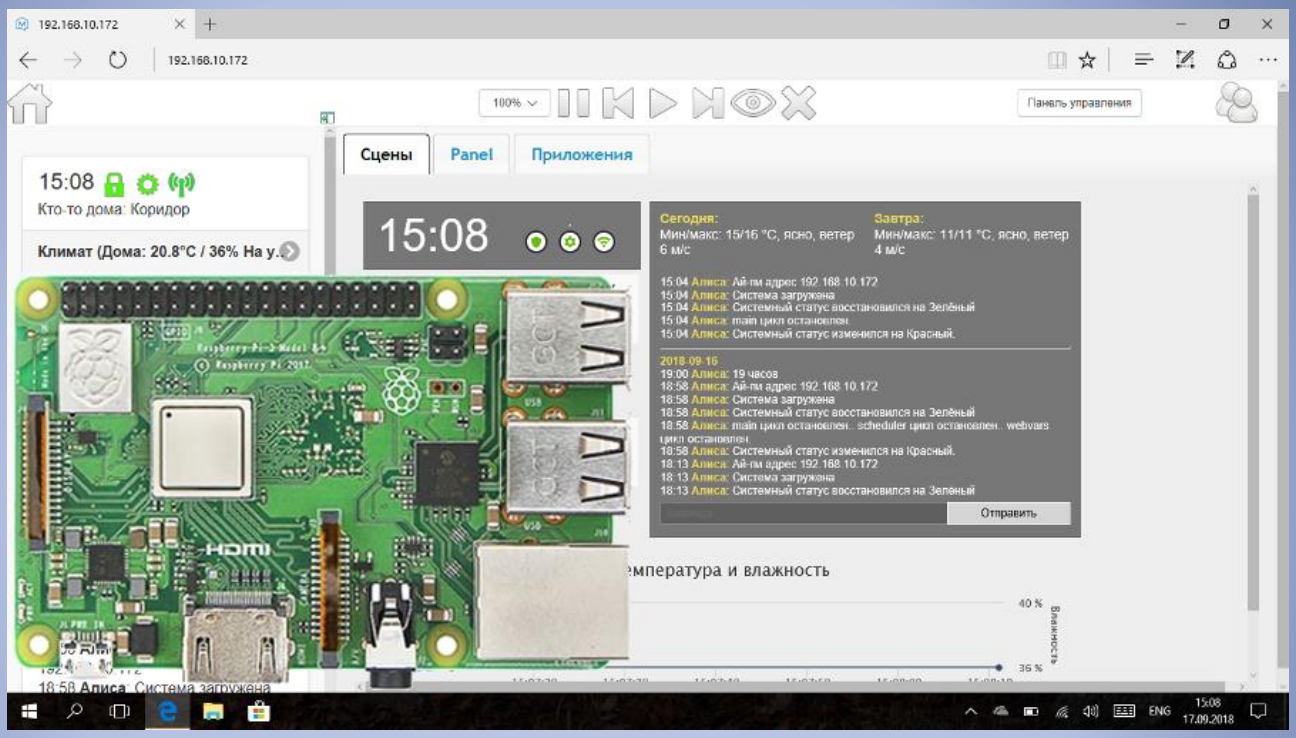
В.Н. Гололобов

IoT как ИНТРАНЕТ вещей



С Raspberry Pi и MajorDoMo

Москва - 2019

Оглавление

Пролог

> *Интернет вещей (англ. Internet of Things, IoT) — концепция вычислительной сети физических предметов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой...*
>
> *Wikipedia*

В Интернете происходит многое из того, что мы не видим, когда запускаем Internet Explorer, Edge, Opera или Firefox. Способность Интернета доставлять на компьютер фильмы и телевизионные программы зависит от пропускной способности сетей, которая растет от года к году. Но, возможно, будут созданы специализированные коммерческие сети на основе WiFi исключительно для IoT, чтобы ваш пылесос мог без вашего участия заказать мешки для сбора пыли, а холодильник заказал ваш любимый йогурт, если за завтраком вы съели последнюю порцию.

Получается, что «Интернет вещей, не за горами. Но это задача для предприятий-производителей техники и специалистов по Интернет-технологиям. А мы попробуем немного подглядеть, что там «не за горами»?

Мы проведем ряд экспериментов без Интернета, проведем их в домашней компьютерной сети под флагом «ИНТРАНЕТ вещей». Используем модули: Arduino, ESP8236, OTA WeMOS D1, которые будут играть роль «умных вещей». В качестве среды их взаимодействия используем MajorDoMo – систему «Умный дом».

Для установки системы MajorDoMo, которую можно использовать и в Windows, и в Linux, и в MacOS, выберем миникомпьютер Raspberry Pi.

Если вам это покажется интересным, то, прочитав рассказ, вы можете провести свои эксперименты, которые подойдут не только для изучения технологии, но смогут найти применение и на практике. Все зависит от ваших возможностей и круга интересов.

Глава 1. Выбор модулей для опытов

Миникомпьютер Raspberry Pi

Это важно!

Не спешите покупать все те модули, что перечислены ниже. Если вы намерены провести эксперименты за столом «своей лаборатории», то их может оказаться гораздо меньше. Подсчитайте стоимость всего, что перечислено ниже, а после этого пролистайте книгу до конца, чтобы спланировать свои закупки. И учтите, что товары отбирались по наименьшей стоимости на осень 2018 года.

В этом рассказе мы используем модуль Raspberry Pi (**рис. 1.1**) для установки среды MajorDoMo в комфортной для экспериментов форме.

Рис. 1.1. Модуль Raspberry Pi 3 на Aliexpress

Примечание.

Следует обратить внимание на модель модуля. В данном рассказе используется Raspberry Pi 3 модель B v.1.2. Это не означает, что нужна только эта модель. Но какие-то настройки и операции для другой модели могут отличаться от тех, что будут представлены ниже. И учтите, что со временем многое может измениться.

Нужно ли покупать Raspberry Pi в полном наборе: модуль, дисплей, блок питания, SD-карта с операционной системой, корпус? Нет, понадобится только модуль и USB-кабель для него. Если нет патч-корда, то его тоже нужно будет либо сделать, либо купить (**рис. 1.2**).

Рис. 1.2. USB-кабель для Raspberry Pi и патч-корд для подключения к роутеру

USB-кабелей, возможно, потребуется два, если ваш Arduino будет иметь гнездо для USB-micro. И это только при условии, что у вас нет таких кабелей для зарядки смартфона или планшета.

А SD-карта объемом 16 Гбайт класса 10 нам действительно понадобится.

Arduino и то, что с ним связано

Если у вас есть Arduino, используйте его, если нет, то не торопитесь покупать (**рис. 1.3**).

Примечание.

Для экспериментов подойдет ряд моделей: Arduino Uno, Arduino Mega 2560 и т.д. Но удобнее использовать те модели, которые позволяют легко подключить готовые насадки (shields).

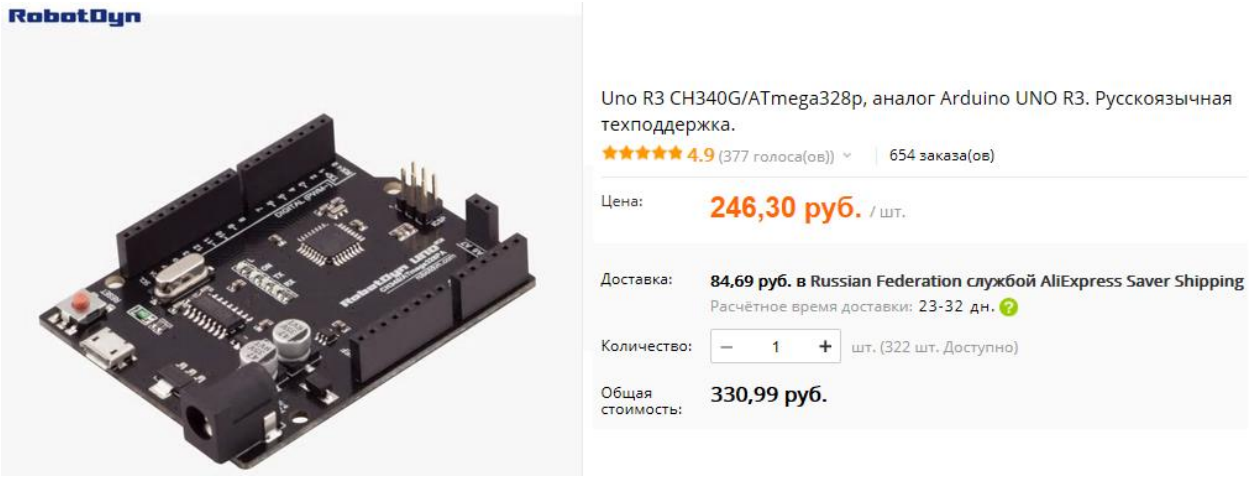


Рис. 1.3. Модуль Arduino Uno с гнездом для USB-micro

Модуль Arduino сам по себе интересен. С его помощью можно лучше понять работу с микроконтроллерами **[2]**. Модуль Arduino прекрасно работает с многочисленными датчиками, используя разные аппаратные интерфейсы. Он имеет хорошо развитую структуру встроенных устройств: USART, АЦП, интерфейсы 1-wire, SPI.

В экспериментах он примет участие, скорее в качестве вспомогательного устройства. Хотя с надстройкой Ethernet его можно подключить к роутеру (**рис. 1.4**).

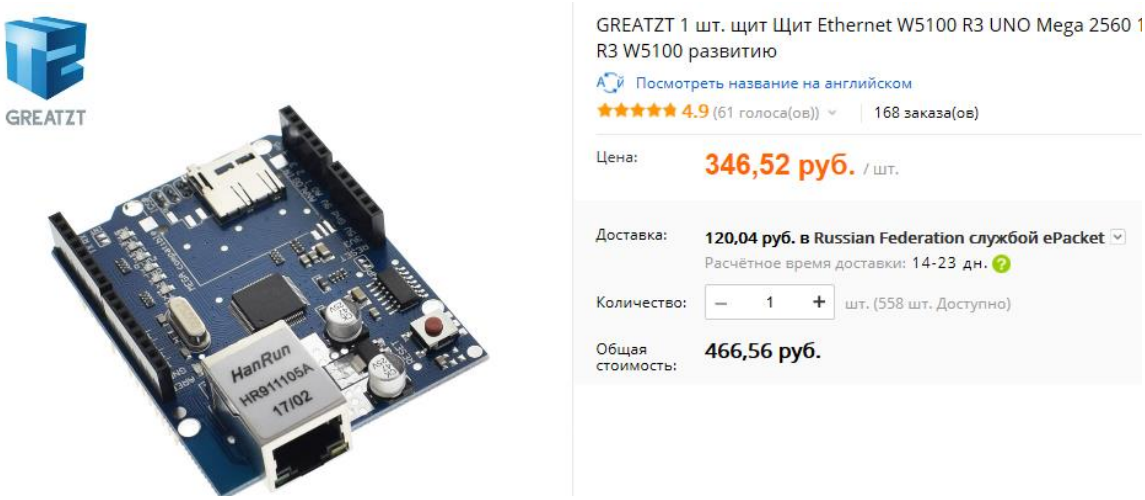


Рис. 1.4. Ethernet надстройка для Arduino, подключаемая к компьютерной сети

Можно использовать WiFi надстройку для Arduino Mega (**рис. 1.5**).

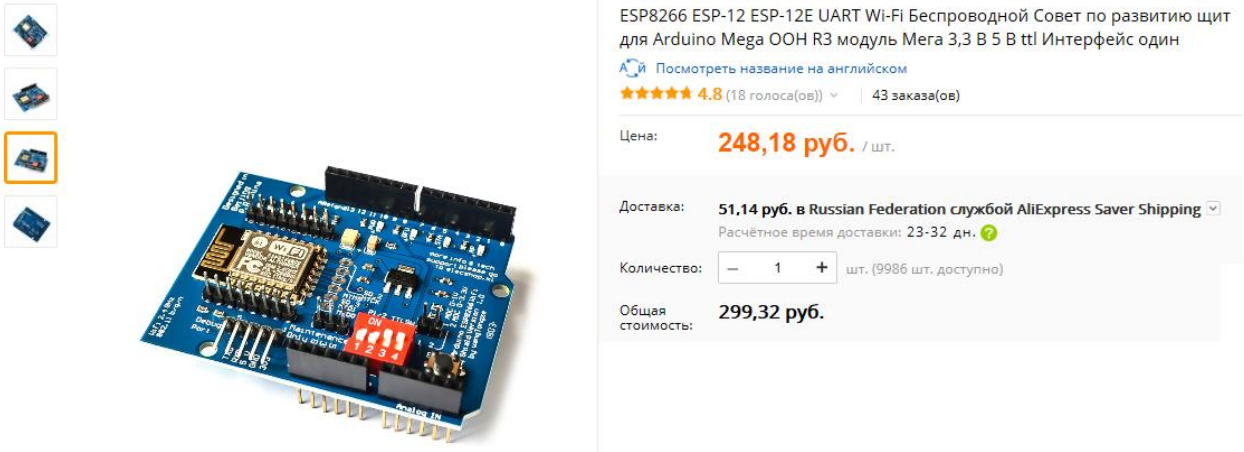


Рис. 1.5. Надстройка для использования WiFi с модулем Arduino Mega

Что же лучше – надстройка Ethernet или WiFi? Удобнее, наверное, WiFi – подключать модуль патч-кородом не так удобно.

Но есть модуль, который может подключаться по WiFi, а код для него при этом можно написать в среде программирования Arduino (**рис. 1.6**).

Рис. 1.6. WiFi модуль, работающий со средой программирования Arduino

Модуль ESP8266

Этот модуль требует небольшого рассказа о себе (**рис. 1.7**).

Рис. 1.7. Модуль ESP8266 с управляемым реле

Справка.

ESP8266 — микроконтроллер китайского производителя Espressif с интерфейсом Wi-Fi. Помимо Wi-Fi, микроконтроллер отличается отсутствием флеш-памяти в SoC, программы пользователя исполняются из внешней флеш-памяти с интерфейсом SPI.
Основное применение ESP8266 находит в управлении разнообразными бытовыми приборами через беспроводные сети. Концепцию такого управления часто называют «Internet of Things» (IoT, «интернет вещей»). Верхний уровень IoT представлен разнообразными приложениями под популярные платформы (Android, iOS, Windows). Эти приложения позволяют разработчику прибора адаптировать приложение под управление его прибором и передать пользователю готовое решение.

Позже мы познакомимся с модулем поближе.

Датчики для опытов

Самый простой датчик – это пара: герконовые контакты и магнит. Геркон (**рис. 1.8**), другими словами, герметизированные контакты, простой, но достаточно эффективный и надежный переключатель. Его порой применяют в приборах, что увеличивает срок службы устройства.

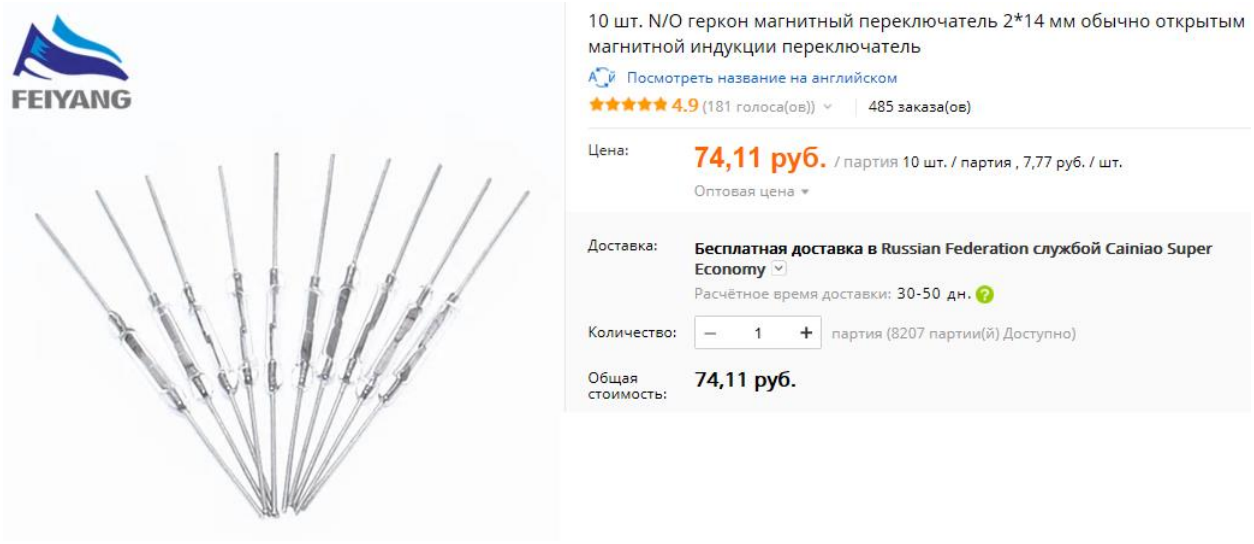


Рис. 1.8. Герконы (партия 10 шт) на Aliexpress

Измерять температуру можно с помощью разных датчиков. Самым простым может стать обычный диод, в производстве часто используют термопары.

А нам для регистрации температуры можно использовать датчик DS18B20 (**рис. 1.9**). Arduino прекрасно с этим справляется, благодаря готовой библиотеке функций.

Рис. 1.9. Датчик температуры производства Dallas

Некоторые эксперименты с датчиком описаны в моей книге **[2]**. Датчик может давать весьма точные значения температуры, если применять его в нешироком диапазоне температур.

Но чаще нас интересуют одновременно такие параметры как температура и атмосферное давление. Измерять эти параметры мы будем с помощью датчика BMP180 (**рис. 1.10**).

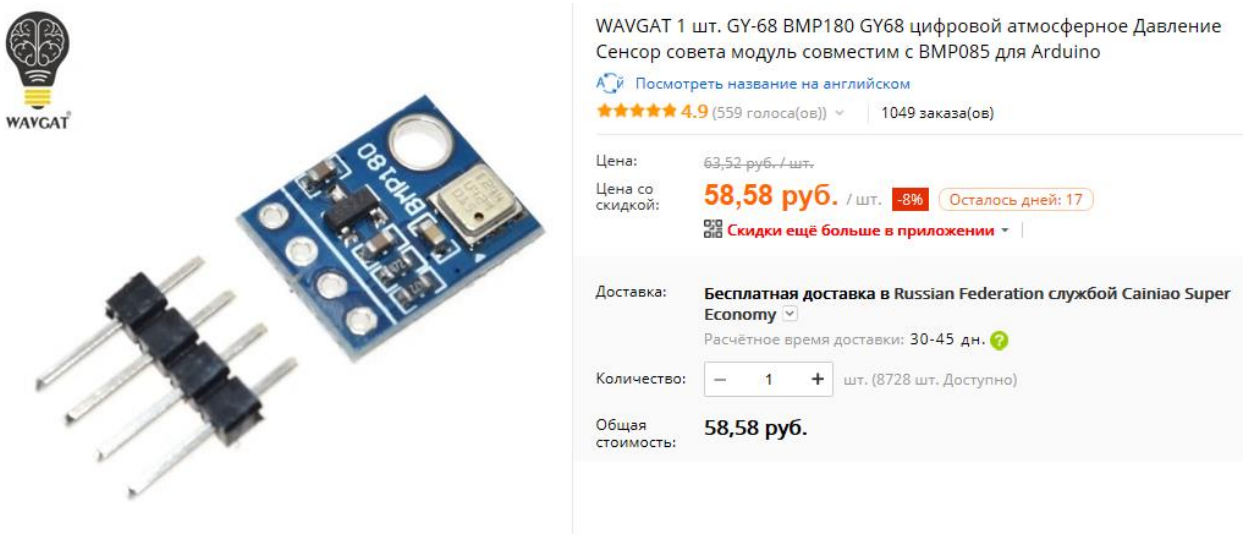


Рис. 1.10. Датчик атмосферного давления

Если понадобится определять освещенность, то с этой задачей хорошо справится фоторезистор (**рис. 1.11**).

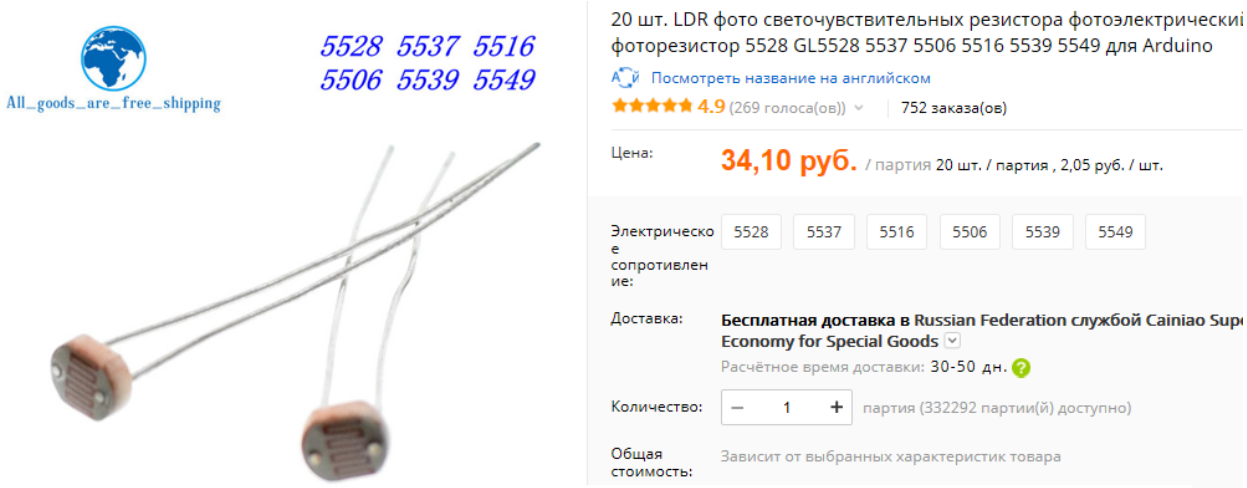


Рис. 1.11. Фоторезистор

Возможно, в процессе работы потребуется пополнить этот список. Если так, пополним.

Глава 2. Установка MajorDoMo с нуля

Подготовка использованной SD-карты в Linux

При достаточном опыте можно обойтись без тех настроек, которые последуют далее, но для начинающих экспериментаторов они, с моей точки зрения, не будут лишними.

В моем предыдущем рассказе о Raspberry Pi **[1]** мне несколько раз удалось запортить операционную систему. Приходилось освобождать SD-карту и начинать все с самого начала. Я не уверен, что, проводя эксперименты, вы не окажетесь в такой же ситуации, поэтому и рассказ о подготовке SD-карты к новой записи не будет лишним. Начать я хочу с того, как это сделать в Linux. После подключения SD-карты к считывателю карт в моем виртуальном дистрибутиве Rosa операции начинаются с запуска программы Gparted, которая потребует пароль для подтверждения прав администратора.

Почему Linux, если у многих Windows? Мы вернемся к Windows чуть позже, а про подготовку карты в Linux мы, мне кажется, в предыдущей книге не было сказано ни слова.

Программа показывает обустройство жесткого диска компьютера, на котором установлена операционная система. То, что записано на SD-карте, не входит в состав жесткого диска. Что требует перехода к выбору устройств (**рис. 2.1**).

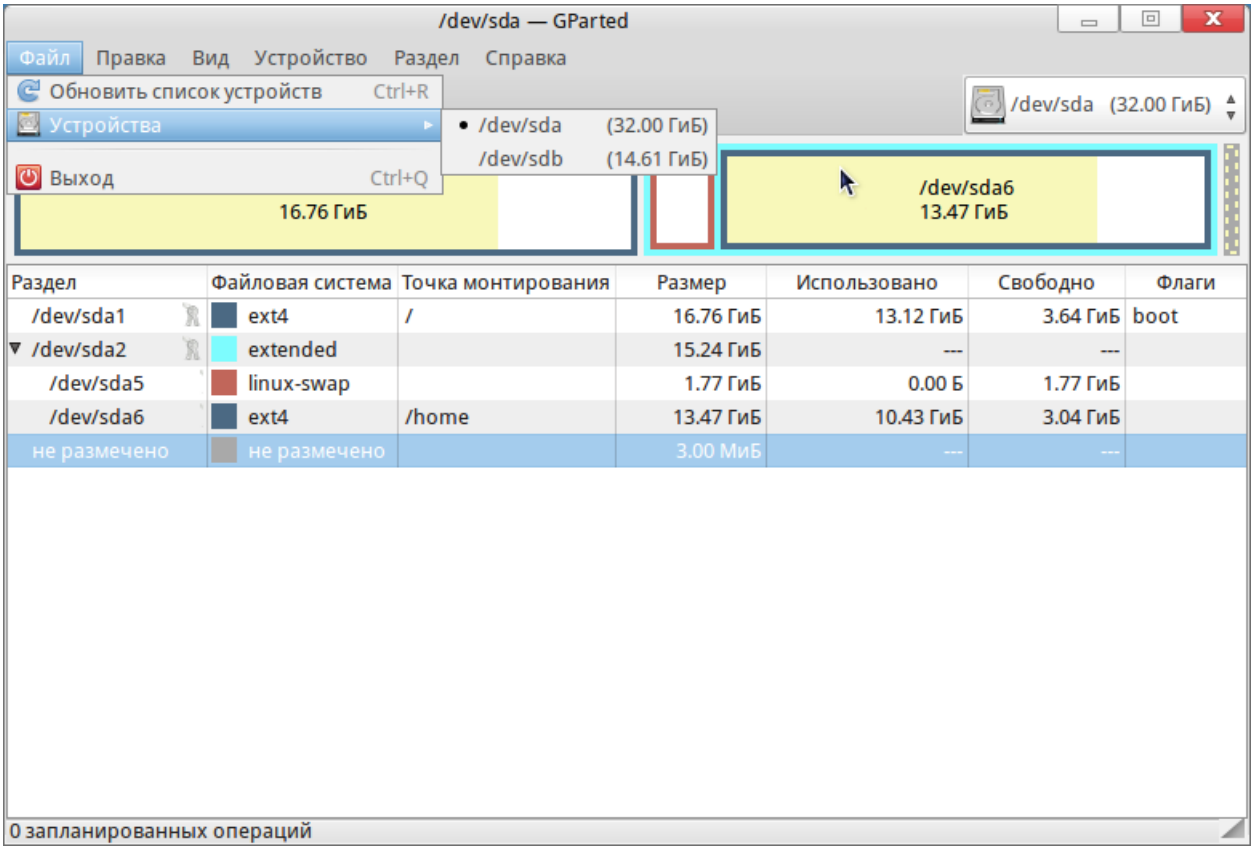


Рис. 2.1. Выбор устройства в Linux Rosa

Требуемое устройство – это sdb (все устройства расположены в директории /dev). После выбора устройства вы увидите SD-карту (**рис. 2.2**).

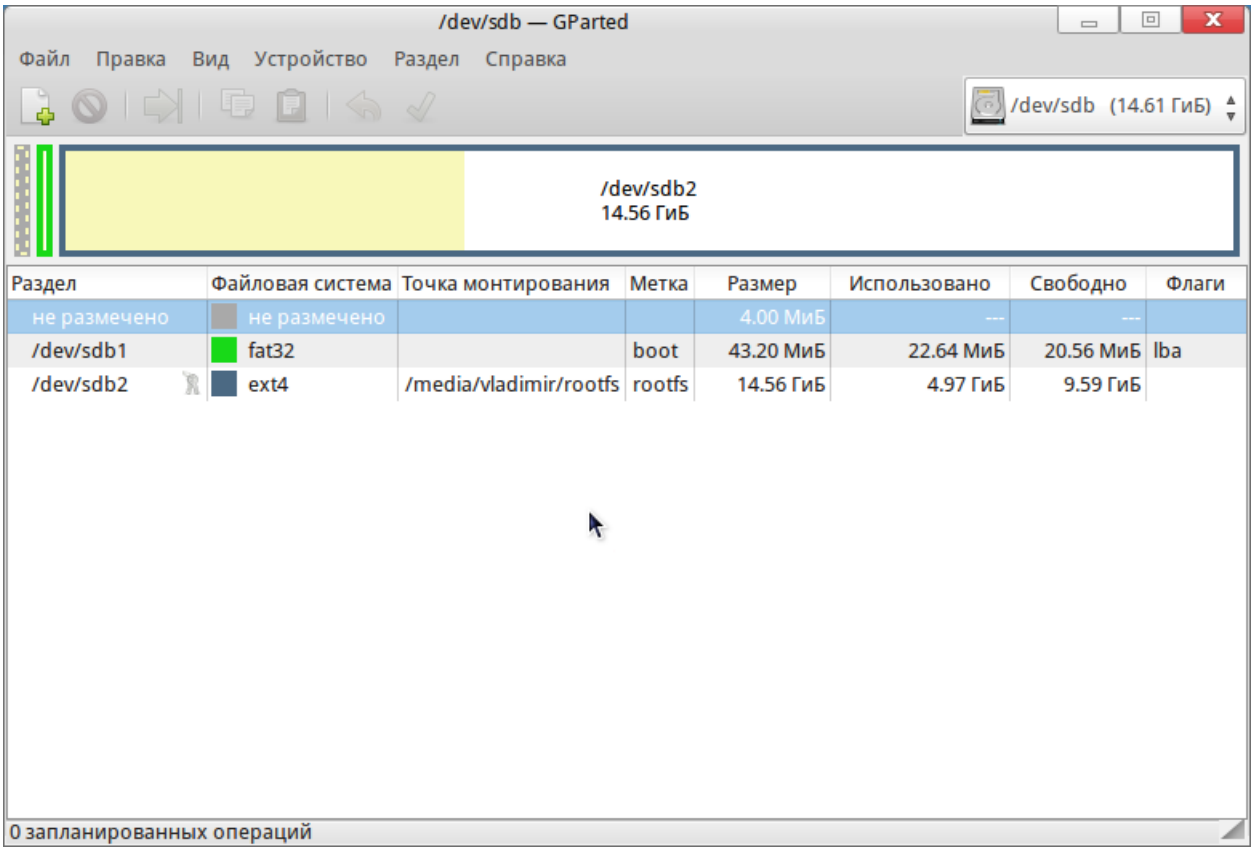


Рис. 2.2. Предыдущая запись на SD-карте в программе Gparted

Это важно!

В процессе подготовки SD-карты следует быть очень внимательным, чтобы не испортить жесткий диск компьютера, то есть, свою операционную систему. Перед продолжением операций удалите SD-карту и повторите вход в программу Gparted. Устройство sdb должно исчезнуть, что подтвердит – вы на правильном пути.

Переместитесь к */dev/sdb2* и используйте либо иконку удаления раздела, либо щелкните правой клавишей мышки, вызывая выпадающее меню с разделом *Удалить* (**рис. 2.3**).

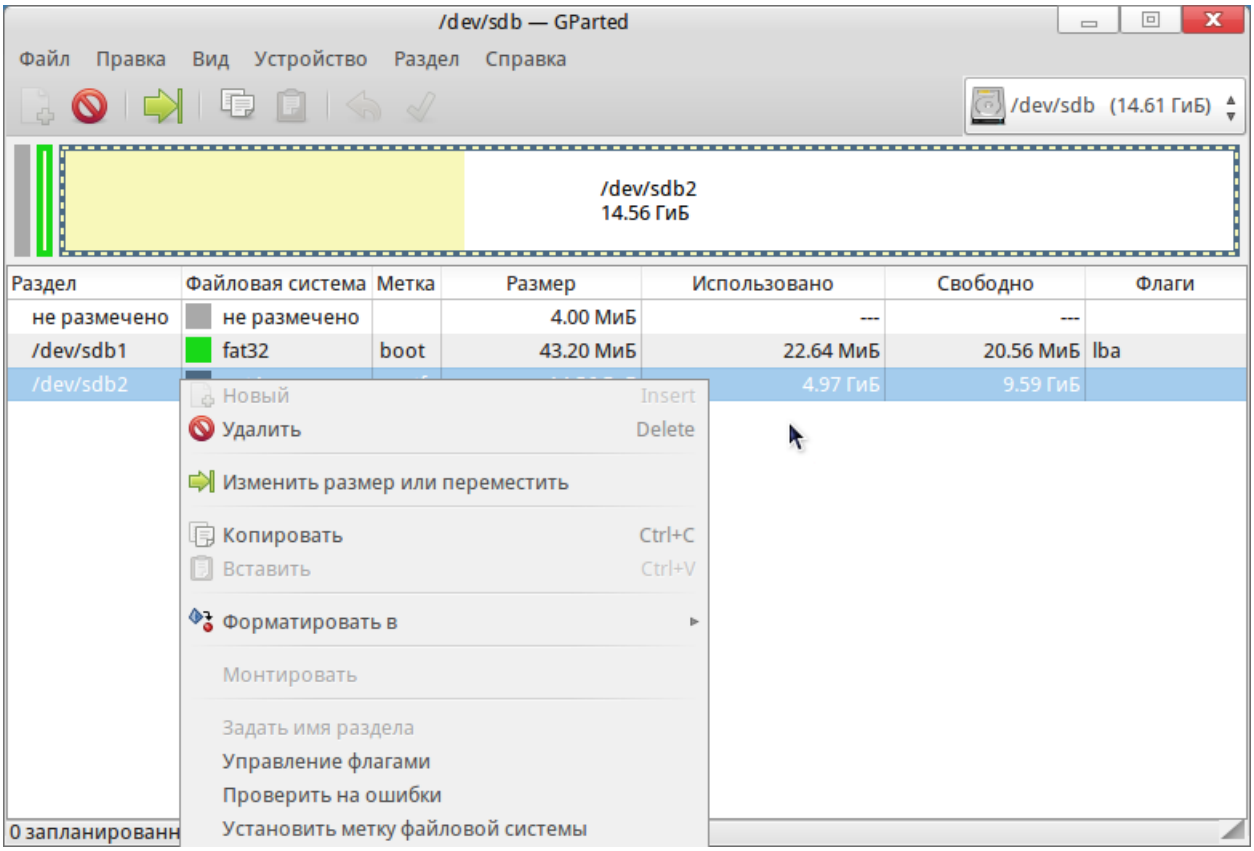


Рис. 2.3. Выпадающее меню с доступными командами

После этого удаления повторите процедуру для */dev/sb1*. Итогом должно стать получение единственного неразмеченного раздела (**рис. 2.4**).

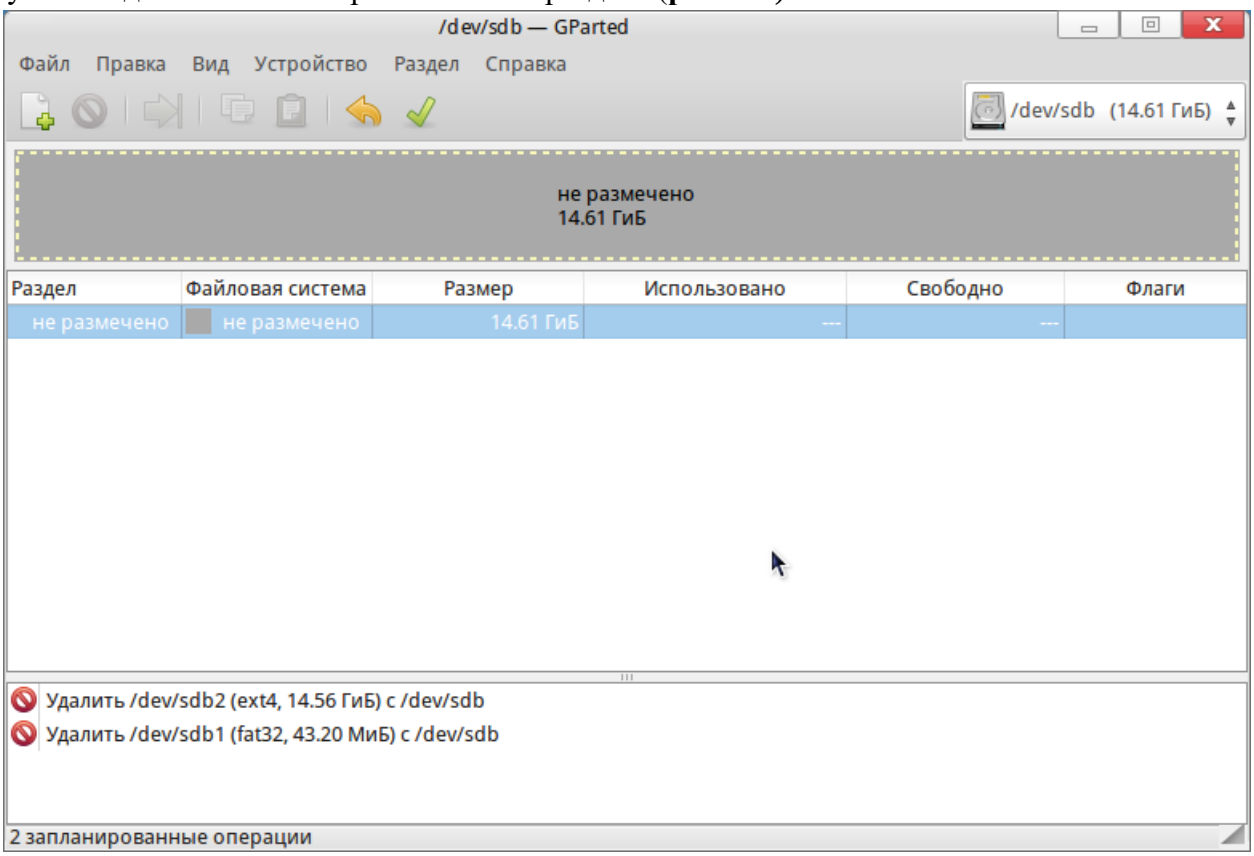


Рис. 2.4. Итоговый раздел на SD-карте

Почему же Linux?

Напомню, что операционная система Raspberry Pi создана на основе одного из дистрибутивов Linux. Чтобы познакомиться с этой операционной системой, можно использовать программу VirtualBox. С ее помощью можно установить любой дистрибутив Linux в операционной системе Windows. А нам предстоит использовать многое из того, что используется в Linux.

И мы еще не закончили с SD-картой.

Пока эти операции только запланированы, но они еще не выполнены. Если возникли сомнения, то можно их отменить, используя иконку со стрелкой возвращения. Теперь следует создать новый раздел. Задав нужные параметры, как показано на **рис. 2.5**, можно продолжить работу.

