serial.begin(9600);
    if (wifi.joinAP(SSID, PASSWORD)) {
        Serial.println("I'm ready! Press the button.");
    } else {
        Serial.println("Wi-Fi connection error");
    }
}

void loop(void) {
    boolean buttonIsUp = digitalRead(BTN_PIN);
    if (buttonWasUp && !buttonIsUp) {
        delay(10);
        buttonIsUp = digitalRead(BTN_PIN);
        if (!buttonIsUp) {
```

```

    sendEmail();
    Serial.println("Notification has been sent");
}
}
buttonWasUp = buttonIsUp;
}

boolean sendEmail() {
    if (wifi.createTCP("maker.ifttt.com", 80)) {
        String value1 = "Hello!";
        String request = "GET /trigger/feed/with/key/"
            + maker_ID + "?value1=" + value1
            + " HTTP/1.1\r\n";
        request += "Host: maker.ifttt.com\r\n\r\n";
        wifi.send(request.c_str(), request.length());
        wifi.releaseTCP();
    } else {
        Serial.println("create tcp error");
    }
}

```

2. Код примера изменен так, чтобы вместе с email отправлялось время нажатия кнопки.

Листинг 4.2 — Листинг программы (модифицированный код):  
arduino

```

#include "ESP8266.h"
#include <SoftwareSerial.h>

#define SSID      "имя_твоего_Wi-Fi"
#define PASSWORD  "пароль_твоего_Wi-Fi"
#define BTN_PIN   12

SoftwareSerial mySerial(4, 5);
ESP8266 wifi(mySerial);
boolean buttonWasUp = true;
String maker_ID = "id_компонента_webhooks";

void setup(void) {
    Serial.begin(9600);
    pinMode(BTN_PIN, INPUT_PULLUP); // Включаем подтягивающий
    резистор для кнопки

    if (wifi.joinAP(SSID, PASSWORD)) {
        Serial.println("I'm ready! Press the button.");
    }
}

```

```

    } else {
        Serial.println("Wi-Fi connection error");
    }
}

void loop(void) {
    boolean buttonIsUp = digitalRead(BTN_PIN);
    if (buttonWasUp && !buttonIsUp) {
        delay(10); // Устранениедребезга контактов
        buttonIsUp = digitalRead(BTN_PIN);
        if (!buttonIsUp) {
            sendEmail();
            Serial.println("Notification has been sent");
        }
    }
    buttonWasUp = buttonIsUp;
}

boolean sendEmail() {
    if (wifi.createTCP("maker.ifttt.com", 80)) {
        String value1 = String(millis() / 1000); // Время нажатия кнопки в секундах
        String request = "GET /trigger/feed/with/key/" + maker_ID + "?value1="
+ value1 + " HTTP/1.1\r\n";
        request += "Host: maker.ifttt.com\r\n\r\n";
        wifi.send(request.c_str(), request.length());
        wifi.releaseTCP();
        return true; // Добавляем возврат значения для индикации успеха
    } else {
        Serial.println("create tcp error");
        return false; // Добавляем возврат значения для индикации ошибки
    }
}

```

Ответы на вопросы:

1. Что такое IFTTT и как его использовать для отправки email?

Ответ: IFTTT (If This Then That) — это веб-сервис, позволяющий автоматизировать задачи между различными онлайн-сервисами и устройствами. В данном случае он используется для

отправки электронного письма при получении HTTP-запроса от Arduino.

2. Для чего нужен подтягивающий резистор (pull-up resistor) при подключении кнопки?

Ответ: Подтягивающий резистор обеспечивает определенный уровень сигнала (высокий или низкий) на входе микроконтроллера, когда кнопка не нажата.

3. Что такоедребезг контактов и как его устранить?

Ответ: Дребезг контактов — это кратковременные колебания сигнала при нажатии или отпускании кнопки. Он устраняется с помощью небольшой задержки (delay) в коде.

4. Как передать несколько параметров (value1, value2, value3) в IFTTT Webhooks?

Ответ: Параметры передаются в URL запроса, добавляя &value2=значение2&value3=значение3.

5. Какие существуют альтернативные сервисы для автоматизации задач, кроме IFTTT?

Ответ: Zapier, Microsoft Power Automate (бывший Microsoft Flow).

6. Как изменить текст письма, отправляемого через IFTTT?

Ответ: Текст письма настраивается в самом рецепте IFTTT при создании апплета.

Вывод: я научился создавать систему уведомлений, отправляющую электронные письма при нажатии кнопки, используя Arduino, ESP8266 и сервис IFTTT Webhooks.

## Лабораторная работа 5: Браузерный Dendy

Цель: создать сервер на Arduino с браузерной игрой, считывая файл игры с SD-карты и передавая его через модуль Wi-Fi для отображения в браузере.

Рисунок: (Здесь должна быть схема подключения SD-карты и ESP8266 к Arduino).

1. Схема собрана, скетчи загружены на Arduino и ESP8266. Файл `race.htm` (содержимое игры) помещен на SD-карту. Параметры SSID и PASSWORD в скетче для ESP8266 заменены на актуальные для вашей сети Wi-Fi.

*Код для Arduino:*

Листинг 5.1 Листинг программы (Arduino  
5\_dendy\_arduino.ino):

