2023.08.03

Использование DataSnap-технологии на примере разработки комплекса взаимодействующих приложений (ОС Windows, ОС Android и ОС Arduino) в среде Delphi 10.2 Tokyo. Иллюстрирующий пример

В этом документе используются материалы из ранее опубликованной статьи «Использование DataSnap-технологии на примере разработки комплекса взаимодействующих приложений (ОС Windows и ОС Android) в среде Delphi 10.2 Tokyo. Последовательность действий» [2].

Оглавление

Термины и сокращения

В таблице ниже приведен перечень используемых в документе терминов и сокращений.

Таблица 1 − Перечень используемых в документе терминов и сокращений

Сокращение	Обозначение
1	2
Android	Операционная система Android
Arduino	Инструмент (программируемый электронный модуль) для создания различных электронных устройств (систем автоматики и робототехники), ориентированная на непрофессиональных пользователей. Устройства на Arduino имеют возможность принимать сигналы от различных датчиков и управлять различными исполнительными устройствами. Arduino может работать автономно или взаимодействовать с компьютером
Delphi	Embarcadero Delphi 10.2 Tokyo
ESP-12F WeMos D1 WiFi	Программируемый электронный модуль (из семейства Arduino), предназначенный для управления различными электронными устройствами как в автономном режиме, так и в режиме информационного взаимодействия с внешними программами в сети Wi-Fi и/или по последовательному COM-порту (см., также, «Интернет вещей»)
ESP	ESP-12F WeMos D1 WiFi
GUI	Graphical user interface (графический интерфейс пользователя)
Windows	Операционная система MS Windows
Интернет вещей	Система взаимосвязанных вычислительных устройств, которые могут собирать и передавать данные по беспроводной сети без участия человека
Клиент	DataSnap-клиент
ОС	Операционная система
Сервер	DataSnap-сервер
Скетч	Программа, сформированная для модулей семейства Arduino
Телефон	Android-устройство (мобильный телефон, работающий под управлением ОС Android)
Техносфера	Среда функционирования программно-аппаратных средств, взаимодействующих по определенным правилам

1 Введение

DataSnap – технология создания многозвенных приложений.

Материалов об этой технологии опубликовано очень много (в том числе и в Интернет). Повторяться не стОит. Следует только отметить, что со временем ее актуальность не теряется.

Очередное огромное спасибо авторам статьи [1] за те идеи, которые возникли в процессе ее проработки.

1.1 Цель этого документа

В последнее время все чаще на слуху такие термины, как «умный дом», «техносфера», «искусственный интеллект», «нейронные сети», «миварные технологии» и т.д.

Примечание – по личному мнению Автора термин «техносфера» включает в себя все, перечисленные выше, термины (как инструментарий для формирования «техносферы»).

Предлагаются различные прикладные решения (например, от Яндекс, Сбер, НИИ «МИВАР» и т.д.).

Существуют соответствующие API (как правило, платные и ориентированные на ВЕБ-разработку).

Т.е., эта сфера бурно развивается. И следующие поколения уже будут жить в соответствующем программно-аппаратном окружении.

Но все, ныне предлагаемые, решения носят (в контексте «техносферы»), пока еще фрагментарный характер.

Т.е., на бытовой уровень «обычной квартиры» системно они «не пришли».

И даже не заявились.

В настоящее время, для рядового обывателя (отягощенного профессией «программист», страдающего синдромом «творчество» и замотивированного на автоматизацию своей личной деятельности как дома, так и вне), существует достаточно программно-аппаратных средств (кубиков), из которых вполне можно формировать что-то, вроде «Домашняя техносфера своими руками» (даже, например, в «обучительно-игровой» форме с участием детей/внуков).

Delphi (все ее «крайние» версии) предоставляет возможность разработки приложений для различных платформ (включая Windows и Android).

Кроме этого, на рынке (Aliexpress) предлагается достаточно много дешевых программируемых электронных модулей семейства Arduino, а также множество типов дешевых датчиков, индикаторов и исполнительных механизмов (которые ориентированы на подключение к Arduino).

Ряд модулей семейства Arduino имеют встроенную, аппаратную, поддержку для организации информационного обмена с внешними программами (включая и взаимодействие через wi-fi).

Наличие «всего этого» предоставляет возможность заинтересованному программисту предпринять практические шаги для «автоматизации» определенных, важных для него, процессов (включая и игровые с детьми/внуками).

Цель этого документа: предоставить минимальную практическую информацию заинтересованным (или, которые потенциально могут стать заинтересованными) Читателям на простом примере использования DataSnap для организации информационного взаимодействия (в сети wi-fi) различных устройств по следующей схеме:

Android (телефон) ⇔ **Windows** (компьютер) ⇔ **ESP** (Arduino)

Автор использует этот подход в рамках игрового проекта «Домашняя техносфера своими руками». См. здесь:
https://roamer55.ru/main_programming/arduino/arduino_technosphere_000/

1.2 Соглашения

1. «Сколько геологов, столько и мнений» или «на вкус и цвет товарищей нет».
2. Автор не ставит перед собой цель – удивить этот мир «красотой» и оптимальностью исходного кода, а также применяемой методики.
3. Автор в этом документе лишь излагает свой собственный подход при решении (программировании) подобных задач на конкретном, простом примере (с целью – помочь тем, кто в этом нуждается).
4. Предполагается, что читатель знает Delphi и умеет программировать на достаточном, прикладном уровне.
5. Прилагаемые к этому документу исходники программного комплекса хотя и рабочие, но носят чисто иллюстративный характер.
6. В данном документе используются материалы из ранее опубликованных статей [1] и [2].

2 «Железо»

В рассматриваемом примере используется модуль **`ESP-12F WeMos D1 WiFi`**.

Это – достаточно простой, функциональный, надежный и дешевый модуль для того, чтобы использовать его в своих проектах.

На рисунке ниже приведены внешний вид и габариты модуля.

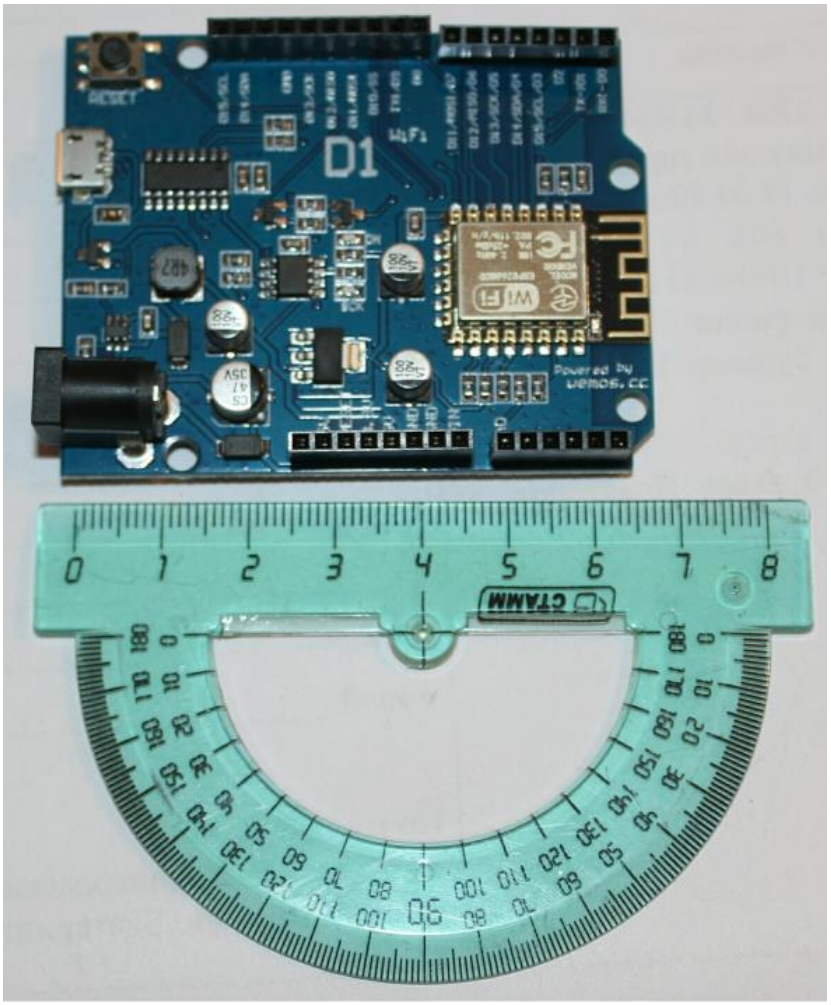


Рисунок 1 – Модуль `ESP-12F WeMos D1 WiFi`

Этот модуль можно свободно найти на **Aliexpress** (https://aliexpress.ru/) по цене до **300 рублей** (по состоянию на 02.08.2023) и с бесплатной доставкой в РФ.

Следует отметить, что есть существенно более компактные и дешевые модули этого семейства (по размерам – в пределах половины спичечного коробка).

Например, `ESP12 WeMos D1 Mini WiFi`, но там требуется пайка (проводов к контактам).

Детальное описание всех этих модулей можно легко найти в интернете.

Поэтому, нет смысла здесь дублировать эту информацию.

В данном документе приведено только то, что является необходимым для рассматриваемого примера.

На рисунке ниже показаны разъемы аппаратного интерфейса, которые могут быть использованы в процессе разработки и эксплуатации соответствующих программно-аппаратных систем.

Рисунок 2 – Аппаратный интерфейс ESP, предназначенный для подключения к нему источника внешнего питания и USB-кабеля для его подключения к компьютеру

Выноской 1 на рисунке выше обозначен стандартный USB-разъем для подключения ESP к компьютеру с использованием стандартного USB-кабеля (см. рисунок 3).

Выноской 2 на рисунке выше обозначен разъем для подключения внешнего источника питания постоянного напряжения 9 вольт – к ESP.