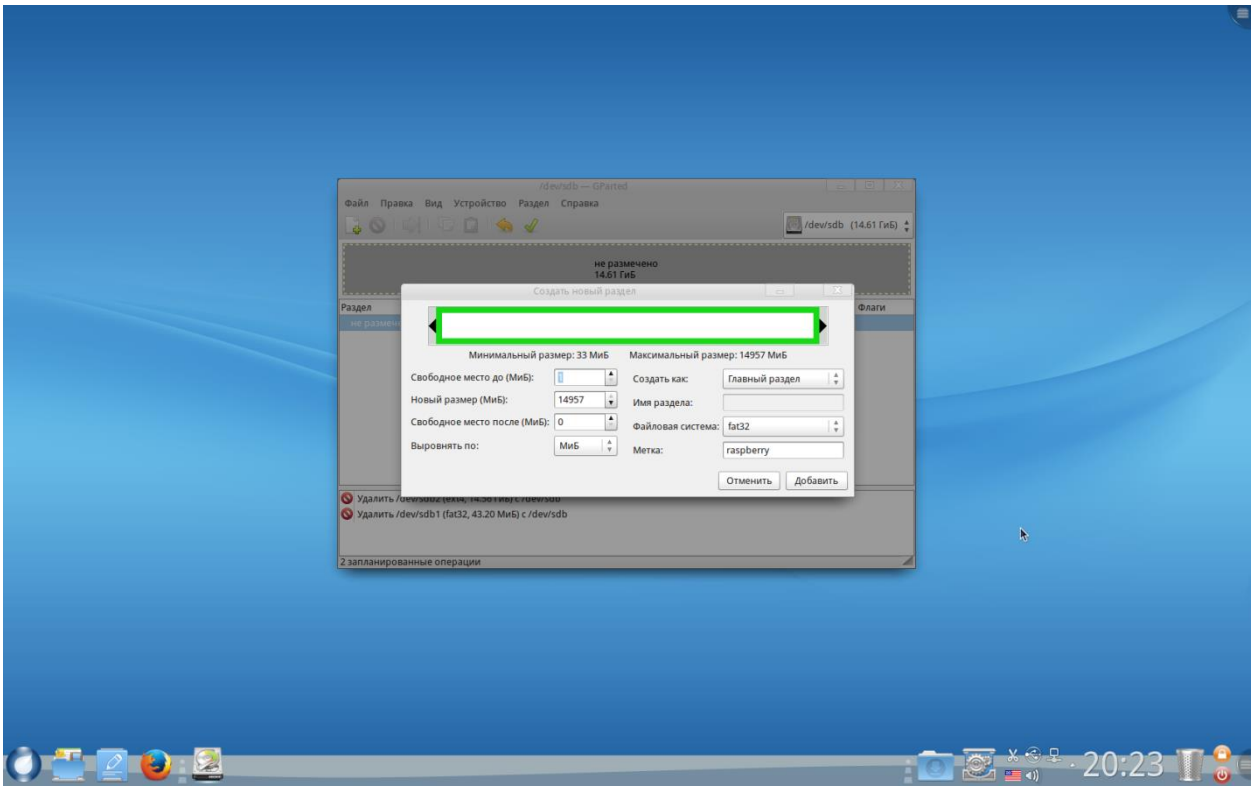


Рис. 2.5. Настройка формата SD-карты

Иконка с «галочкой» позволит применить все запланированные операции (**рис. 2.6**).

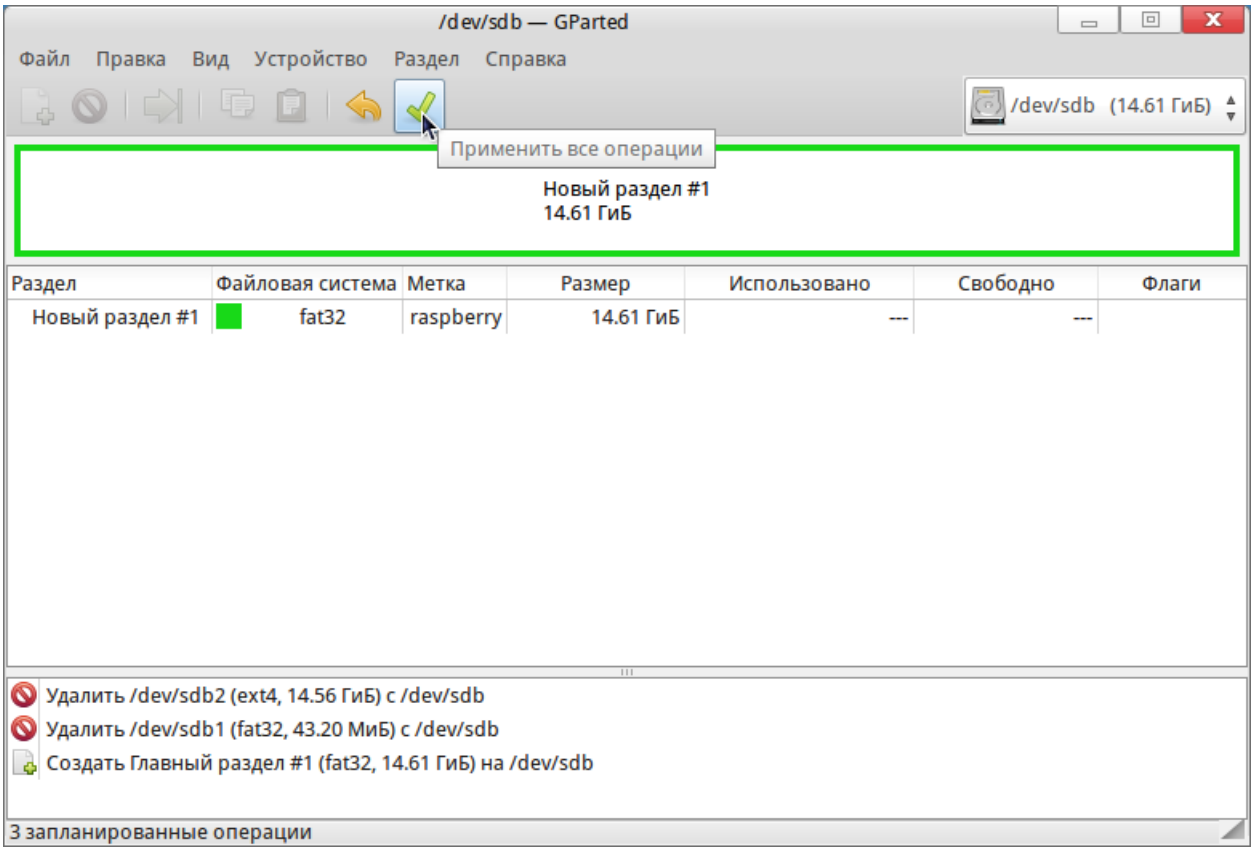


Рис. 2.6. Запуск выполнения всех запланированных операций

В списке операций будет удаление всех разделов, создание нового раздела в формате FAT32 и его форматирование. Операции занимают некоторое время, но следует дождаться завершения операций (**рис. 2.7**).

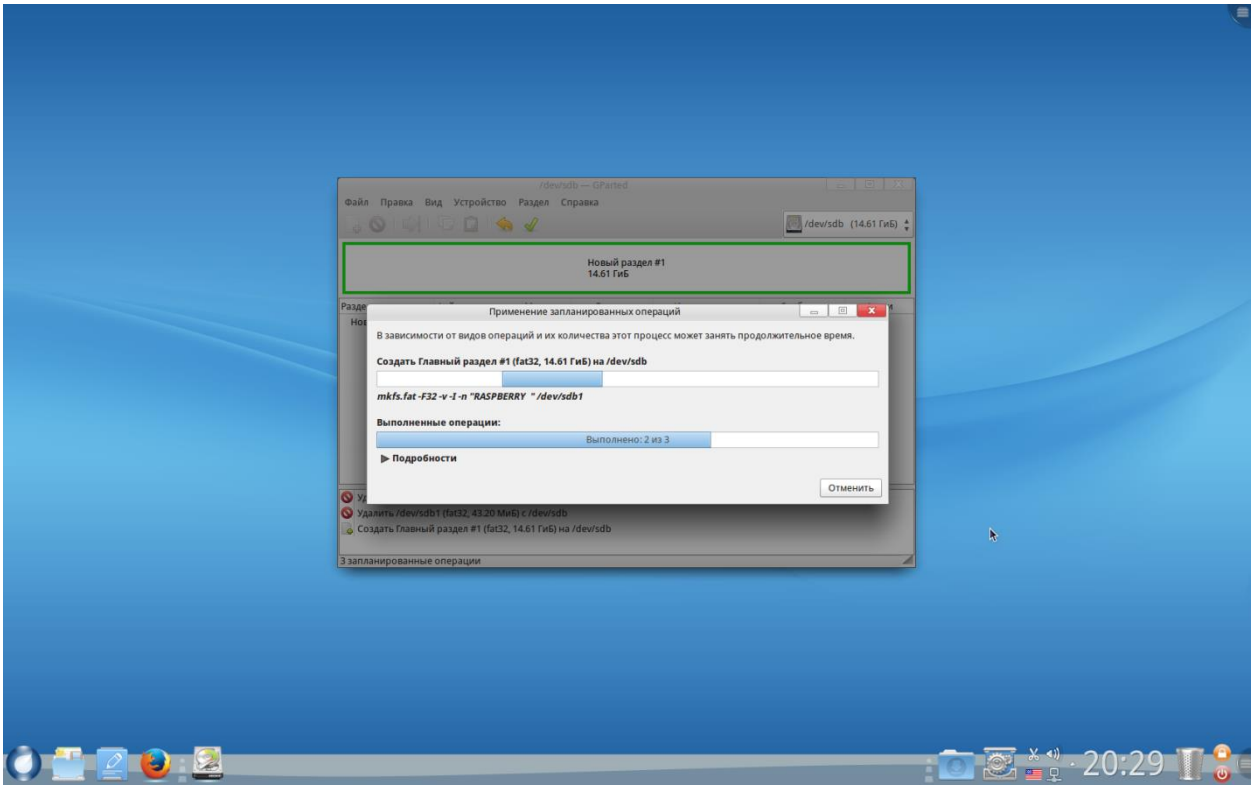


Рис. 2.7. Форматирование SD-карты

Завершаются подготовительные операции с появлением следующего вида диска sdb (**рис. 2.8**).

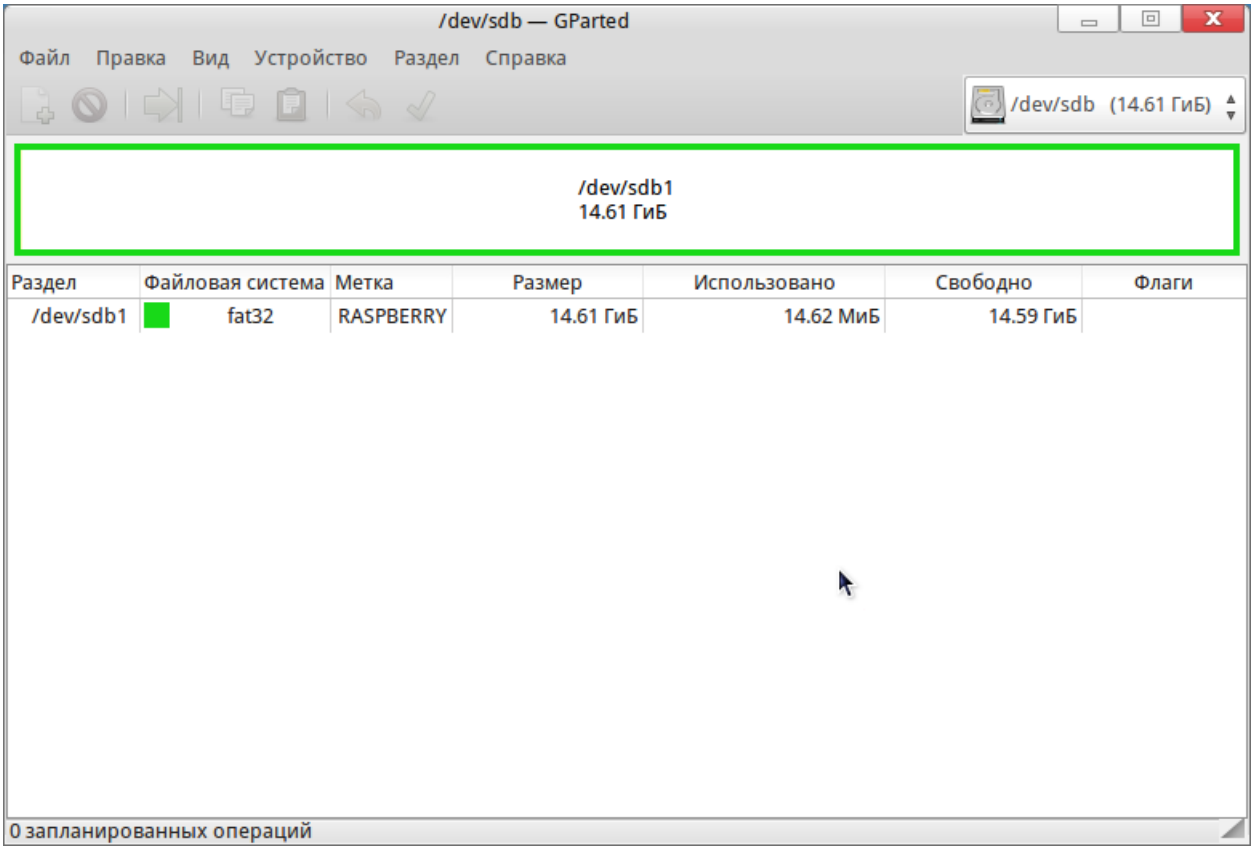


Рис. 2.8. Отформатированная SD-карта

Записать образ MajorDoMo на SD-карту в Linux по отзывам опытных пользователей проще, чем в Windows. Достаточно загрузить образ для своей модели Raspberry Pi, распаковать его и воспользоваться командой:

```
sudo dd if=~/путь/к/образу диска.img /dev/sdb1
```

У меня до записи образа дело не дошло. При распаковке файла с образом диска мой дистрибутив Rosa сообщил, что места на жестком диске для выполнения операции не хватает. Порой аналогичный вопрос возникает у меня и в Windows. Удалив закачанный файл, я возвращаюсь в Windows.

Подготовка использованной SD-карты в Windows

В Windows используется утилита управления дисками, которую можно в Windows 10 найти, если щелкнуть правой клавишей мышки по иконке меню. В окне управления дисками первыми будут показаны разделы жесткого диска. Нужно переместиться ниже, где появится SD-карта (**рис. 2.9**).

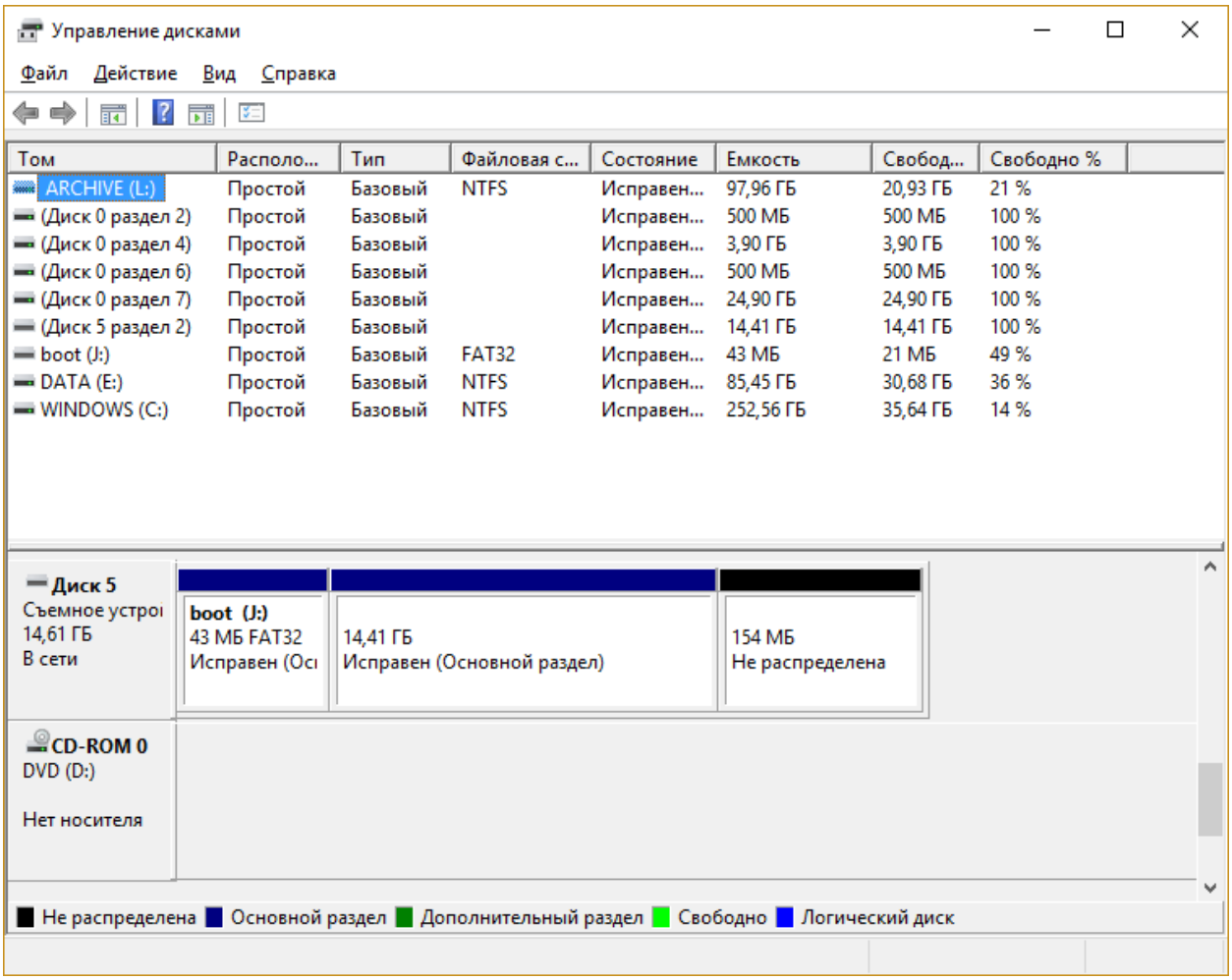


Рис. 2.9. SD-карта в окне управления дисками Windows

Если SD-карта чистая, то можно пропустить это описание. Однако, когда в процессе экспериментов операционная система будет испорчена, а такое может случиться, придется вернуться к описанным здесь операциям.

Это важно!

Еще раз напоминаю, будьте осторожны при работе с SD-картой, чтобы не испортить основную операционную систему!

Выделяя разделы и удаляя их (**рис. 2.10**), освободим SD-карту от предыдущей записи, что позволит провести установку MajorDoMo на отформатированную заново SD-карту.

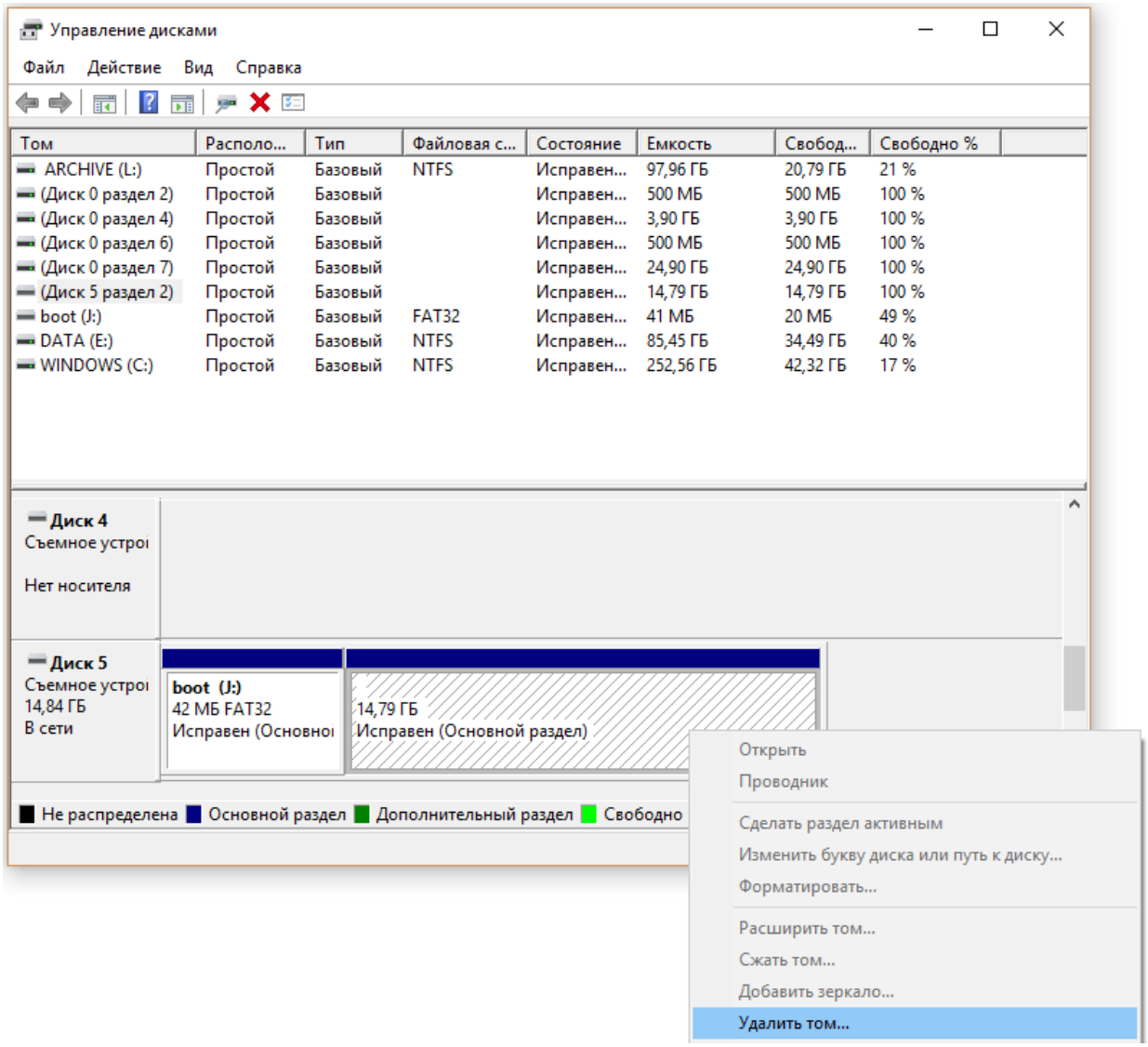


Рис. 2.10. Удаление ненужных разделов Sd-карты

После удаления разделов следует создать новый том, используя выпадающее меню, которое появится после щелчка правой клавишей мышки по свободному разделу. В диалоге создания *Простого тома* указать формат FAT32 и дать имя новому тому. На всякий случай я, как правило, выполняю еще команду *Сделать раздел активным*.

Для записи образа диска MajorDoMo его нужно скачать **[3]** для своей модели Raspberry Pi. В Windows используется утилита Win32DiskImger (**рис. 2.11**).

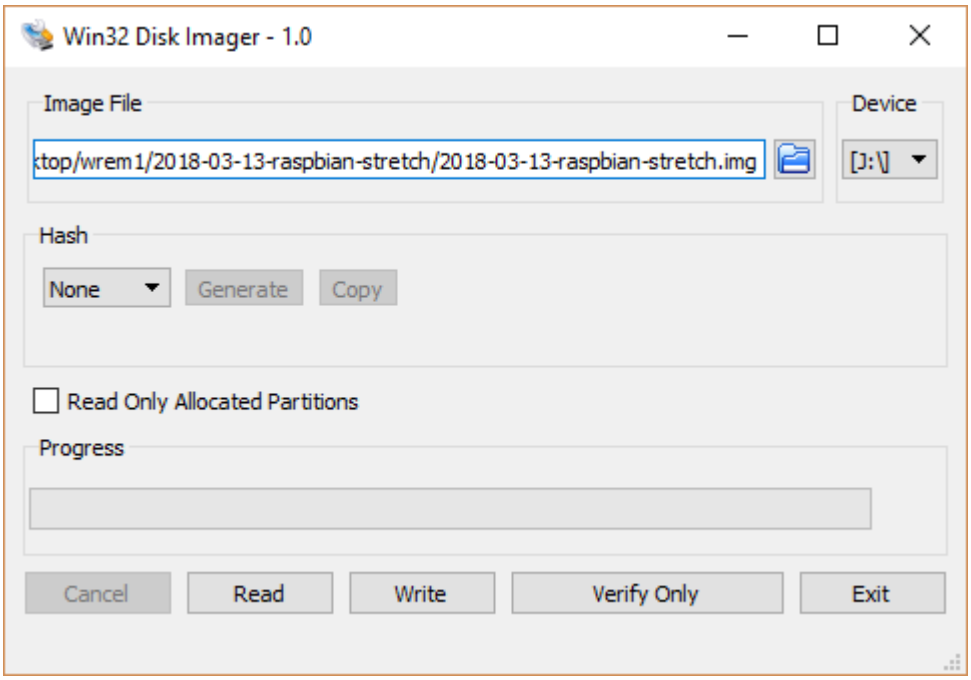


Рис. 2.11. Запись образа на SD-карту

Вход в систему MajorDoMo

В минимальной конфигурации Raspberry Pi самый простой способ определить IP-адрес, без которого нет смысла что-то делать, это заглянуть в настройки роутера.

Примечание.

Если у вас к Raspberry Pi подключена аудиосистема, то ваш ip-адрес вам сообщит информатор MajorDoMo Алиса.

Подключите модуль Raspberry Pi патч-кордом к роутеру. Включите Raspberry Pi. Когда загрузка завершится, в разделе клиентов DHCP должен появиться клиент majordomo. Запишите его IP-адрес. Теперь, если вы введете этот адрес в ваш web-браузер, то увидите следующее (**рис. 2.12**).

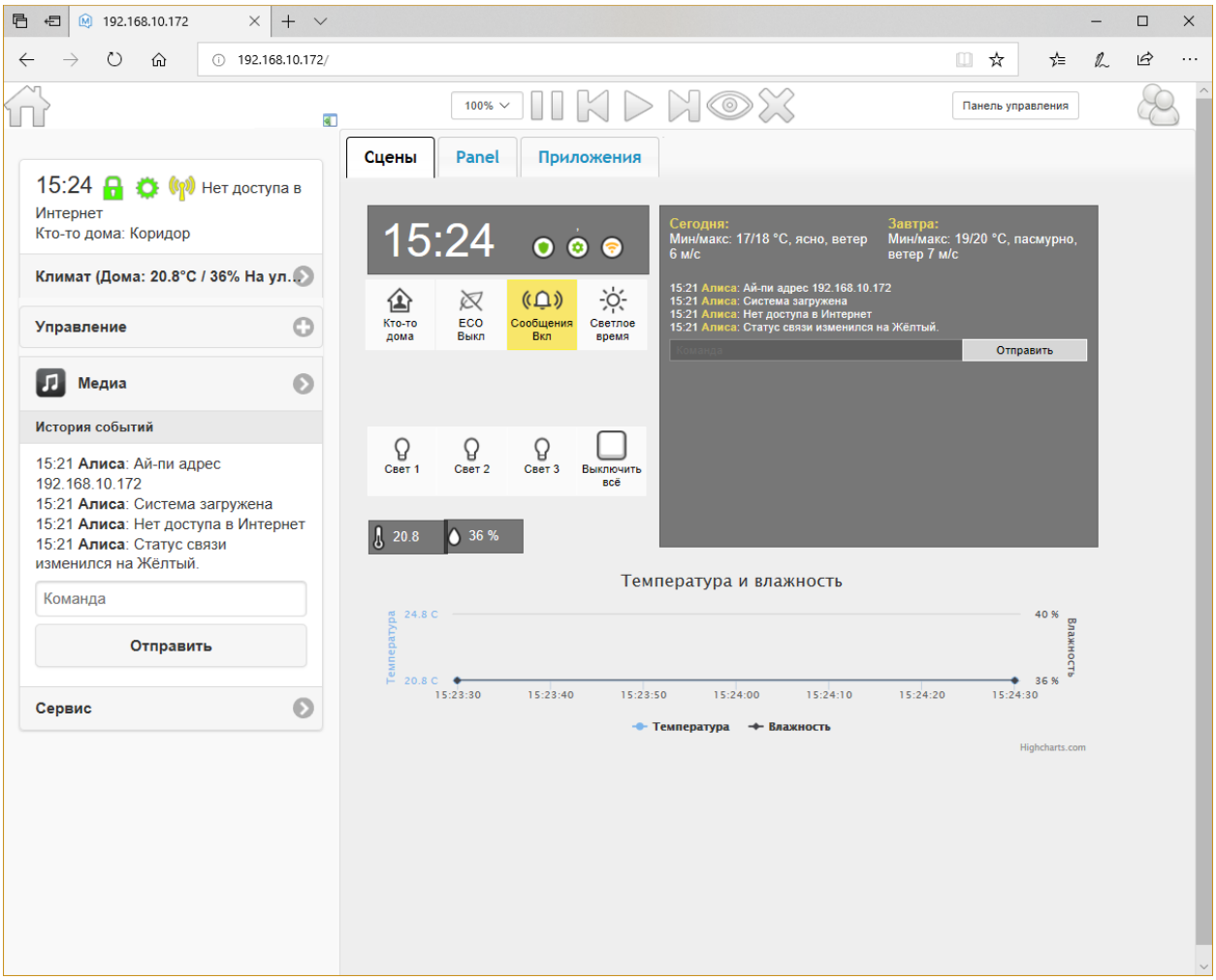


Рис. 2.12. Сервер MajorDoMo

При достаточном опыте этого достаточно для продолжения работы.

Мы не будем притворяться, поэтому попробуем обустроить операционную систему на Raspberry Pi для комфортной работы, к которой мы привыкли в других операционных системах.

Мы можем использовать программу PuTTY (программа свободная, ее можно скачать) для входа на Raspberry Pi (**рис. 2.13**).

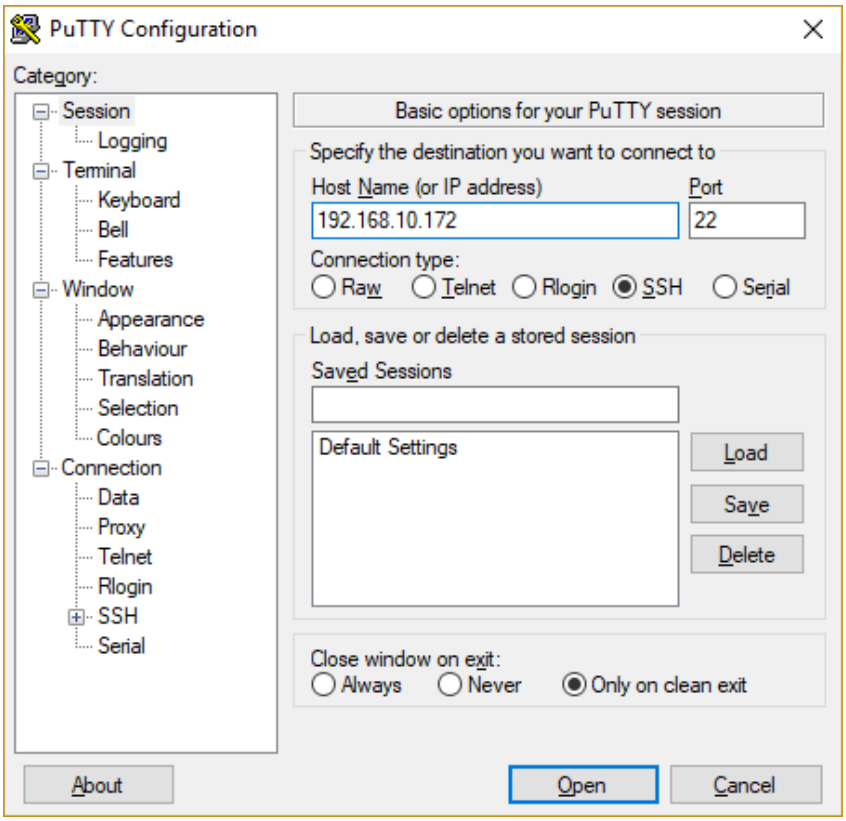


Рис. 2.13. Вход в операционную систему Raspberry Pi из программы PuTTY

Для входа потребуется ввести пользователя pi и пароль raspberry (**рис. 2.14**).

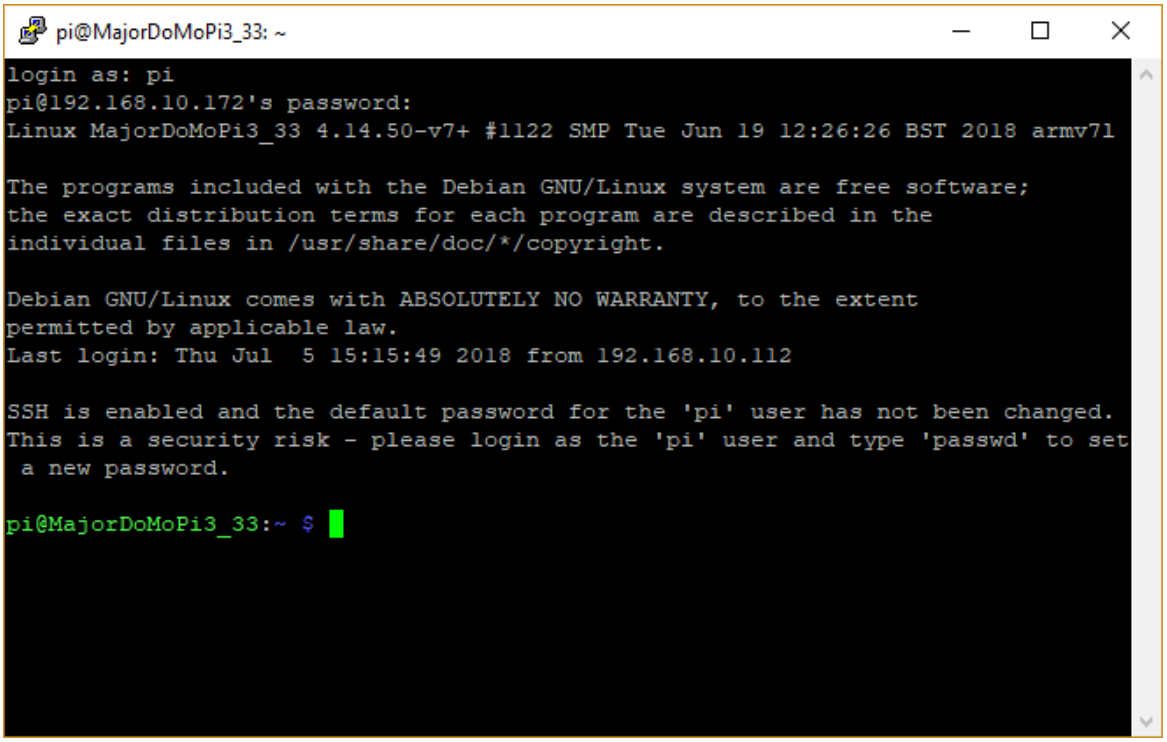


Рис. 2.14. Программа PuTTY в системе MajorDoMo

Это важно!

Перед тем, как начать опыты, я советую сделать резервную копию, как это описано в конце главы 6.

Глава 3. Избыточные настройки в MajorDoMo

Графическая оболочка

Для опытных пользователей MajorDoMo эти настройки, как я полагаю, не нужны. Но далеко не все, включая меня, столь опытны, чтобы использовать только сервер MajorDoMo или командную строку в PuTTY. Напомню, что MajorDoMo система «Умный дом».

Это важно!

*Если вы намерены превратить свой дом в умный, то лучшее решение – обратиться к специалистам, контактные данные вы найдете на сайте **[3]**.*

Но нас интересуют эксперименты.

Нам понадобится выход в Интернет, чтобы загрузить нужные пакеты программ. Попробуйте в командной строке PuTTY выполнить команду:

```
ping –c4 www.yandex.ru.
```

Зачем нам нужен Интернет, если мы будем использовать домашнюю компьютерную сеть? Через Интернет будет обновляться операционная система. Через Интернет мы будем устанавливать нужные нам приложения. Для этого нам и нужен Интернет. Продолжим.

Если команда не проходит, то выход в Интернет следует настроить. Проверьте командой `dpkg -l | grep resolv.conf` наличие нужного файла конфигурации. Он должен высветиться красным цветом в окне терминала PuTTY.

Примечание.

Программа PuTTY использует для связи протокол SSH. В образе диска MajorDoMo этот протокол уже включен. Иначе пришлось бы включить его, используя файл конфигурации Raspberry Pi.

Для исправления файла конфигурации можно воспользоваться редактором *nano*, а можно вызвать редактор надстройки программы Midnight Commander, которая, кстати, работает. Выполним команду `sudo mcedit /etc/resolv.conf`. В текстовом редакторе выполним добавление правильного IP-адреса сервера DNS, без которого соединение не возможно по привычному для нас адресу `www.yandex.ru` (**рис. 3.1**). После исправления проверим командой `ping` доступ к нужному ресурсу.