```
arduino
#include <SPI.h>
#include <SD.h>

void setup() {
  Serial.begin(115200);
  SD.begin(8);
}

void loop() {
  if (Serial.find("race.htm")) {
    File myFile = SD.open("race.htm");
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    myFile.close();
  }
}
```

*Код для Wi-Fi модуля:*

Листинг 5.2 Листинг программы (ESP8266 - 5\_dendy\_wifi.ino):

```
arduino
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
```

```
String page = "";
ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/html", page);
}

void setup(void) {
  Serial.begin(115200);
  WiFi.begin("имя_твоего_Wi-Fi", "пароль_твоего_Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.println();
  Serial.println(WiFi.localIP());

  server.on("/race.htm", handleRoot);
  server.begin();

  Serial.println("race.htm");
}

void loop(void) {
  if (Serial.available()) {
    page += (char)Serial.read();
  }
  server.handleClient();
}
```

2. Код изменен таким образом, чтобы вместо файла `race.htm` отображался другой HTML-файл с более сложной графикой и интерактивностью. Для этого на SD-карту помещен новый файл `game.htm`.

*Код для Arduino:*

Листинг 5.3 - Листинг программы (Arduino - 5\_dendy\_arduino\_modified.ino):

```
arduino
#include <SPI.h>
#include <SD.h>

void setup() {
  Serial.begin(115200);
```

```

SD.begin(8);
}

void loop() {
  if (Serial.find("game.htm")) { // Изменено с race.htm на game.htm
    File myFile = SD.open("game.htm");
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    myFile.close();
  }
}

```

Код для Wi-Fi модуля:

Листинг 5.4 - Листинг программы (ESP8266 —  
5\_dendy\_wifi\_modified.ino):

```

arduino
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

String page = "";
ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/html", page);
}

void setup(void) {
  Serial.begin(115200);
  WiFi.begin("имя_твоего_Wi-Fi", "пароль_твоего_Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.println();
  Serial.println(WiFi.localIP());

  server.on("/game.htm", handleRoot); // Изменено с /race.htm на
/game.htm
  server.begin();

  Serial.println("game.htm");
}

```



```
void loop(void) {  
  if (Serial.available()) {  
    page += (char)Serial.read();  
  }  
  server.handleClient();  
}
```

Ответы на вопросы:

1. Зачем в этом эксперименте нужны два микроконтроллера (Arduino и ESP8266)?

Ответ: Arduino используется для чтения файла с SD-карты, а ESP8266 — для организации Wi-Fi-соединения и создания веб-сервера, который будет отдавать содержимое файла. Разделение позволяет Arduino сосредоточиться на чтении с SD-карты, а ESP8266 — на сетевых функциях..

2. Что такое `server.on("/race.htm", handleRoot)` и для чего это нужно?

Ответ: `server.on("/race.htm", handleRoot)` регистрирует функцию `handleRoot` для обработки HTTP-запросов к адресу `/race.htm`. Когда браузер запрашивает этот адрес, вызывается функция `handleRoot`, которая отправляет содержимое страницы .

3. Почему используется скорость 115200 для Serial-порта?

Ответ: . Более высокая скорость последовательного порта (115200) позволяет быстрее передавать данные между Arduino и ESP8266, что особенно важно при передаче больших файлов (например, HTML-страницы с игрой)

4. Как убедиться, что ESP8266 успешно подключился к сети Wi-Fi?

Ответ: В Serial Monitor ESP8266 должен вывести свой IP-адрес.

5. Что делать, если браузер не отображает страницу с игрой, даже если IP-адрес ESP8266 получен?

Ответ: Убедиться, что файл `race.htm` присутствует на SD-карте.

а. Убедитесь, что Arduino и ESP8266 правильно подключены друг к другу.

б. Убедитесь, что Arduino правильно считывает файл с SD-карты и передаёт его в последовательный порт.

- c. Убедитесь, что в скетче ESP8266 указан правильный путь к файлу (/race.htm).
- d. Попробовать перезагрузить ESP8266.

Вывод: я изучил процесс создания простого веб-сервера на базе Arduino и ESP8266, который может отображать контент, хранящийся на SD-карте. Я понял, как организовать взаимодействие между двумя микроконтроллерами для решения задачи.

## Лабораторная работа 6. Умный дом

Цель: Собрать выключатель света, управляемый из веб-интерфейса, используя Wi-Fi модуль.

Рисунок: (Здесь должна быть схема подключения реле и ESP8266 к Arduino).

1. Собрана схема и разработана программа, которая позволяет управлять реле (выключателем света) через веб-интерфейс. При помощи команды "On" реле включается, а при команде "Off" — выключается. Также реализована возможность отправки HTML-страницы "home.htm" через Serial.

*Код для Arduino:*

Листинг 6.1. Листинг программы для Arduino  
(6\_smarthome\_arduino.ino):