Следует отметить, что USB-разъем (выноска-1) может использоваться:

- для подключения внешнего источника питания;
- для подключения к компьютеру с целью:
  - использование компьютера, как внешнего источника питания;
  - «прошивка» ESP (запись программы, сформированной Пользователем);
  - информационный обмен между ESP и внешней программой.

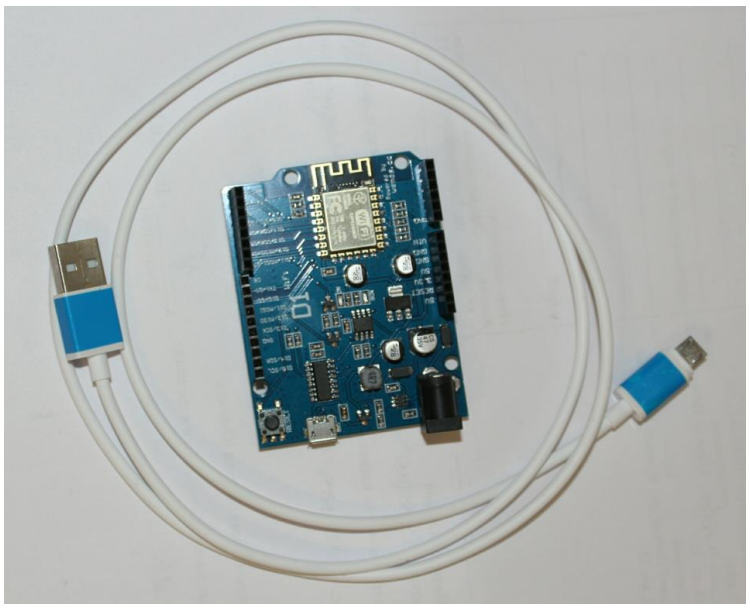


Рисунок 3 – Для подключения ESP к внешним устройствам используется стандартный USB-кабель

На рисунке 4 показано подключение ESP к розетке ~220 вольт через стандартный USB-адаптер (для телефона) с использованием стандартного USB-кабеля.

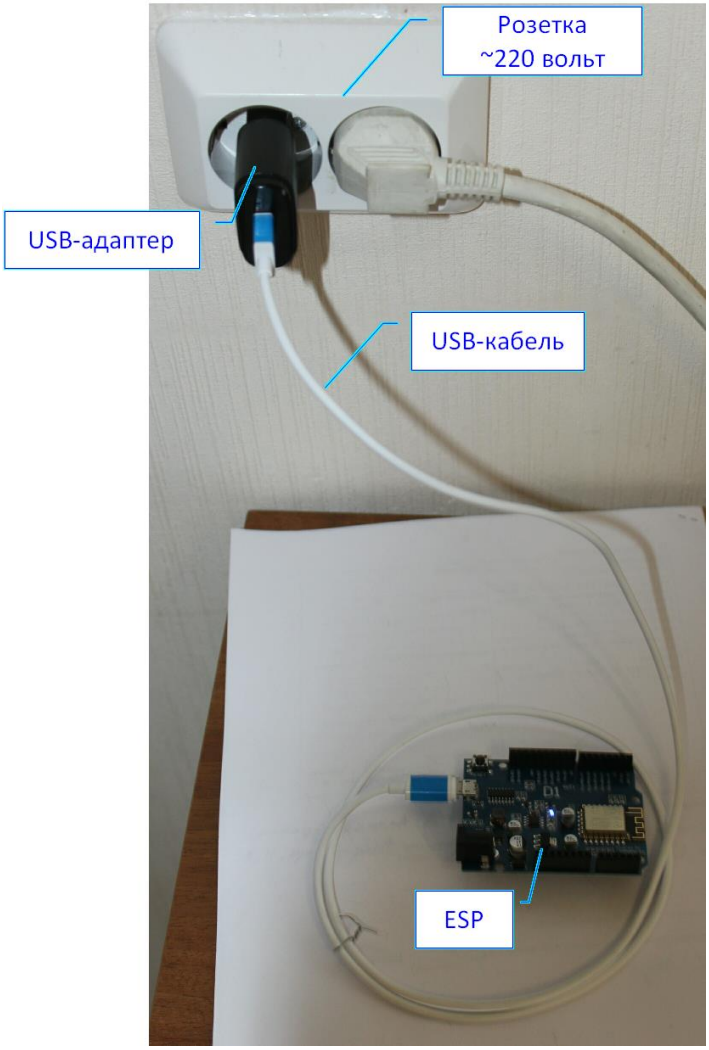


Рисунок 4 – Подключение ESP к розетке ~220 вольт

3 Постановка задачи

Задача (иллюстрирующий пример):

1. Разработать простой комплекс из 3-х программ, позволяющих организовать информационный обмен по следующей схеме:

   `Android` (телефон) ⇔ **`Windows`** (компьютер) ⇔ **`ESP`** (Arduino)
2. В качестве управляемого объекта должен быть использован встроенный светодиод ESP (см. рисунок 5).
3. Цель проекта: обеспечить включение/выключение встроенного светодиода ESP с помощью Телефона (Android) и/или DataSnap-сервера (Windows).
4. В качестве средств, предназначенных для управления встроенным светодиодом ESP, сформировать соответствующие программы (см. пункт 1).
5. Программу для Windows реализовать, как DataSnap-сервер.
6. Программу для Android реализовать, как DataSnap-клиент.
7. Скетч (для ESP) должен быть реализован с учетом режима функционирования ESP: WebServer.
8. Информационное взаимодействие программ (в комплексе) должно производиться с использованием Wi-Fi.
9. Для информационного обмена между DataSnap-сервером (Windows) и ESP должен быть использован HTTP-протокол.
10. При реализации информационного обмена по HTTP-протоколу использовать компонент `TNetHTTPClient` из палитры компонентов Delphi.

Рисунок 5 – Встроенный светодиод (LED_BUILTIN) ESP

4 Решение задачи

4.1 Требования к разрабатываемому комплексу программ

Исходя из поставленной задачи, можно принять, что:

1. DataSnap-сервер (Windows) должен выполнять следующие функции:
   1.1 «Диспетчер» для поддержки информационного обмена между Телефоном и ESP;
   1.2 Автономный пульт управления состоянием встроенного светодиода ESP;
2. На приложение, функционирующее в среде ОС Android (DataSnap-клиент), возлагается только роль «пульта» управления состоянием встроенного светодиода ESP.
3. ESP должен функционировать в режиме WebServer (пассивный режим) и лишь только реагировать (отрабатывать) на пришедшие ему команды.
4. Необходимо реализовать функционал следующих команд:
   4.1 Получить подтверждение, что ESP работоспособен и доступен;
   4.2 Включить встроенный светодиод ESP;
   4.3 Выключить встроенный светодиод ESP;
   4.4 Получить текущее состояние встроенного светодиода ESP;
5. Предусмотреть вариант, когда Пользователь забыл (или не знал) список доступных команд и/или их наименования. В этом случае, на ESP должен быть реализован функционал, возвращающий пользователю соответствующую информацию.

4.2 Этапы разработки комплекса программ

Разработка комплекса программ будет производиться в следующем порядке:

1. Установка и настройка программы Arduino IDE (при необходимости).
2. Формирование и «заливка» соответствующего скетча на ESP.
3. Тестирование функционала ESP с помощью любого ВЕБ-браузера (например, Google Chrome).
4. Разработка DataSnap-сервера под Windows (с соответствующим функционалом).
5. Тестирование функционала ESP с помощью DataSnap-сервера.
6. Разработка DataSnap-клиента под Android (с соответствующим функционалом).
7. Тестирование функционала ESP с помощью DataSnap-клиента.

4.2.1 Этап-1. Установка и настройка программы Arduino IDE (при необходимости)

Необходимо выполнить следующие операции (действия):

1. Скачать и установить (если на компьютере еще нет) программу Arduino IDE для программирования и прошивки Arduino.
   Это программа – в свободном доступе. Найти в интернете ссылки для ее скачивания – не трудно.
   Например:
   https://amperka.ru/page/arduino-ide ;
   https://arduino.ru/Arduino_environment ;
   https://arduino-ide.com/ ;
   https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE ;
   и т.д.…

2. Настроить программу Arduino IDE для работы с `ESP-12F WeMos D1 WiFi`.
   В интернете об этом достаточно много информации.
   Например:
   https://wiki.iarduino.ru/page/wemos_start/;
   https://portal-pk.ru/news/140-chto-takoe-nodemcu-programmiruem-v-srede-arduino-ide.html;
   https://arduinomaster.ru/datchiki-arduino/esp8266-wemos-d1-mini-raspinovka/#__IDE__WeMos;
   https://istarik.ru/blog/esp8266/110.html;
   https://arduino-esp8266.readthedocs.io/en/2.7.4_a/;
   https://www.drive2.ru/b/473267738819690589/;
   https://arduino-kit.ru/product/kontroller-s-wi-fi-wemos-d1-r2-na-esp8266-esp-12e;
   и т.д.
   В конечном итоге, необходимо выбрать плату (модуль):
   “**`WeMos D1 R1`**” (см. рисунок ниже).

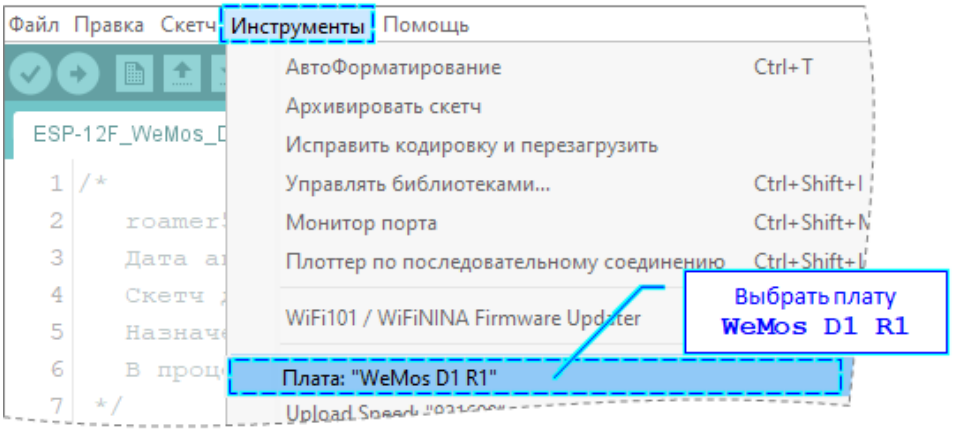


Рисунок 6 – Настройка Arduino IDE. Выбор платы `WeMos D1 R1`

3. Закрыть программу Arduino IDE.