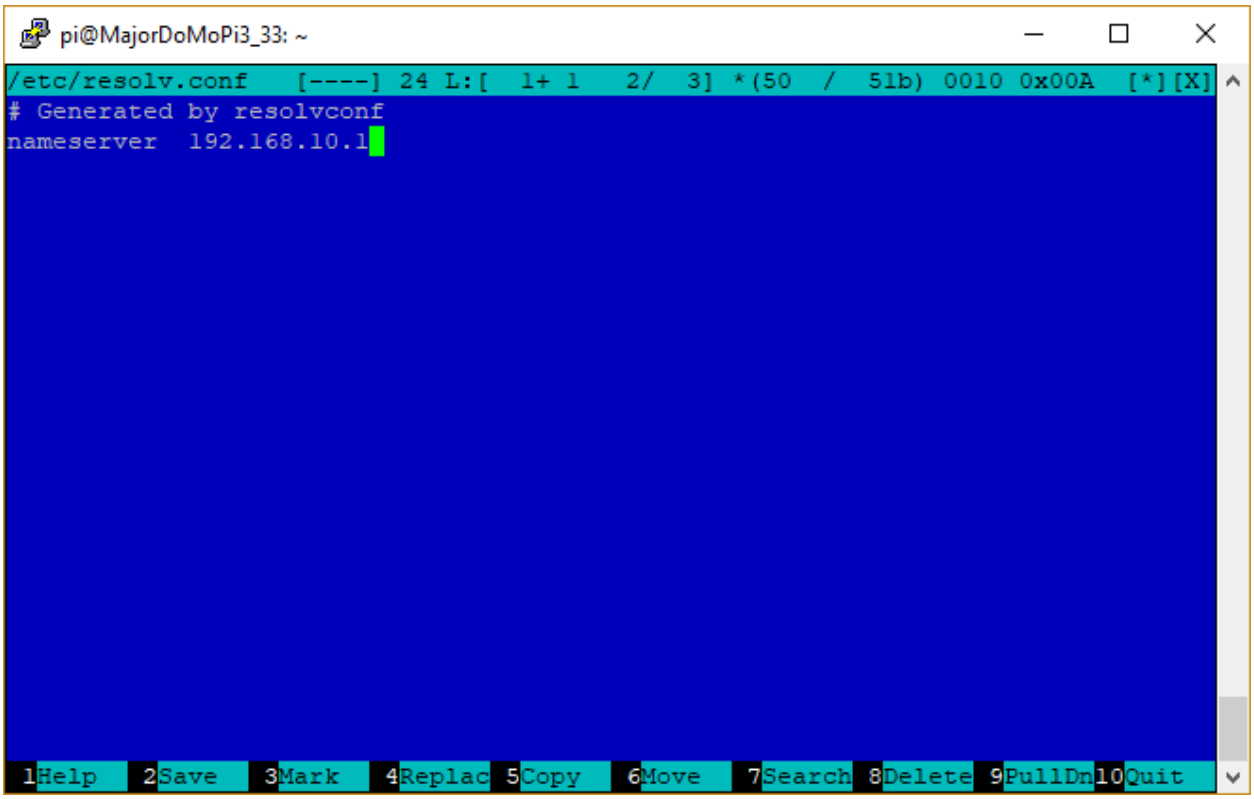


Рис. 3.1. Настройка конфигурации для выхода в Интернет

Сохраняется файл клавишей на клавиатуре **F2**, а для выхода из mc служит клавиша **F10**.

Примечание.

После перезагрузки Raspberry Pi файл конфигурации переписывается, приобретая исходный вид. Это связано с защитой вашего «Умного дома» от проникновения извне.

По совету «бывалых» сейчас нужно обновить всю систему. Это выполняется последовательными командами:

`sudo apt-get update` и `sudo apt upgrade`

По завершению обновления, что потребует времени, нужно перезагрузить Raspberry Pi:

```
sudo reboot
```

После перезагрузки зайдем на Raspberry Pi через PuTTY и проверим возможность обращения по имени ресурса командой `ping -c4 www.yandex.ru`.

Примечание.

-c4 в команде означает только повторение команды четыре раза. Попробуйте ее выполнить без этой добавки, остановить команду можно клавишами Ctrl+x на клавиатуре.

Вероятнее всего результат окажется таким (**рис. 3.2**).

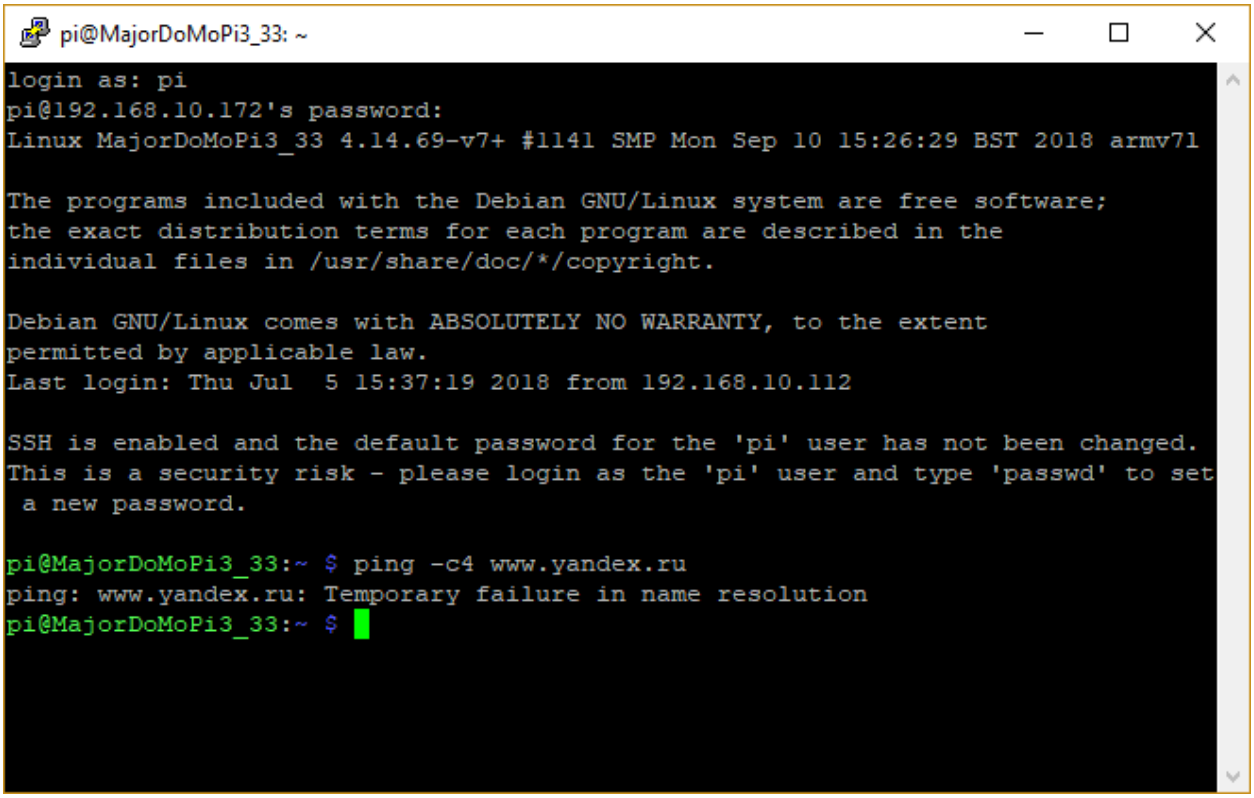


Рис. 3.2. Попытка обращения к Яндексу после перезагрузки

Поправим файл конфигурации еще раз. И мы убедились, что он переписывается после перезагрузки.

Поскольку мы только экспериментируем с Raspberry Pi MajorDoMo, мы будем обращаться к Интернету, видимо, неоднократно. Поэтому стабилизируем файл конфигурации, выполнив команду:

```
sudo chattr +i /etc/resolv.conf
```

Чтобы вернуться, если это будет нужно, к перезаписи файла нужно выполнить команду:

```
sudo chattr -i /etc/resolv.conf
```

Кстати, выключить Raspberry Pi, а это важно, чтобы не испортить операционную систему простым выключением питания, можно командой:

```
sudo shutdown –h now
```

Зачем эти сложности? Что мы хотим сделать?

В первую очередь постараемся зайти на Raspberry Pi с помощью VNC – программы, которая позволяет войти на компьютер с удаленного компьютера, но работать так же, как мы привыкли работать за своим компьютером. Программу VNC, она свободная, следует установить на Windows (введите в поисковике *vnc viewer* и скачайте программу). После ее запуска нужно ввести IP-адрес, чтобы зайти на другой компьютер (на наш Raspberry Pi,

который вне всяких сомнений компьютер). Запустив программу, используйте основное меню, раздел *File→New connection*. Но без дополнительных настроек результат, вероятно, будет такой (**рис. 3.3**).

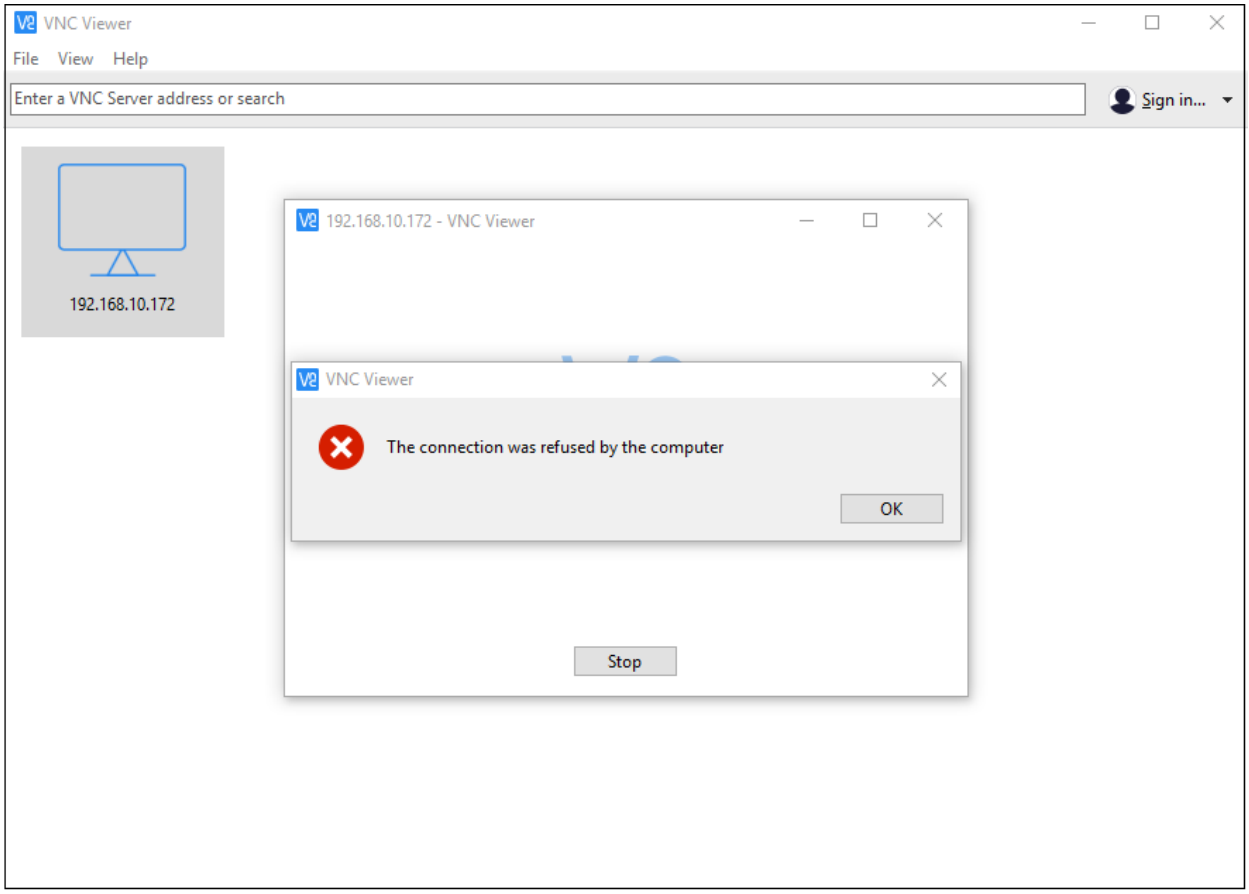


Рис. 3.3. Первая попытка использовать VNC вход на Raspberry Pi

Так, в принципе, можно зайти с любого компьютера на любой, но в нашем случае так можно зайти на любой компьютер домашней сети. Или, если Raspberry Pi подключен напрямую к Интернету, и вы знаете его ip-адрес, то можете зайти с домашнего компьютера через Интернет на Raspberry Pi. Но для этого…

Начнем решать задачу работы с VNC командой:

```
sudo apt-get install realvnc-vnc-server
```

Она должна установить сервер VNC на Raspberry Pi. Попытка зайти на Raspberry Pi, можете проверить, вновь неудачна. Попробуем настроить конфигурацию, разрешив вход через VNC с помощью команды:

`sudo raspi-config` (**рис. 3.4**).

Выберем раздел *Interfacing Options*, где есть *P3 VNC* разрешение на работу этого интерфейса. Подтвердим это (переход к нижней части с кнопками клавишей Tab клавиатуры, а перемещения стрелками на клавиатуре). Перезагрузим Raspberry Pi.

Вход на Raspberry Pi через VNC

Оказалось, что всей проделанной работы еще недостаточно для входа через VNC. Вновь входим с помощью PuTTY. Я не готов выяснять, почему ничего не получилось, поэтому следую советам и выполняю несколько установок:

```
sudo apt-get install --no-install-recommends xserver-xorg
sudo apt-get install --no-install-recommends xinit
sudo apt-get install lxde-core lxappearance
```

Но и после этих добавлений понадобилось загрузить дисплейный менеджер LightDM командой `sudo apt install lightdm`, выполнить `sudo apt-get update`, еще раз вернуться к файлу конфигурации `sudo raspi-config`, чтобы разрешить работу по VNC, перезагрузить Raspberry Pi.

И эти добавления дают результат (**рис. 3.4**). Пользователь остается *pi*, а пароль *raspberry*.

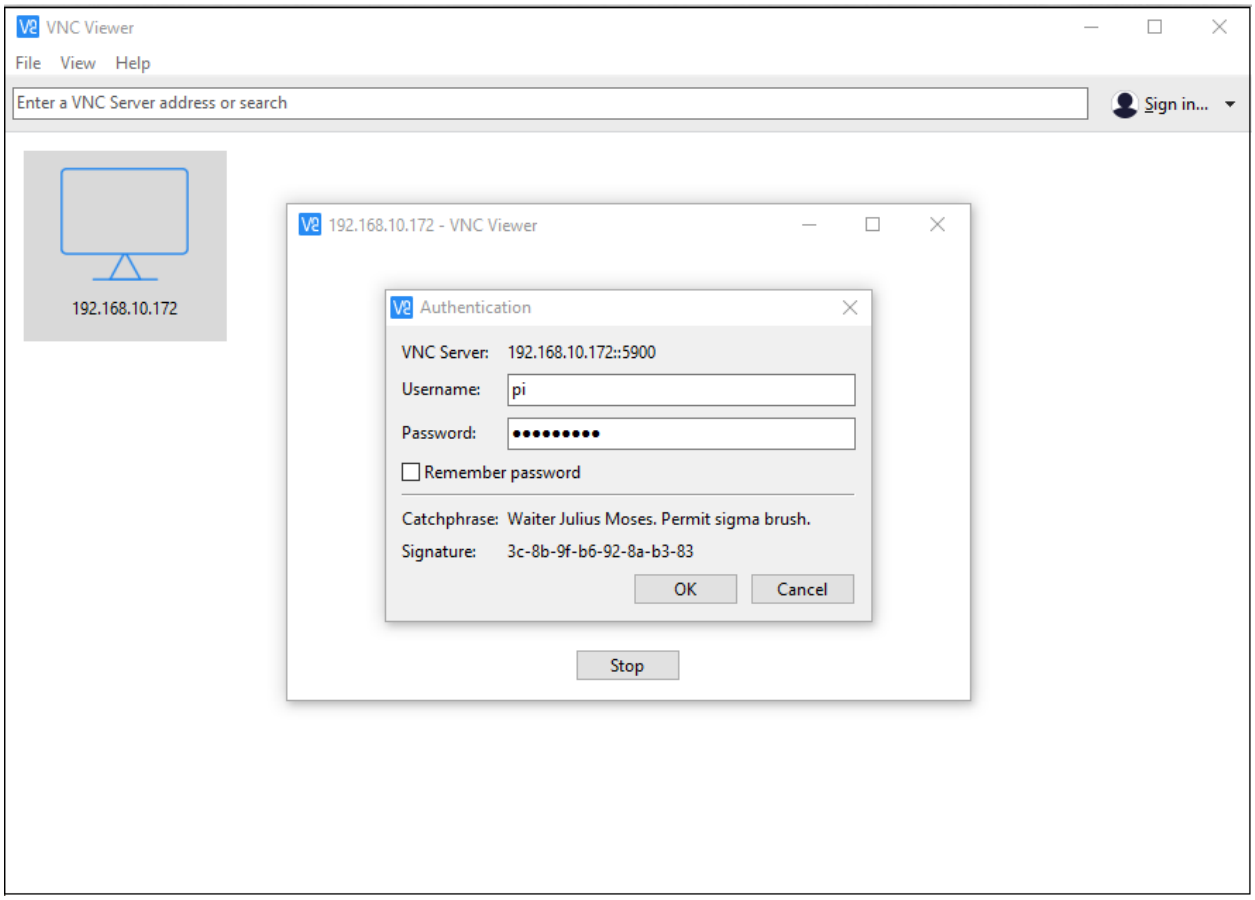


Рис. 3.4. Вход через VNC

Все ли это? Не уверен. Вот результат входа через VNC (**рис. 3.5**).

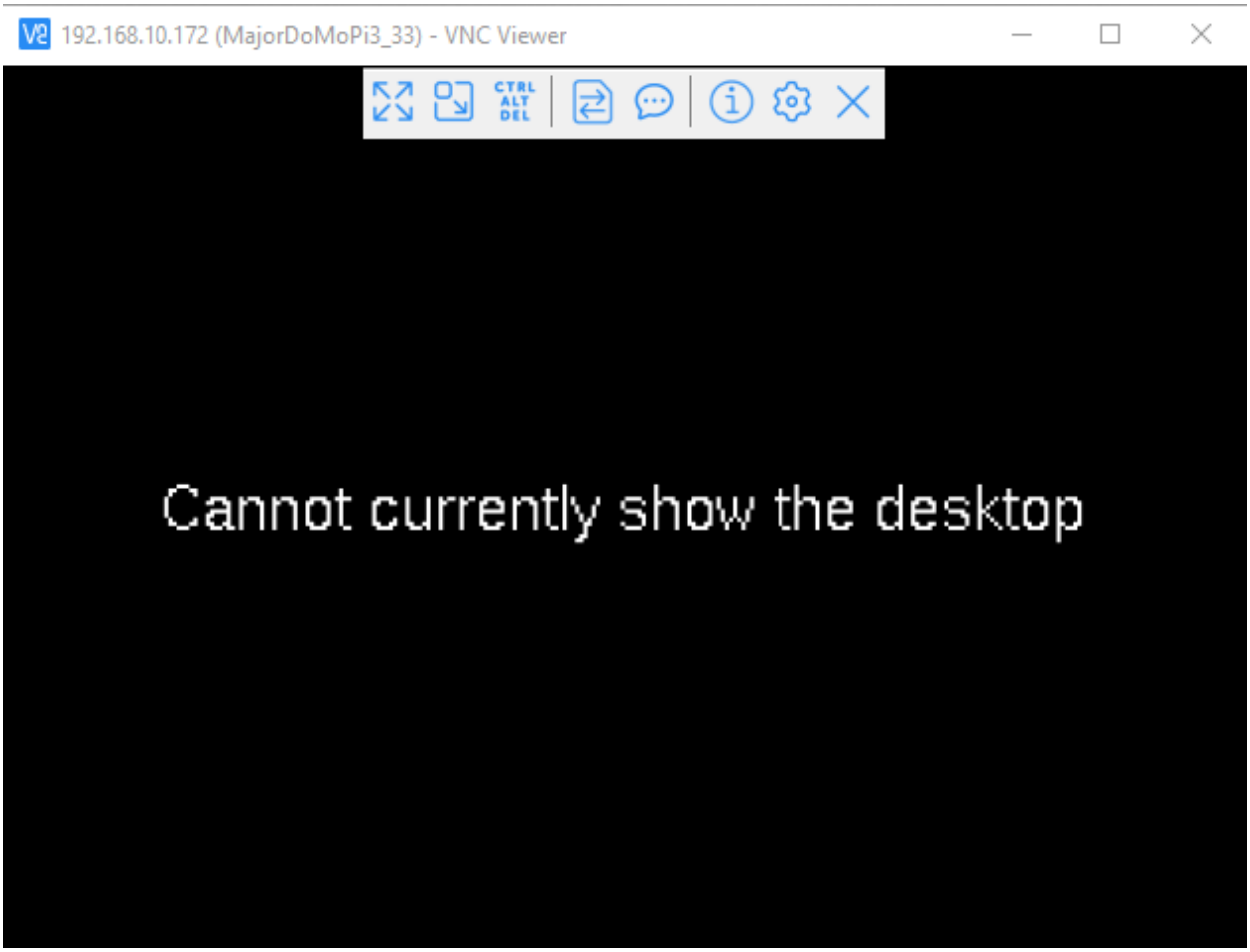


Рис. 3.5. Первый результат входа через VNC

И вновь повторим советы, которые нашлись в Интернете:

```
vncserver :1
netstat -pan|grep 'vnc'
```

Выполнение первой команды дает совет использовать IP-адрес 192.168.10.172:1. Но это, увы, тоже, похоже, не все: `sudo apt-get install lxde`.

По завершении этих операций, с этим адресом, наконец, появляется то, что хотелось бы видеть (**рис. 3.6**).

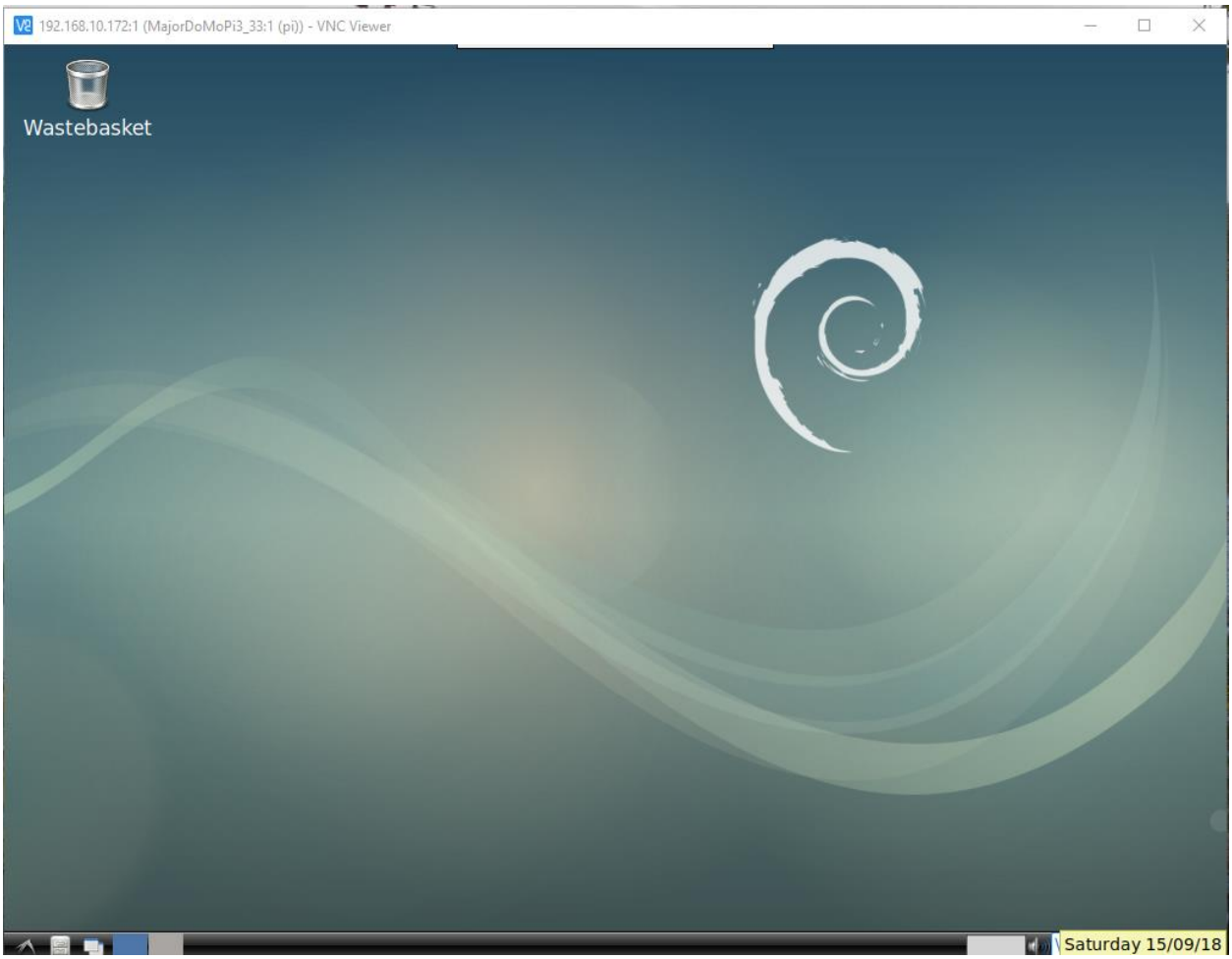


Рис. 3.6. Окончательный вид Raspberry Pi после надстройки

Сделав попытку выключить модуль с помощью меню графической оболочки, я не понял, выключил я или нет, повторил перезагрузку в PuTTY, но завершилось все тем, что пришлось просто выключить питание.

Следующий запуск Raspberry Pi дал возможность войти с помощью терминала, но не через VNC. Видимо, я не учел еще один совет – настроить автозагрузку графической оболочки при включении Raspberry Pi:

```
cd /home/pi/.config
mkdir autostart
cd autostart
nano realvnc.desktop
```

В редакторе записываются следующие строки:

```
[Desktop Entry]
Type=Application
Name=RealVNCServer
Exec=vncserver :1
StartupNotify=false
```

Для выхода из текстового редактора *nano* следует на клавиатуре выполнить последовательность: **Ctrl+o**, **Enter**, **Ctrl+x**. Затем следует перезагрузить Raspberry Pi. И это еще не все.

Нужна еще такая последовательность действий:

- зайти через SSH на Raspberry Pi (с помощью PuTTY);
- в домашней папке создать файл *vncserver* командой: `nano vncserver`;
- записать следующий скрипт:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          VNC
# Required-Start:    $local_fs
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start or stop the VNC server
### END INIT INFO

PATH=/sbin:/usr/sbin:/bin:/usr/bin
eval cd ~pi
case "$1" in
  start)
        su pi -c "/usr/bin/vncserver :1 -geometry 1024x728 -
depth 24"
        echo "Started VNC server."
        ;;
  stop)
        su pi -c "/usr/bin/vncserver -kill :1"
        echo "Stopped VNC server."
        ;;
  *)
        echo "Usage: vncserver [start|stop]" >&2
        exit 3
        ;;
esac
:
```

- закрыть редактор *nano*: **Ctrl+o**, **Enter**, **Ctrl+x**;
- выполнить команды по переносу файла и его атрибутам:

```
sudo mv vncserver /etc/init.d/
sudo chown root:root /etc/init.d/vncserver
sudo chmod 755 /etc/init.d/vncserver
sudo update-rc.d vncserver defaults
```

Теперь можно, обходя терминал, войти на Raspberry Pi через VNC.

Дополнения к избыточным настройкам

Началось все с простой проблемы – есть иконка выхода из операционной системы LXDE. Если на нее нажать, то появится очень красивое меню (**рис. 3.7**).

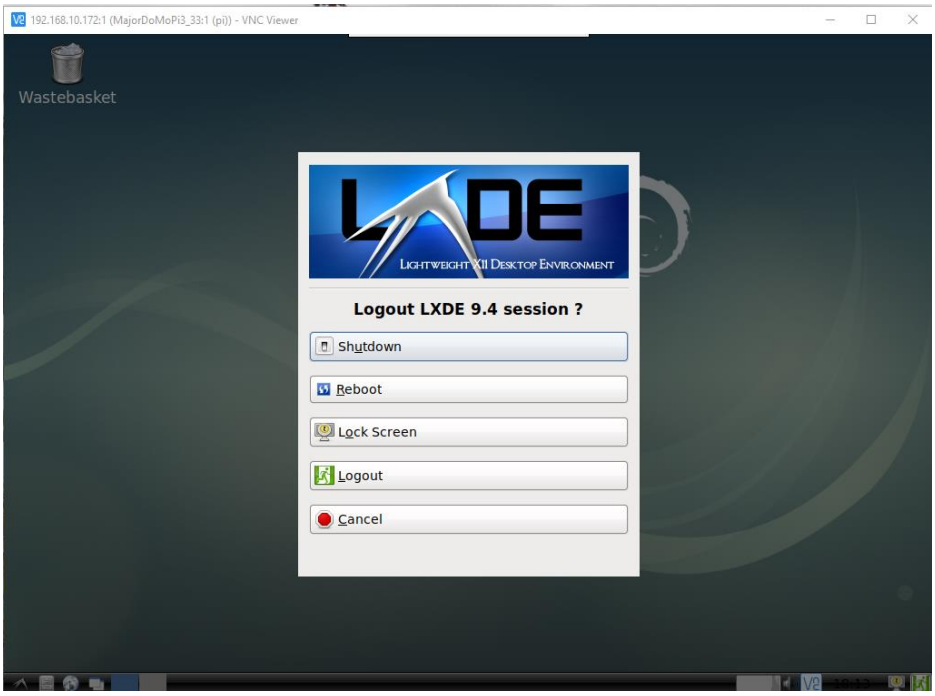


Рис. 3.7. Меню выхода из системы

Но вот беда, если нажать на кнопку Shutdown, то появляется не менее красивое сообщение (**рис. 3.8**).

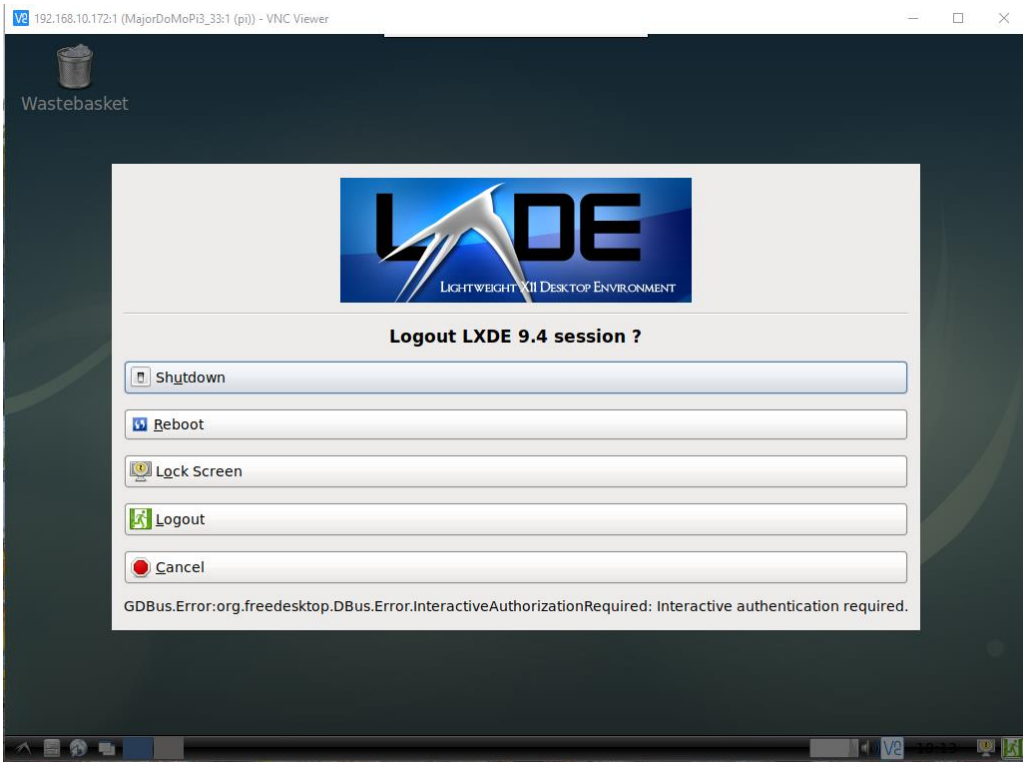


Рис. 3.8. Сообщение о необходимости авторизации в качестве администратора

И я не знаю, как авторизоваться. Несколько часов поиска решения проблемы в Интернете, несколько попыток последовать советам, результат нулевой. Я подозреваю, что, когда знаешь решение, эта проблема пустяк, однако решения я не знаю.

Попутно я выполняю несколько полезных установок:

`sudo apt-get install rc-gui` – устанавливается полезное меню настроек Raspberry Pi;

`sudo apt-get install leafpad` – устанавливается текстовый редактор;

`sudo apt-get install ark` – устанавливается архиватор;

`sudo apt-get install python3-thonny` – устанавливается среда разработки на Питоне;

`sudo apt-get install chromium-browser` – устанавливается web-браузер.

Последний (и архиватор) нужен мне, чтобы обойти проблему выключения. Есть, согласен простое решение – запустить терминал и ввести команду:

```
sudo shutdown -h now
```

Но для этого не нужна графическая оболочка.

А нужна ли она? Не нужна, если достаточно Midnight Commander, если опыт работы с этой оболочкой сохранился со времен MS DOS.

Иначе, используя web-браузер, скачаем пакет со страницы:

https://github.com/hakerdefo/lepo-logout.

Есть кнопка скачивания справа. Пакет называется *lepo-logout-master.zip*. Перед установкой загрузим еще одно необходимое дополнение, которое потребуется для нового меню выхода:

```
sudo apt-get install yad
```

После распаковки (прямо в папке Downloads) нужно выполнить несколько команд.

Сначала переместиться в папку, где лежит распакованная программа, затем выполнить еще одну команду:

```
cd /home/pi/Downloads/lepo-logout-master
sudo ./install-lepo-logout
```

После установки этой небольшой утилиты она появится в меню (**рис. 3.9**).

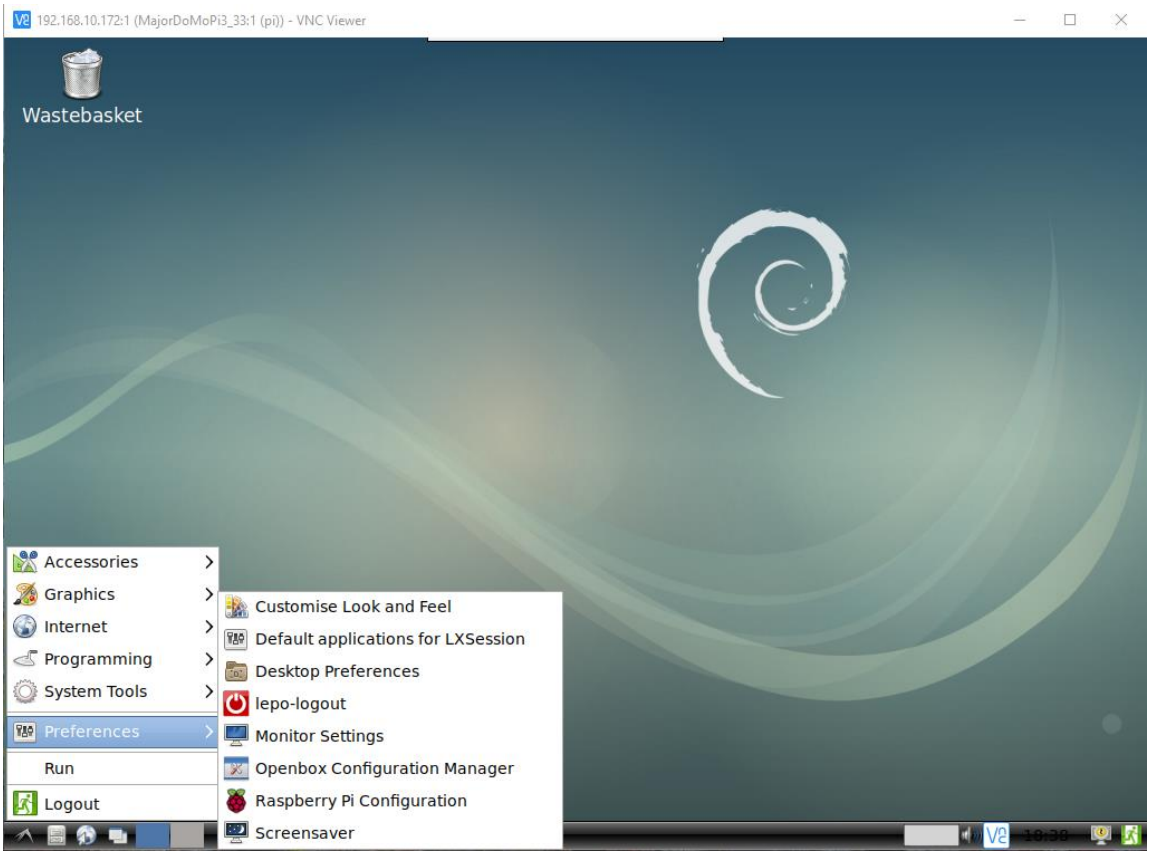


Рис. 3.9. Утилита выключения и перезагрузки Raspberry Pi в основном меню

Меню выхода из операционной системы выглядит не так красочно, но вполне понятно (**рис. 3.10**).

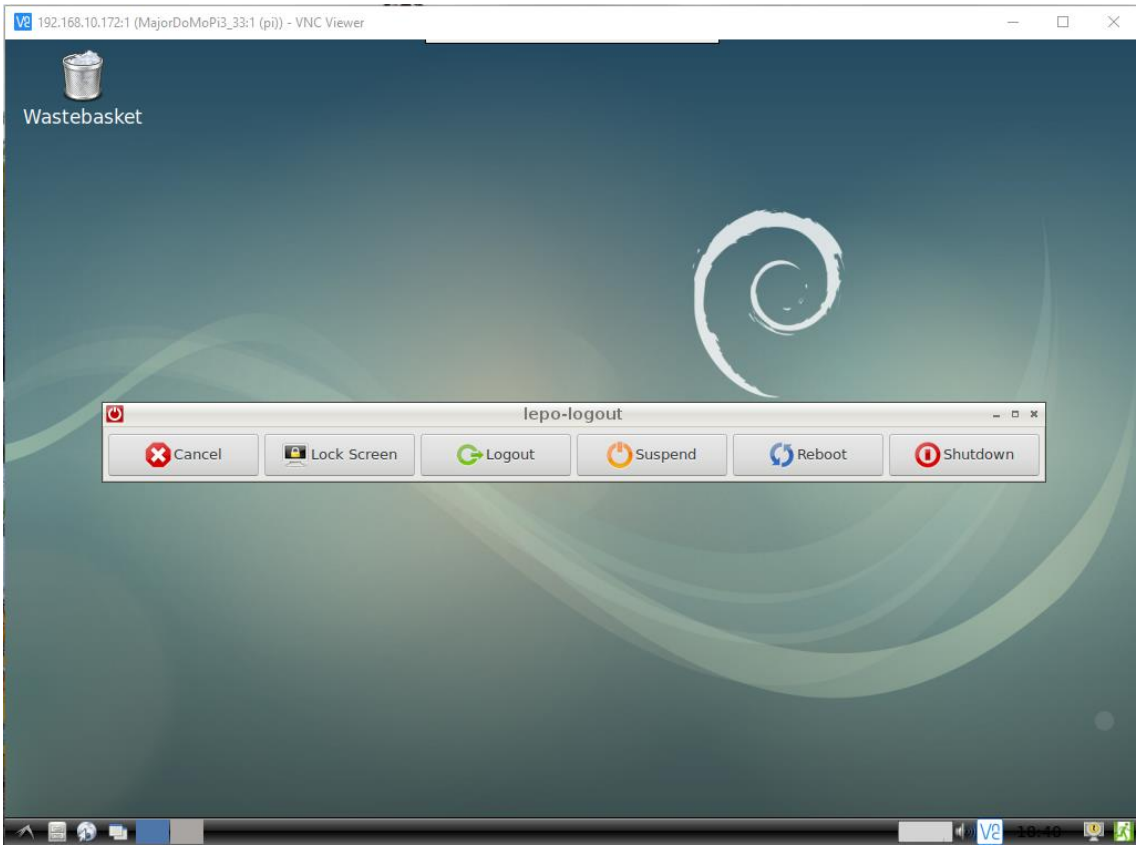


Рис. 3.10. Меню выхода из операционной системы

Теперь, если нажать кнопку Shutdown, появляется окошко, где можно ввести пароль для выхода.

Я не знаю, понадобится ли что-то из этих настроек в дальнейшем, но доведем настройки до конца. Используя меню настроек Raspberry Pi, выполним еще несколько изменений, используя закладку Localisation (**рис. 3.11**). Главная кнопка **Set Locale**, выбрав русский язык (кодировку я выбираю UTF-8), можно проверить остальные настройки, но их, возможно, не придется менять.

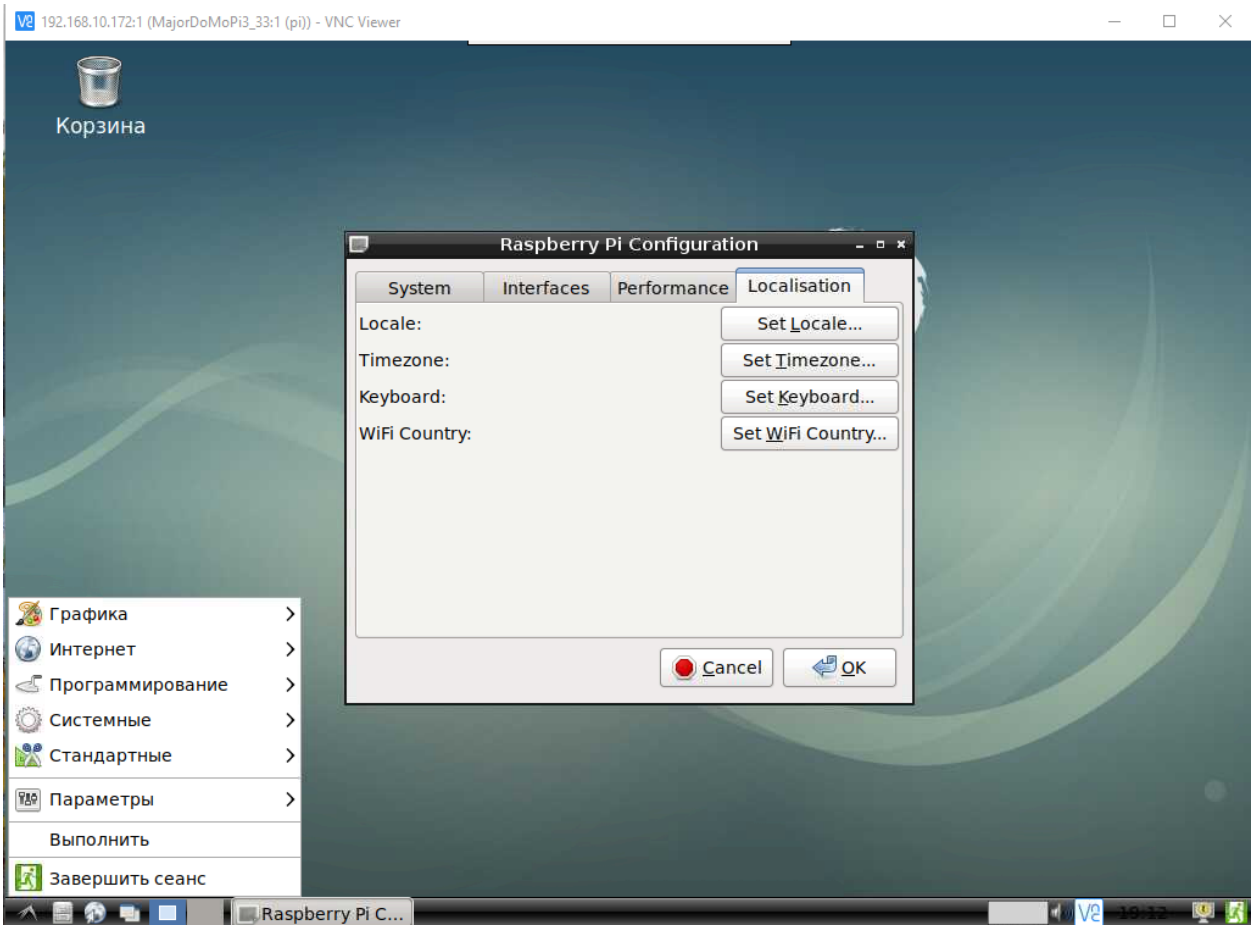


Рис. 3.11. Изменение настроек к русскоязычной версии

Если выключение и перезагрузка Raspberry Pi с помощью установленной программы покажется вам не самой удачной, то вы можете поступить иначе, благо основную работу мы будем выполнять в MajorDoMo: зайдите в раздел *Сервис*, выберите *Выключение системы* (**рис. 3.12**).

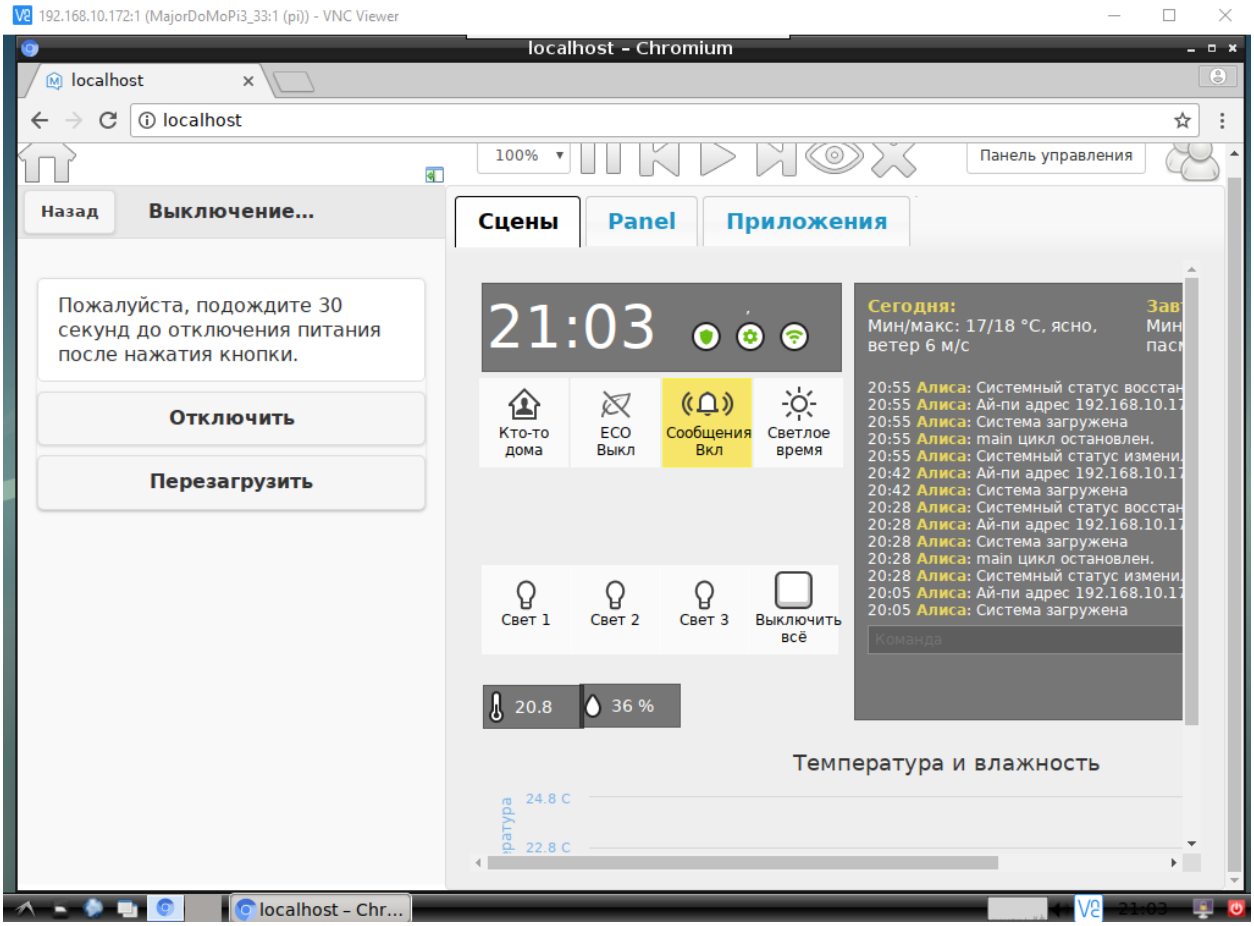


Рис. 3.12. Выключение Raspberry Pi из MajorDoMo

Это важно!

Операционные системы и программы часто обновляются. Не исключение, надо полагать, операционная система Raspberry Pi и MajorDoMo. Поэтому приведенные выше операции могут измениться или устареть. Об этом не следует забывать!

Глава 4. MajorDoMo – «Умный дом» или Intranet of Things

Знакомимся с MajorDoMo

Систему «Умный Дом» можно представить как компьютер, на котором работает программное обеспечение (у нас MajorDoMo) **[4]**, и к которому могут обращаться другие компьютеры, планшеты, смартфоны; который контролирует состояние ряда датчиков, принимая решение о необходимости включения и выключения бытовых приборов, таких как кондиционеры, обогреватели, осветители и т.д., следуя заданным нами условиям.

Почему «Умный дом», ведь компьютер – он компьютер и не более того?

Любой компьютер полезным и умным делает программное обеспечение, создаваемое умными людьми. Это люди делятся своим умом с компьютером, чтобы он стал нашим помощником. В данном случае помощь оказывает сервер Apache.

Сервер может работать на планшете или компьютере, а у нас он будет работать на миникомпьютере Raspberry Pi. Ему предстоит получать данные от различных датчиков и отдавать команды различным устройствам, о которых речь пойдет в главе 5.

Сервер для компьютера Raspberry Pi состоит из нескольких составляющих (обратимся к Wiki).

Справка.

Apache HTTP-сервер. Это он позволяет зайти на MajorDoMo из любого web-браузера.
MySQL – свободная реляционная система управления базами данных.
PHP – « препроцессор гипертекста»; первоначально Personal Home Page Tools, «Инструменты для создания персональных веб-страниц» – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений.

Есть еще один компонент – брокер (посредник) Mosquitto MQTT.

Справка.

Mosquitto – это открытый брокер сообщений, который использует MQTT версии 3.1 и 3.1.1. MQTT – протокол, применяемый для общения между устройствами. Этот протокол используется для передачи сообщений поверх протокола TCP/IP.

В нашем случае нет необходимости все это устанавливать, поскольку, записав образ диска на SD-карту, мы получим все необходимое. После включения питания Raspberry Pi можно зайти на страницу MajorDoMo со стационарного компьютера, планшета или смартфона (**рис. 4.1**). Нужно только знать IP-адрес.

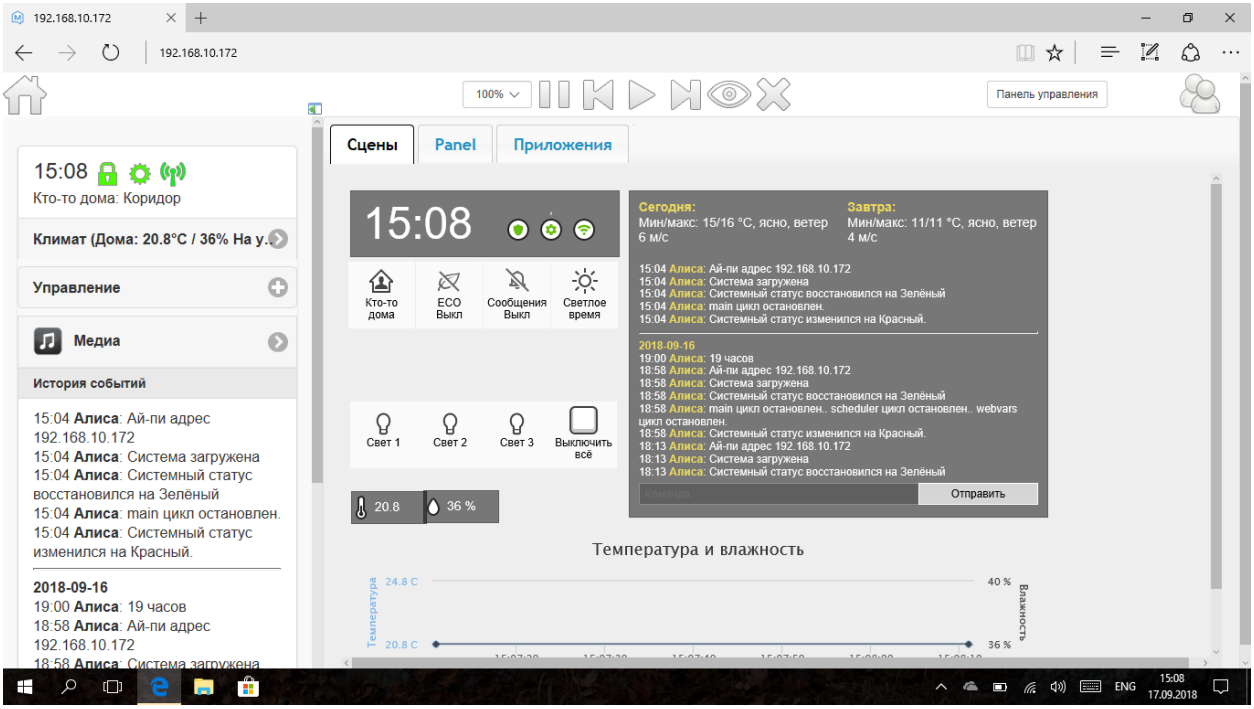


Рис. 4.1. MajorDoMo на планшете

Система MajorDoMo придерживается концепции объектно-ориентированного языка программирования: есть объекты, им присущи свойства, есть методы и функции.

Любая система «Умный дом» – это автоматизация рутинных операций по дому. Если домашние процедуры допускают полную автоматизацию, они осуществляются без участия человека, иначе используется ручное управление.

С другой стороны, IoT – система «Интернет вещей» – подразумевает общение разных «умных» устройств через Интернет. Умный помощник в этой системе следит за количеством, например, продуктов, а при их нехватке заказывает что-то в Интернет-магазине.

Поскольку формально общение устройств идет за счет работы в компьютерной сети, мы используем домашнюю сеть, имитируя систему IoT.

Примечание.

Интернет – это объединение множества компьютеров в единую сеть, которую некогда обозначили с помощью аббревиатуры «www», что в переводе означало «всемирную паутину». На деле Интернет объединяет компьютеры, а всемирная паутина – это коллекция web-страниц. HTTP – это протокол передачи гипертекста (HTML web-страниц), а HTML – это язык гипертекстовой разметки.

Сцена

Все события, происходящие в доме, можно считать действиями на сцене. Из чего следует, начать можно с создания своей сцены в MajorDoMo. Достаточно включить Raspberry Pi, зайти с помощью web-браузера с основного компьютера (так, вероятно, удобнее всего) в MajorDoMo, и начать создание сцены (**рис. 4.2**).

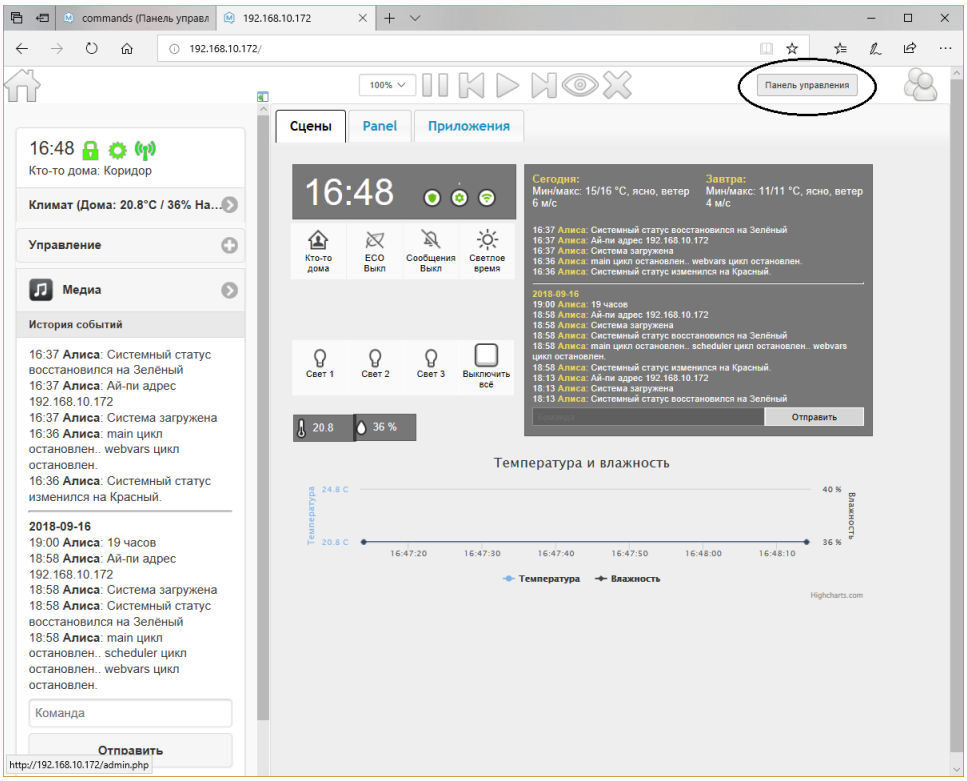


Рис. 4.2. Начало операций по созданию сцены

Нужно ли нам создавать сцену? Нет, но, если и когда это потребуется, мы будем к этому готовы, не так ли?

Все начинается с кнопки **Панель управления**. Этим совершается переход к новой странице, где есть объекты, есть управление, есть сцены (**рис. 4.3**).

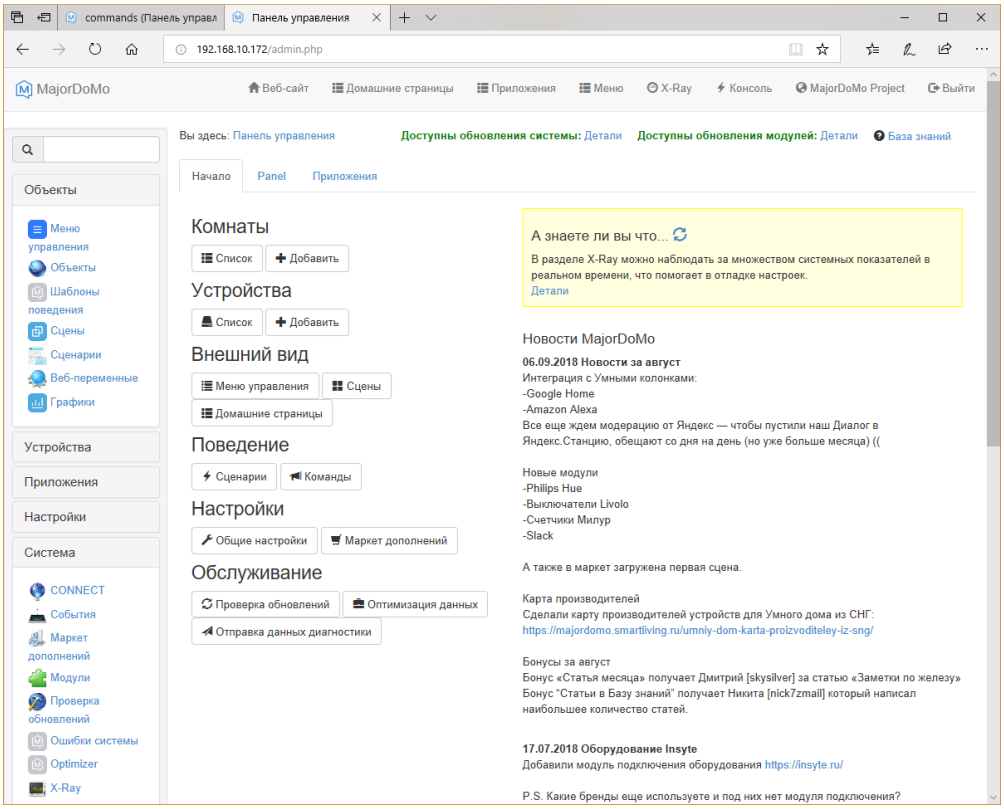


Рис. 4.3. Панель управления MajorDoMo

Выберем на панели слева пункт *Сцены*. Есть надежда, что там найдутся нужные нам средства для создания своей сцены (**рис. 4.4**). Так и получается – есть кнопка **+Добавить новую сцену**.

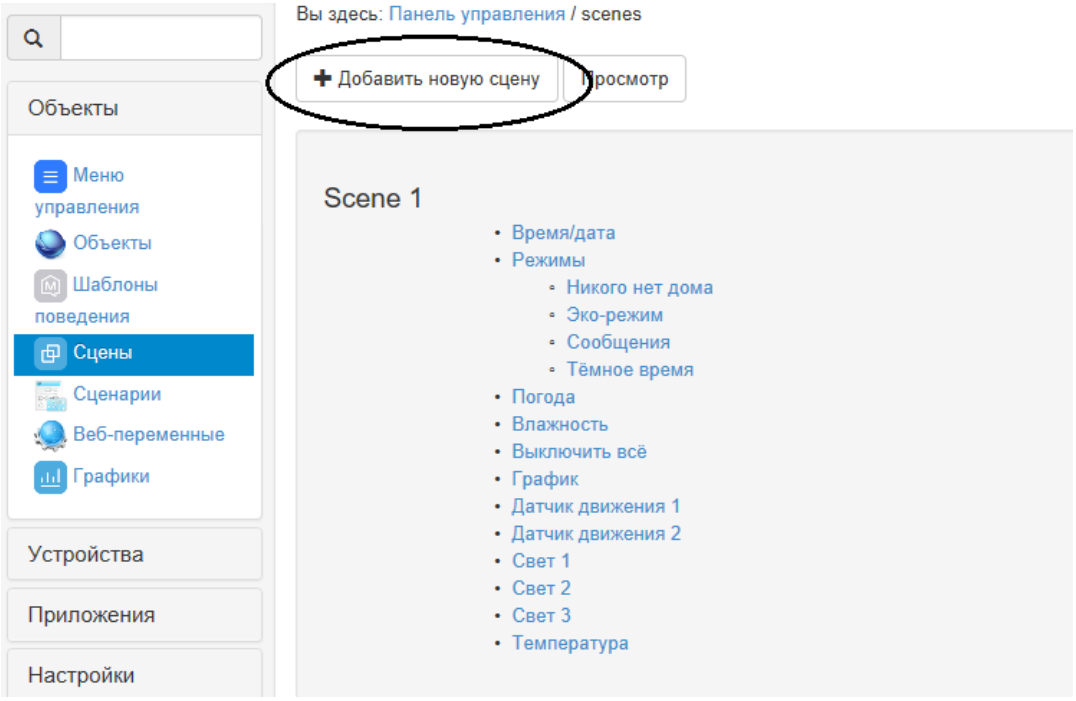


Рис. 4.4. Страница сцен

Одна сцена (**рис. 4.4**) уже есть, с нее начиналось наше знакомство. Не обладая достаточным опытом, сохраним ее. Если что-то будет непонятно, можно заглянуть в разделы этой сцены. Нажав кнопку **+Добавить новую сцену**, мы окажемся на странице заполнения данных для сцены (**рис. 4.5**). Для наших экспериментов предполагается использовать MajorDoMo, поэтому название можно вписать такое Scene 2 (зачем выдумывать велосипед?!).

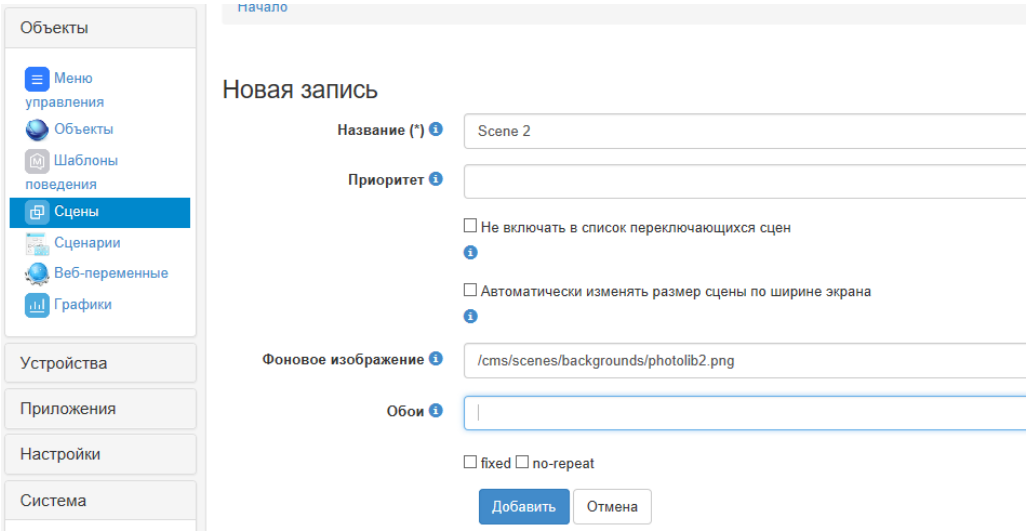


Рис. 4.5. Заполнение данных при создании сцены

Относительно фонового изображения – щелчок левой клавишей мышки в окне открывает проводник, где есть несколько папок, одна из которых называется backgrounds.

В ней можно выбрать подходящую картинку, но можно создать и свою картинку, добавив ее к существующим.

Кнопка **Добавить** покажет вид сцены (пока пустой) на следующей странице, а кнопка **Сохранить** сохранит наши настройки. Теперь при входе на сервер MajorDoMo будет две сцены. Вторую мы только что создали (**рис. 4.6**).

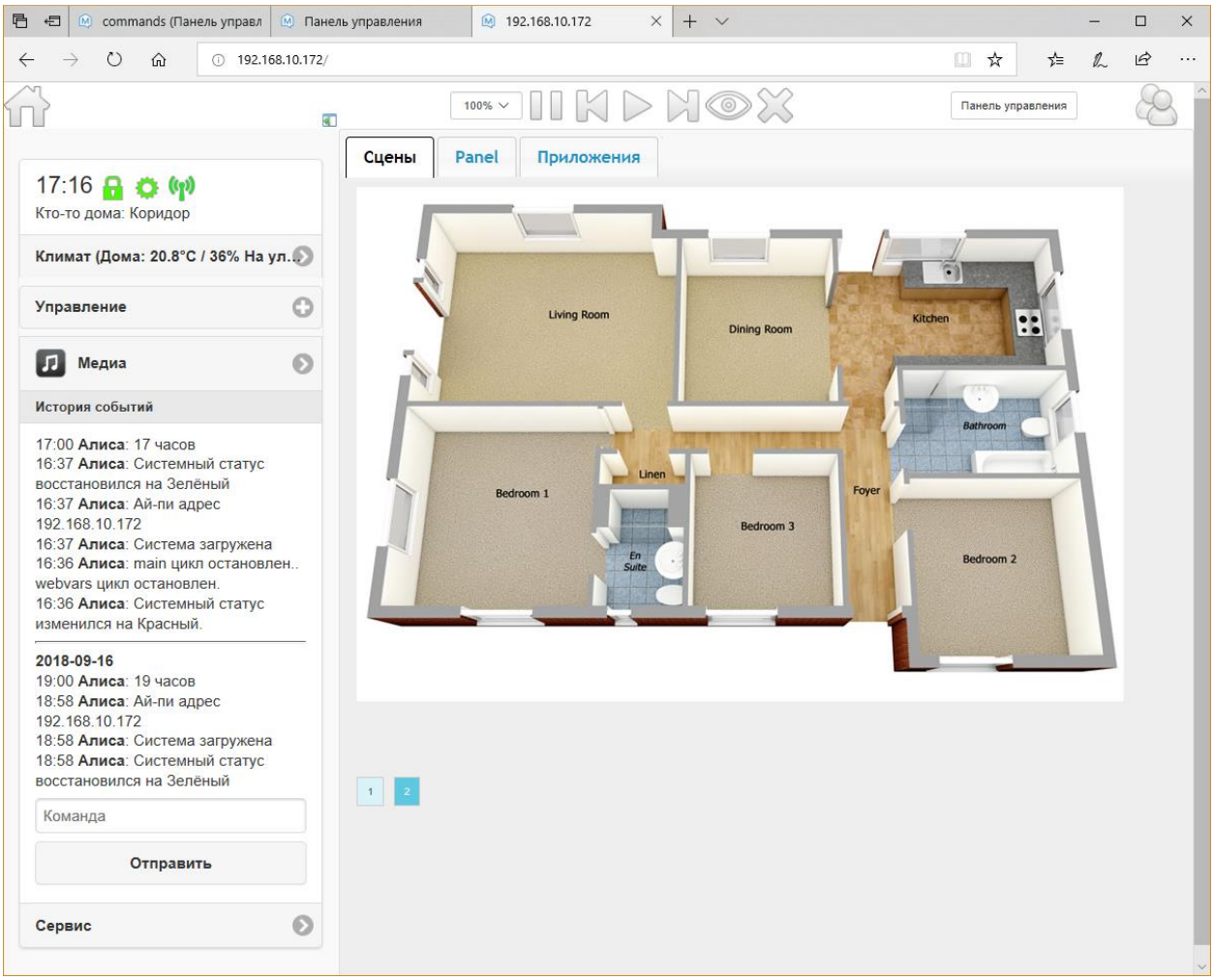


Рис. 4.6. Появление новой сцены

В разных комнатах можно разместить разные модули. Но сначала нужно с ними разобраться.

Глава 5. Модули для экспериментов в MajorDoMo

Arduino без надстройки Ethernet

Самым первым опробуем модуль Arduino, который знаком многим. Это удачный проект, помогающий не только делать первые шаги в программировании микроконтроллеров, но и создавать полезные устройства. Так недорогой вариант Arduino Nano – это небольшая плата, которую легко встроить вместе с другими элементами в подходящий корпус, и стоит этот модуль недорого.

Сам модуль Arduino подключить к компьютерной сети не получится, но я прочитал в базе знаний MajorDoMo **[5]**, что для операционной системы Windows есть приложение arduino_gw. Его можно скачать и использовать, приложение не требует установки, достаточно его запустить. И я хочу посмотреть, получится ли у меня использовать это приложение для подключения Arduino к Raspberry Pi?

Сам модуль Arduino, похоже, следует подготовить к работе в «Умном доме». Загрузим в Arduino рекомендованную программу:

```
int cycle_counter=0;

int old_garage=0;
int old_entry=0;
int old_movement_1=0;
int old_movement_2=0;

int old_button_1=0;
int old_button_2=0;

char buf[80];

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
}


void setup()
{

 pinMode(4, INPUT); // Датчик гаражной двери на 4 выводе
 old_garage=digitalRead(4);

 pinMode(5, INPUT); // Датчик въездных ворот на 5 выводе
 old_entry=digitalRead(5);

 pinMode(6, INPUT); // Датчик движения 1 на 6 выводе
 old_movement_1=digitalRead(6);
```

```
  pinMode(7, INPUT); // Датчик движения 2 на 7 выводе
  old_movement_2=digitalRead(7);

  pinMode(8, INPUT); // Кнопка пульта 1 на 8 выводе
  old_button_1=digitalRead(8);

  pinMode(9, INPUT); // Кнопка пульта 2 на 9 выводе
  old_button_2=digitalRead(9);


  Serial.begin(115200); // Скорость работы порта 115200
 }

 void loop()
 {

   int valid_sensor=0;

   delay(100); // задержка в 0.1 сек.
   cycle_counter++;

   if (cycle_counter==600) {
     cycle_counter=0;
   }

   //датчик гаражной двери
   Serial.println("G");
   int current_garage=digitalRead(4);
   //Serial.println(current_garage);
   if (current_garage!=(int)old_garage) {
     old_garage=(int)current_garage;
     sprintf(buf, "GET
/objects/?object=sensorGarage&amp;op=m&amp;m=statusChanged&amp;s
tatus=%i HTTP/1.0", (int)current_garage);
     sendHTTPRequest();
   }

  //датчик входа

   Serial.println("E");
   int current_entry=digitalRead(5);
   //Serial.println(current_entry);
   if (current_entry!=(int)old_entry) {
     old_entry=(int)current_entry;
```

```
      sprintf(buf, "GET
/objects/?object=sensorEntry&amp;op=m&amp;m=statusChanged&amp;st
atus=%i HTTP/1.0", (int)current_entry);
      sendHTTPRequest();
    }

   //датчик движения 1
    Serial.println("M1");
    int current_movement_1=digitalRead(6);
    //Serial.println(current_movement_1);
    if (current_movement_1!=(int)old_movement_1) {
      old_movement_1=(int)current_movement_1;
      sprintf(buf, "GET
/objects/?object=sensorMovement1&amp;op=m&amp;m=statusChanged&am
p;status=%i HTTP/1.0", (int)current_movement_1);
      sendHTTPRequest();
    }

   //датчик движения 2
    Serial.println("M2");
    int current_movement_2=digitalRead(7);
    //Serial.println(current_movement_2);
    if (current_movement_2!=(int)old_movement_2) {
      old_movement_2=(int)current_movement_2;
      sprintf(buf, "GET
/objects/?object=sensorMovement2&amp;op=m&amp;m=statusChanged&am
p;status=%i HTTP/1.0", (int)current_movement_2);
      sendHTTPRequest();
    }


  //кнопка управления 1
    Serial.println("R1");
    int current_button_1=digitalRead(8);
    if (current_button_1!=(int)old_button_1 &
current_button_1==1) {
    delay(2000);
    int current_button_1=digitalRead(8);
    if (current_button_1==1) {
     // long press
       sprintf(buf, "GET
/objects/?object=remoteButton1&amp;op=m&amp;m=statusChanged&amp;
status=%i HTTP/1.0", 2);
    } else {
     // click
```

```
        sprintf(buf, "GET
/objects/?object=remoteButton1&amp;op=m&amp;m=statusChanged&amp;
status=%i HTTP/1.0", 1);
     }
     old_button_1=(int)current_button_1;
       sendHTTPRequest();
     }

   //кнопка управления 2
     Serial.println("R2");
     int current_button_2=digitalRead(9);
     if (current_button_2!=(int)old_button_2 &
current_button_2==1) {
     delay(2000);
     int current_button_2=digitalRead(9);
     if (current_button_2==1) {
      // long press
        sprintf(buf, "GET
/objects/?object=remoteButton2&amp;op=m&amp;m=statusChanged&amp;
status=%i HTTP/1.0", 2);
     } else {
      // click
        sprintf(buf, "GET
/objects/?object=remoteButton2&amp;op=m&amp;m=statusChanged&amp;
status=%i HTTP/1.0", 1);
     }
     old_button_2=(int)current_button_2;
       sendHTTPRequest();
     }
     delay(4000); // пауза только для эксперимента
   }
```

Программу для Arduino пришлось немного подправить.

Это важно!

Если к модулю Arduino подключены датчики, имеющие подтягивающие резисторы к общему проводу (GND), то программа будет работать правильно, без подключенных датчиков открытые входы будут давать случайные значения.

После запуска шлюза arduino_gw введем данные в окна настройки и посмотрим на результат (пауза в 4 секунды в программе Arduino для опытов) (**рис. 5.1**).

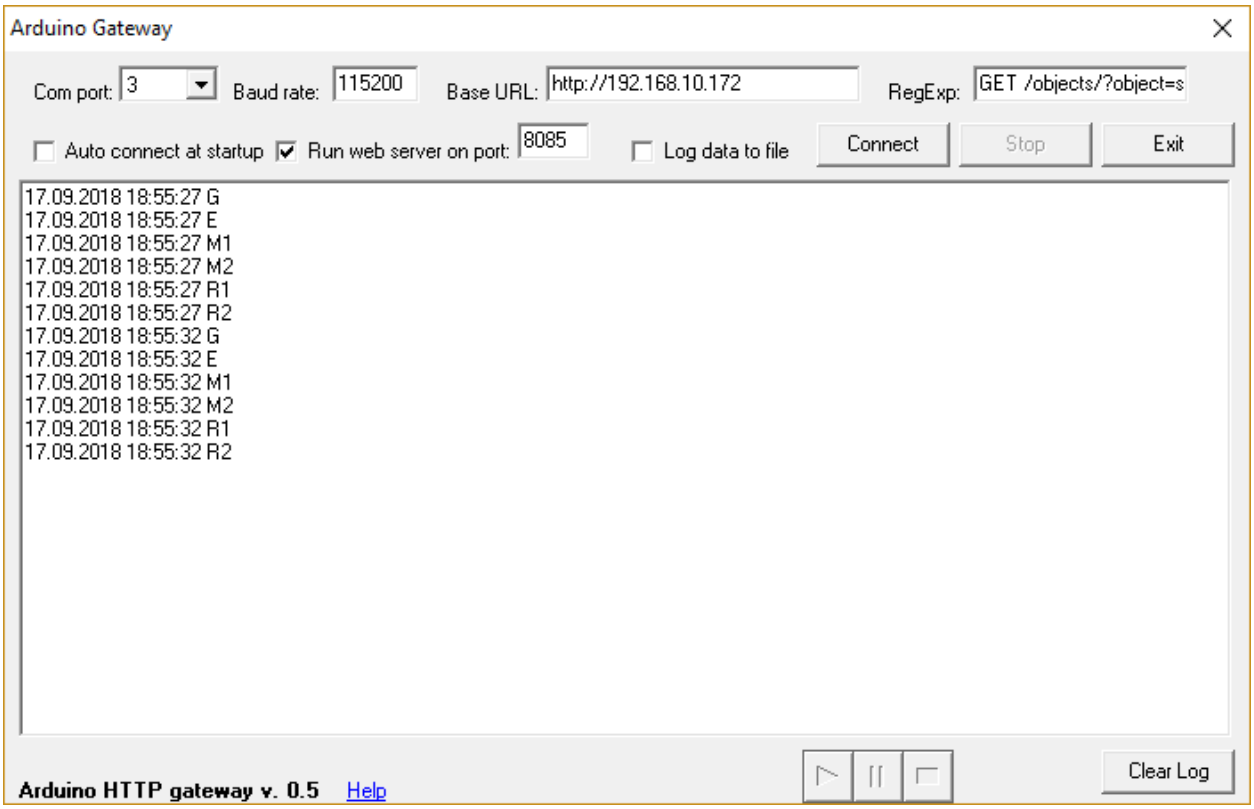


Рис. 5.1. Первый опыт с Arduino и шлюзом arduino_gw

По умолчанию и скорость в этой программе другая, и адрес сервера другой.

Программа для Arduino работает, так, но есть ли связь с MajorDoMo? Это еще предстоит проверить. Начнем с того, что упростим программу:

```
int garage=0; // Переменная для чтения состояния датчика
char buf[80]; // Массив для строки запроса

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
}

void setup()
{
 pinMode(4, INPUT); // Датчик гаражной двери на 4 выводе
 Serial.begin(115200); // Скорость работы порта 115200
}

void loop()
{
 garage=digitalRead(4); // Читаем состояние датчика
  sprintf(buf, "GET
/objects/?object=garage&op=m&m=status_input&status=%i HTTP/1.0",
(int)garage); // Записываем строку запроса
```

```
sendHTTPRequest(); // Отправляем запрос на сервер
delay(2000); // Задержка в 2 сек.
}
```

В таком виде с программой работать легче, но теперь она дает несколько иной результат (**рис. 5.2**).

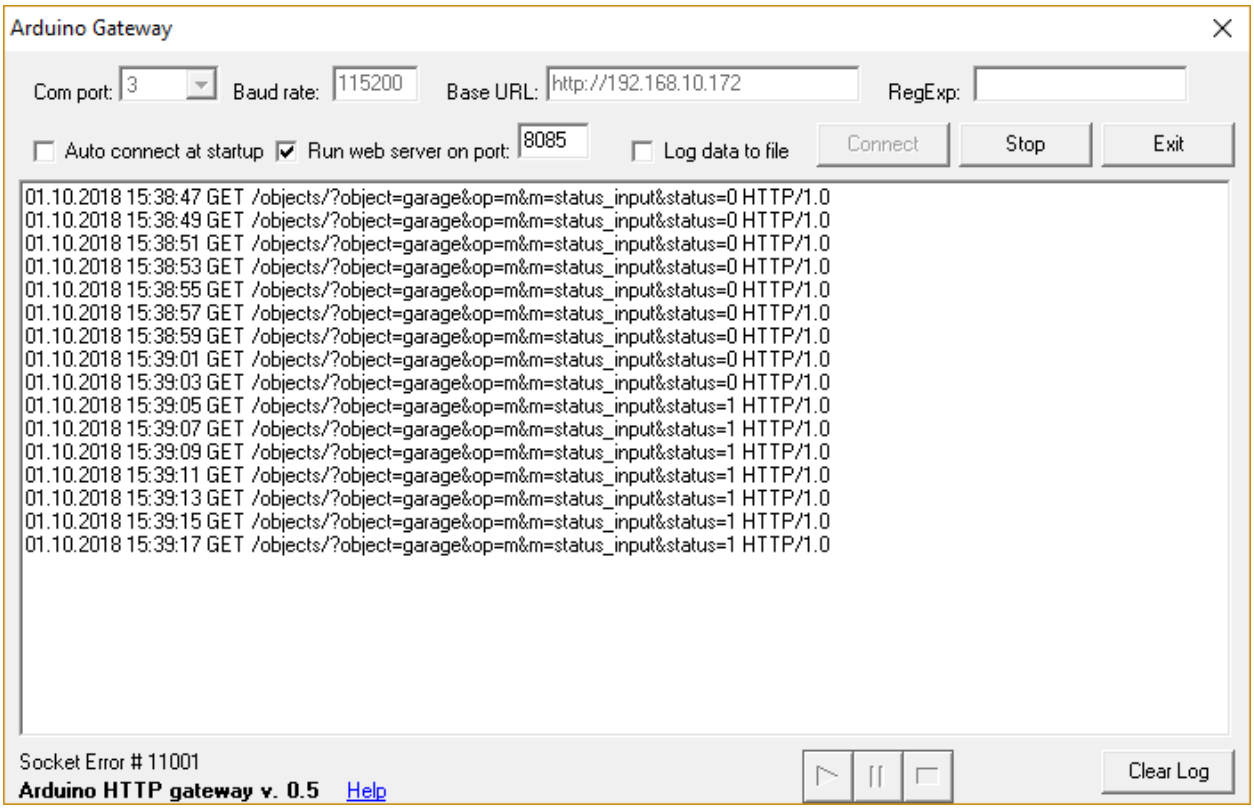


Рис. 5.2. Новый результат с программой arduino_gw

Следуя указаниям, которые можно найти на форуме **[6]**, я много раз пытаюсь оживить что-то в сцене; дошло до того, что пришлось заново записывать образ MajorDoMo на SD-карту, но, как не работала, так работать и не захотела связка Arduino и MajorDoMo на Raspberry Pi.

Я не люблю оставлять незавершенную работу. И вам не советую. Однако в этот раз я прекратил все попытки, поскольку есть одно простое соображение: сервер работает на Raspberry Pi, так?

Чтобы использовать Arduino с программой *arduino_gw* придется включать компьютер с операционной системой Windows. А это разве хорошо?

Следуя советам форумчан, я делаю попытку подключить Arduino к USB-порту Raspberry Pi, записав такую программу:

```
byte inByte = 0; // Переменная для чтения состояния датчика

void setup() {
  Serial.begin(115200); // Скорость работы порта
  pinMode(13, OUTPUT); // Задаем вывод 13 на выход
  digitalWrite(13, LOW); // Устанавливаем ноль на выходе
```

```
}

void loop()
{
  if (Serial.available() > 0) // Если порт готов к работе
  {
    inByte = Serial.read(); // Прочитаем состояние датчика
      if (inByte == 1) { // Если на входе единица
        digitalWrite(13, HIGH);} // Устанавливаем 1 на выходе
      else if (inByte == 0) // Если на входе ноль
         digitalWrite(13, LOW); // Устанавливаем 0 на выходе
   }
}
```

Программа подразумевает, что с сервера MajorDoMo будут отправляться сигналы, 1 и 0, которыми будет включаться и выключаться светодиод на плате Arduino, (вывод 13).

Чтобы отправить сигналы с Raspberry Pi, следует в панели управления MajorDoMo выбрать раздел *Меню управления*. В появившемся списке (**рис. 5.3**) выбрать подраздел *Управление*.

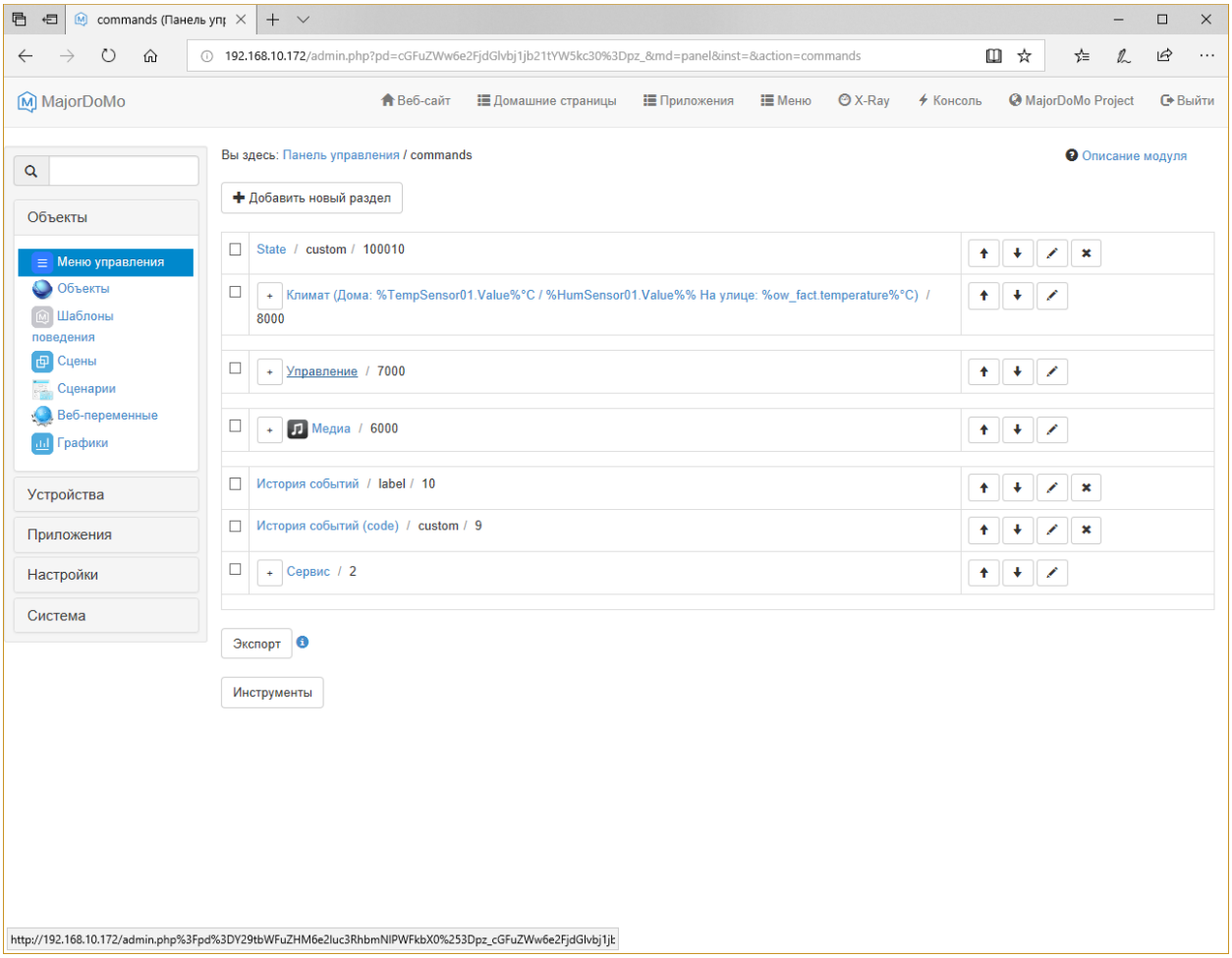


Рис. 5.3. Подготовка MajorDoMo к работе с Arduino, первый шаг

В новом окне в нижней части справа есть список *Дочерние пункты*, под которым есть кнопка **Добавить**. В новом диалоговом окне впишем название LED, объявим тип нового компонента *Выключателем*, впишем следующий код PHP (**рис. 5.4**):

```
if ($params['VALUE']) { // Если переменная TRUE
// Выполняем настройку порта
exec("mode COM3: BAUD=115200 PARITY=N data=8 stop=1 xon=off");
$fp =fopen("COM3", "w"); // Открываем порт для записи
fwrite($fp, chr(1)); // Записываем в порт «1»
fclose($fp); // Закрываем порт
say("Светодиод включен",5); // Для Алисы
} else { // Иначе все повторяем для записи «0»
exec("mode COM3: BAUD=115200 PARITY=N data=8 stop=1 xon=off");
$fp =fopen("COM3", "w");
fwrite($fp, chr(0));
fclose($fp);
say("Светодиод выключен",5);
}
```

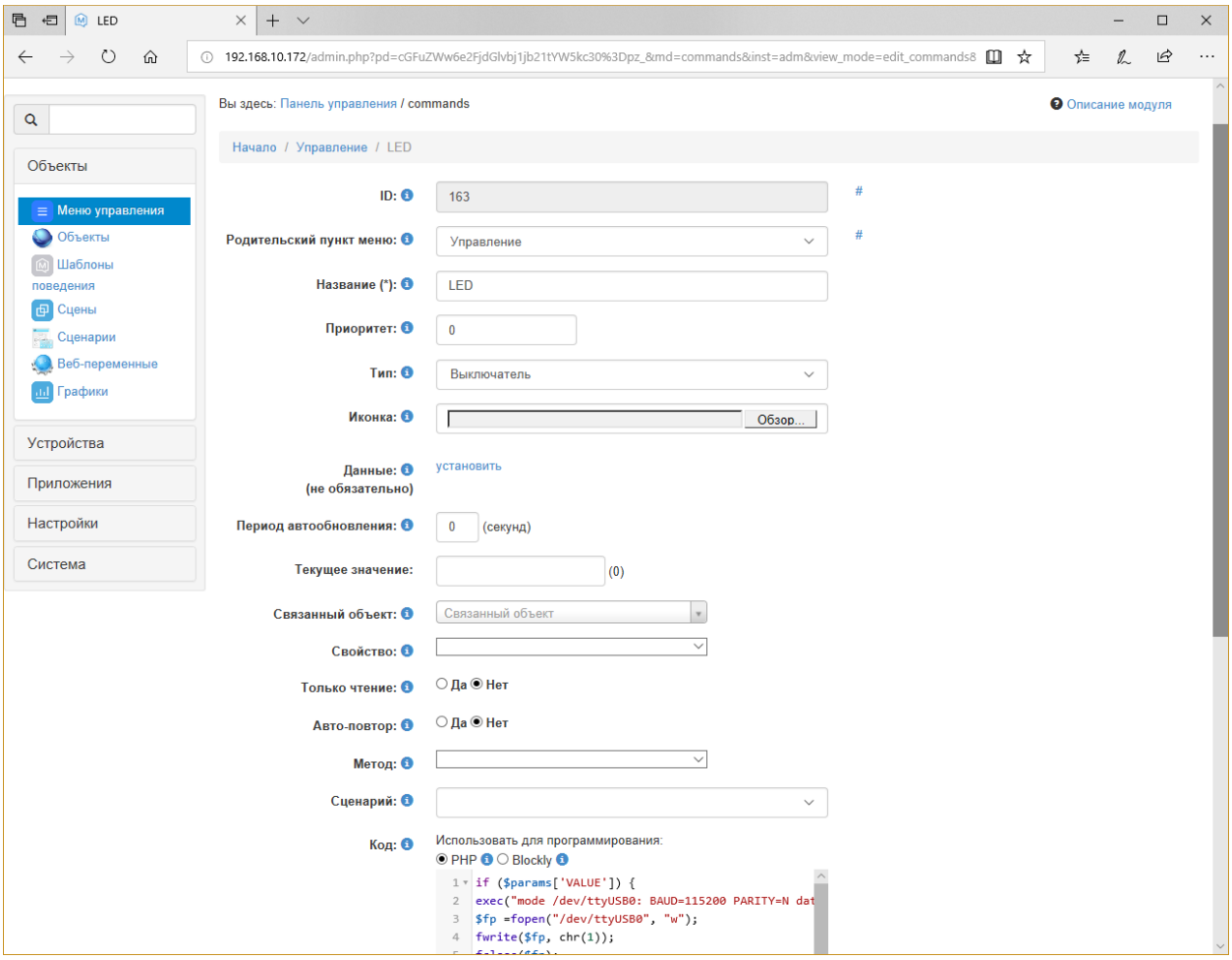


Рис. 5.4. Подготовка MajorDoMo для работы с Arduino, следующий шаг

В самом низу есть кнопка **Сохранить**. Ее и используем для завершения этого этапа подготовки. Если теперь вернуться к основной сцене, открыв раздел *Управление* на панели слева, то обнаружится новый выключатель по имени LED (**рис. 5.5**).

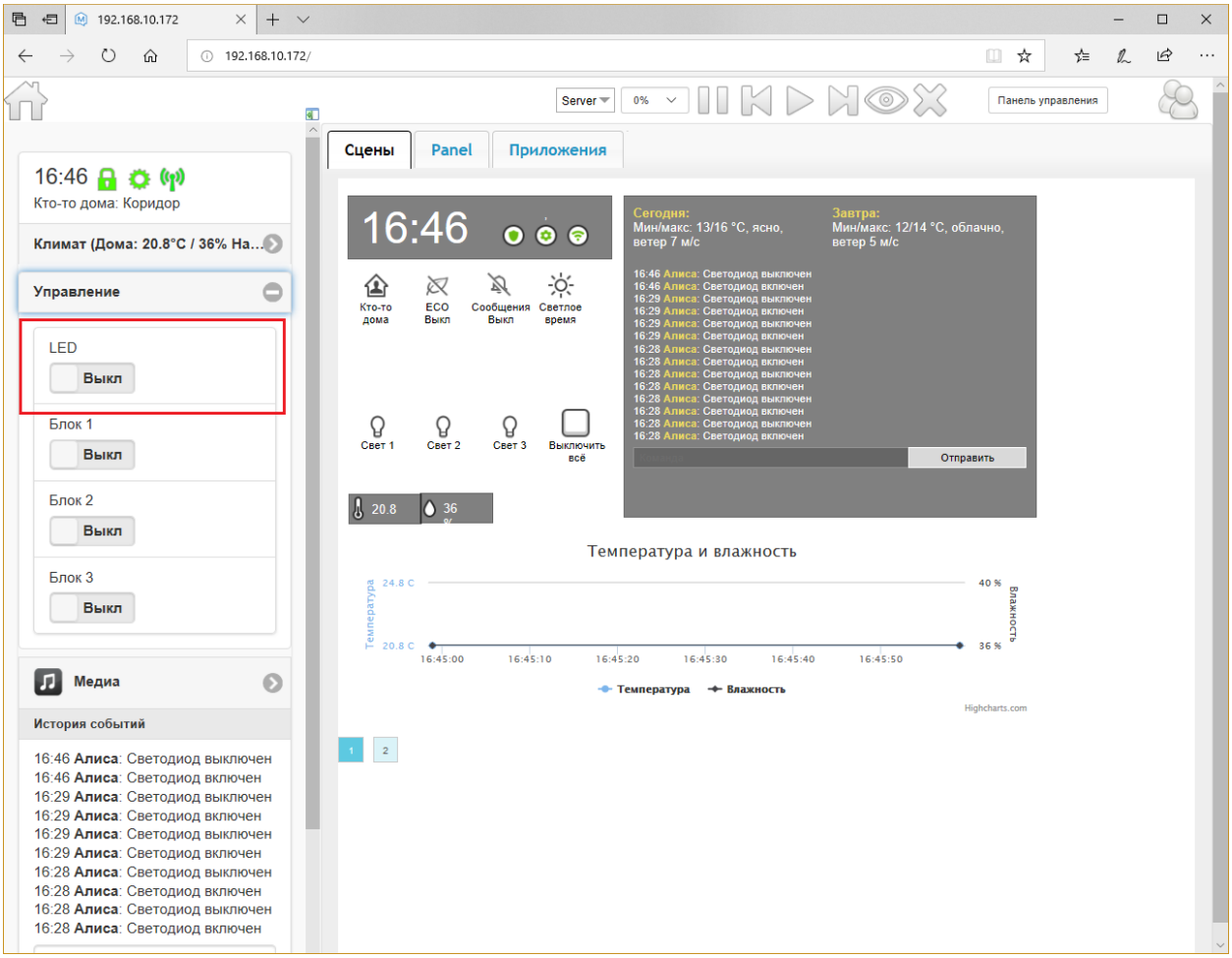


Рис. 5.5. Появление нового выключателя

Почему бы не воспользоваться выключателями, которые уже есть?

Хорошо, но, когда нам понадобится добавить выключатель, нам все равно придется создавать выключатель, это раз. И два, я уже с опаской отношусь к переделкам – после предыдущей пришлось заново записывать образ MajorDoMo. Я готов сформулировать ситуацию так: я еще не дорос до больших переделок в MajorDoMo.

Вновь созданным выключателем можно пощелкать. При подключенном к USB-порту Raspberry Pi модуле Arduino светодиод на выводе 13 несколько раз быстро мигает, делает паузу, затем мигает последний раз. Так происходит и при включении, и при выключении нового выключателя, хотя Алиса (информатор MajorDoMo) уверяет, что светодиод включен (или выключен).

Я не знаю, в чем дело, поэтому устанавливаю на Raspberry Pi программу Arduino:

```
sudo apt-get install arduino
```

Но этого мало, поскольку программа не запускается. Приходится продолжить установку:

```
sudo apt-get install oracle-java7-jdk
```

Не все устанавливается из пакета java, но теперь программа Arduino запускается. С ее помощью можно зажечь и погасить светодиод с помощью выключателя LED. Но!

После перезагрузки сервера Arduino вновь не выполняет команды включение-выключение, а только мигает. Нужно запустить программу Arduino и включить монитор порта. После этих «притопов» выключатель LED на сервере работает, можно выключить программу Arduino. Что ж, для наших опытов, возможно, это и подойдет.

Мое решение игнорировать предыдущую загадку, похоже, привело к появлению новой загадки. Одна из них – необходимость запускать программу Arduino. Я предположил, что нужно установить драйвер для работы с микросхемой CH340, которая преобразует USB-порт в виртуальный COM-порт. Я попытался установить такой драйвер, загрузив его на флэшку. И обнаружил, что и флэшка, которая подключена к порту USB, вне зоны доступа.

Конечно, эта проблема решается настройкой файла fstab **[7]**. Однако, и это только увеличило количество загадок, флэшка «видна» при использовании Midnigt Commander, но «не видна» в менеджере файлов.

Сомнения в наличии драйвера для Arduino. можно проверить, используя команды `lsusb` и `lsmode` (**рис. 5.6**).

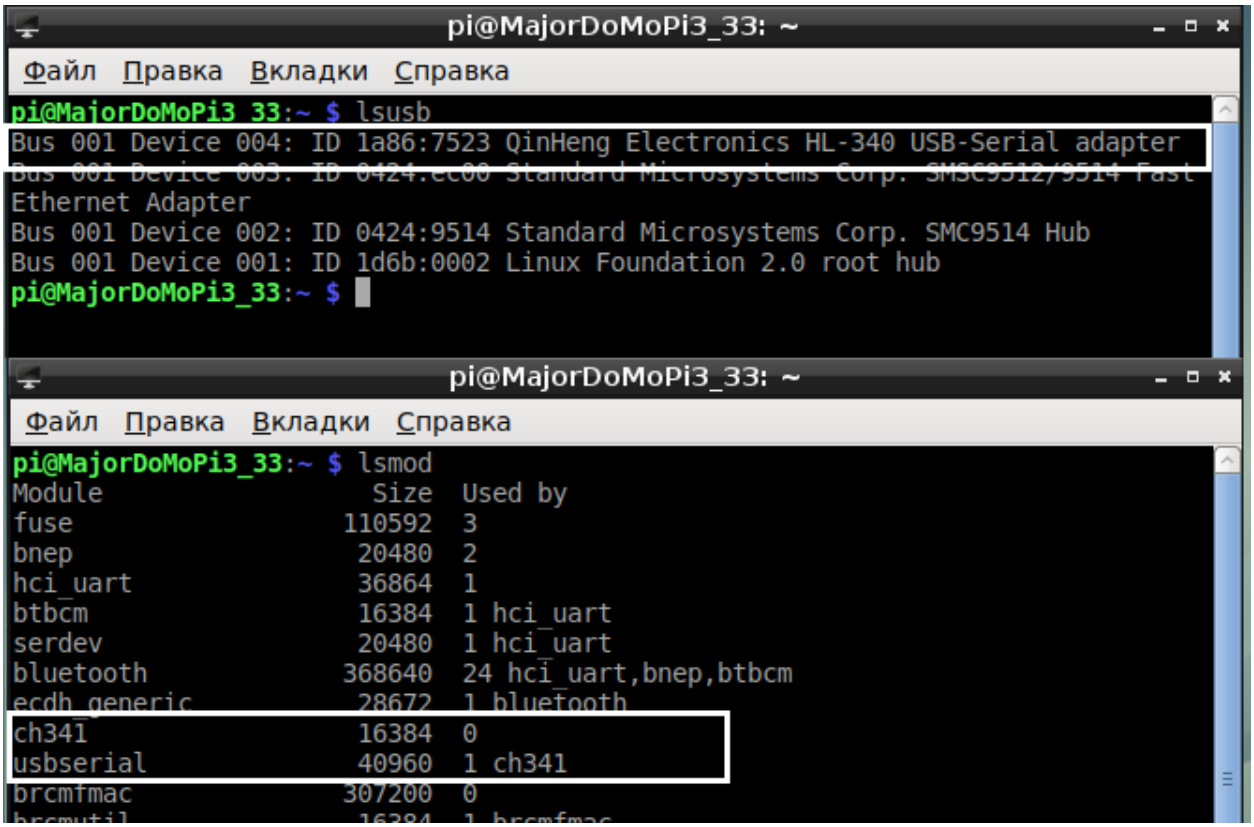


Рис. 5.6. Проверка подключения Arduino

Тот факт, что программа, приведенная выше, предназначена для Windows (COM3!), был очевиден сразу, но перевод программы на языке PHP для использования с Linux не изменил ситуацию:

```
if ($params['VALUE']) { // Если значение переменной TRUE
```

```
// Выполняем настройку порта в Linux
exec("stty -F /dev/ttyUSB0 115200 cs8");
$fp =fopen("/dev/ttyUSB0", "r+"); // Открываем порт для записи
fwrite($fp, chr(1)); // Записываем «1»
fclose($fp); // Закрываем порт
say("Светодиод включен",5); // Пусть Алиса подтвердит операцию
} else { // Иначе повторяем все для записи «0»
exec("stty -F /dev/ttyUSB0 115200 cs8");
$fp =fopen("/dev/ttyUSB0", "r+");
fwrite($fp, chr(0));
fclose($fp); ;
say("Светодиод выключен",5);
}
```

При этом легко убедиться, что *ttyUSB0* – это правильное обращение к Arduino: при выключении Arduino это устройство пропадает, при включении появляется. Происходит все в папке */dev*. Были сомнения в правильной настройке скорости обмена и других параметров порта. Проверить их помогла программа Arduino. Проверяя настройки до включения программы и после ее запуска и обращения к монитору порта, можно легко привести параметры в полное соответствие (**рис. 5.7**).

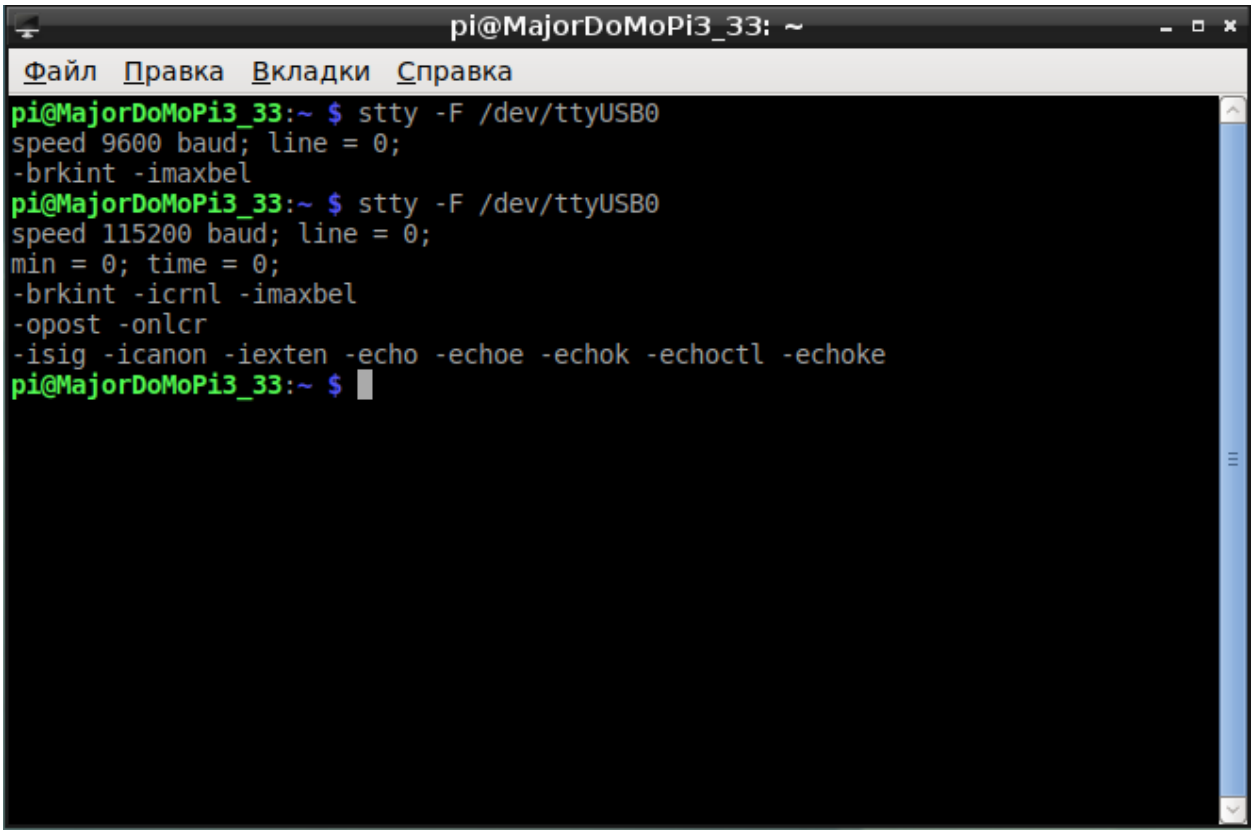


Рис. 5.7. Параметры порта до и после запуска программы Arduino

Загадка не осталась неразгаданной, решение было найдено поиском в Интернете **[8]**. Теперь программа выглядит так:

```
  if ($params['VALUE']) { // Если переменная TRUE
  $port = "/dev/ttyUSB0"; // Указываем порт
  $command =`stty -F /dev/ttyUSB0 cs8 115200 ignbrk -brkint -
icrnl -imaxbel -opost -onlcr -isig -icanon -iexten -echo -echoe
-echok -echoctl -echoke noflsh -ixon -crtscts -hupcl`;
  print $command;  // Инициализация порта
  $fp =fopen("/dev/ttyUSB0", "r+"); // Открываем порт для записи
  fwrite($fp, chr(1)); // Записываем в порт символ «1»
  fclose($fp); // Закрываем порт
  say("Светодиод включен",5); // Алиса подтвердит включение
  } else { // Иначе повторяем все для записи символа «0»
  $port = "/dev/ttyUSB0";
  $command =`stty -F /dev/ttyUSB0 cs8 115200 ignbrk -brkint -
icrnl -imaxbel -opost -onlcr -isig -icanon -iexten -echo -echoe
-echok -echoctl -echoke noflsh -ixon -crtscts -hupcl`;
  print $command;  // инициализация порта
  $fp =fopen("/dev/ttyUSB0", "r+");
  fwrite($fp, chr(0));
  fclose($fp); ;
  say("Светодиод выключен",5);
  }
```

Первое включение выключателя LED не срабатывает, но последующие включения и выключения правильно управляют пресловутым светодиодом на выводе 13 платы Arduino.

Arduino с Ethernet-шилдом

Каждый раз, когда что-то оставляешь в стадии не решенной проблемы, эта проблема имеет свойство возвращаться в самый неподходящий момент. Когда не получилась работа Arduino через шлюз *arduino_gw*, я обоснованно отказался от поиска решения, но в этот раз попытка поработать с помощью Ethernet-шилда вернулась к тем же результатам, что и в прошлый раз.

После включения модуля Arduino с установленной надстройкой Ethernet полезно проверить, работает ли модуль. Для этого используем программу из примера, которая называется *WebClient* в разделе *Ethernet*. Правильная работа программы в мониторе порта выглядит так (**рис. 5.8**).

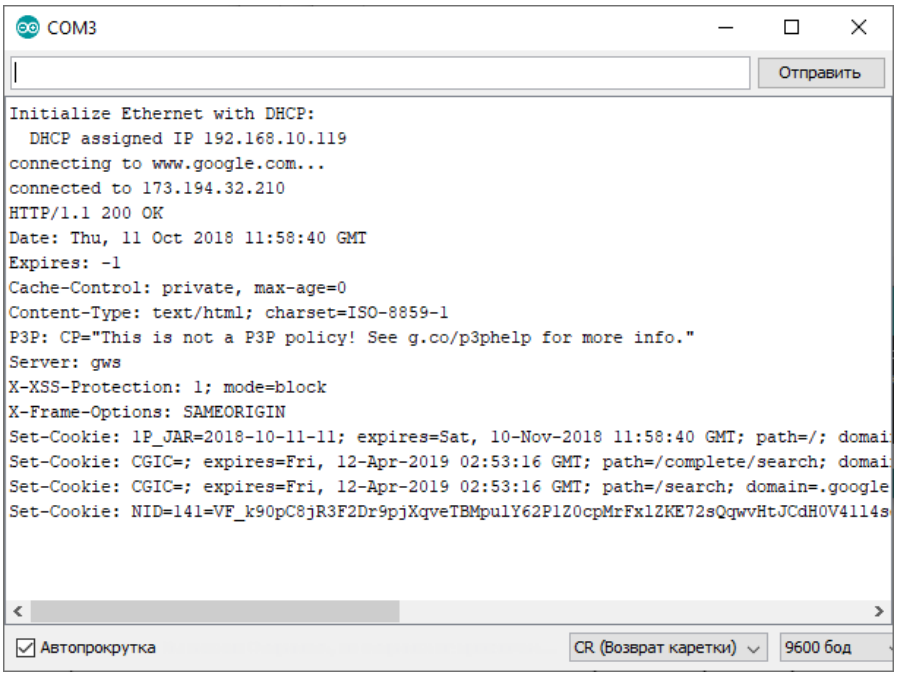


Рис. 5.8. Работа программы WebClient

Подсмотрев IP-адрес, выделенный роутером для модуля Arduino, можно продолжить работу с ним. В этом опыте (см. Приложение Б, **рис. Б.1**) я намереваюсь использовать один из входов Arduino для подключения к напряжению логической единицы (вывод 3V3) и логического нуля (вывод GND). Эти изменения я хочу увидеть либо на созданной мною сцене, либо на панели, либо и там, и там.

При этом необходимо внести дополнения в MajorDoMo, которые я беспрекословно повторяю по рекомендациям на форуме. А именно:

- Создаем новый класс Input (**рис. 5.9**). То есть, переходим в панель управления, выбираем раздел *Объекты*, где выбираем подраздел *Объекты* (на рисунке класс *Input* уже добавлен). Используется кнопка **+Добавить новый класс**.

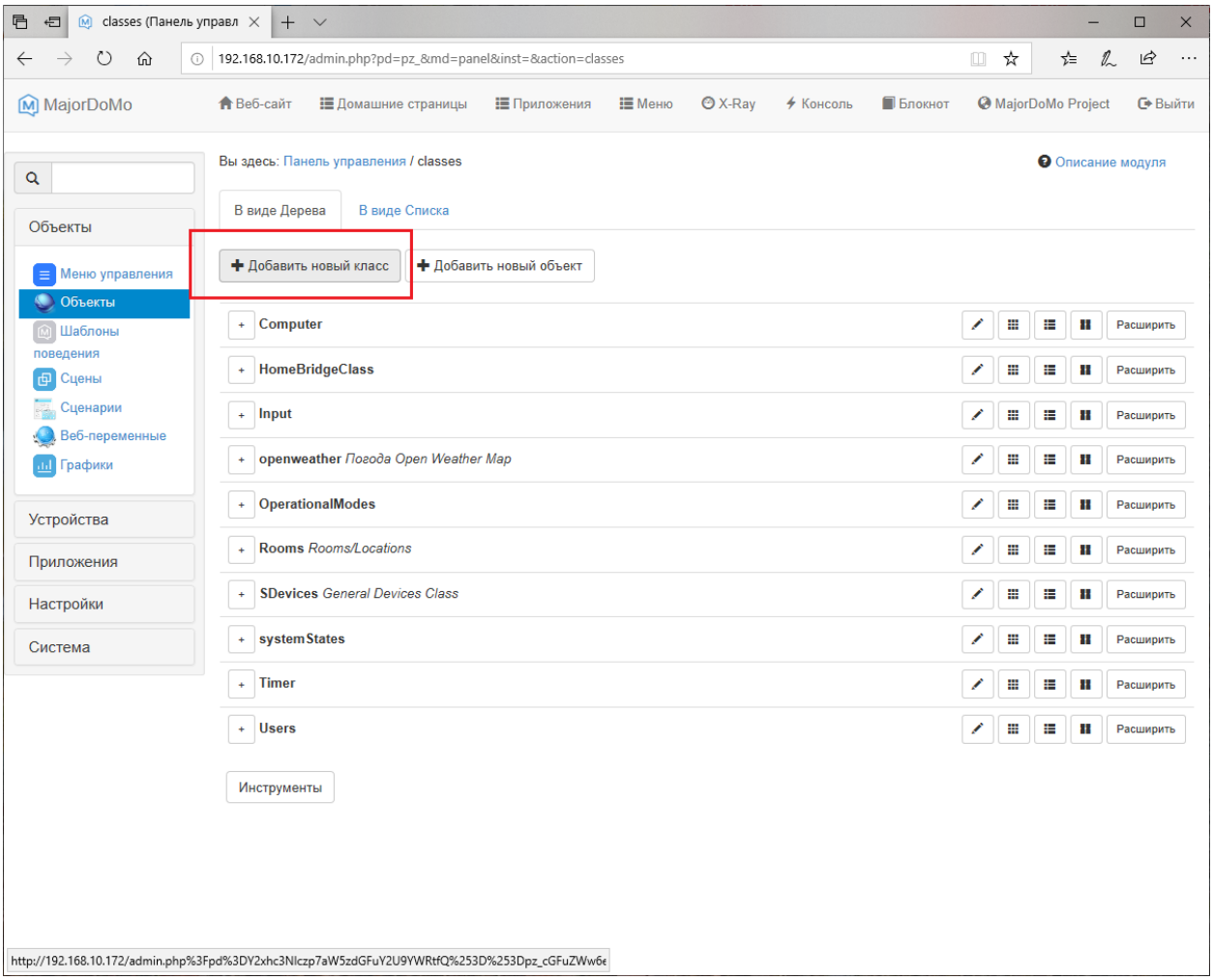


Рис. 5.9. Создание нового класса в системе MajorDoMo

Любой класс интересен не сам по себе, но своими свойствами и возможностями, то есть, методами. Как и классу, им можно присвоить любые имена, но лучше такие, которые хорошо отражают их назначение. В данном случае используются те, что я увидел на форуме, чтобы легче было восстановить ход событий. Итак.

На странице класса есть закладки, где описываются (и создаются) все атрибуты класса. А на закладках диалога класса есть кнопки, позволяющие создать или добавить новое свойство, метод, объект. Для свойства я повторяю увиденное название – *status*. На странице свойства не будет ничего, кроме названия (**рис. 5.10**).

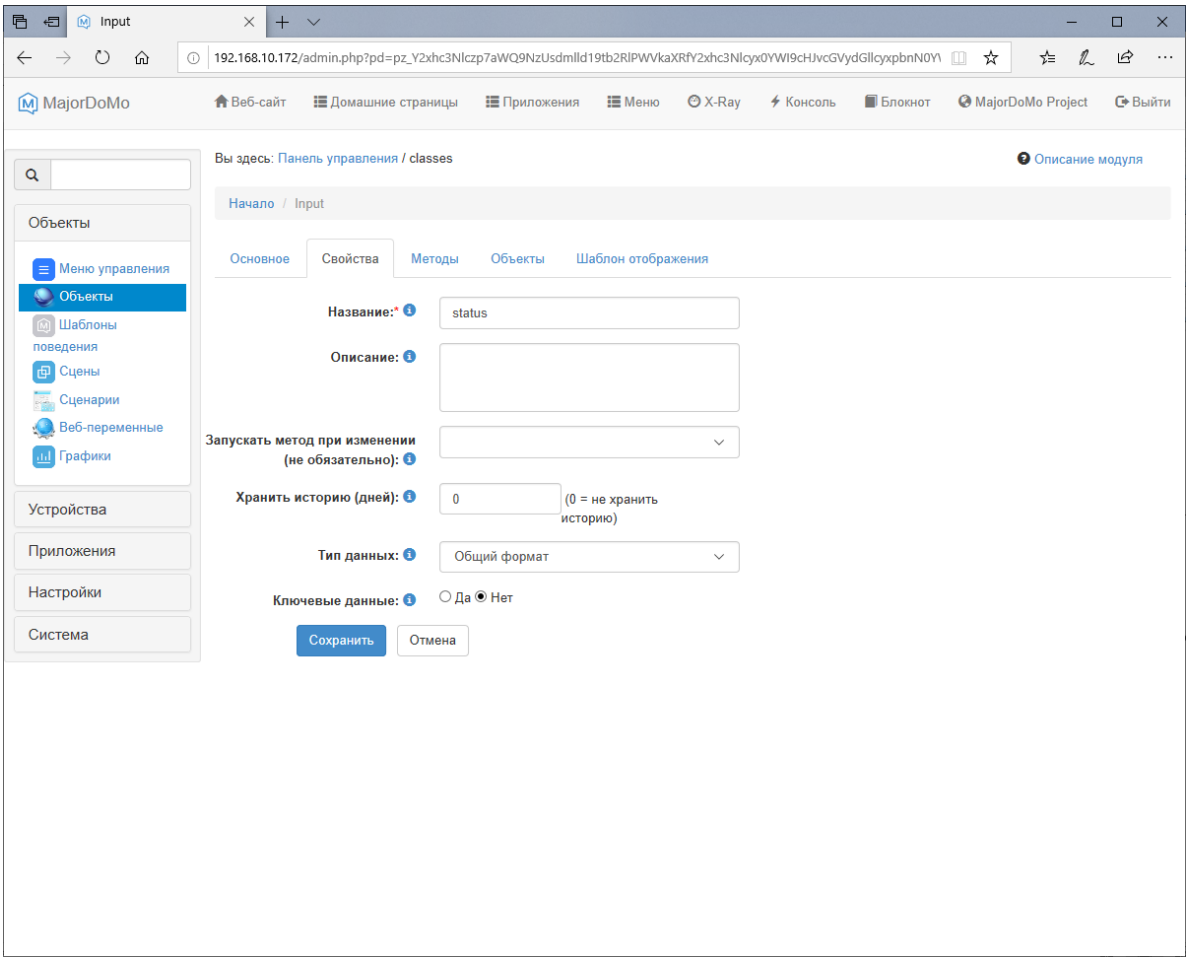


Рис. 5.10. Страница нового свойства нового класса

Свойства могут иметь другие параметры, если вы их знаете. Иначе достаточно нажать кнопку **Сохранить**.

Если нажать закладку *Основное*, а следом вернуться на закладку *Свойства*, вы получите возможность редактировать или удалять свойства (**рис. 5.11**). Кнопка с изображением карандаша (или ручки) позволяет начать редактирование предмета, а кнопка с крестиком удалит этот предмет. Сейчас это свойство, но такие же возможности имеют и другие элементы, включая сам класс.

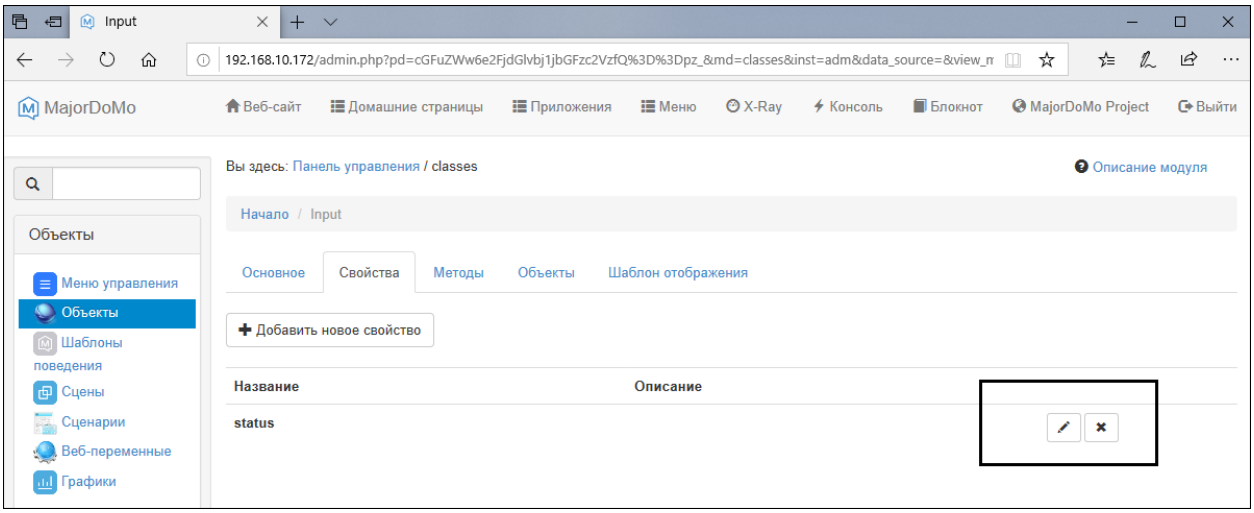


Рис. 5.11. Закладка *Свойства* после создания нового свойства

Кроме добавленного свойства наш класс должен иметь метод. Его название я тоже взял с форума, вы можете использовать свое название. На закладке *Методы*, используя кнопку добавления нового метода, открываем диалог параметров метода (**рис. 5.12**).

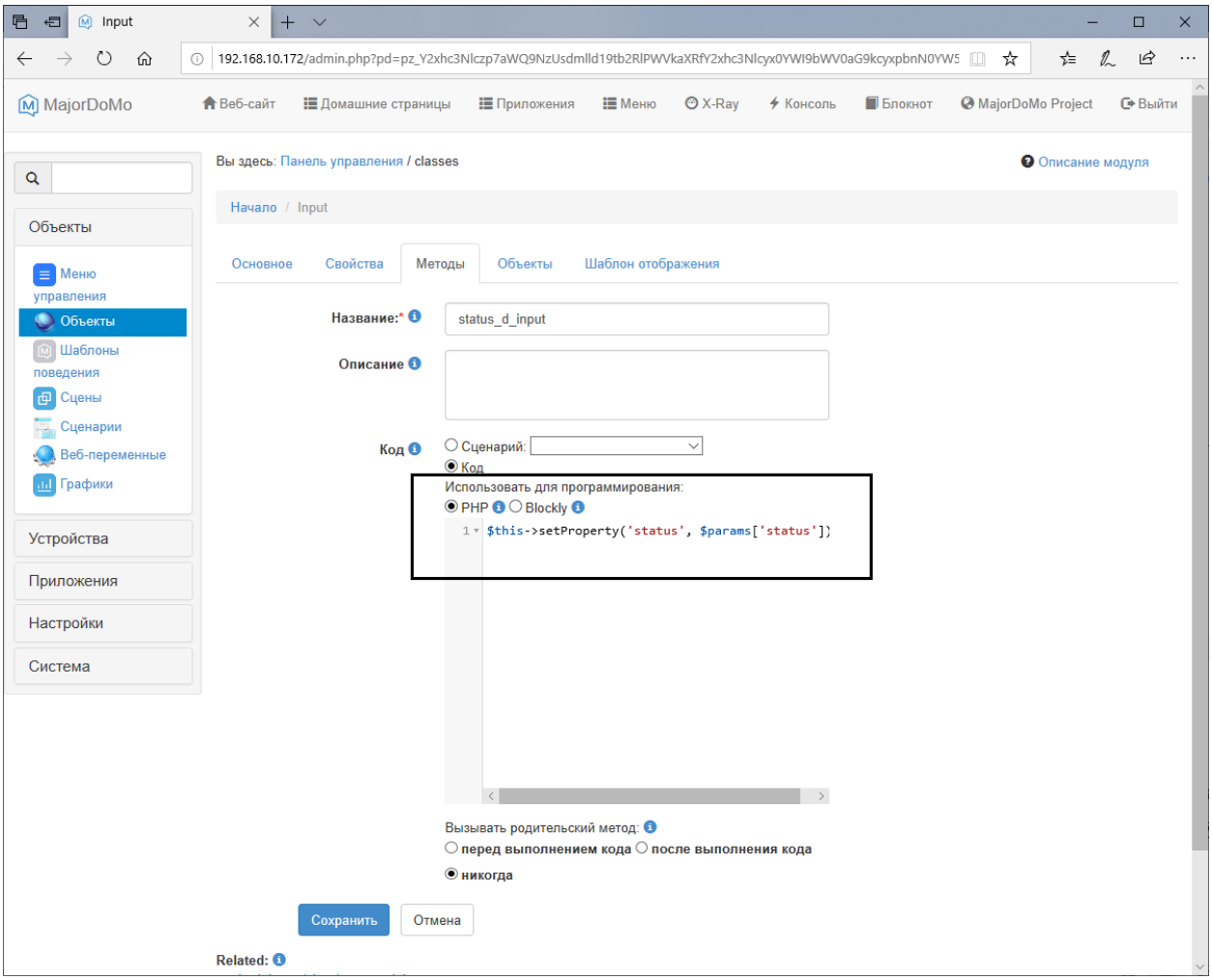


Рис. 5.12. Диалог параметров метода

Здесь помимо названия *status_d_input* нужно в разделе *Код* включить метку PHP, а в окне кода ввести строку:

```
$this->setProperty('status', $params['status']);
```

Смысл этой команды простой – установить свойство *status* в наборе параметров *params*. Осталось добавить объект, я назвал его *garage*. На странице диалога добавлено название объекта и его приписка к классу. Объект наследует свойства и методы класса, но может иметь и дополнительные свойства, о чем свидетельствует закладка свойства (**рис. 5.13**). Завершается диалог нажатием на кнопку **Сохранить**.

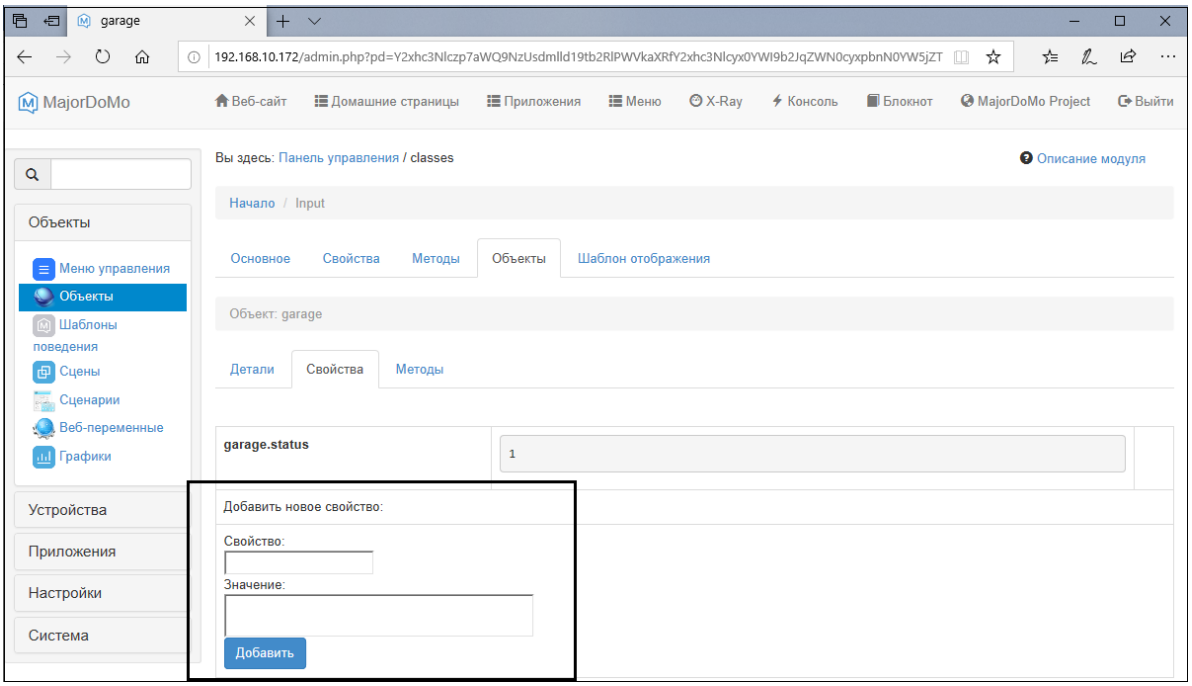


Рис. 5.13. Диалог добавления объекта

Не менее интересно заглянуть на закладку *Методы* после создания объекта. То есть, после использования кнопки **Сохранить** (**рис. 5.14**). По возвращении достаточно рядом с этим методом *Input-> status_d_input* нажать кнопку **Настроить**.

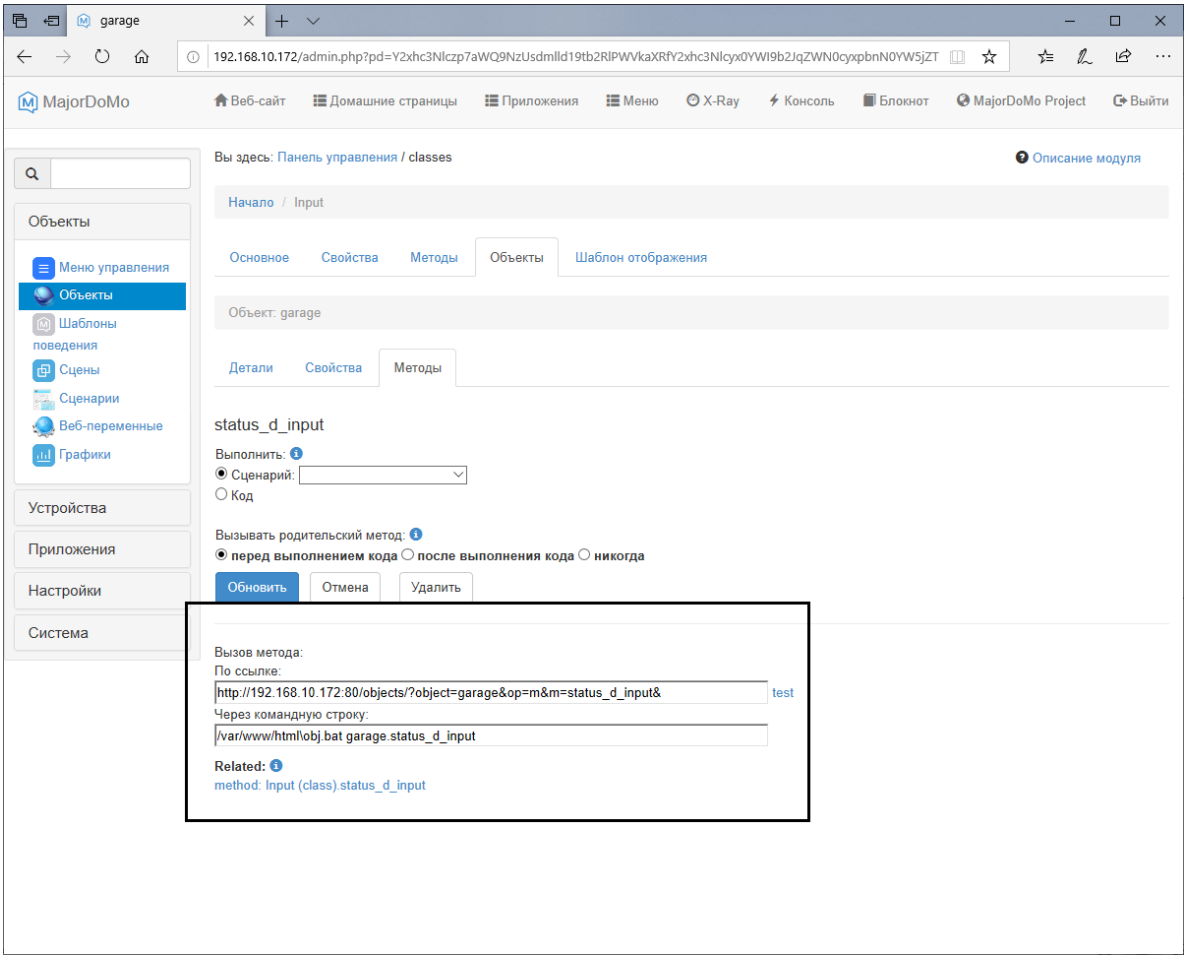


Рис. 5.14. Описание метода в настройках объекта

На странице есть подсказки по вызову метода: по ссылке и через командную строку.

С командной строкой, забегая вперед, сознаюсь у меня ничего не получилось, поскольку используется файл, предназначенный для Windows, а на вызов метода обратите внимание – он нам еще пригодится.

Далее, просматривая беседу на форуме, где приводится описание всего, что я сделал, я попытался повторить так, как понял, иными словами, создать новый объект для своей сцены.

Примечание.

После перезаписи образа диска для MajorDoMo, можно было не повторять все сделанное раньше, полезнее сделать что-то новое, например, добавить свою картинку для сцены. Полученный опыт позднее пригодится.

Если обратиться к сцене: выбрать в разделе *Объекты* подраздел *Сцены*, то достаточно щелкнуть левой клавишей мышки по новой сцене, чтобы перейти к диалогу свойств сцены (**рис. 5.15**). А там есть закладка *Элементы*.

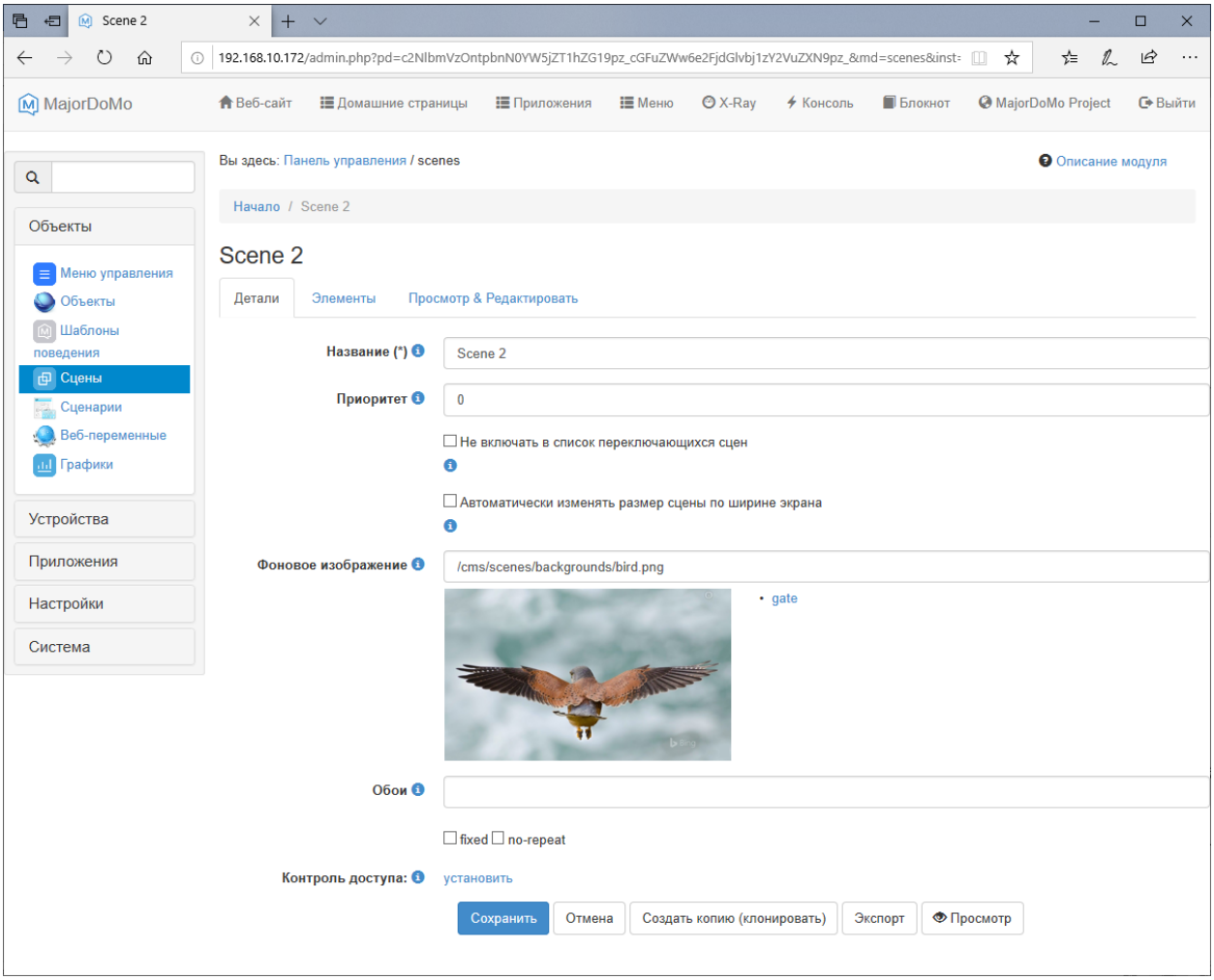


Рис. 5.15. Диалоговое окно работы со сценой

В этом диалоге и добавим новый элемент. Я назвал его *gate*, выбрал его тип как объект и связал с объектом *garage* (**рис. 5.16**).

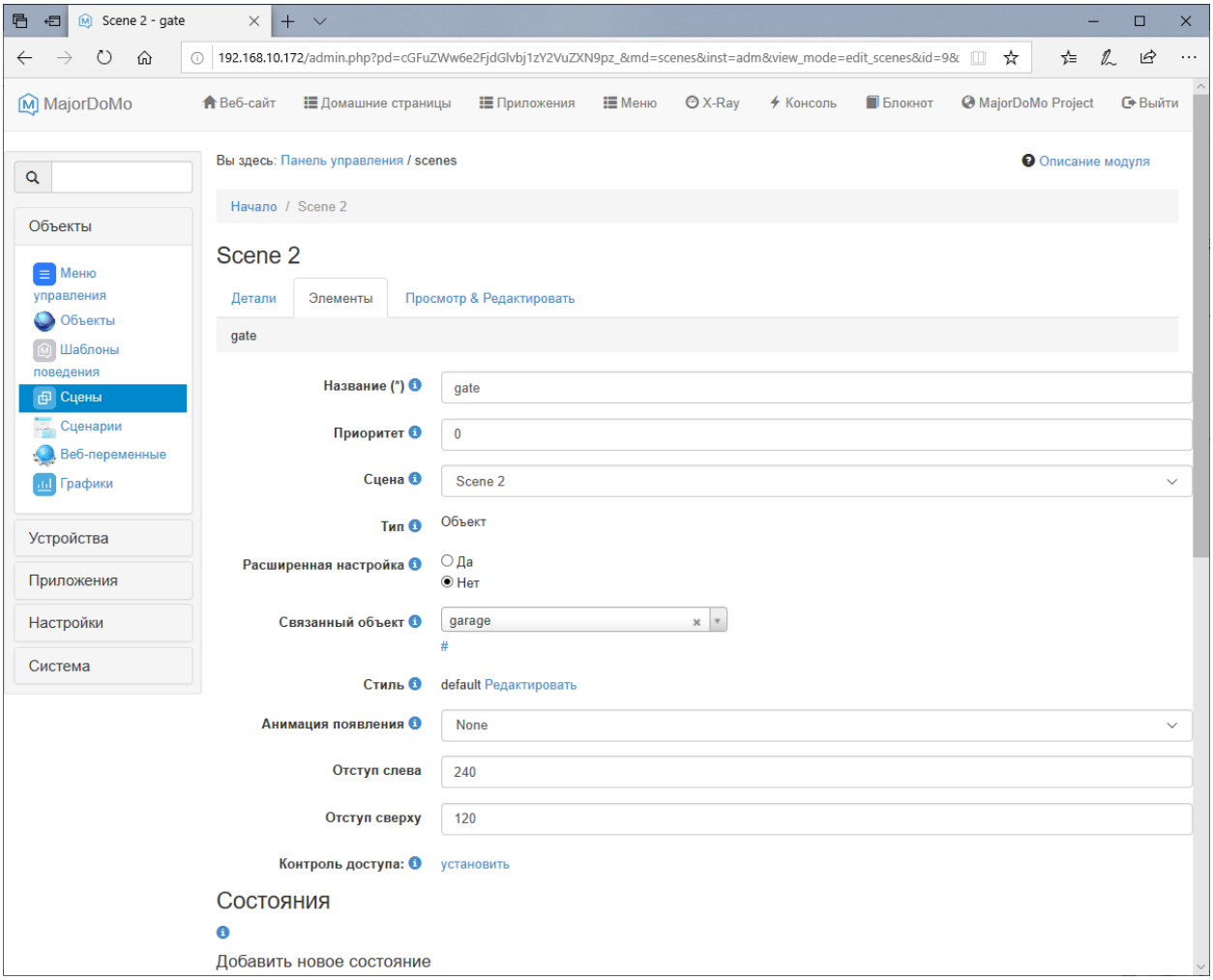


Рис. 5.16. Новый элемент сцены

Программу для работы с входами Arduino я честно взял в указанном на форуме месте, но оставил использование только одного входа, чтобы упростить проверку и добавил длинную паузу между запросами:

```
/**
* Контроллер-сборщик данных (к проекту http://smartliving.ru/)
* Platform: Arduino UNO R3 + EthernetShield W5100
* IDE: Arduino 1.0.1
* Остальные датчики:
* Digital 4 - Датчик гаражной двери
**/

#include <Ethernet.h>
#include <SPI.h>

// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```
  byte ip[] = { 192, 168, 10, 119 }; // Его ip-адрес
  byte subnet[] = { 255, 255, 255, 0 }; // Маска подсети
  byte gateway[] = { 192, 168, 10, 1 }; // Адрес шлюза
  byte dns_server[] = { 192, 168, 10, 1 }; // Адрес DNS сервера

  // ip-адрес удалённого сервера
  byte server[] = { 192, 168, 10, 172 };

  EthernetClient rclient;

  // Инициализация начальных значений
  int old_garage=0; // Предыдущее состояние датчика
  char buf[80]; // Массив для строки запроса
  char ipbuff[16]; // Массив для ip-адреса

  // Функция отправки HTTP-запроса на сервер
  void sendHTTPRequest() {
    Serial.println(buf); // Вывод на монитор порта
    if (rclient.connect(server, 80)) { // Есть ли подключение?
     Serial.println("OK");
     rclient.print(buf); // Отправка запроса
     rclient.println(" HTTP/1.0");
     rclient.print("Host: ");
     sprintf(ipbuff, "%u.%u.%u.%u", ip[0], ip[1], ip[2], ip[3]);
     rclient.println(ipbuff); // ip адрес нашего контроллера в
текстовом виде
     rclient.print("Content-Type: text/html\n");
     rclient.println("Connection: close\n");
     delay(2000);
     rclient.stop(); // Остановка клиента
    } else { // Если соединения нет
     Serial.println("FAILED");
    }
  }

  void setup()
  {
    // Для отладки отладочные сообщения выводятся на монитор
    Serial.begin(9600); // Скорость работы порта
    Serial.println("Start");
   Ethernet.begin(mac, ip, dns_server, gateway, subnet); //
Инициализируем Ethernet Shield
   pinMode(4, INPUT); // Датчик гаражной двери
   old_garage=digitalRead(4); // Прочитываем состояние датчика
  }
```

```
  void loop()
  {
    //Датчик гаражной двери
    Serial.println("G");
    int current_garage=digitalRead(4); // Еще раз читаем
состояние датчика
    if (current_garage!=(int)old_garage) { // Если состояние
изменилось
      old_garage=(int)current_garage; // Теперь это будет старое
состояние
      sprintf(buf, "GET
/objects/?object=garage&op=m&m=status_d_input&&status=%i",
(int)current_garage); // Строка запроса
      sendHTTPRequest(); // Отправляем запрос
    }
  delay(4000);
  }
```

Проверка работы показала, что при переключении входа Arduino монитор порта исправно показывает, что состояние входа изменяется, но этого нельзя сказать про MagorDoMo. Как отображалось заданное в настройках значение, так оно и оставалось неизменным (**рис. 5.17**). Но обращение к серверу было, иначе при отключении Raspberry Pi в мониторе порта не появлялось бы сообщение о неудачном обращении к серверу.

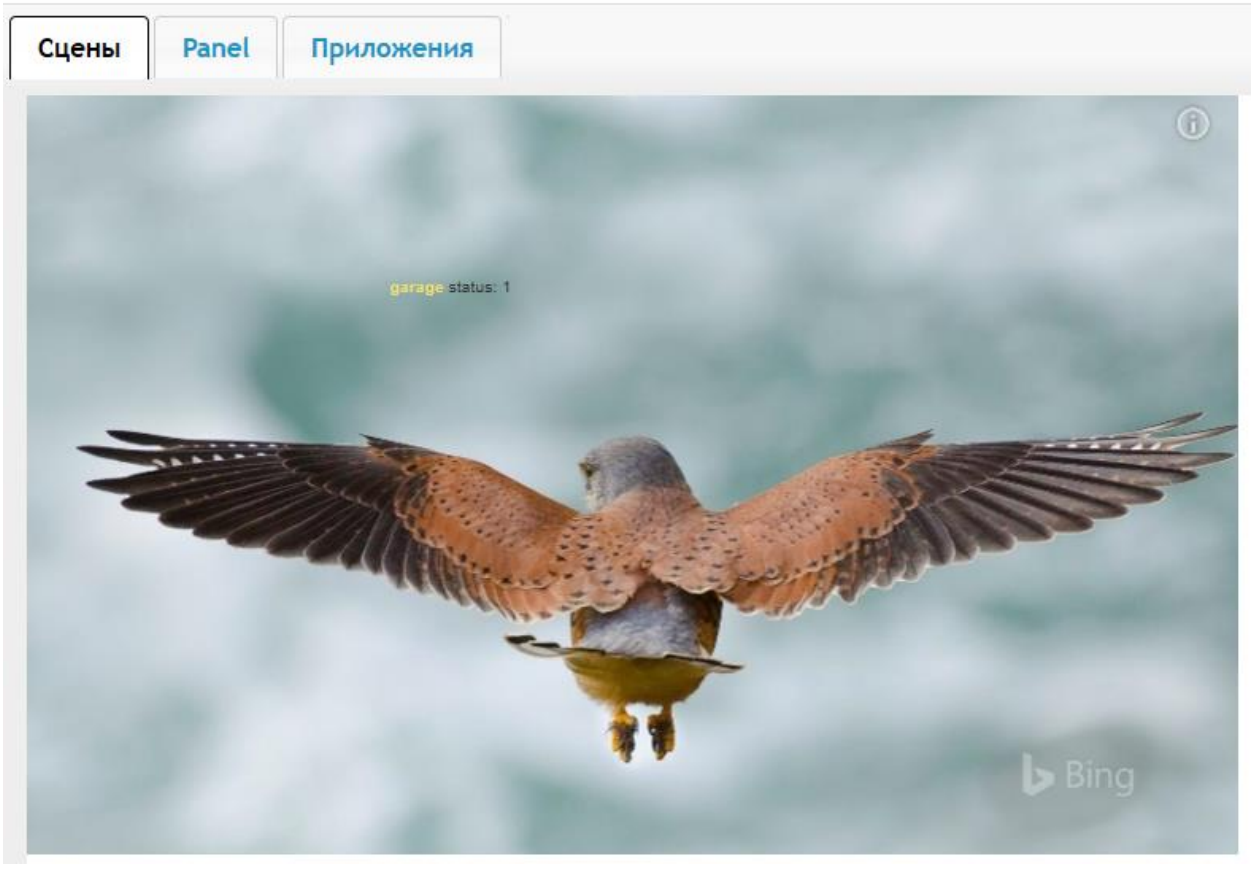


Рис. 5.17. Неудачный результат опыта с входом Arduino

Я не стал бы приводить этот рисунок, если бы не «наступил на те же грабли», что и ранее. Из песни слова не выкинешь. Но любые ошибки, если в них разобраться, могут дать новые знания. На дальнейшую работу меня вдохновляет тот факт, что… Помните, я просил обратить внимание на **рис. 5.14** на строку вызова метода по ссылке? Если ввести эту строку в браузере стационарного компьютера, добавив *status=1* или *status=0*, то на странице MajorDoMo *garge.ststus* меняет значение.

Поэтому, наверное, не сразу, я решил попробовать программу из примеров пакета Arduino, подстроив ее под свои нужды:

```
  #include <SPI.h>
  #include <Ethernet.h>

  // Введите MAC-адрес вашего контроллера ниже.
  // Последние Ethernet shields имеют MAC-адрес на плате
  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

  // если вы не хоитите использовать DNS (и уменьшить программу)
  // используйте числовой IP вместо имени сервера:
  //IPAddress server(74,125,232,128);  // числовой IP для Google
(без DNS)
  byte server[] = { 192, 168, 10, 172 };    // имя сервера
Google (с использованием DNS)

  // Задаем статический IP-адрес для использования, если DHCP не
срабатывает
  IPAddress ip(192, 168, 10, 119); // IP-адрес модуля
  IPAddress myDns(192, 168, 10, 1); // IP-адрес DNS

  // Инициализация библиотеки Ethernet клиента
  // с IP-адресом и портом сервера, к которому вы хотите
  // подключиться (порт 80 определен для HTTP):
  EthernetClient client;

  // Переменные для измерения скорости
  unsigned long beginMicros, endMicros;
  unsigned long byteCount = 0;
  bool printWebData = true;  // устанавливается в false для
лучшего измерения скорости

  void setup() {
    // Вы можете использовать Ethernet.init(pin) для
конфигурирования вывода CS
    //Ethernet.init(10);  // Most Arduino shields
    //Ethernet.init(5);   // MKR ETH shield
    //Ethernet.init(0);   // Teensy 2.0
```

```
    //Ethernet.init(20);  // Teensy++ 2.0
    //Ethernet.init(15);  // ESP8266 with Adafruit Featherwing
Ethernet
    //Ethernet.init(33);  // ESP32 with Adafruit Featherwing
Ethernet

    // Открываем последовательный порт и ждем, когда он
откроется:
    Serial.begin(9600);
    while (!Serial) {
      ; // ждем подключения к последовательному порту. Нужно
только для собственного USB порта
    }

    // начинаем Ethernet подключение:
    Serial.println("Initialize Ethernet with DHCP:");
    if (Ethernet.begin(mac) == 0) {
      Serial.println("Failed to configure Ethernet using DHCP");
      // Проверим наличие оборудования Ethernet
      if (Ethernet.hardwareStatus() == EthernetNoHardware) {
        Serial.println("Ethernet shield was not found.  Sorry,
can't run without hardware. :(");
        while (true) {
          delay(1); // ничего не делаем, нет смысла работать без
Ethernet оборудования
        }
      }
      if (Ethernet.linkStatus() == LinkOFF) {
        Serial.println("Ethernet cable is not connected.");
      }
      // пробуем сконфигурировать, используя IP-адрес вместо
DHCP:
      Ethernet.begin(mac, ip, myDns);
    } else {
      Serial.print("  DHCP assigned IP ");
      Serial.println(Ethernet.localIP());
    }
    // дадим немного времени Ethernet shield для инициализации:
    delay(1000);
    Serial.print("connecting to ");
  //  Serial.print(server);
    Serial.println("...");

    // если соединение есть, сообщим через монитор порта:
    if (client.connect(server, 80)) {
      Serial.print("connected to ");
```

```
      Serial.println(client.remoteIP());
  //    client.print("Content-Type:text/html,charset=iso8859-
1");
      client.println();
      // Make a HTTP request:
      client.println(" HTTP/1.0");
      client.println("Host: 192.168.10.172");
      client.println("Connection: close");
      client.println();
    } else {
      // если подключения к серверу нет:
      Serial.println("connection failed");
    }
    beginMicros = micros();
  }

  void loop() {
    // если есть доступные входящие байты
    // от сервера, прочитаем их и выведем:
    int len = client.available();
    if (len > 0) {
      byte buffer[80];
      if (len > 80) len = 80;
      client.read(buffer, len);
      if (printWebData) {
        Serial.write(buffer, len); // покажем на мониторе
      }
      byteCount = byteCount + len;
    }

    // если сервер отключился, остановим клиента:
    if (!client.connected()) {
      endMicros = micros();
      Serial.println();
      Serial.println("disconnecting.");
      client.stop();
      Serial.print("Received ");
      Serial.print(byteCount);
      Serial.print(" bytes in ");
      float seconds = (float)(endMicros - beginMicros) /
1000000.0;
      Serial.print(seconds, 4);
      float rate = (float)byteCount / seconds / 1000.0;
      Serial.print(", rate = ");
      Serial.print(rate);
      Serial.print(" kbytes/second");
```

```
    Serial.println();