```
arduino
#include <SPI.h>
#include <SD.h>

#define RELAY A4

File myFile;

void setup() {
  Serial.begin(115200);
  SD.begin(8);
  pinMode(RELAY, OUTPUT);
}

void loop() {

  String command = Serial.readStringUntil('\n');
  command.trim();

  if (command == "On") {
    digitalWrite(RELAY, HIGH);
  }
  if (command == "Off") {
    digitalWrite(RELAY, LOW);
  }
  if (command == "home.htm") {
    myFile = SD.open("home.htm");
```

```

while (myFile.available()) {
  Serial.write(myFile.read());
}
myFile.close();
}
}

```

2. Код для Wi-Fi модуля позволяет управлять реле через HTTP-запросы. Команды "turnOn" и "turnOff" включают и выключают реле соответственно.

*Код для Wi-Fi модуля:*

Листинг 6.2. Листинг программы для Wi-Fi модуля (6\_smarthome\_wifi.ino):

```

arduino
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

String page = "";
ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/html", page);
}

void handleOn() {
  Serial.println("On");
  server.send(200, "text/plain", "On");
}

void handleOff() {
  Serial.println("Off");
  server.send(200, "text/plain", "Off");
}

void setup(void) {
  Serial.begin(115200);
  WiFi.begin("имя_твоего_Wi-Fi", "пароль_твоего_Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.println();
}

```

```
Serial.println(WiFi.localIP());
```

```
server.on("/home.htm", handleRoot);
```

```
server.on("/turnOn", handleOn);
```

```
server.on("/turnOff", handleOff);
```

```
server.begin();
```

```
Serial.println("home.htm");
```

```
}
```

```
void loop() {
```

```
  if (Serial.available()) {
```

```
    page += (char)Serial.read();
```

```
  }
```

```
  server.handleClient();
```

```
}
```

Ответы на вопросы:

1. Как работает команда `digitalWrite(RELAY, HIGH)`?

Ответ: Эта команда включает реле, подключенное к пину RELAY, подавая на него высокий уровень сигнала.

2. Что происходит при получении команды `command.trim()`?

Ответ: Метод `trim()` удаляет пробелы в начале и конце строки, что помогает избежать ошибок при сравнении команд.

3. Как осуществляется связь между Arduino и Wi-Fi модулем?

Ответ: Связь осуществляется через последовательный порт (Serial), где Arduino отправляет команды на Wi-Fi модуль.

4. Для чего используется функция `server.on()` в коде Wi-Fi модуля?

Ответ: Функция `server.on()` регистрирует обработчики для определённых URL-адресов, которые будут вызываться при получении соответствующих HTTP-запросов.

5. Как можно расширить функциональность данного проекта?

Ответ: Можно добавить управление несколькими реле, интегрировать датчики движения или освещенности для автоматического управления освещением.

**Вывод:** Я изучил процесс создания системы умного дома с управлением освещением через веб-интерфейс с использованием Arduino и Wi-Fi модуля.

## Лабораторная работа 7. Telegram-бот

**Цель:** Управлять устройствами прямо из мессенджера Telegram с помощью собственного бота.

**Рисунок:** (Здесь должна быть схема подключения реле и ESP8266 к Arduino).

1. Собрана схема и разработана программа для Arduino, которая позволяет управлять реле через команды, получаемые от Wi-Fi модуля.

*Код для Arduino:*

Листинг 7.1. Листинг программы для Arduino  
(7\_telegram\_arduino.ino):