4.2.2 Этап-2. Формирование и «заливка» соответствующего скетча на ESP

Важно! Предполагается, что ESP уже есть в наличии.

1. Открыть программу Arduino IDE (см. рисунок ниже);

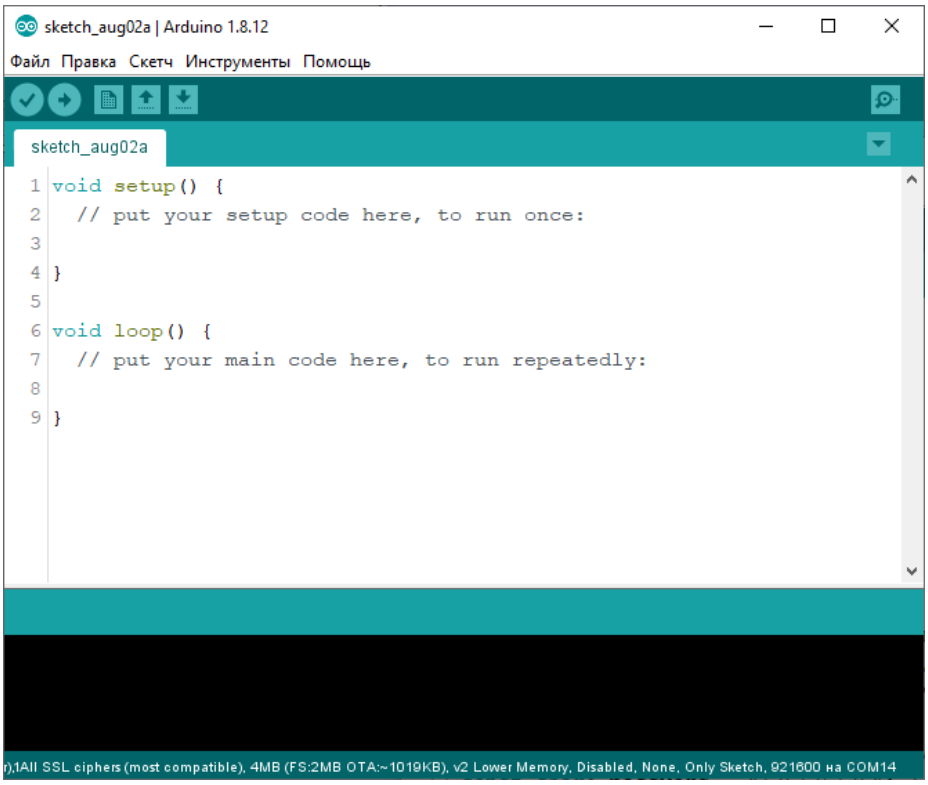
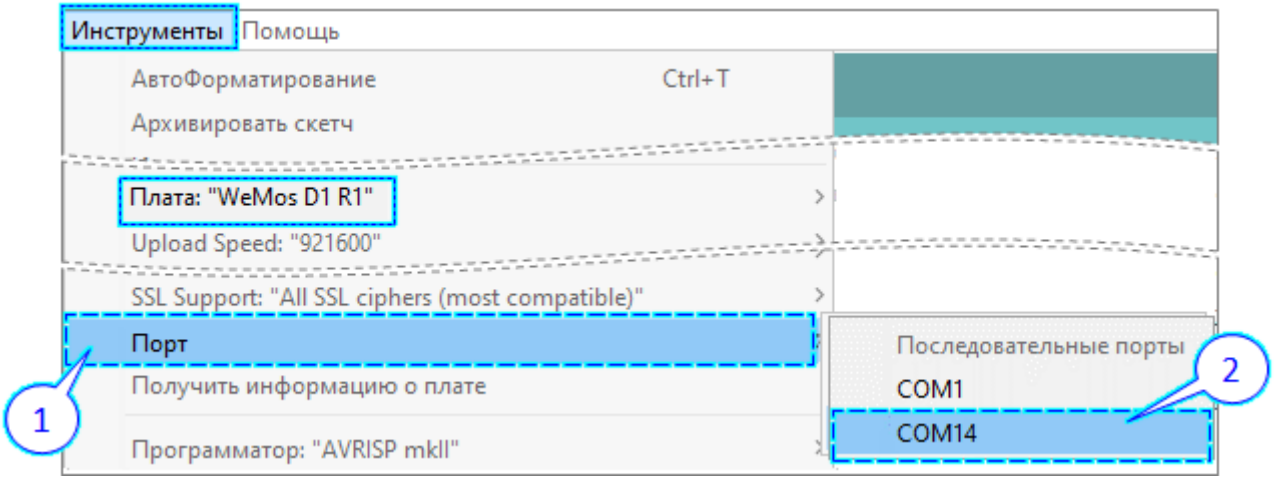


Рисунок 7 – Окно программы Arduino IDE

2. Подключить модуль `ESP-12F WeMos D1 WiFi` через USB-разъем (см. рисунок 2) к компьютеру.

3. Далее, необходимо для Arduino IDE указать COM-порт (на который «сел» ESP).
   Для этого, следует выбрать пункт главного меню программы:
   «`Инструмент`» -> «`Порт`» (см. выноску-1, на рисунке ниже).
   А затем, выбрать соответствующий COM-порт из списка доступных (см. выноску-2).



В результате идентификатор порта отобразится в меню программы:

Порт: "COM14"

4. «Залить» скетч в ESP.

Примечание – если Arduino IDE корректно установлена и настроена, то процесс «заливки скетча» («прошивки» модуля) не представляет никаких сложностей.

Для этого необходимо открыть файл со скетчем (воспользовавшись пунктом меню «Файл» -> «Открыть», см. рисунок ниже) или вставить текст скетча в окно Arduino IDE через буфер обмена (Clipboard).

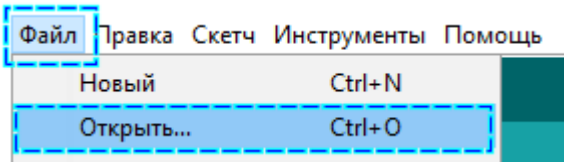


Рисунок 8 – Пункт меню «Файл» -> «Открыть»

Текст скетча:

```
/*
   roamer55.ru.
   Дата актуальности: 02.08.2023.
   Скетч для "ESP-12F WeMos D1 WiFi" (далее - ESP).
   Назначение - иллюстрирующий пример обеспечения информационного обмена
ESP с внешней программой через Wi-Fi;
  В процессе информационного обмена (по соответствующим командам)
включается и выключается встроенный светодиод ESP.
 */

#include <ESP8266WebServer.h>

//.................................
//Идентификация wi-fi сети и этого ESP
const char* ssid = "ИмяТвоейСети";    // Указываем имя существующей точки
доступа (wi-fi сеть)
const char* password = "ПарольКСети"; // Указываем пароль существующей
точки доступа (wi-fi сеть)

const String id_this = "001"; // Уникальный ID этого ESP (если
предполагается работа с несколькими ESP, то для каждого - уникальный в
рамках конкретной сети)
//.................................

//.................................
//"Настроечные" переменные
bool ItsMode_Debug = true;       //Режим отладки. Значение true актуально
только, если этот ESP подсоединен к ПК через USB (COM-порт)
//.................................

ESP8266WebServer server(80);

void setup(void)
{
  // ----------------------------------------
  //Инициализация устройства
  //Установка режима встроенного светодиода: OutPut
  pinMode(LED_BUILTIN, OUTPUT);
  //"Выключить" встроенный светодиод
  digitalWrite(LED_BUILTIN, HIGH);
```

```
  //Инициализация последовательного соединения (COM-порт через USB) со
скоростью 9600
  Serial.begin(9600);

  //Инициализация Wi-Fi модуля
  WiFi.mode(WIFI_STA); // Установка Wi-Fi модуля в режим клиента (STA)
  WiFi.begin(ssid, password); // Подключение к сети
  // Ожидание подключения к сети
  while (WiFi.status() != WL_CONNECTED) { delay(500); }

  //...................
  //Если включен режим отладки, то по последовательному соединению
  //отправляется соотв. информ. блок
  if (ItsMode_Debug==true)
  {
     Serial.println("");
     Serial.println("-----------------------");
     Serial.print("WI-FI network: ");
     Serial.println(ssid);
     Serial.print("IP address: ");
     Serial.println(WiFi.localIP());
     Serial.print("My ID: ");
     Serial.println(id_this);
     Serial.println("-----------------------");
  }
  //...................

  // -----------------------------------------
  // Обработчики HTTP-запросов
  //Отработка запроса на получение общей информации
  server.on("/", handleRoot);
  //Отработка запроса на получение ID этого ESP
  server.on("/who_is_it", []() { who_is_it(); });
  server.on("/led_on", []() { led_on(); });
  server.on("/led_off", []() { led_off(); });
  server.on("/led_state_get", []() { led_state_get(); });
  // -----------------------------------------

  // Вызывается, когда обработчик не назначен
  server.onNotFound(handleNotFound);

  // Запуск сервера
  server.begin();
}

void loop(void)
{
  server.handleClient();
}

void handleRoot()
{
  // Обработчик запроса клиента по корневому адресу - ответ сервера
(возвращается список команд)
  server.send(200,   "text/plain",   "ID="+id_this+".\n"+"Commands:   \n
who_is_it \n led_on \n led_off \n led_state_get");
}


//=========================================
void handleNotFound()
{ // Ошибка 404. Некорректный URL (несуществующая команда)
  String message = "File Not Found\n\n";
  message += "URI: ";
```

```
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++)
  {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", "ID="+id_this+":\n"+"Command:"+message +
" not found");
}
//=========================================

//=========================================
void who_is_it()
{ //Запрос на получение ID этого ESP
    //Пример:   http://192.168.0.103/who_is_it
    server.send(200, "text/plain", "ID="+id_this);
}
//=========================================

//=========================================

void led_state_get()
{ //Запрос на возврат состояния встроенного светодиода
    //Пример:  http://192.168.0.103/led_state_get
    //Прочитать состояние встроенного светодиода
    int v = digitalRead(LED_BUILTIN);
    //Ответ сервера
    server.send(200,      "text/plain",      "ID="+id_this      +      ";
command=led_state_get; pin="+LED_BUILTIN+"; state="+v);
}
//=========================================


//=========================================
void led_on()
{ //Запрос на изменение состояния встроенного светодиода: ВКЛЮЧИТЬ
    //Пример:  http://192.168.0.103/led_on
    //Включить встроенный светодиод
    digitalWrite(LED_BUILTIN, LOW);
    //Ответ сервера
    server.send(200,  "text/plain",  "ID="+id_this  +  ";  command=led_on;
pin="+LED_BUILTIN+"; state is ON");
}
//=========================================

//=========================================
void led_off()
{ //Запрос на изменение состояния встроенного светодиода: вЫключить
    //Пример:  http://192.168.0.103/led_off
    //вЫключить встроенный светодиод
    digitalWrite(LED_BUILTIN, HIGH);
    //Ответ сервера
    server.send(200,  "text/plain",  "ID="+id_this  +  ";  command=led_off;
pin="+LED_BUILTIN+"; state is OFF");
}
//=========================================
```