    // больше нечего делать:
    while (true) {
      delay(1);
    }
  }
}
```

Результат работы программы в мониторе порта выглядел так (**рис. 5.18**).

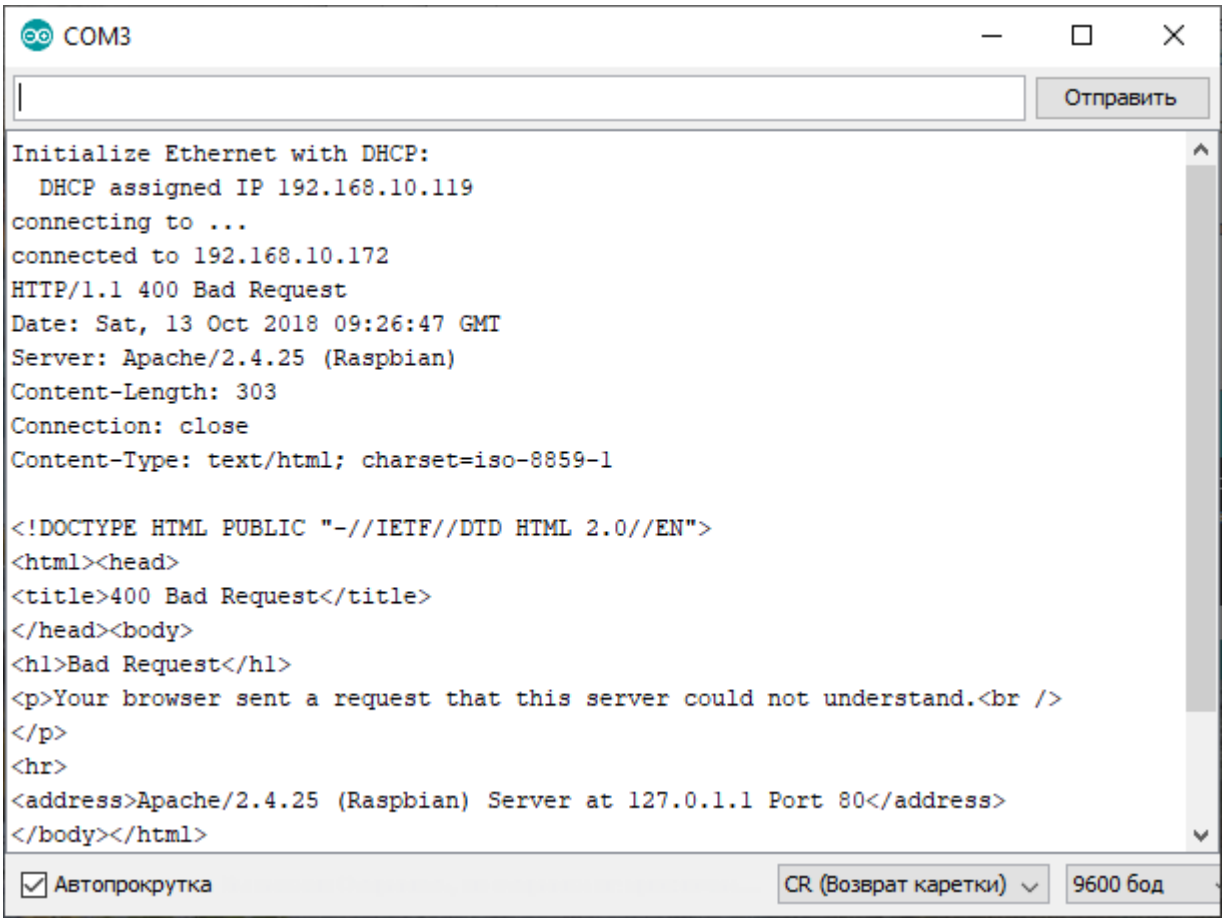


Рис. 5.18. Отображение работы переделанной программы WebClient

Проблема, похоже, оказалась в плохом запросе. Я не знаток запросов к серверу, на мой взгляд с запросом все было хорошо, но и с программой не поспоришь. Тогда я решил убрать в функции запроса все строки, постепенно приводя программу к такому виду:

```
#include <SPI.h>
#include <Ethernet.h>

char buf[80];
char ipbuff[16];
int old_garage=0;
```

```
// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 119 };
byte subnet[] = { 255, 255, 255, 0 };
byte gateway[] = { 192, 168, 10, 1 };
byte dns_server[] = { 192, 168, 10, 1 };

// ip-адрес удалённого сервера
byte server[] = { 192, 168, 10, 172 };

EthernetClient rclient;

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
  if (rclient.connect(server, 80)) {
   Serial.println("OK");
     rclient.println(buf);
    rclient.println("HOST: ");
    rclient.println("Connection: clos\n");
    rclient.println();
   delay(2000);
   rclient.stop();
  } else {
   Serial.println("FAILED");
  }
}

void setup()
{
  // Для отладки будем выводить сообщения на монитор
  Serial.begin(9600);
  Serial.println("Start");
//Ethernet.init(10);  // Most Arduino shields
 Ethernet.begin(mac, ip, dns_server, gateway, subnet); //
Инициализируем Ethernet Shield
 pinMode(4, INPUT); // Датчик гаражной двери
 old_garage=digitalRead(4);
}

void loop()
{
  //Датчик гаражной двери
  Serial.println("G");
  int current_garage=digitalRead(4);
  //Serial.println(current_garage);
```

```
    if (current_garage!=(int)old_garage) {
      old_garage=(int)current_garage;
     sprintf(buf, "GET
/objects/?object=garage&op=m&m=status_d_input&status=%i",
(int)current_garage);
      sendHTTPRequest();
    }
    delay(4000);
  }
```

И это помогло (**рис. 5.19**).

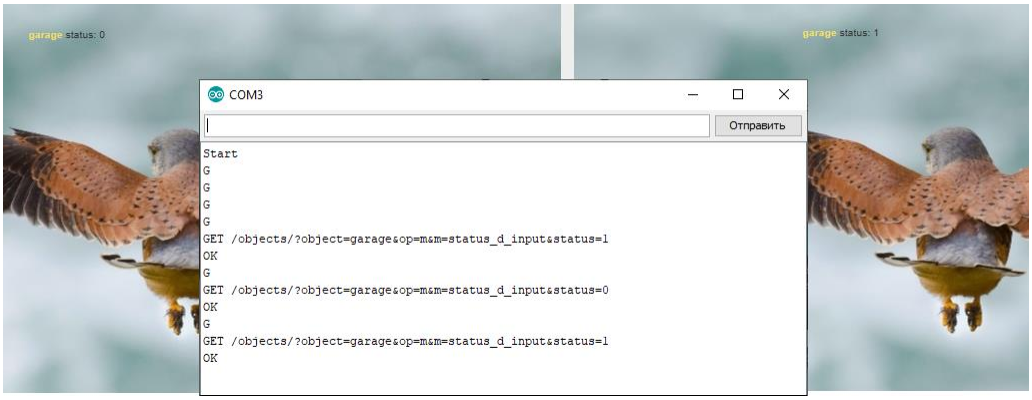


Рис. 5.19. Заработавший вариант

Вывод состояния появляется мелким шрифтом, а я пока не знаю, как это изменить.

Из-за этого я решил добавить вывод состояния на панель. Если нажать иконку с изображением гаечного ключа (таким мне он показался), то вы переходите в режим настройки панели, а, выбрав нижнюю панель, с помощью иконки со значком плюс, можно добавить что-то свое (**рис. 5.20**).

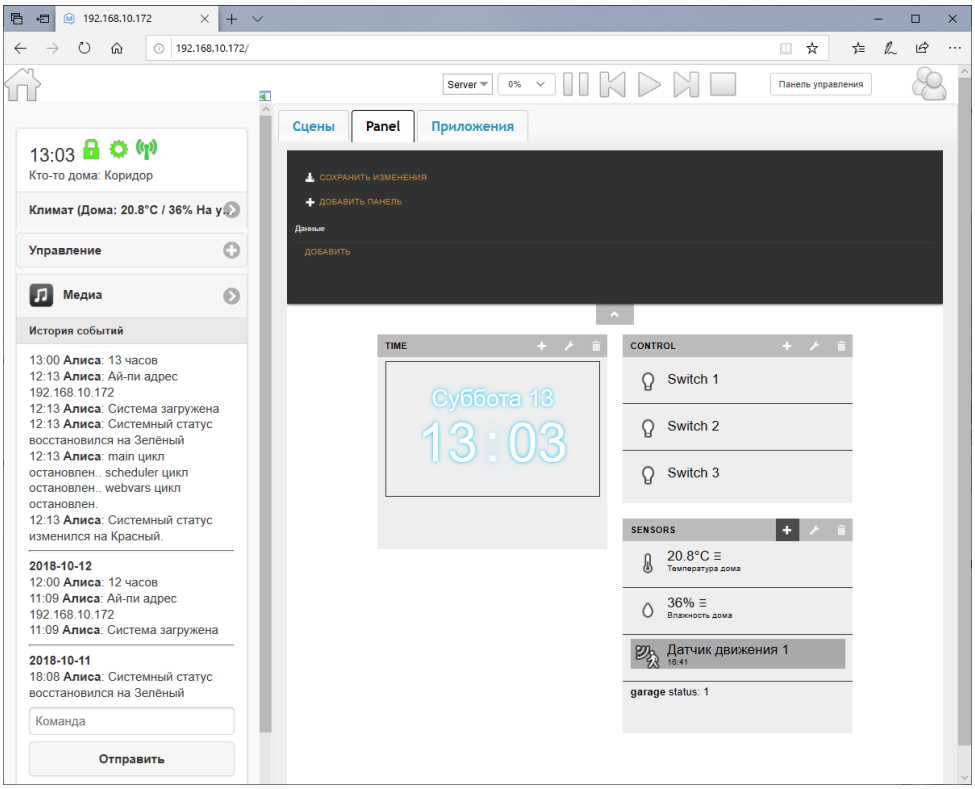


Рис. 5.20. Добавление вывода состояния на панель

Модуль OTA WeMOS D1

Для работы с этим модулем нужно настроить программу Arduino.

Хотя это ESP8266, но удобная среда разработки программ – это Arduino.

Начинается настройка с проверки подключения этого модуля (кабель USB-USBmicro), должен появиться виртуальный COM-порт. Затем в программе Arduino следует выбрать пункт *Файл→Настройки*. В окне настроек переходим к разделу «Дополнительные ссылки для Менеджера плат». Справа от окна есть кнопка, которая поможет вставить нужную ссылку на плату (**рис. 5.21**):

http://arduino.esp8266.com/stable/package_esp8266com_index.json

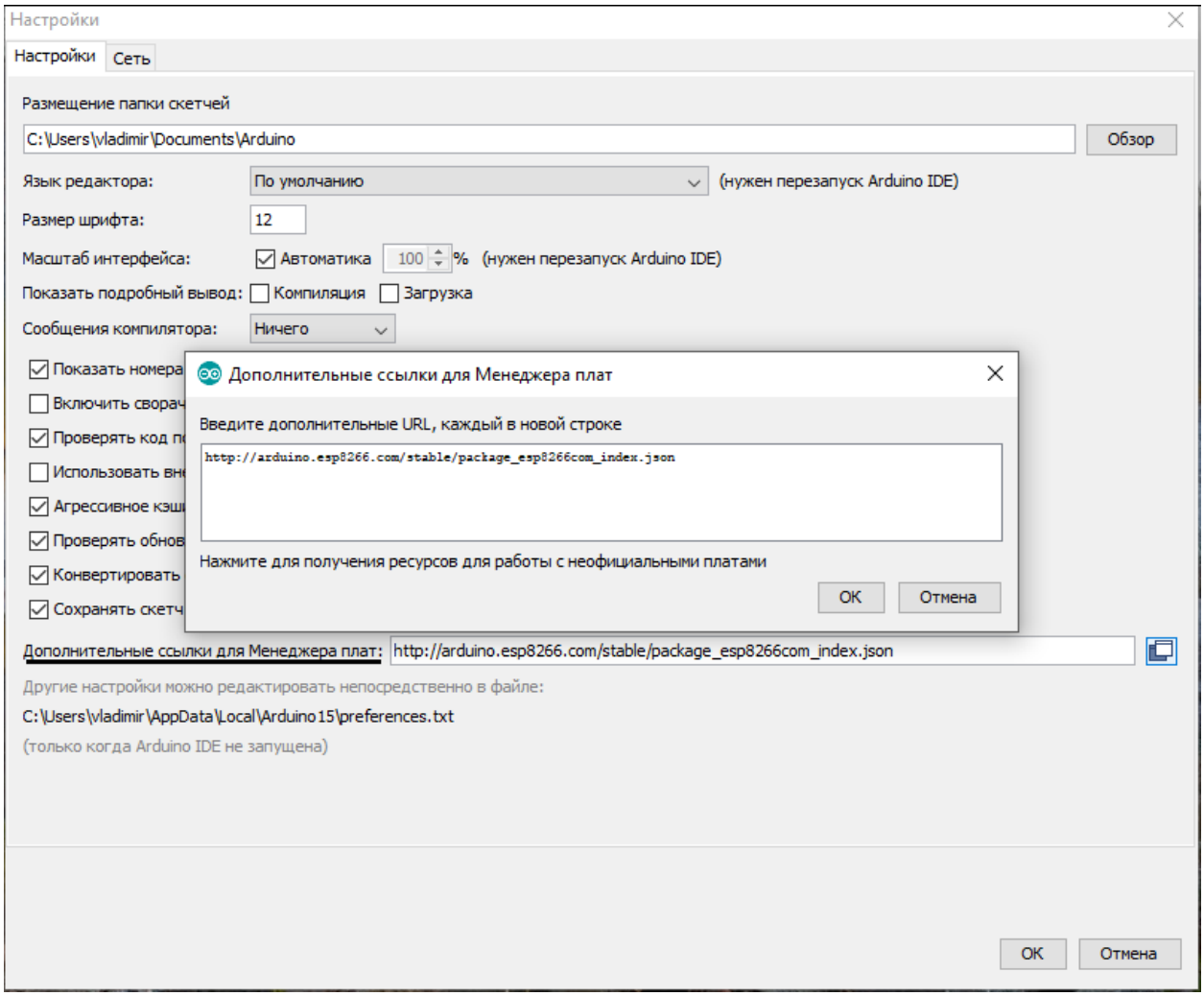


Рис. 5.21. Подготовка для добавления платы

Завершается эта часть настройки кнопкой **ОК** в первом и втором окне. Можно ввести адрес и в первом окне, но во втором, наверное, удобнее. Затем нужно перейти в основном меню к пункту *Инструменты→Плата→Менеджер плат*. В самой нижней части списка появится новый раздел, где следует выбрать внизу ссылку *More info*, вызывающую появление кнопки **Установить**. Нажав кнопку, следует дождаться загрузки платы (**рис. 5.22**).

Рис. 5.22. Установка новой платы в программе Arduino

После установки новой платы она появится в списке плат (**рис. 5.23**). И среди примеров, ранее установленных в программе, появятся примеры для работы с этой платой.

Заметьте, что примеры появятся после выбора платы: *Инструменты→Плата→WeMos D1 R1*.

Для проверки можно загрузить в модуль программу Blink. Мигающий светодиод будет рядом с антенной модуля WiFi. Работа с платой в программе Arduino аналогична работе с другими модулями Arduino.

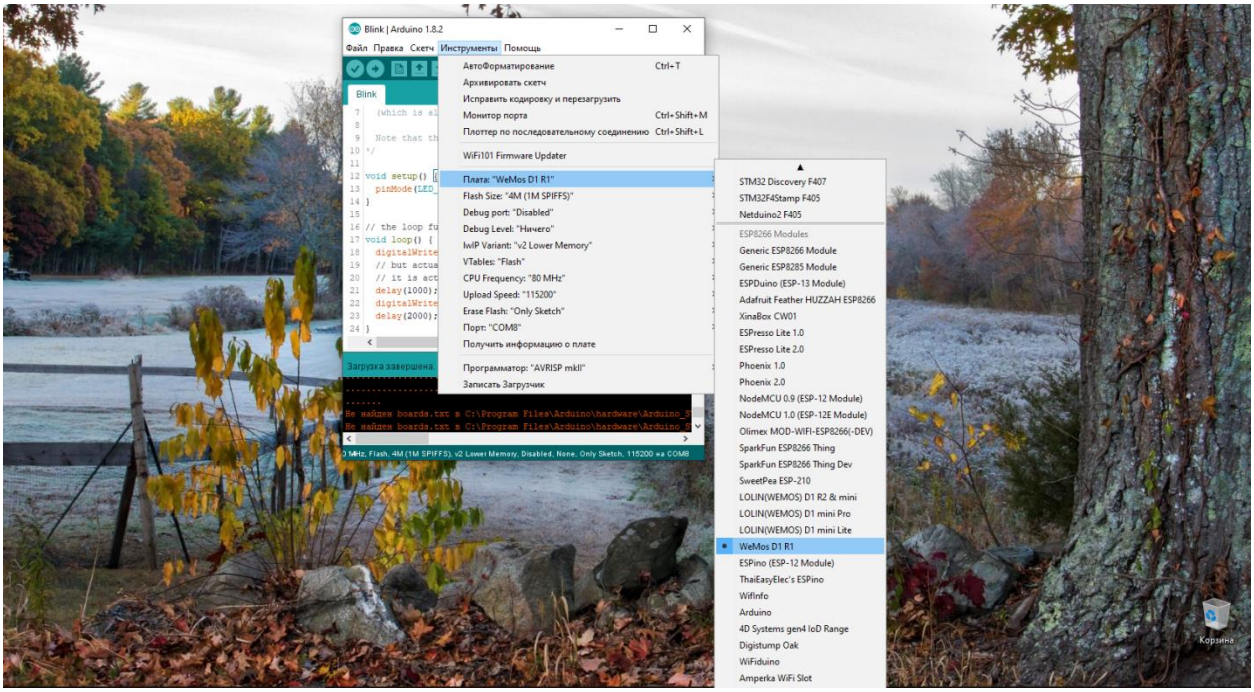


Рис. 5.23. Новая плата в программе Arduino

Производитель модуля, впрочем, предлагает проверить с помощью программы:

```
  void setup() {
    // добавьте ваш код настройки здесь, он выпонится один раз:
  Serial.begin(9600);
  }

  void loop() {
    // добавьте здесь ваш основной код для многократного
повторения:
  Serial.println("Hello World");
  delay(2000);
  }
```

Результат можно увидеть на мониторе порта (**рис. 5.24**).

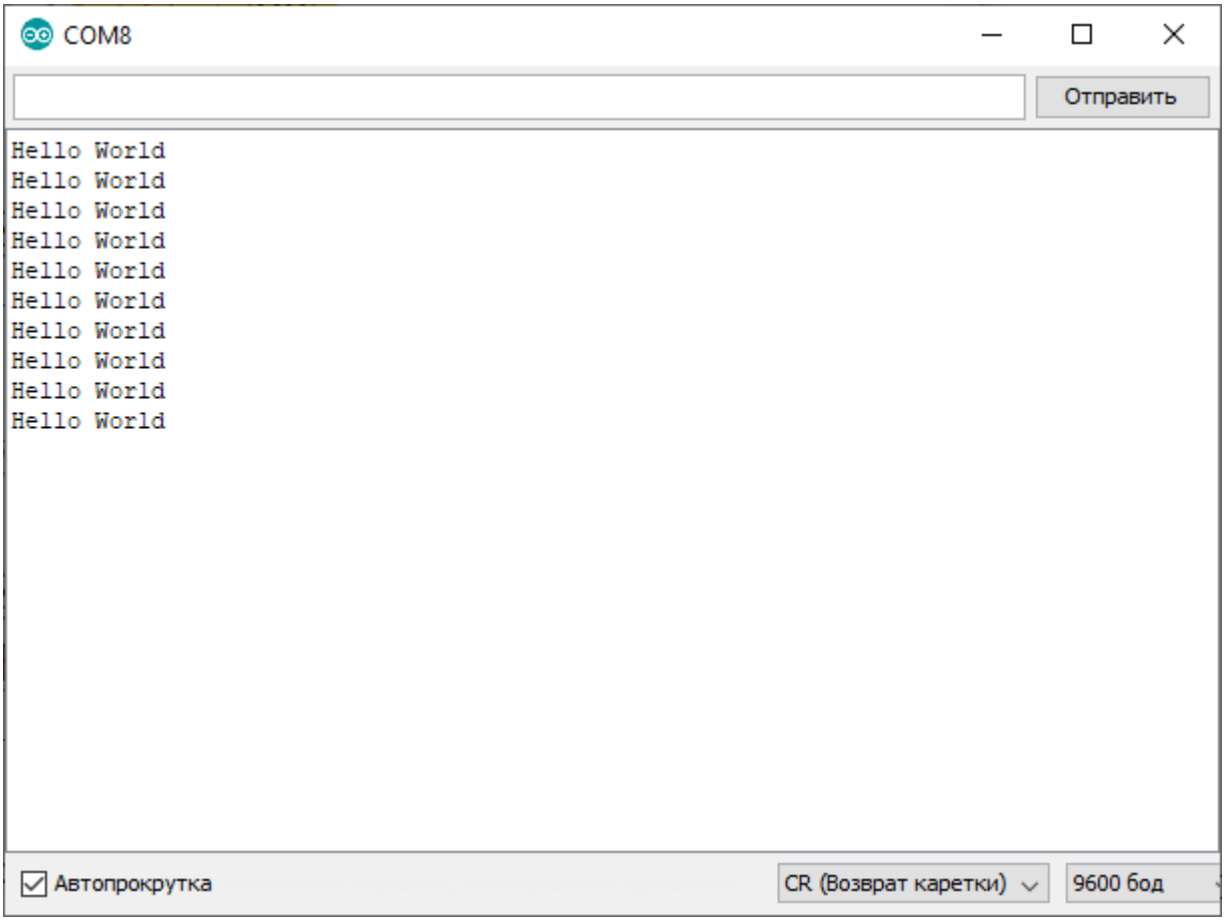


Рис. 5.24. Проверка работы модуля

Обрадованный предыдущим опытом, я решил, что теперь обойдусь без проблем.

Это я так решил, что без проблем, но не модуль OTA WeMOS D1.

Поскольку модуль в сети работает с WiFi, а модуль Raspberry Pi ранее был подключен только через Ethernet, я решил включить WiFi. В прошлый раз, работая над книгой «Raspberry Pi для любознательных», я подключал WiFi с помощью ОС Raspbian, мышки и клавиатуры – не заладилось иное подключение к роутеру. И опыт подключения клавиатуры оказался полезен, однако в этот раз я решил испробовать подключение через утилиту raspi-config. К ней можно обратиться и с помощью программы PuTTY, и в терминале (что я и

сделал) операционной системы Raspberry Pi. Выбрав раздел коммуникации, в диалоге подключения по WiFi достаточно ввести SSID – имя подключения, ввести passphrase – пароль для подключения (**рис. 5.25**).

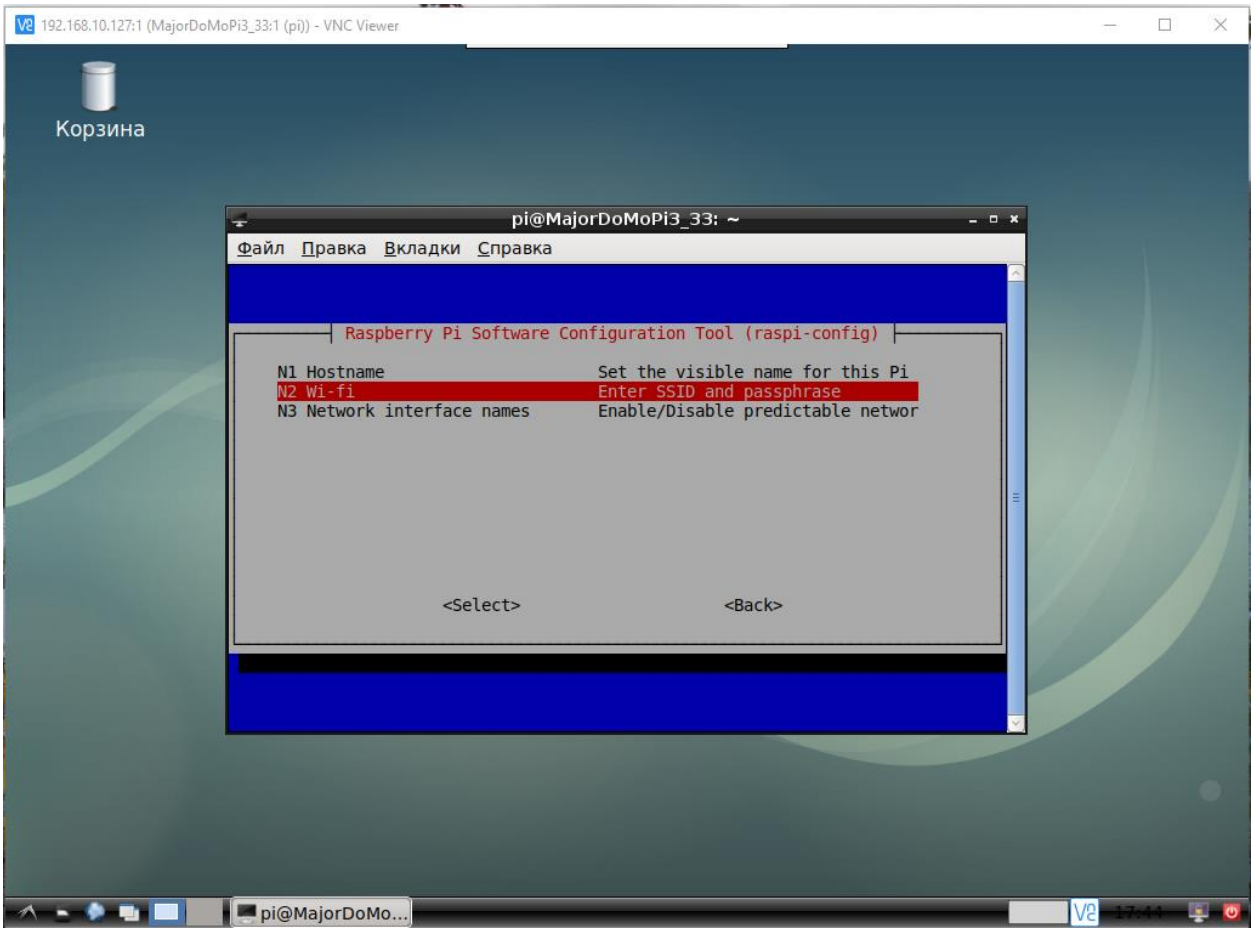


Рис. 5.25. Подключение по WiFi к домашней сети

Видимо, меня смутил тот факт, что среда разработки та же, что для модуля Arduino, поэтому я решил повторить программу, работавшую с Arduino Uno, скорректировав библиотеку функций.

Не буду рассказывать, что потратил полдня (вот и проговорился) на попытки отправить sendRequest() в переделанной программе WebClient. Видимо, запрос запросу рознь. Потом, махнув рукой на местечковые разборки, поискал подходящую статью в Интернете **[9]**. Программа для WeMOS D1 получилась такой:

```
#include <ESP8266WiFi.h>
WiFiClient client;
const char* ssid = "your-ssid"; // Ваш логин
const char* password = "your-password"; // Ваш пароль
String host = "192.168.10.127"; // Строка ip-адреса сервера
int stat=0; // Переменная для состояния датчика

void setup() {
  Serial.begin(115200); // Скорость последовательного порта
  delay(10);
```

```
    pinMode(4, INPUT); // Вывод 4 для чтения состояния датчика

    // Connect to WiFi network
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) { // Пока нет
соединения
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected"); // Есть подключение по
WiFi

  if (!client.connect(host, 80)) { // Если нет подключения к
серверу
    Serial.println("connection failed");
    return;
    }
  }