```
arduino
#include <SoftwareSerial.h>

#define REL_PIN 12

SoftwareSerial esp(4, 5);

void setup() {
    esp.begin(9600);
    pinMode(REL_PIN, OUTPUT);
}

void loop() {
    String command = esp.readStringUntil('\n');
    command.trim();

    if (command == "turnOn") {
        digitalWrite(REL_PIN, HIGH);
    }

    if (command == "turnOff") {
        digitalWrite(REL_PIN, LOW);
    }
}
```

*Код для Wi-Fi*

Код для Wi-Fi модуля позволяет боту получать команды от пользователей Telegram и управлять реле.

Листинг 7.2 — Листинг программы для Wi-Fi модуля  
(7\_telegram\_wifi.ino):

**7\_telegram\_wifi.ino**

**#include** <ESP8266WiFi.h>

**#include** <WiFiClientSecure.h>

**#include** <UniversalTelegramBot.h>

char ssid[] = "имя\_твоего wi-fi";

char password[] = "пароль\_wi-fi";

**#define** BOTtoken "твой\_токен"

WiFiClientSecure client;

UniversalTelegramBot bot(BOTtoken, client);

**String** keyboardJson = "[[\"/ledon\", \"/ledoff\"]]";

**void** setup() {

**Serial**.begin(9600);

  client.setInsecure();

**WiFi**.begin(ssid, password);

**while** (**WiFi**.status() != WL\_CONNECTED) {

    delay(500);

  }

}

**void** loop() {

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

  handleNewMessages(numNewMessages);

}

**void** handleNewMessages(**int** numNewMessages) {

**for** (**int** i = 0; i < numNewMessages; i++) {

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

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

**if** (text == "/ledon") {

**Serial**.println("turnOn");

      bot.sendMessage(chat\_id, "Relay is ON", "");

    }



```

if (text == "/ledoff") {
    Serial.println("turnOff");
    bot.sendMessage(chat_id, "Relay is OFF", "");
}

if (text == "/start") {
    bot.sendMessageWithReplyKeyboard(chat_id,
        "Choose from one of the following options",
        "", keyboardJson, true);
}
}
}

```

Ответы на вопросы:

1. Как работает библиотека UniversalTelegramBot?

Ответ: Библиотека UniversalTelegramBot предоставляет интерфейс для взаимодействия с API Telegram и позволяет отправлять и получать сообщения от бота.

2. Почему используется client.setInsecure()?

Ответ: Эта команда отключает проверку сертификата SSL при подключении к Telegram API, что упрощает настройку на этапе разработки.

3. Как бот обрабатывает новые сообщения?

Ответ: Бот использует метод getUpdates(), чтобы получать новые сообщения от пользователей и затем обрабатывает их в функции handleNewMessages().

4. Что происходит при получении команды /ledon?

Ответ: При получении этой команды бот отправляет сообщение о том, что реле включено, и передает команду на Arduino для включения реле.

5. Как можно расширить функциональность бота?

Ответ: Можно добавить дополнительные команды для управления другими устройствами или интегрировать функции мониторинга состояния датчиков.

Вывод: Я научился создавать Telegram-бота для управления устройствами с помощью Arduino и Wi-Fi модуля.

## Лабораторная работа 8. Блик

**Цель:** Познакомиться с сервисом Blynk и управлять RGB-светодиодом со смартфона.

Схема подключения RGB-светодиода к Arduino на макетной плате

Компоненты:

Arduino UNO;

RGB-светодиод с общим катодом;

3 резистора 220 Ом;

Макетная плата;

Соединительные провода.

Визуализация схемы:

Описание подключения:

Общий катод RGB-светодиода подключен к шине GND макетной платы, которая соединена с GND Arduino.

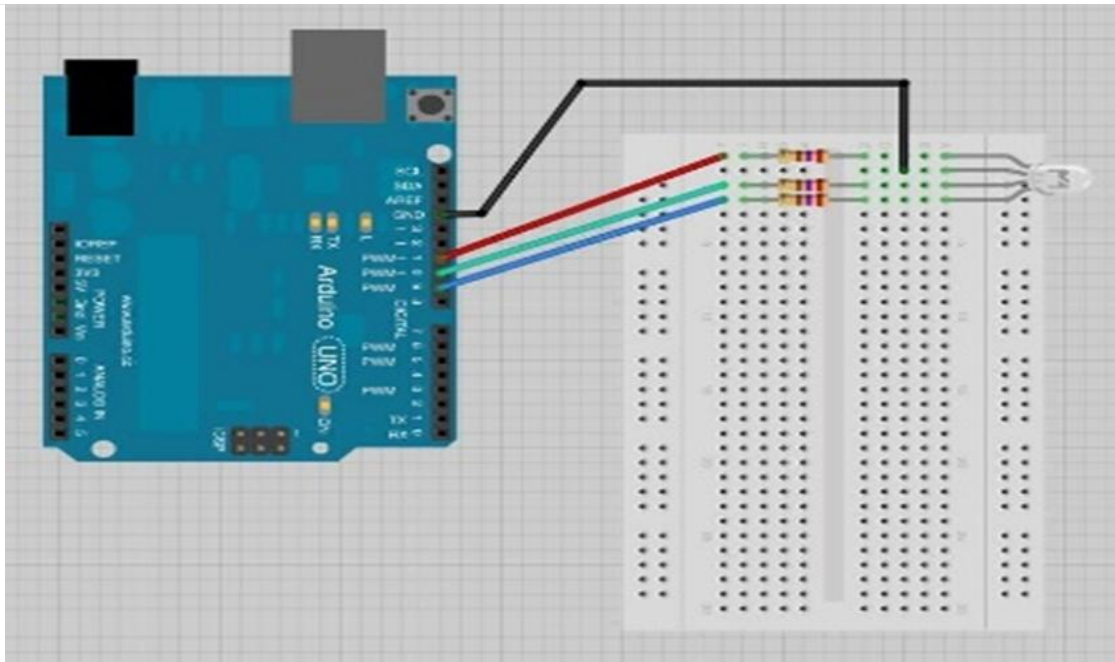
Красный контакт подключен через резистор (220 Ом) к пину 3 Arduino.

Зеленый контакт подключен через резистор (220 Ом) к пину 5 Arduino.

Синий контакт подключен через резистор (220 Ом) к пину 6 Arduino.

Примечание: Макетная плата используется для удобства и организации

**Рисунок:** (Здесь должна быть схема подключения RGB-светодиода к Arduino).



1. Собрана схема и разработана программа для управления RGB-светодиодом через последовательный порт на основе команд от Wi-Fi модуля.

*Код для Arduino:*

Листинг 8.1 Листинг программы для Arduino

(8\_blynk\_arduino.ino):

arduino