Исходник скетча хранится в файле
!_Arduino\ESP-12F_WeMos_D1_WiFi**ESP-12F_WeMos_D1_WiFi.ino**

5. Нажать на кнопку  , см. рисунок ниже.

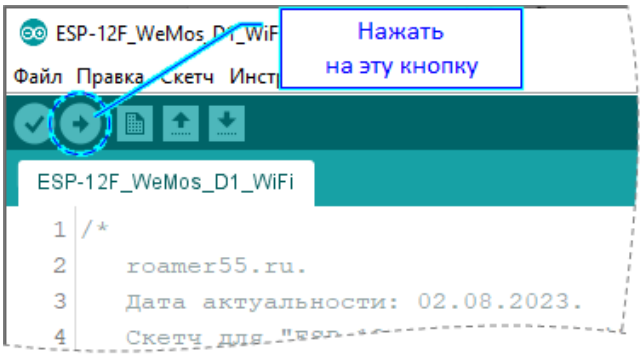


Рисунок 9 – Выполнить загрузку скетча в ESP

6. Дождаться окончания процесса загрузки (см. рисунок ниже).

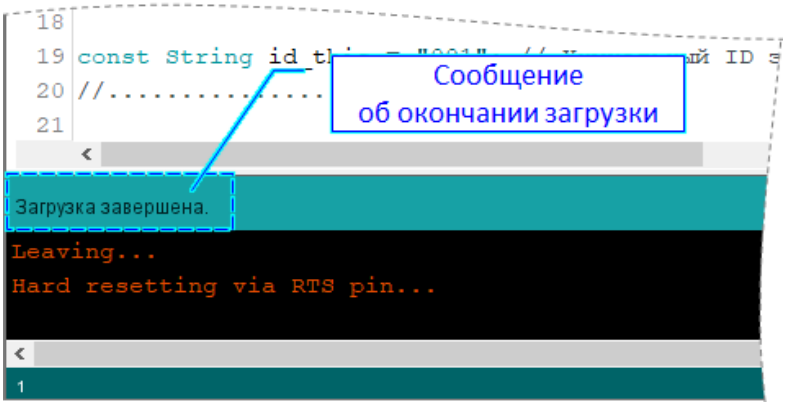


Рисунок 10 – Окончание загрузки скетча в ESP

7. Открыть монитор COM-порта (см. рисунок ниже).

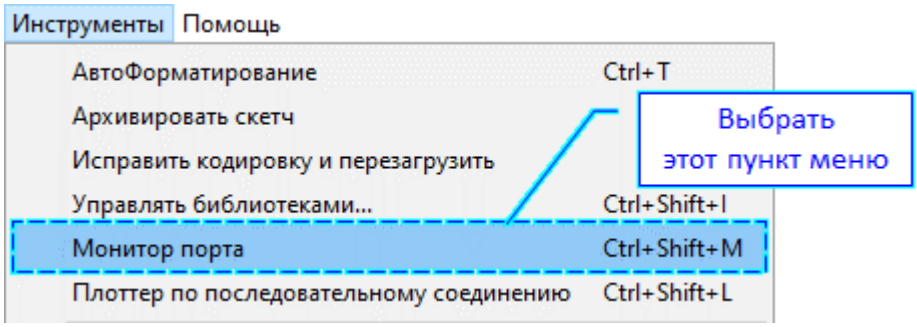


Рисунок 11 – Открыть монитор порта

В монитор COM-порта ESP вернет имя сети, свой IP и идентификатор (см. рисунок ниже).

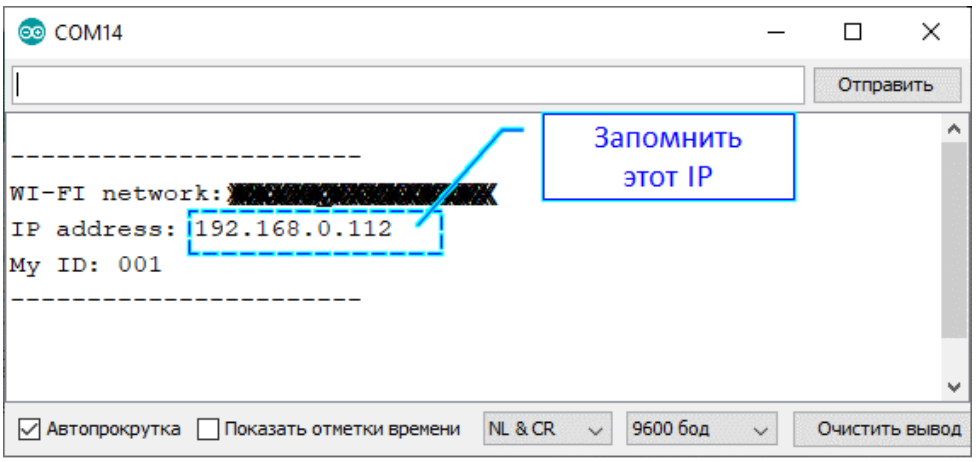


Рисунок 12 – Ответ ESP, отсылаемый на компьютер при выполнении процедуры `void setup(void)`. См. текст скетча

Важно! IP-адрес (см. рисунок выше) нужно запомнить. Он потребуется при тестировании.

На этом все действия с «железом», необходимые для рассматриваемого примера, можно считать законченными.

8. Закрыть программу Arduino IDE (она больше нам не потребуется).

4.2.3 Этап-3. Тестирование функционала ESP с помощью любого ВЕБ-браузера (например, Google Chrome)

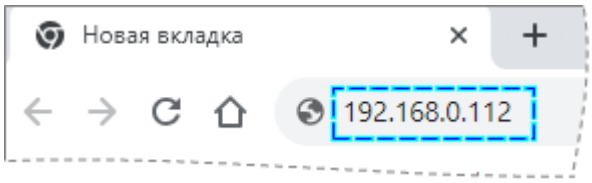
Операции, связанные с тестированием функционала ESP приведены ниже.

1. Для «чистоты эксперимента» отключить ESP от компьютера.

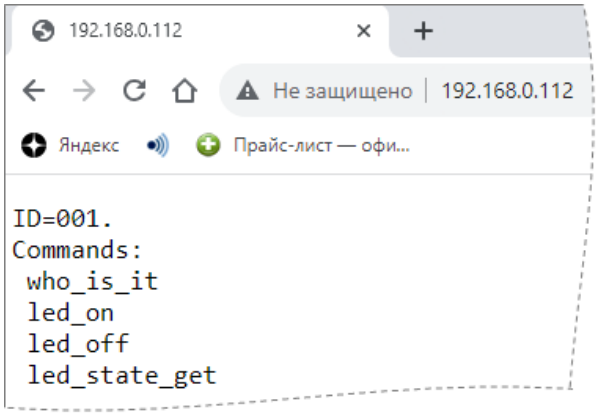
2. Выполнить подключение ESP к другому (внешнему) источнику питания например так, как показано на рисунке 4.

3. Открыть ВЕБ-браузера Google Chrome.

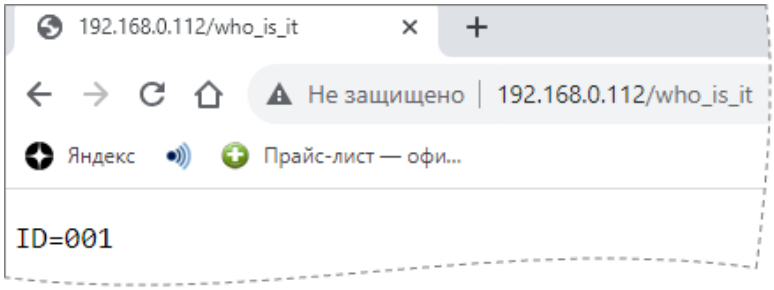
4. В адресную строку браузера ввести IP-адрес ESP (см. рисунок 12 и рисунок ниже) и нажать на кнопку `Enter` (на клавиатуре).



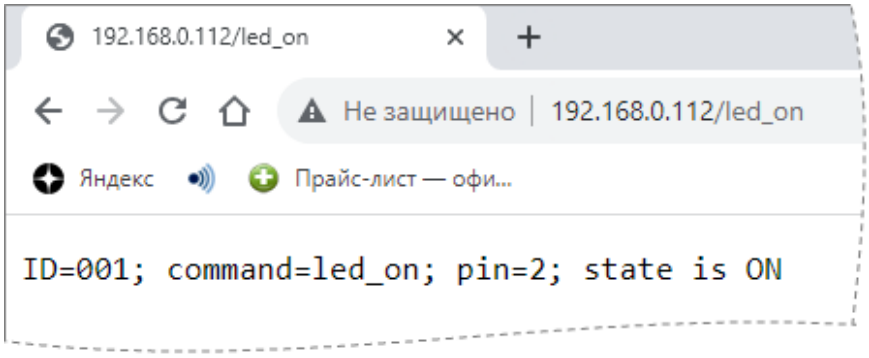
5. Дождаться ответа ESP (см. рисунок ниже).



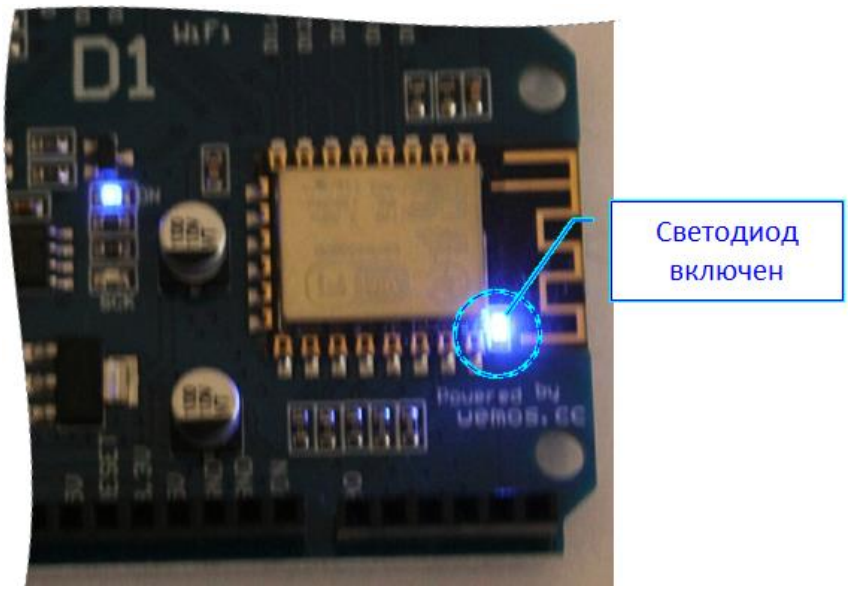
6. В адресную строку браузера ввести IP-адрес ESP и команду **`who_is_it`** и нажать на кнопку `Enter` (на клавиатуре). Дождаться ответа ESP (см. рисунок ниже).



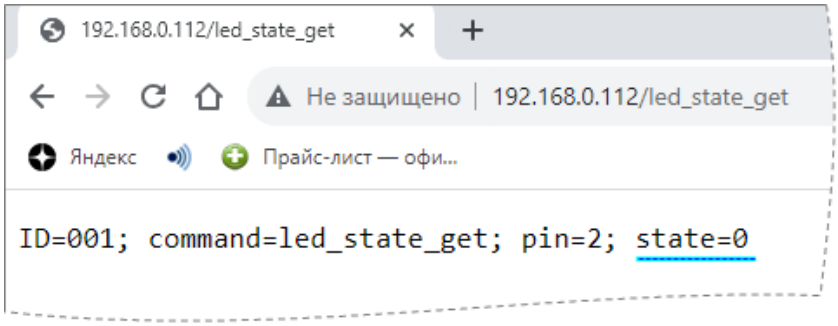
7. В адресную строку браузера ввести IP-адрес ESP и команду **`led_on`** и нажать на кнопку `Enter` (на клавиатуре). Дождаться ответа ESP (см. рисунок ниже).



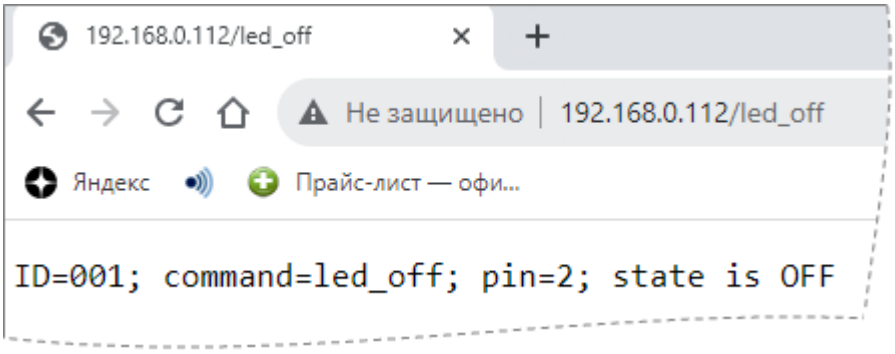
Встроенный светодиод будет включен (см. рисунок ниже).



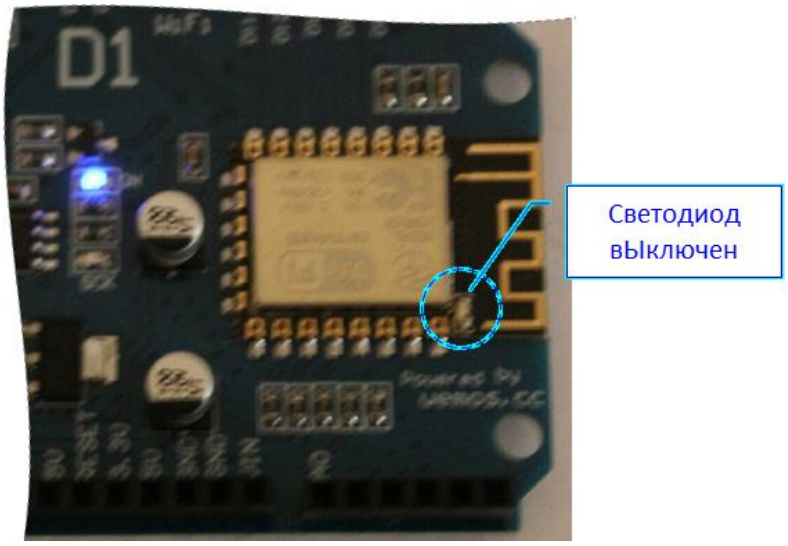
8. В адресную строку браузера ввести IP-адрес ESP и команду **`led_state_get`** и нажать на кнопку `Enter` (на клавиатуре). Дождаться ответа ESP (см. рисунок ниже).



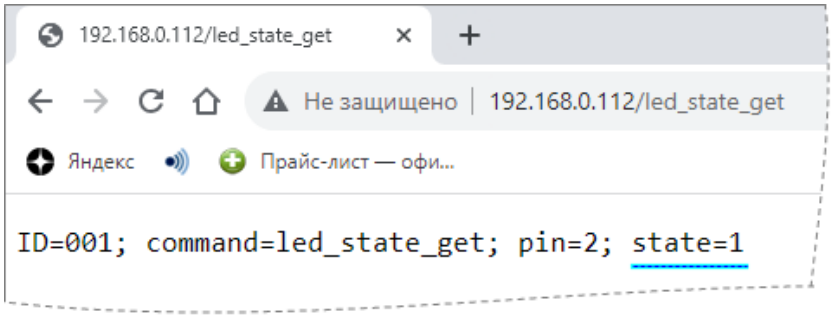
9. В адресную строку браузера ввести IP-адрес ESP и команду **`led_off`** и нажать на кнопку `Enter` (на клавиатуре). Дождаться ответа ESP (см. рисунок ниже).



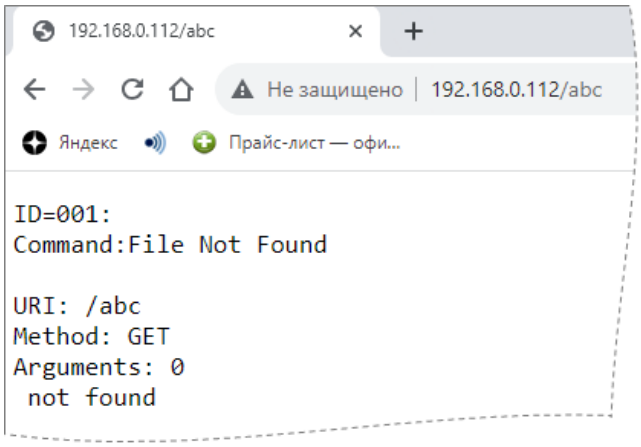
Встроенный светодиод будет вЫключен (см. рисунок ниже).



10. В адресную строку браузера ввести IP-адрес ESP и команду **`led_state_get`** и нажать на кнопку `Enter` (на клавиатуре). Дождаться ответа ESP (см. рисунок ниже).



11. В адресную строку браузера ввести IP-адрес ESP и не существующую команду (например, **abc**) и нажать на кнопку `Enter` (на клавиатуре). Дождаться ответа ESP (см. рисунок ниже).



На этом процесс тестирования функционала ESP можно считать завершенным.

4.2.4 Этап-4. Разработка DataSnap-сервера под Windows (с соответствующим функционалом)

Важно! В данном разделе используются материалы из ранее опубликованных статей [1] и [2].

В таблице ниже перечислены методы DataSnap-сервера (ориентированные на поддержку информационного обмена с DataSnap-клиентом).

Таблица 2 – Методы DataSnap-сервера (ориентированы на Клиента)

Метод	Назначение
1	2
`LogString`	Добавить принятую от Клиента строку – в LOG Сервера. Этот метод необходим только для отладочных работ
`Btn_Presh`	Реакция Сервера на нажатие какой-либо функциональной кнопки на Клиенте

В таблице ниже перечислены методы DataSnap-сервера (ориентированные на поддержку информационного обмена с ESP).