```
int ledR = 3;
```

```
int ledG = 5;
```

```
int ledB = 6;
```

```
void setup() {
```

```
  Serial.begin(115200);
```

```
}
```

```
void loop() {
```

```
  String command = Serial.readStringUntil('\n');
```

```
  if (command.length() > 0) {
```

```
    int led;
```

```
    if (command[0] == 'r') {
```

```
      led = ledR;
```

```
    }
```

```
    if (command[0] == 'g') {
```

```
      led = ledG;
```

```
    }
```

```
    if (command[0] == 'b') {
```

```

    led = ledB;
}
String brightness = command.substring(1);
analogWrite(led, brightness.toInt());
}
}

```

2. Код для модуля Wi-Fi позволяет одключаться к Blynk-сервису и управлять цветами RGB-светодиода через мобильное приложение.

*Код для Arduino:*

Листинг 8.2 Листинг программы для Wi-Fi модуля (8\_blynk\_wifi.ino):

```

arduino
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "токен_blynk";
char ssid[] = "имя_твоего wi-fi";
char pass[] = "пароль_wi-fi";

void setup() {
    Serial.begin(115200);
    Blynk.begin(auth, ssid, pass);
}

void loop() {
    Blynk.run();
}

BLYNK_WRITE(V1) { // Обработчик изменения значения виртуального пина V1
    int redValue = param[0].asInt(); // Получаем значение красного цвета
    int greenValue = param[1].asInt(); // Получаем значение зеленого цвета
    int blueValue = param[2].asInt(); // Получаем значение синего цвета

    Serial.print("r");
    Serial.println(redValue); // Отправляем значение красного цвета
    Serial.print("g");
    Serial.println(greenValue); // Отправляем значение зеленого цвета
    Serial.print("b");

```

```
Serial.println(blueValue); // Отправляем значение синего цвета  
}
```

Контрольные вопросы:

1. Что такое Blynk и как он используется в этом проекте?

Ответ: Blynk — это платформа для создания мобильных приложений IoT с простым интерфейсом управления устройствами через Интернет.

2. Как работает функция BLYNK\_WRITE(V1)?

Ответ: Функция BLYNK\_WRITE(V1) работает следующим образом: она автоматически вызывается, когда сервер сообщает устройству, что значение виртуального пина изменилось. Обычно это происходит, когда нажимают кнопку виджета в приложении.

Сервер отправляет текущее значение виртуального пина на оборудование в качестве параметра. Это значение можно получить внутри функции BLYNK\_WRITE(vPin).