Таблица 3 – Методы DataSnap-сервера (ориентированы на Arduino)

Метод	Назначение
1	2
`ESP_CommandsList_Get`	Получить список доступных команд для управления ESP
`ESP_WhoIsIt`	Получить ID ESP (проверка доступности и работоспособности ESP)
`ESP_LED_on`	Включить встроенный светодиод ESP
`ESP_LED_off`	Выключить встроенный светодиод ESP
`ESP_LED_State_Get`	Получить текущее состояние встроенного светодиода ESP

Процесс разработки DataSnap-сервера детально рассмотрен в статьях [1] и [2].

Здесь, поэтому, не рассматривается.

Существенные фрагменты исходного кода приведены ниже.

Полный текст исходников всего проекта − размещен в отдельном архивном файле (прилагаемом к этому документу): `sp_WeMos_LedBuiltin_OnOff.zip`.

Важно! номер порта (используемого DataSnap-сервером) = **64560** (так решил Автор).

4.2.4.1 Метод LogString

Важно! Функционал этого метода может быть реализован и позже (при разработке DataSnap-клиента).

```
function TServerMethods1.LogString(sValue: string): string;
//Добавить в LOG сообщение от DataSnap-клиента
begin
  Result:='Err';
  sValue:=trim(sValue);
  if sValue<>'' then begin
     fMain.AddToLog(sValue);
     Result:='';
  end;
end;
```

```
function TfMain.AddToLog(sMsg:string; YesInsert:boolean=true):integer;
//Добавить сообщение в LOG
begin
  Result:=-1;
  if LOG_YesUse then begin
     sMsg:=trim(sMsg);
     if sMsg<>'' then begin
        if YesInsert then begin
           RE_Log.Lines.Insert(0, TimeToStr(NOW)+' | '+sMsg);
        end
        else begin
           RE_Log.Lines.Add(TimeToStr(NOW)+' | '+sMsg);
        end;
        Result:=RE_Log.Lines.Count;
     end;
  end;
end;
```

4.2.4.2 Метод Btn_Press

Важно! Функционал этого метода может быть реализован и позже (при разработке DataSnap-клиента).

```
function TServerMethods1.Btn_Presh(sBtnName: string): string;
//Обработать запрос на обработку нажатия на кнопку в DataSnap-клиенте
begin
  Result:='';
  sBtnName:=trim(sBtnName);
  if sBtnName<>'' then begin
     //Отработка нажатия кнопки на Клиенте
     if fMain.Client_Btn_Presh(sBtnName) then begin
        //Возвращаем Клиенту ответ Сервера
        Result:=fMain.RE_ESP_Answer.Lines.Text;
     end;
  end;
end;
```

```
function TfMain.Client_Btn_Presh(sBtnName: string): boolean;
//Обработать запрос на обработку нажатия на кнопку в DataSnap-клиенте
begin
  Application.ProcessMessages;
  Result:=false;
  RE_ESP_Answer.Lines.Clear;
  Edit_HTTP.Text:='';
  sBtnName:=trim(sBtnName);
  if sBtnName<>'' then begin
     Screen.Cursor:=crHourGlass;
     TRY
        sBtnName := AnsiUpperCase(sBtnName);
        if sBtnName=AnsiUpperCase('btn_ListComm') then begin
           //Дай список доступных команд
           Result:=ESP_CommandsList_Get;
        end;
        if sBtnName=AnsiUpperCase('btn_WhoIsIt') then begin
           //Дай свой ID
           Result:=ESP_WhoIsIt;
```

```
        end;
        if sBtnName=AnsiUpperCase('btn_LedOn') then begin
           //Светодиод включи
           Result:=ESP_LED_on;
        end;
        if sBtnName=AnsiUpperCase('btn_LedOff') then begin
           //Светодиод вЫключи
           Result:=ESP_LED_off;
        end;
        if sBtnName=AnsiUpperCase('btn_LedState') then begin
           //Дай остояние светодиода
           Result:=ESP_LED_State_Get;
        end;
     FINALLY
        AddToLog('Android: ' + Edit_HTTP.Text);
        Screen.Cursor:=crDefault;
     END;
  end;
end;
```

4.2.4.3 Метод ESP_CommandsList_Get

Получить список доступных команд для управления ESP.

```
function  TfMain.ESP_CommandsList_Get:boolean;
//Получить список доступных команд для управления ESP
Var
  sAddr:string;
begin
  sAddr:=Comm_for_HTTP_Prepare(Edit_IP.Text, '');
  Edit_HTTP.Text:=sAddr;
  Result:=http_content_from_addr_get(
                                     sAddr,
                                     RE_ESP_Answer.Lines
                                     );
end;
```

4.2.4.4 Метод ESP_WhoIsIt

Получить ID ESP (проверка доступности и работоспособности ESP).

```
function  TfMain.ESP_WhoIsIt:boolean;
//Получить ID ESP (проверка доступности и работоспособности ESP)
Var
  sAddr:string;
begin
  sAddr:=Comm_for_HTTP_Prepare(Edit_IP.Text, 'who_is_it');
  Edit_HTTP.Text:=sAddr;
  Result:=http_content_from_addr_get(
                                     sAddr,
                                     RE_ESP_Answer.Lines
                                     );
end;
```

4.2.4.5 Метод ESP_LED_on

Включить встроенный светодиод ESP.

```
function  TfMain.ESP_LED_on:boolean;
//Включить встроенный светодиод ESP
Var
  sAddr:string;
begin
  sAddr:=Comm_for_HTTP_Prepare(Edit_IP.Text, 'led_on');
  Edit_HTTP.Text:=sAddr;
  Result:=http_content_from_addr_get(
                                     sAddr,
                                     RE_ESP_Answer.Lines
                                     );
end;
```

4.2.4.6 Метод ESP_LED_off

Выключить встроенный светодиод ESP.

```
function  TfMain.ESP_LED_off:boolean;
//Выключить встроенный светодиод ESP
Var
  sAddr:string;
begin
  sAddr:=Comm_for_HTTP_Prepare(Edit_IP.Text, 'led_off');
  Edit_HTTP.Text:=sAddr;
  Result:=http_content_from_addr_get(
                                     sAddr,
                                     RE_ESP_Answer.Lines
                                     );
end;
```

4.2.4.7 Метод ESP_LED_State_Get

Получить текущее состояние встроенного светодиода ESP.

```
function  TfMain.ESP_LED_State_Get:boolean;
//Получить текущее состояние встроенного светодиода ESP
Var
  sAddr:string;
begin
  sAddr:=Comm_for_HTTP_Prepare(Edit_IP.Text, 'led_state_get');
  Edit_HTTP.Text:=sAddr;
  Result:=http_content_from_addr_get(
                                     sAddr,
                                     RE_ESP_Answer.Lines
                                     );
end;
```

4.2.4.8 Вспомогательные процедуры и функции

```
   Uses
     System.Net.HttpClientComponent,
     Classes,
     SysUTILS;

function Comm_for_HTTP_Prepare(sAddr:string;
                               sCommand:string;
                               sHTTPprefics:string='http://'
                              ):string;
//Подготовить HTTP-запрос для ESP
begin
  Result:='';
  sAddr:=trim(sAddr);
  if sAddr<>'' then begin
     if sAddr[length(sAddr)]<>'/' then sAddr:=sAddr+'/';
     Result:=trim(sHTTPprefics)+sAddr+trim(sCommand);
  end;
end;


function http_content_from_addr_get(sAddr:string;
                                    ListRes:TStrings
                                   ):boolean;
//Получить ответ ВЕБ-сервера на запрос (команду)
Var
  NHTTPC: TNetHTTPClient;
begin
  Result:=false;
  if Assigned(ListRes) then begin
     ListRes.Clear;
     sAddr:=trim(sAddr);
     if sAddr<>'' then begin
        NHTTPC := TNetHTTPClient.Create(nil);
        TRY
          TRY
            ListRes.LoadFromStream(NHTTPC.Get(sAddr).ContentStream);
            Result:=true;
          EXCEPT
          END;
        FINALLY
          FreeAndNil(NHTTPC);
        END;
     end;
  end;
end;
```

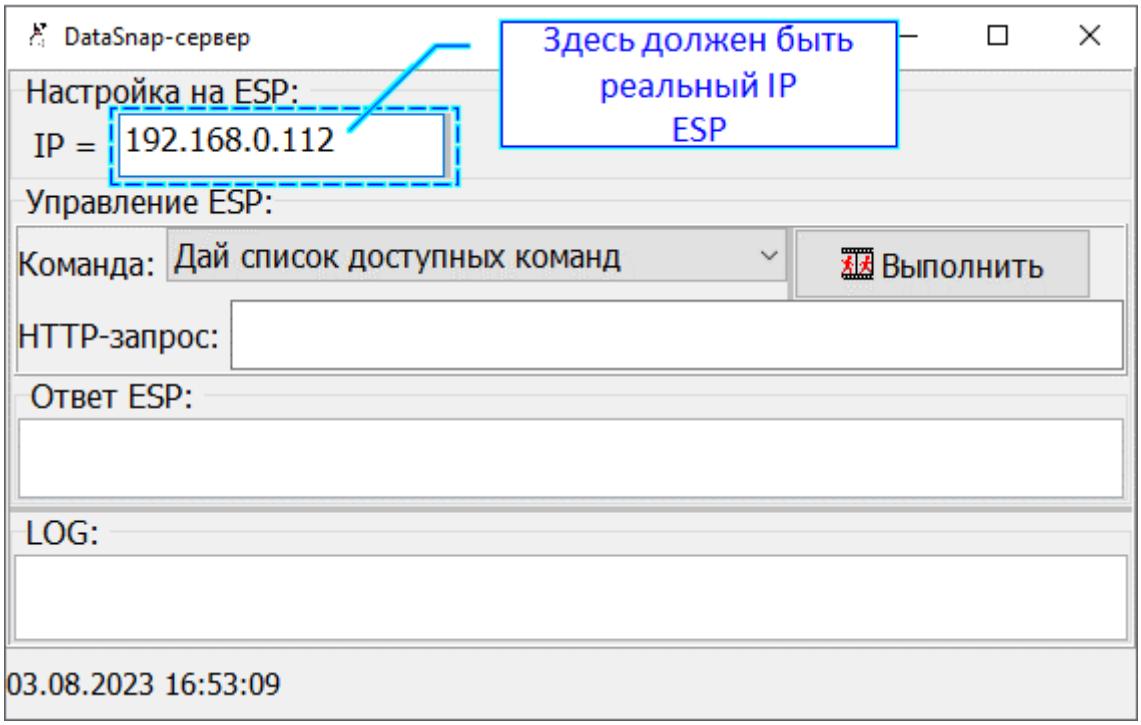
4.2.5 Этап-5. Тестирование функционала ESP с помощью DataSnap-сервера

Операции, связанные с тестированием функционала ESP приведены ниже.

1. Для «чистоты эксперимента» отключить ESP от компьютера.

2. Выполнить подключение ESP к другому (внешнему) источнику питания например так, как показано на рисунке 4.

3. Открыть DataSnap-сервер (см. рисунок ниже).



4. Внести в поле IP (см. рисунок выше) реальный IP-адрес ESP (см. рисунок 12).

5. Выбрать из выпадающего списка команду «*Дай список доступных команд*» и нажать на кнопку «Выполнить» (см. рисунок ниже).

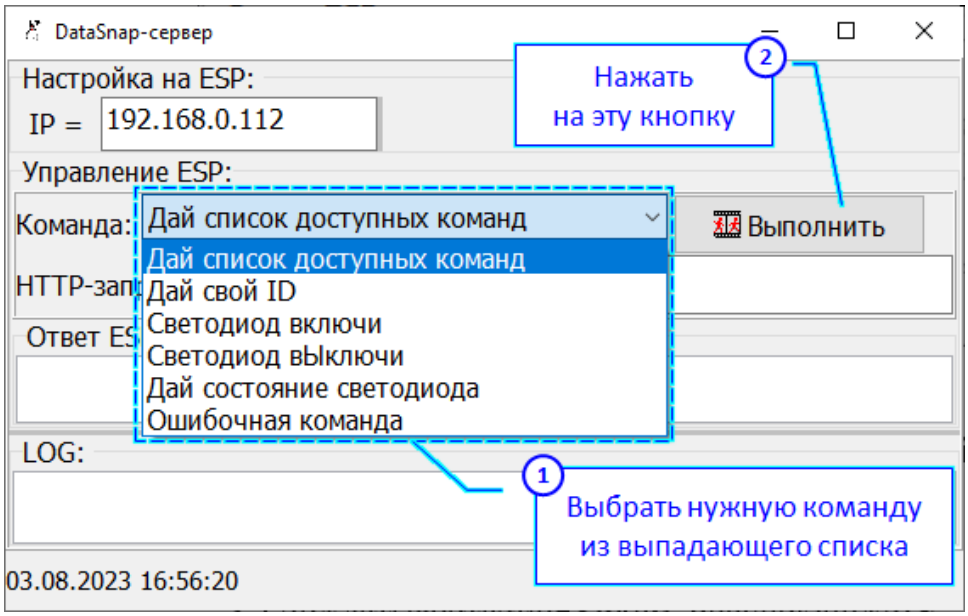


Рисунок 13 – Отправить команду на ESP

6. Дождаться ответа ESP (см. рисунок ниже).

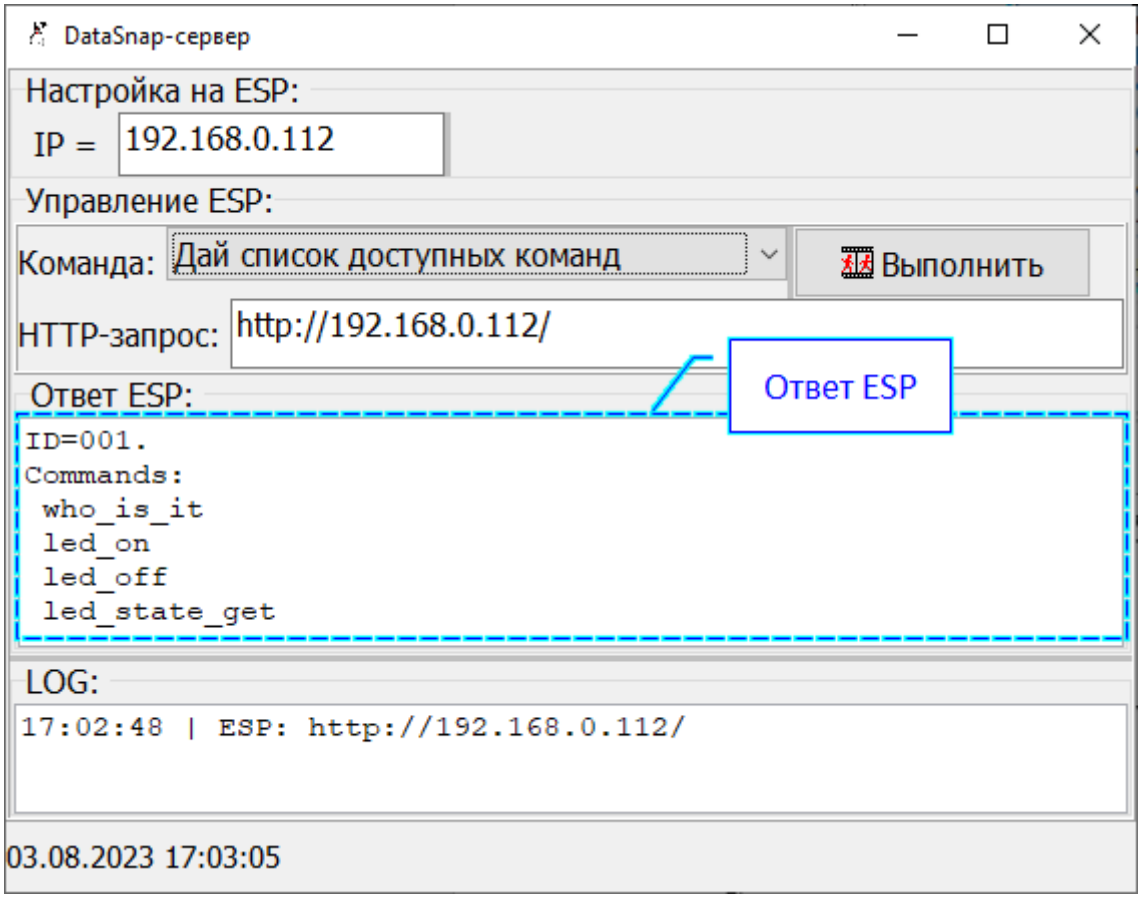


Рисунок 14 – Ответ ESP

7. Убедиться, что команда выполнена корректно.

8. Выполнить пункты 5, 6 и 7 для каждой команды из выпадающего списка (см. рисунок 13).

На этом процесс тестирования функционала ESP и DataSnap-сервера (в этой части) можно считать завершенным.

4.2.6 Этап-6. Разработка DataSnap-клиента под Android (с соответствующим функционалом)

Важно!

1. Процесс разработки DataSnap-клиента детально рассмотрен в статьях [1] и [2]. Здесь, поэтому, не рассматривается.
2. В данном примере (в исходниках клиента) жестко зашиты номер порта и IP-адрес DataSnap-сервера.

   А именно:

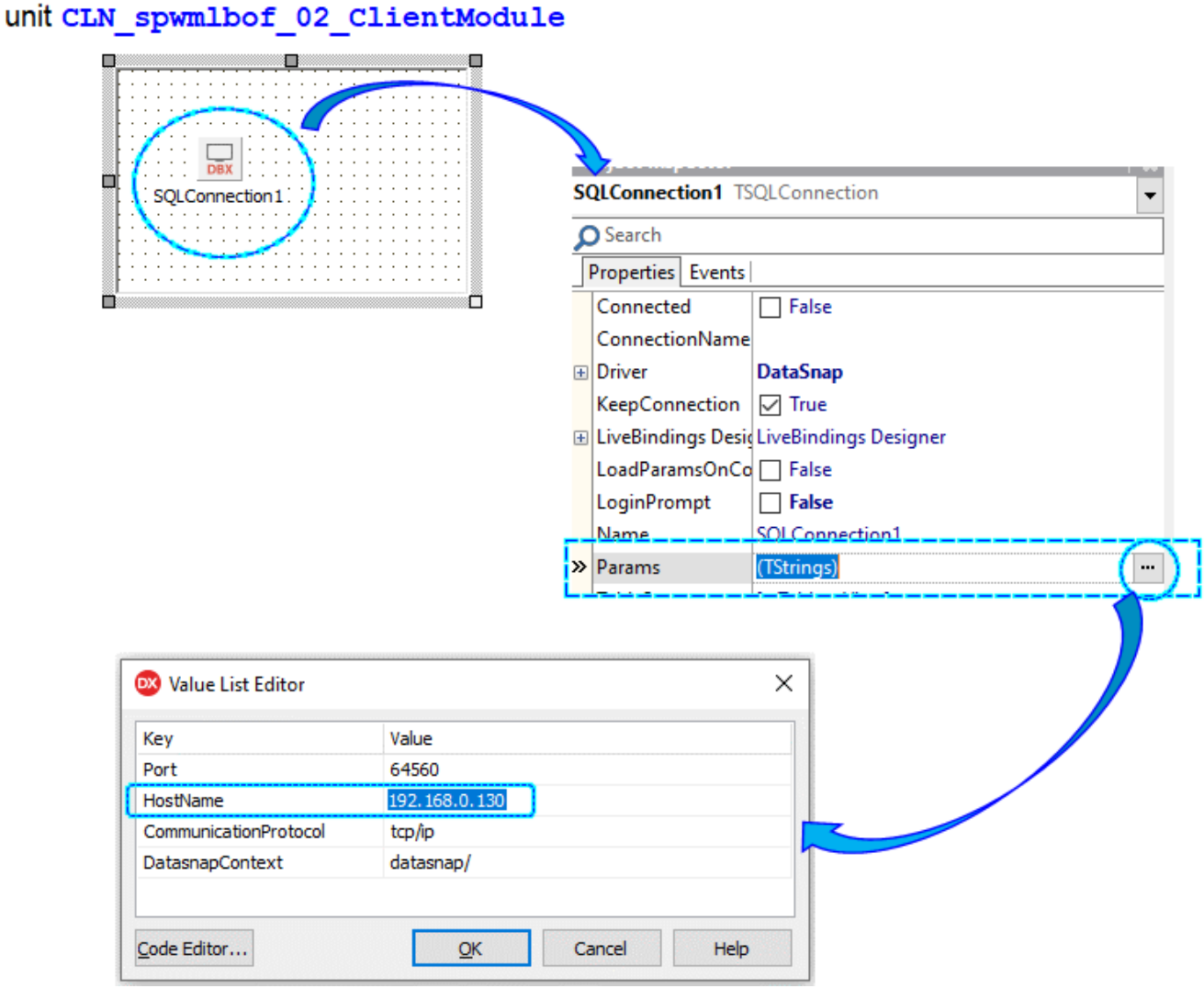
   ```
   Port=64560
   HostName=192.168.0.130
   ```

Понятное дело, что номер порта еще как-то «можно пережить».

А вот жестко зашитый IP-адрес – никого не может устроить совсем.

Есть, конечно, методы «борьбы» с «этим злом» на уровне Клиента в RunTime режиме. Но это выходит за рамки данного примера.

А вот «подрихтовать» IP-адрес в Design mode (чтобы все «пощупать» на практике) вполне можно. «Копать» нужно по цепочке:



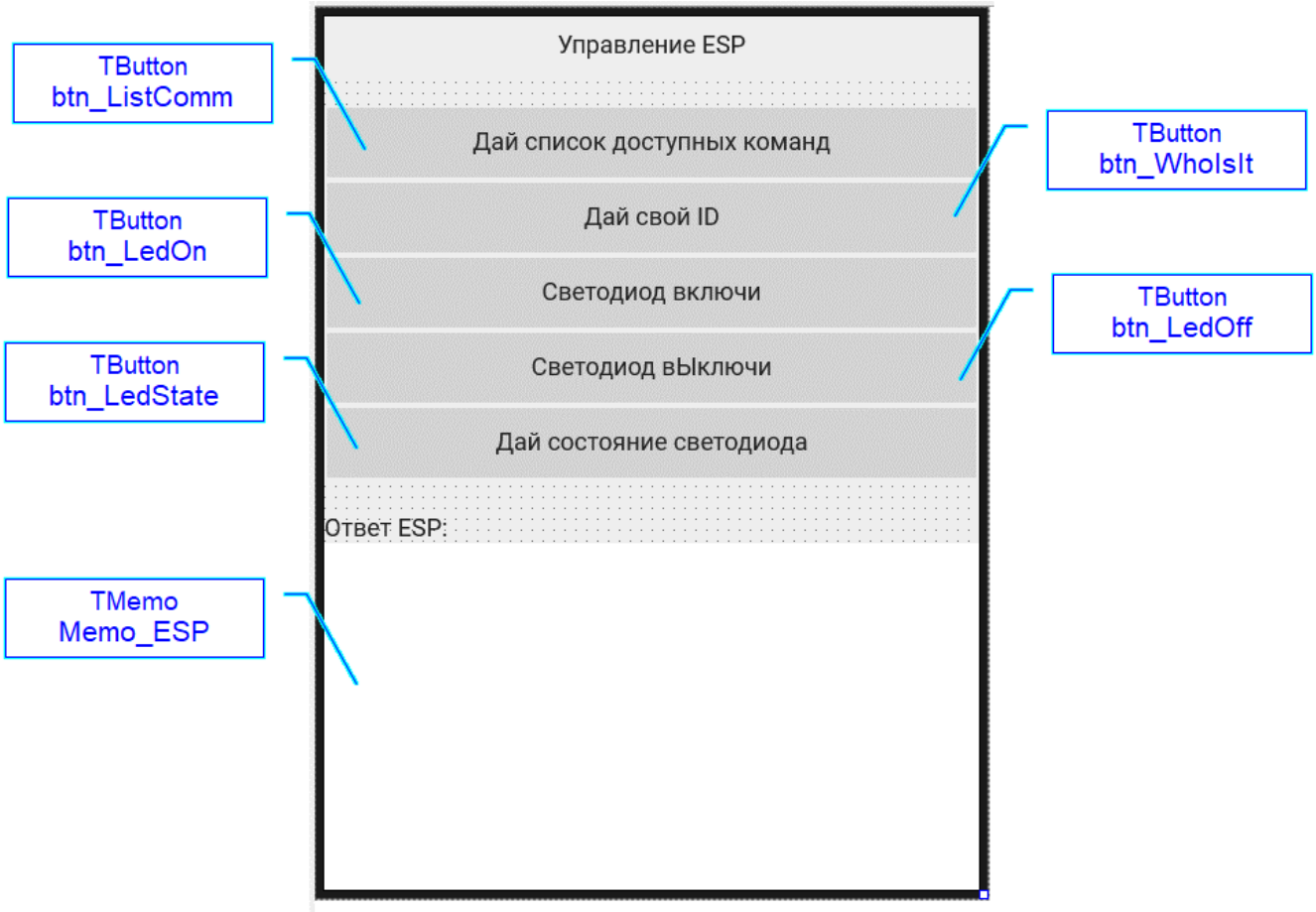
Еще один важный момент. Из всего множества доступных «разрешений» следует оставить только два:

Change wifi state	true
Internet	true

Существенные фрагменты исходного кода приведены ниже.

Полный текст исходников всего проекта – размещен в отдельном архивном файле (прилагаемом к этому документу): `sp_WeMos_LedBuiltin_OnOff.zip`.

На рисунке ниже представлен внешний вид формы DataSnap-клиента.



Для всех кнопок предусмотрен один обработчик (на нажатие):

```
procedure TfMain.btn_ListCommClick(Sender: TObject);
begin
  Memo_ESP.Lines.Clear;
  Memo_ESP.Lines.Text:=ClientModule1.ServerMethods1Client.Btn_Presh(
                                                        (Sender as TButton).Name
                                                       );
end;
```

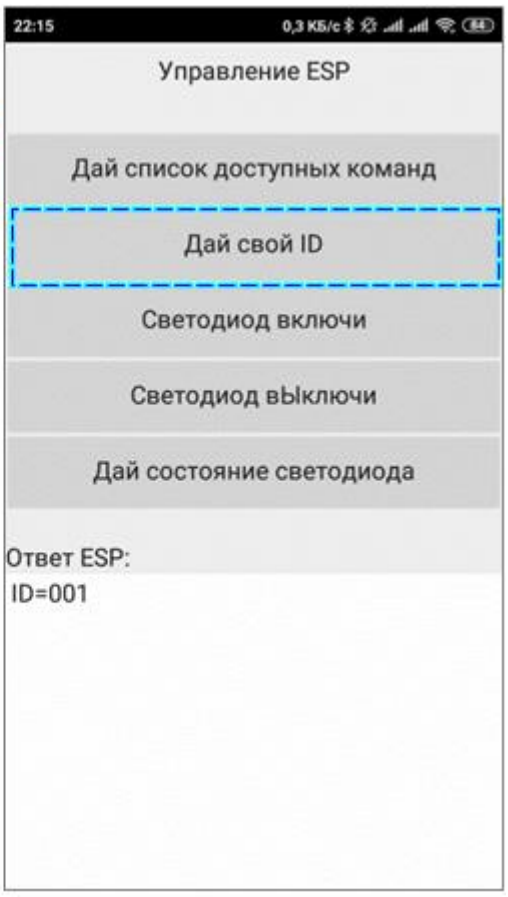
Для рассматриваемого примера этого достаточно.

Внешний вид главной формы приложения DataSnap-клиента сразу после запуска показан на рисунке ниже:

4.2.7 Этап-7. Тестирование функционала ESP с помощью DataSnap-клиента

На рисунках ниже приведены скрины результатов тестирования информационного обмена между программами по схеме:

`Android` (телефон) ⇔ **`Windows`** (компьютер) ⇔ **`ESP`** (Arduino)



Т.е., все так, как и должно быть…

На этом процесс тестирования информационного обмена между ESP, DataSnap-сервером и DataSnap-клиентом можно считать завершенным.

5 Выводы

Летят Петька с Василием Иванычем в самолете:
− Петька, прибор?!
− Тридцать!
− Что «тридцать»?
− А что «прибор»?

А что «выводы»?

1. Возможности DataSnap-технологии существенно интересны и полезны для практического применения. И с течением времени ее актуальность не уменьшается.

2. Мощь Delphi настолько упрощает и ускоряет процесс разработки такого рода Систем Автоматизации, что любой Delphi-программист может легко (играючи) «замахнуться на Вильяма, понимаете ли, нашего Шекспира» и создать «домашнюю техносферу своими руками» (при желании, конечно).

3. Очередное огромное спасибо − авторам статьи [1] за те идеи, которые возникли в процессе ее проработки.

6 Используемые источники

В таблице ниже перечислены основные источники информации, используемые в этом документе.

Таблица 4 – Используемые источники

№ п/п	Источник	Ссылка
1	2	2
1	Статья «Delphi (C++Builder) Android Mobile Client DataSnap Server»	https://community.embarcadero.com/write-blog-post/entry/delphi-cbuilder-android-mobile-client-datasnap-server-1840
2	Статья «Использование DataSnap-технологии на примере разработки комплекса взаимодействующих приложений (ОС Windows и ОС Android) в среде Delphi 10.2 Tokyo. Последовательность действий»	https://roamer55.ru/main_programming/delphi/delphi_10_2_hybrid/delphi_10_2_hybrid_datasnap/delphi_10_2_hybrid_datasnap_spBallCollision/