  void loop() {
  client.connect(host, 80); // Подключаемся к серверу, порт 80
    stat=digitalRead(4); // Читаем состояние датчика
    if (stat==1) { // Если состояние единица
  // Отправляем строку запроса с состоянием 1
   client.print(String("GET
/objects/?object=garage&op=m&m=status_d_input&status=1") + "
HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection:
close\r\n\r\n");
    }
    else { // Иначе отправляем строку запроса с состоянием 0
     client.print(String("GET
/objects/?object=garage&op=m&m=status_d_input&status=0") + "
HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection:
close\r\n\r\n");
    }
    delay(4000);
   }
```

Программы похожи, но не одинаковы. Возможно, в первом случае модуль использовал проводное подключение к роутеру, а во втором случае подключение по WiFi, но предположение требует проверки. Не следует забывать и то, что этот модуль не Arduino.

Загрузка в модуль выглядит несколько необычно – она происходит с выводом множества точек в окно сообщений (**рис. 5.26**).

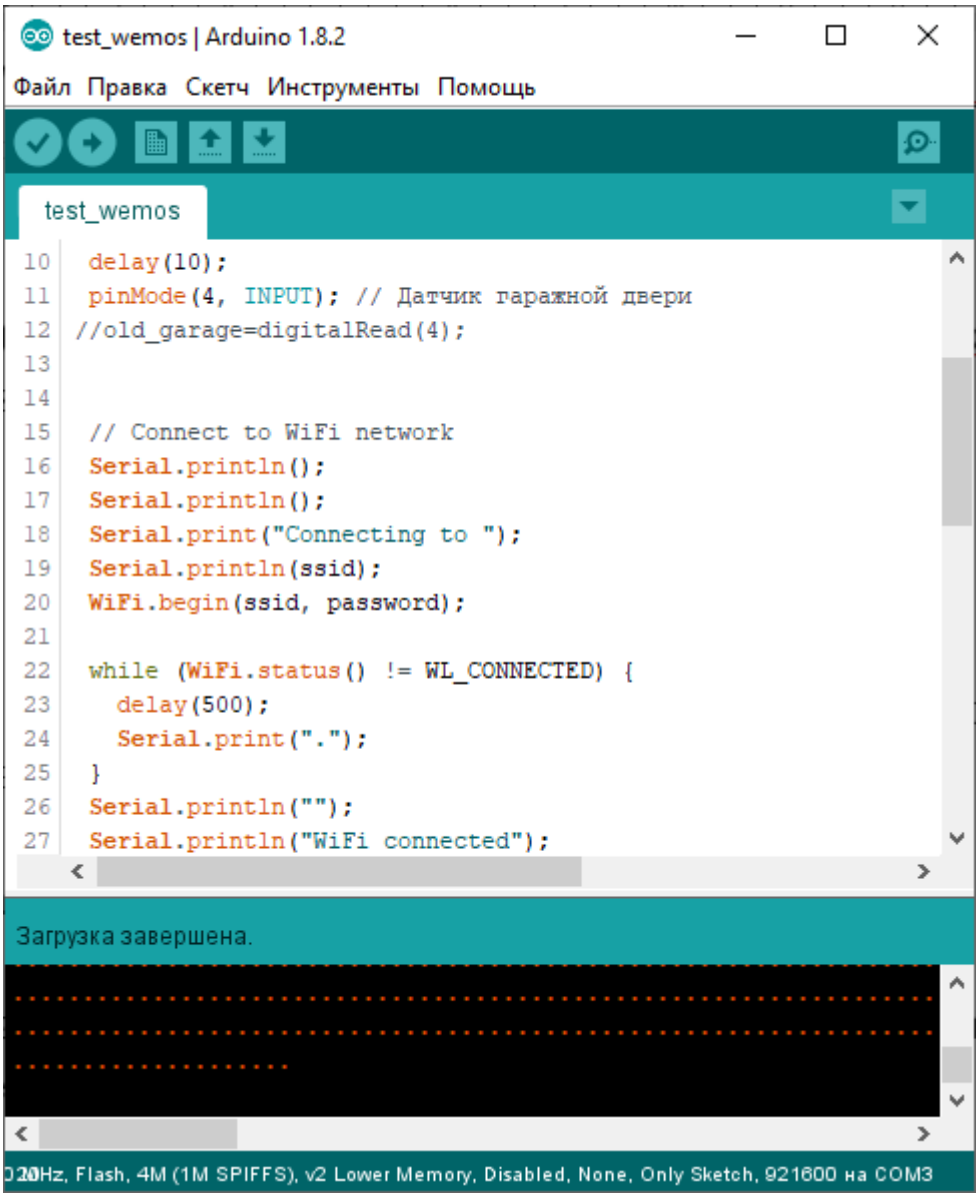


Рис. 5.26. Загрузка скетча в модуль WeMOSS D1

Но теперь переключение входа 4 модуля WeMOSS D1 дает ту же картину, что на **рис. 5.19**.

Модуль ESP8266 с реле

До начала экспериментов я советую дочитать этот раздел.

Этот модуль имеет выводы, которые позволяют подключить его к Arduino, настроенному как переходник USB-TTL (**Рис. 5.27**).

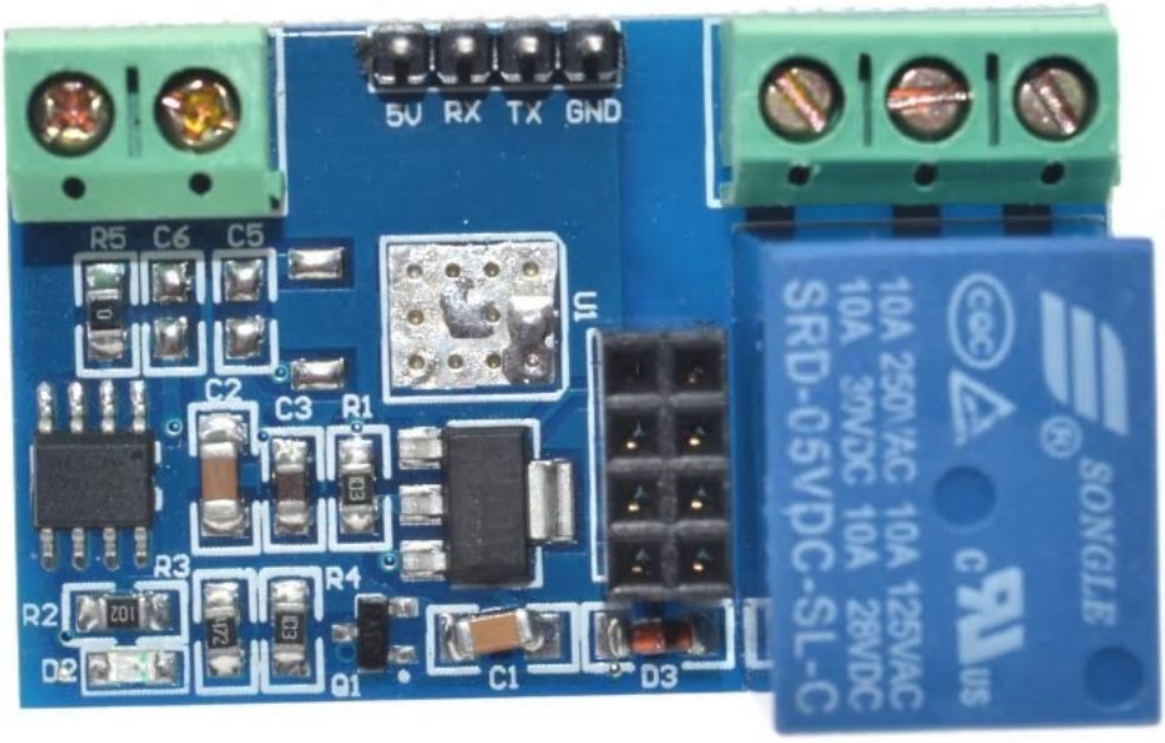


Рис. 5.27. Выводы модуля ESP8266 с реле

С этим релейным модулем, как выяснилось позже, не все так безоблачно.

У Arduino Uno (я использую его) следует соединить вывод сброса (он обозначен как RES) с GND. Как говорят знающие люди, этим Arduino переводится в режим преобразователя USB-TTL. Выводы TX и RX соединяются с выводами TX и RX, а не наоборот, как было бы привычно (см. Приложение Б, **рис. Б.2**). В программе Arduino выбираем плату (библиотеку, полагаю, вы загрузили ранее) Genric ESP8266 Module (**рис. 5.28**).

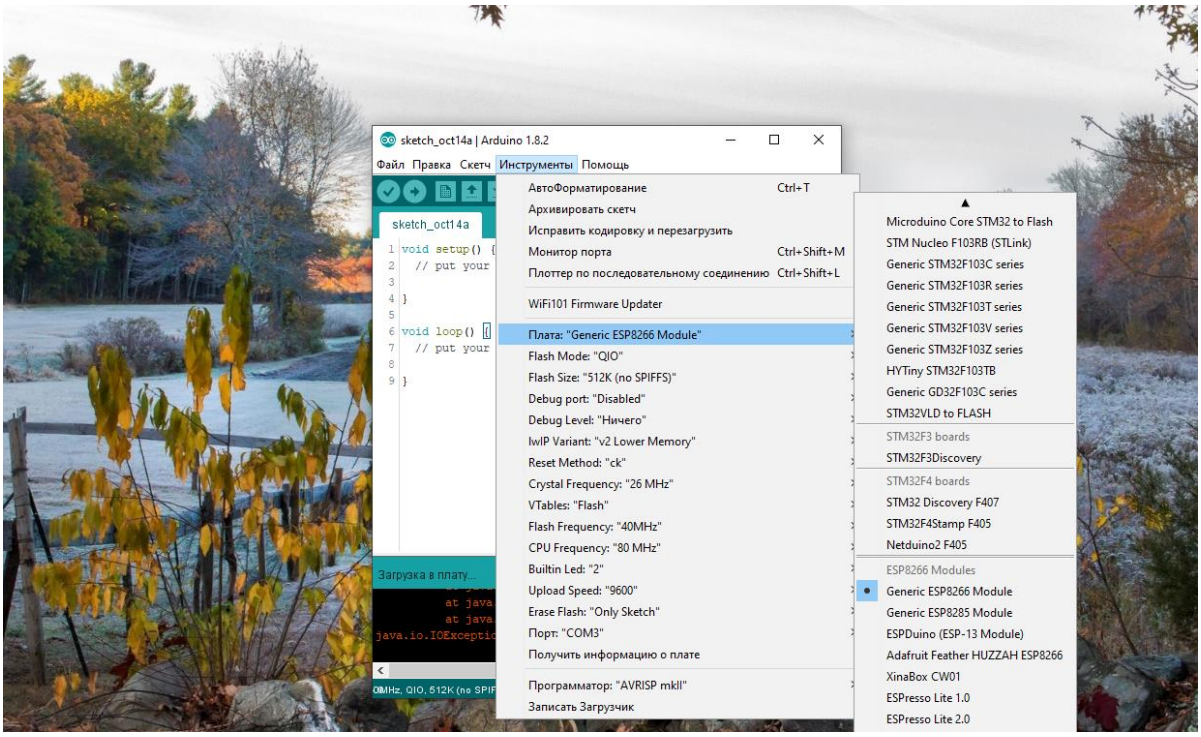


Рис. 5.28. Выбор платы в программе Arduino

Если открыть монитор порта, настроив его на скорость 74880 бод с NL&CR, подключить вывод 5V на разъеме Arduino с выводом 5V модуля ESP8266, то на мониторе должна появится такая информация (**рис. 5.29**).

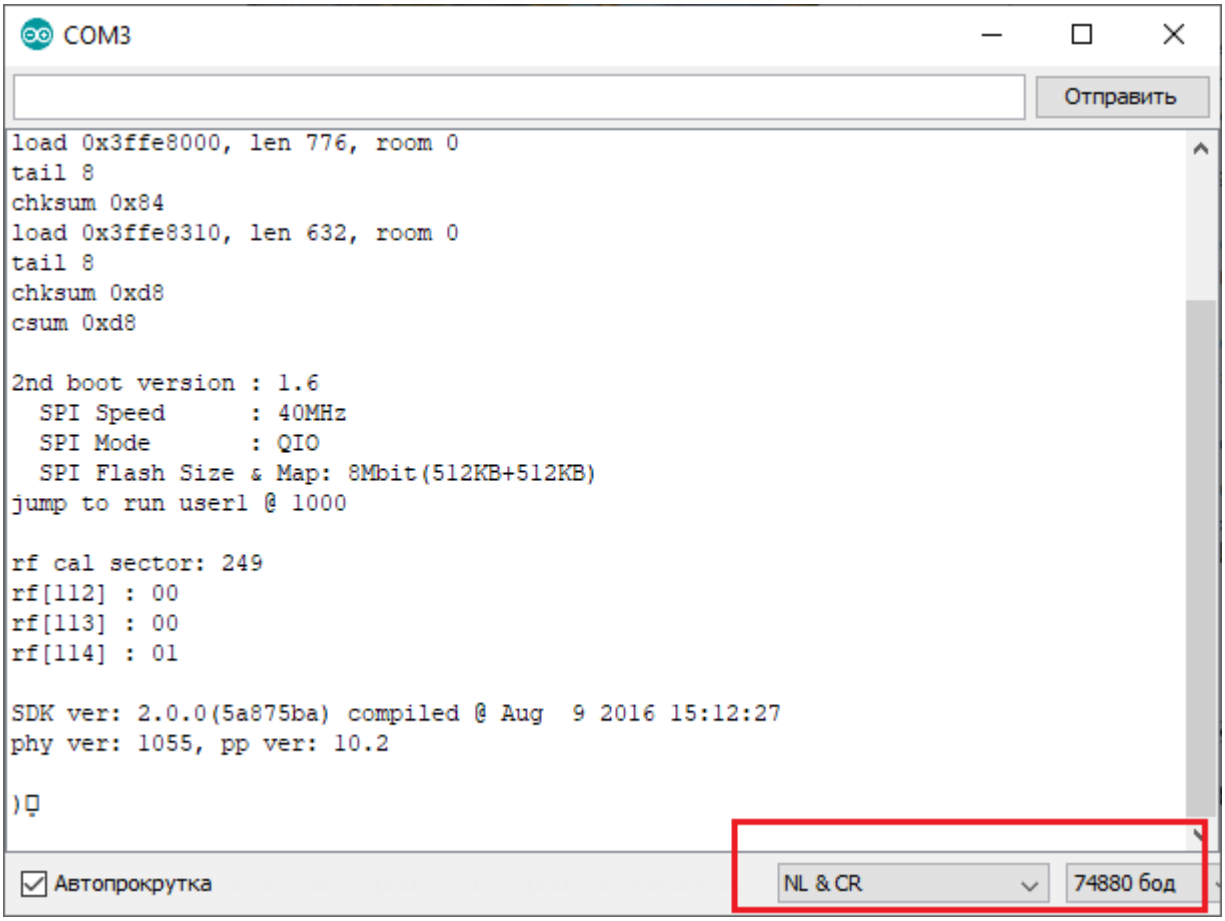


Рис. 5.29. Монитор порта Arduino при подключении ESP8266

Что такое NL&CR? Это команды перевода на новую строку и возврата каретки. Монитор порта добавляет их автоматически. Продолжим.

Следующее действие – нужно изменить скорость работы на 115200 бод, отключить и подключить питание. Эти манипуляции приводят к появлению сообщения о готовности модуля, предваряемого непонятным набором символов (**рис. 5.30**). Для убедительности я повторил процедуру несколько раз.

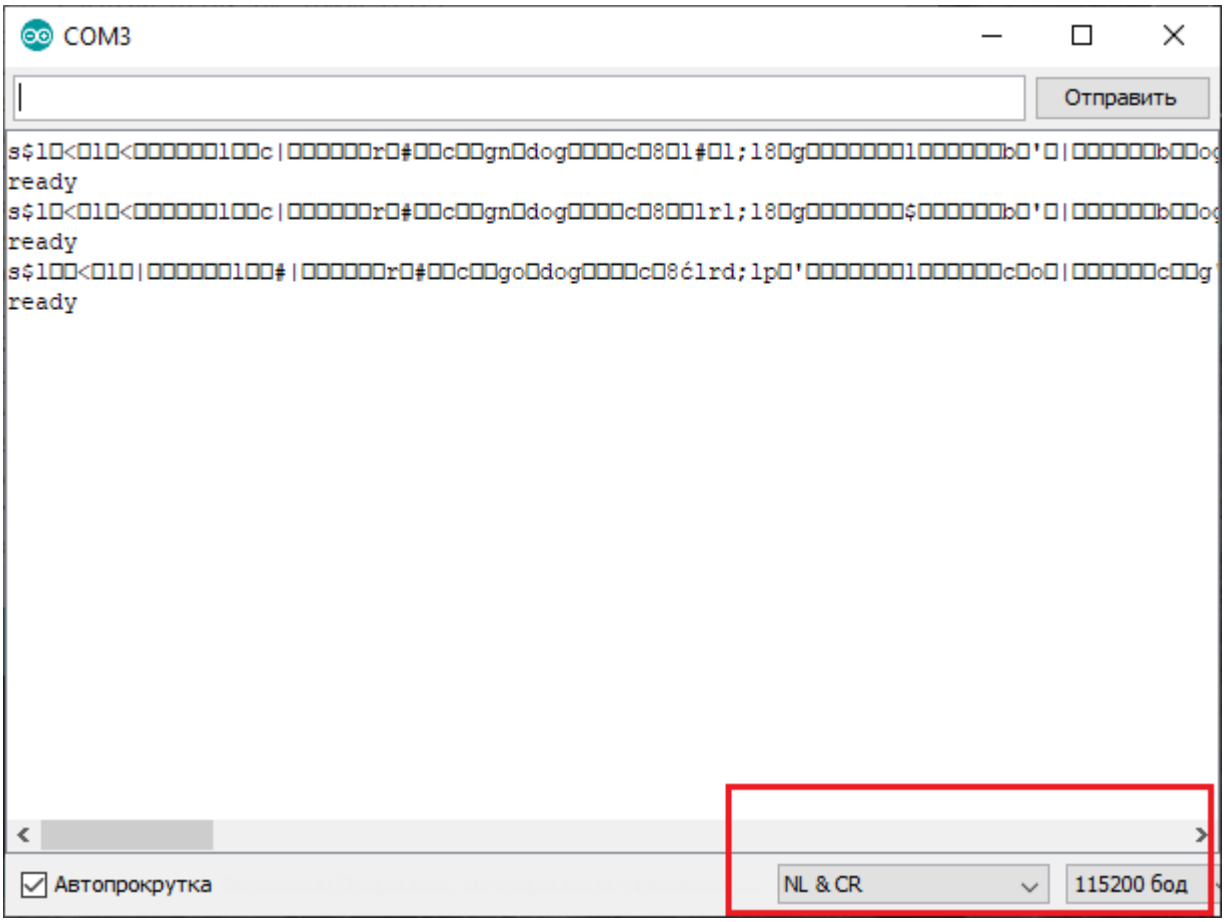


Рис. 5.30. Готовность модуля к продолжению работы

К моему глубокому разочарованию дальше этого я не продвинулся ни на шаг. Я прочитал с десяток статей в Интернете, перепробовал ряд советов, но результат был… да не было результата, если сказать честно. Всю информацию и настройки модуля ESP8266 осуществляют командами AT, которых много (см. Приложение А). Но начинать следует с отправки префикса AT, на который модуль должен ответить OK. Я пытался это сделать, но команды не проходили. К этому времени у меня скопилось множество полезных (но не мне) программ. Иной раз, когда щелкаешь по разным кнопкам программы что-то появляется на мониторе программы, но никак не то, что нужно.

Небольшой сдвиг произошел только тогда, когда я подключил USB-WiFi модуль к своему стационарному компьютеру, обычно он подключен у меня кабелем к роутеру. Теперь компьютеру стали доступны и сети WiFi. Среди них появилась и такая точка доступа (**рис. 5.31**).

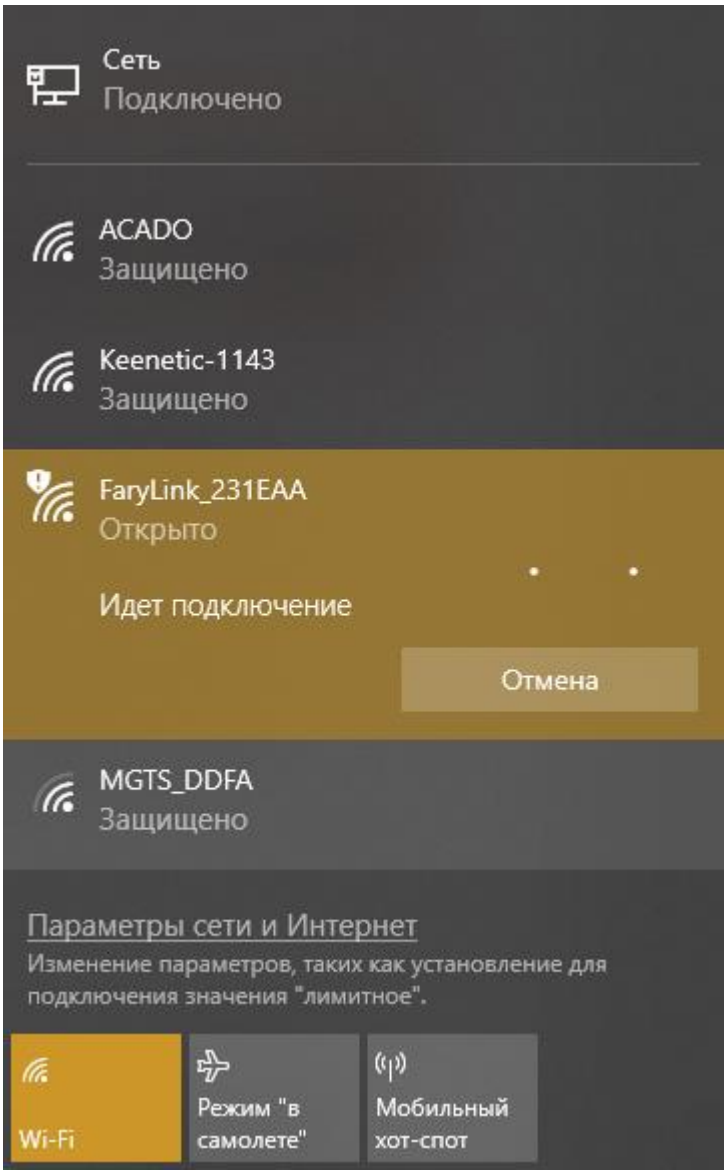


Рис. 5.31. Появление модуля ESP8266 в сети

Мои модули, их два, работают точкой доступа с именем FaryLink_231EAA (второй с постфиксом 231BAB). В статьях я встречал другие имена модулей, видимо, разные производители по-разному называют свои изделия. Если теперь в командной строке ввести запрос: `arp -a`, то можно увидеть свой модуль (**рис. 5.32**).

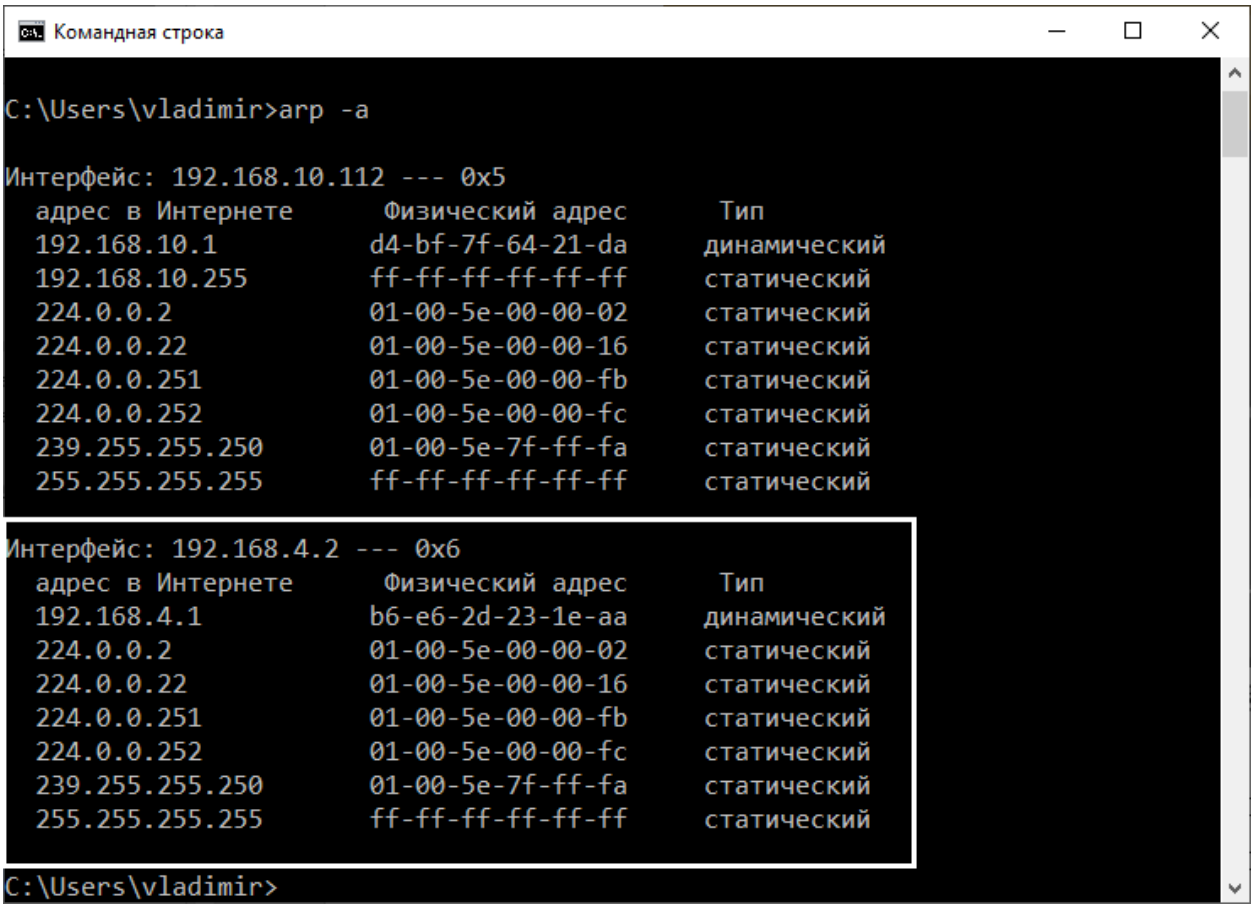


Рис. 5.32. Информация о модуле и подключенном компьютере

Модуль ESP8266 с реле, как мне кажется, разрабатывался для использования со смартфоном. Во многих статьях я встречал совет установить на смартфон программу EasyTCP_20, что и сделал. Смартфон тоже обнаруживает модуль ESP8266, но подключаться к нему не желает. В прошлый раз, когда не получалось установить драйвер для дисплея Raspberry Pi, я посетовал об этом продавцу, получив от него быстрый ответ со ссылкой на нужный драйвер. Но не в этот раз.

Я не уверен, что мой продавец сам знает, в чем причина, или знает, но не хочет об этом говорить.

Пришлось прибегнуть к запасному варианту. Модуль ESP8266 WiFi давно известен в любительских кругах, с ним работают многие, и работают успешно. Поэтому я снял его платформы для реле, выбрал одну из статей, которая мне приглянулась (**рис. 5.33**), и повторил описанные в ней действия. Статья достаточно познавательная, чтобы потратить немного времени на ее чтение **[10]**.

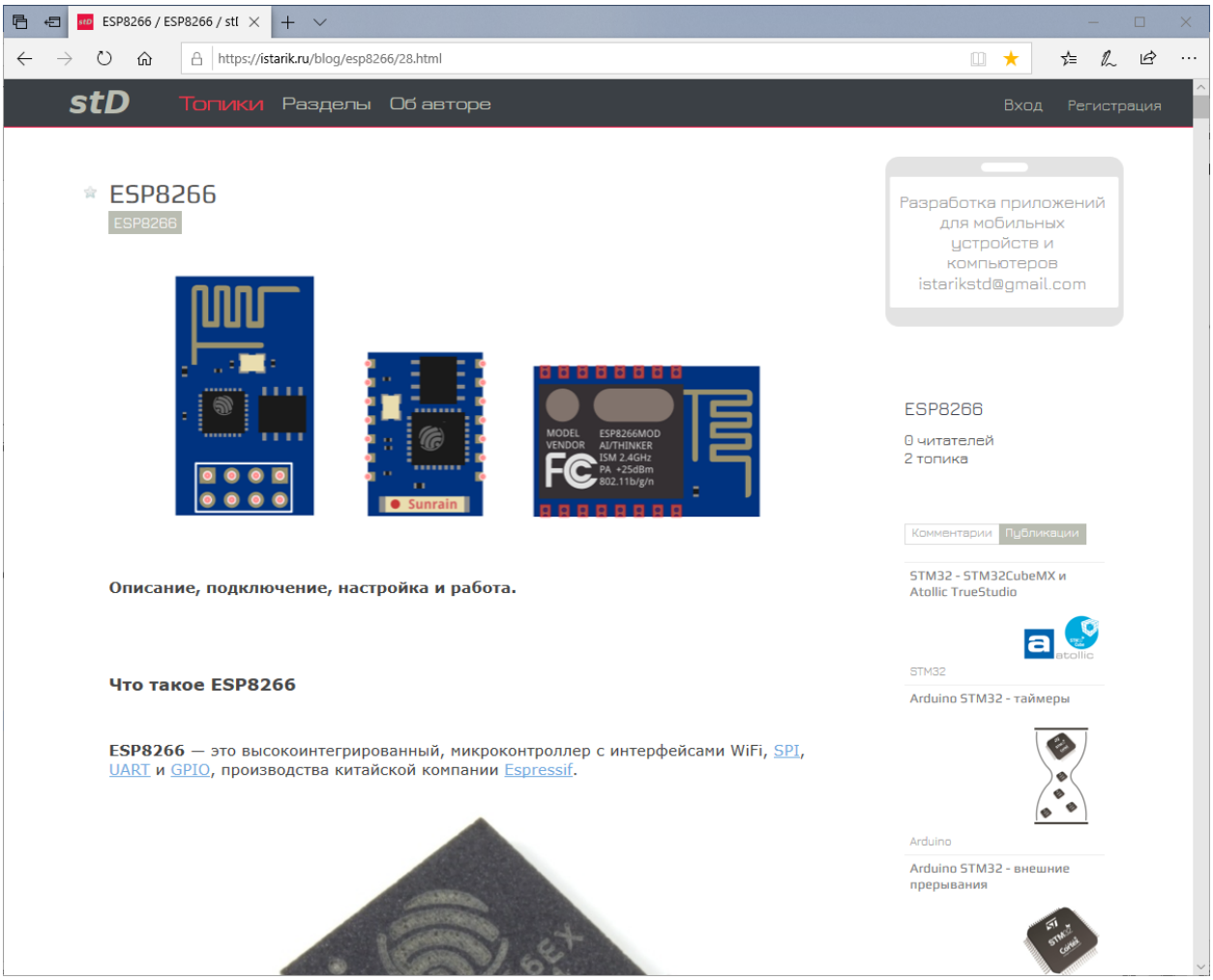


Рис. 5.33. Статья о работе с модулем ESP8266

Включение Arduino остается прежним, а подключение к ESP8266 осуществляется согласно с назначениями его выводов (**рис. 5.34**).

Это важно!

Будьте внимательны, питание 3,3 В!

Я в какой-то момент перепутал с включением релейного варианта, переходя от одного подключения к другому, что привело к тому, что модуль перестал отвечать на команды (со временем, впрочем, он одумался), когда я подключил его к 5 В. Еще более точную картину подключения модуля к Arduino, вы увидите в статье, которую я рекомендовал.

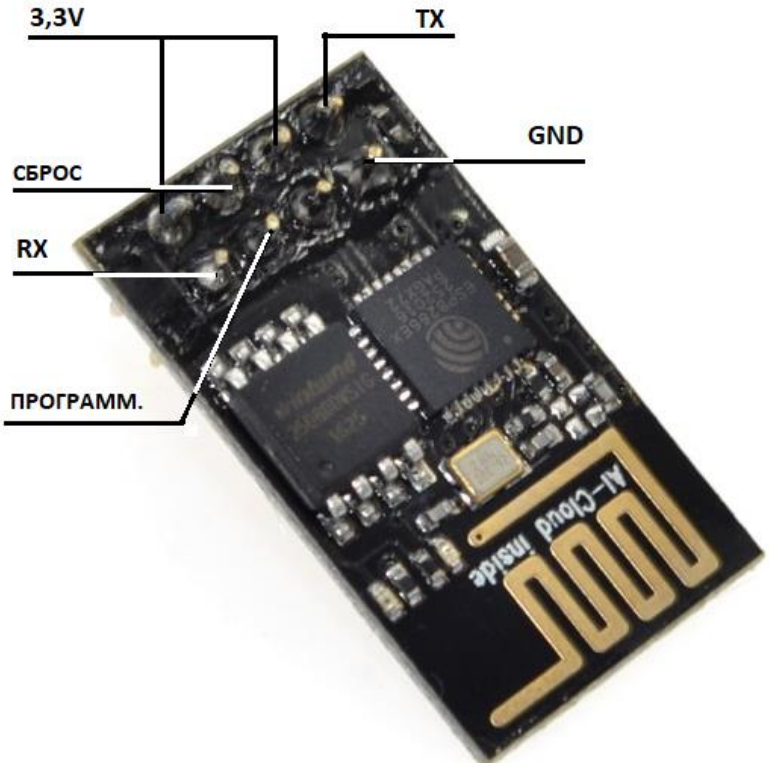


Рис. 5.34. Названия выводов платы WiFi модуля ESP8266 с реле

Теперь, как и пишут об этом, модуль отвечает на АТ-команды (**рис. 5.35**). Я повторил многие из команд, что и вам советую. Наиболее интересные из них я покажу в приложении А.

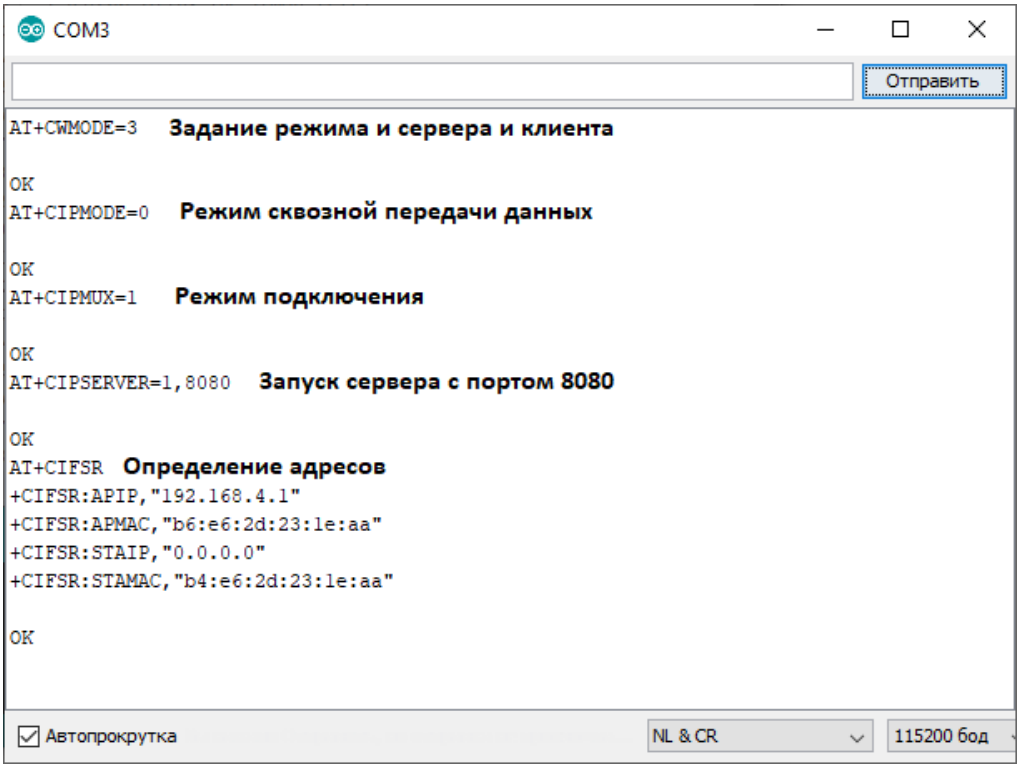


Рис. 5.35. Последовательность команд перед продолжением опытов

Теперь можно проверить соединение моего компьютера (с WiFi модулем) и модуля ESP8266. Открыв браузер, введем адрес модуля: http://192.168.4.1:8080. Браузер ничего не отобразит, но в мониторе порта Arduino появится запрос к ESP8266 (**рис. 5.36**).

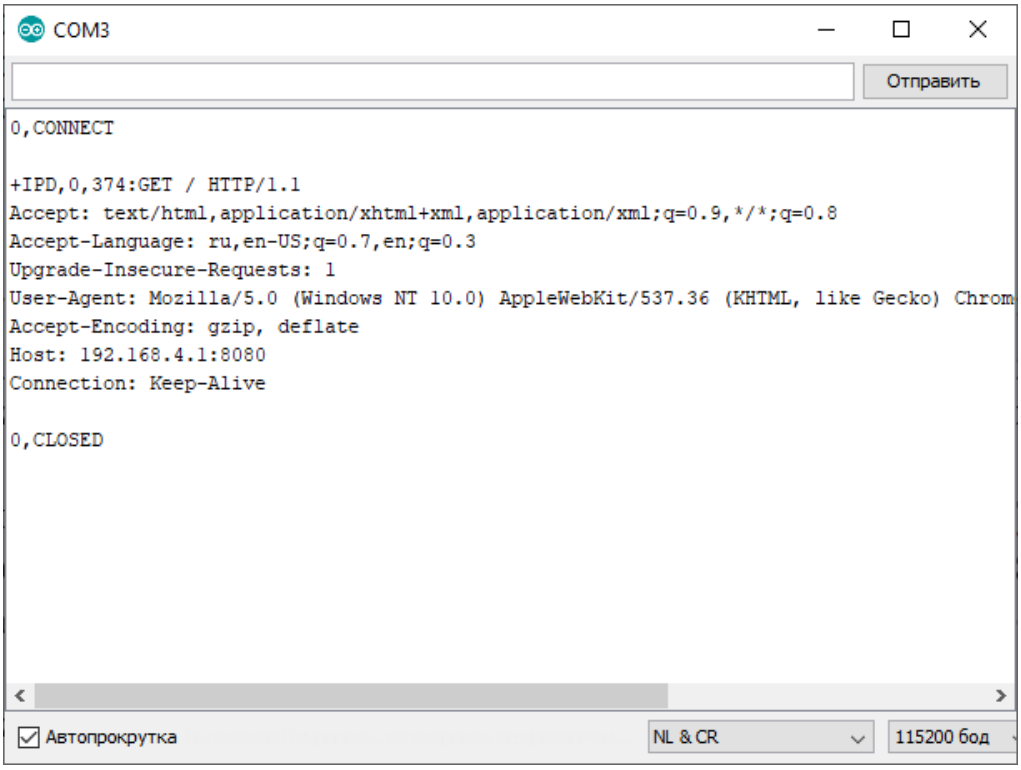


Рис. 5.36. Отображение обращения к модулю ESP8266

Есть такая программа SocketTest **[11]**. Запустив эту программу, можно ввести на закладке *Client* адрес и порт модуля, что отобразится в мониторе порта Arduino. Можно ввести в строку сообщения что-нибудь (**рис. 5.37**).

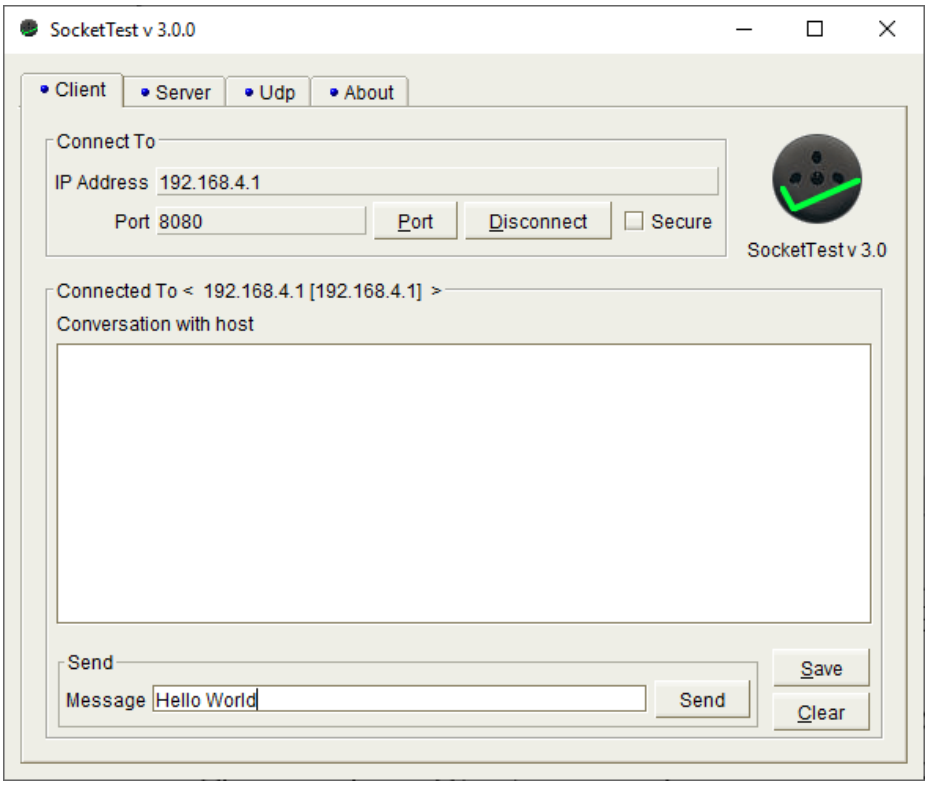


Рис. 5.37. Использование программы

Монитор порта Arduino отобразит такую информацию (**рис. 5.38**).

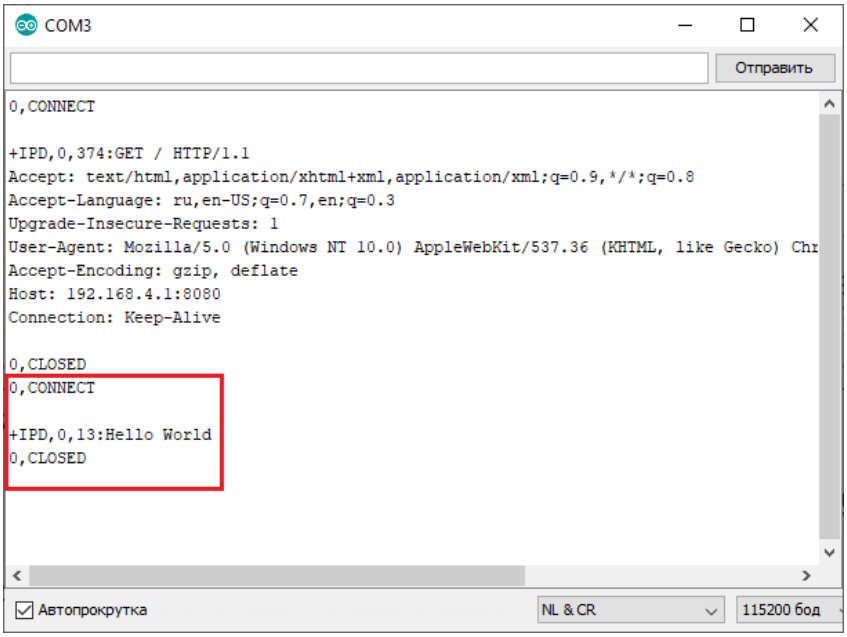


Рис. 5.38. Реакция модуля на сообщение

И теперь подключение со смартфона проходит, программа работает, можно переместиться к закладке *Switch*, чтобы управлять реле.

До этого момента удачная работа касалась WiFi модуля, а релейная часть лежала в стороне. Причина этого – после соединения платы WiFi с релейной половиной ни одна из команд AT не проходит. Пришлось отключить модуль WiFi от реле и вернуться к работе с ним.

В ряде статей я встречал замечание – необходимо установить скорость работы WiFi модуля 9600. Но попытка осуществить это штатной командой `AT+CIOBAUD=9600` давала результат, на который я поначалу не обратил внимания (**рис. 5.39**).

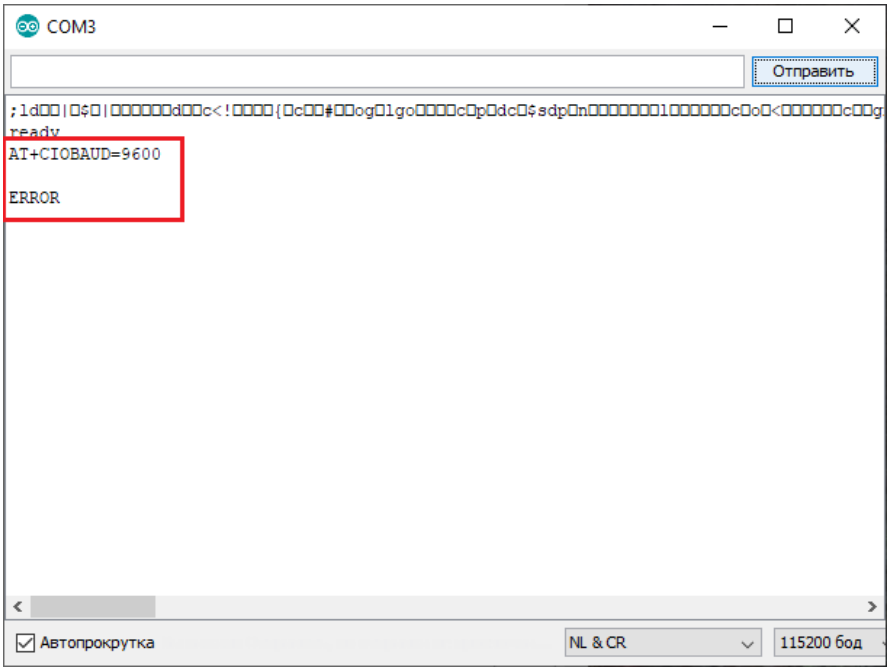


Рис. 5.39. Неудачная попытка изменить скорость работы порта

Необходимость смены скорости последовательного порта объясняется наличием в релейной части ESP8266 контроллера STC15F104W, который может работать только на скорости 9600 бод.

Для продолжения рассказа, конечно, достаточно WiFi части, это так, но, наученный горьким опытом ряда экспериментов, которые я не доводил до конца, то есть, до выяснения причины появившихся проблем, я решил попробовать программы, что накопились за это время, вдруг поможет. Штатной программой можно считать USR-TCP232-Test. На нее ссылаются часто, ее высылают некоторые продавцы ESP8266, если обратиться к ним за помощью.

Сразу скажу, у меня с программой по началу отношения не сложились совсем – ни одной АТ команды отправить модулю не получилось (**рис. 5.40**).

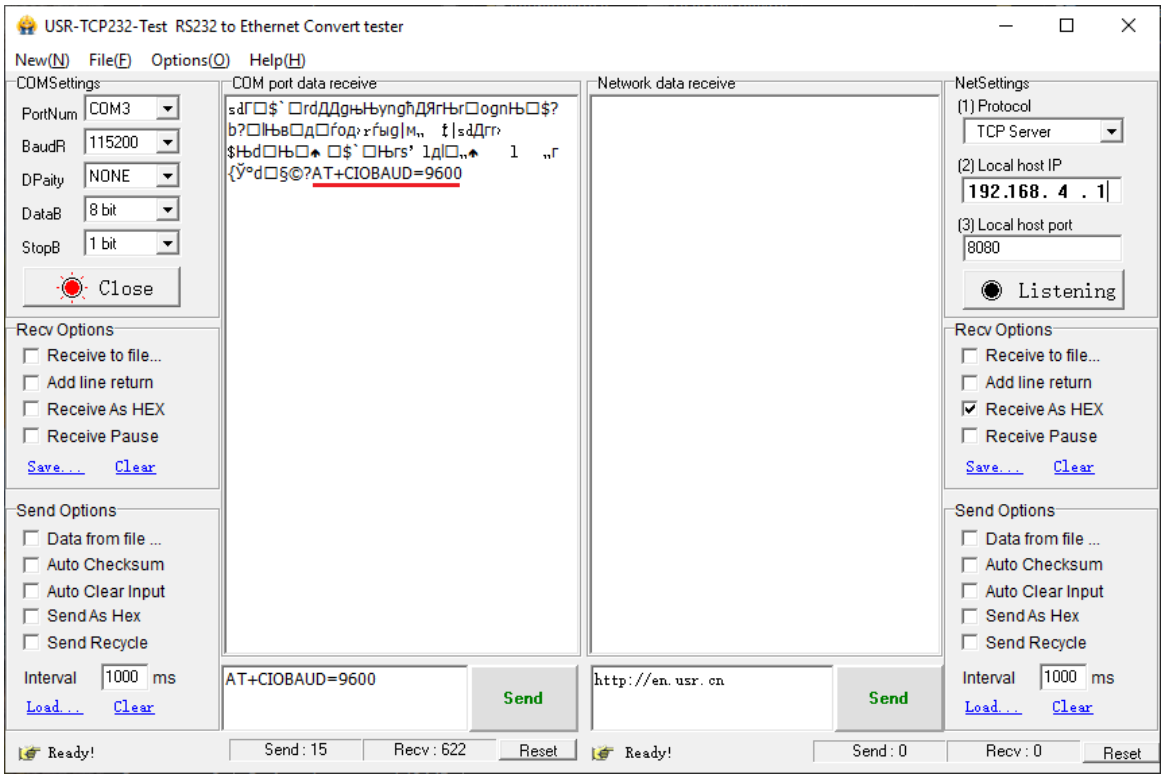


Рис. 5.40. Программа USR-TCP232-Test

Возможно, у вас сразу получится лучше.

Еще одна программа, ESPlorer, позволяет вводить команды в окошко команд или использовать кнопки для выполнения этих команд. Если я правильно понимаю, она написана на языке Java, и может использоваться с разными операционными системами. Но и она оказалась бессильна в борьбе за воссоединение двух плат модуля ESP8266 с реле (**рис. 5.41**).

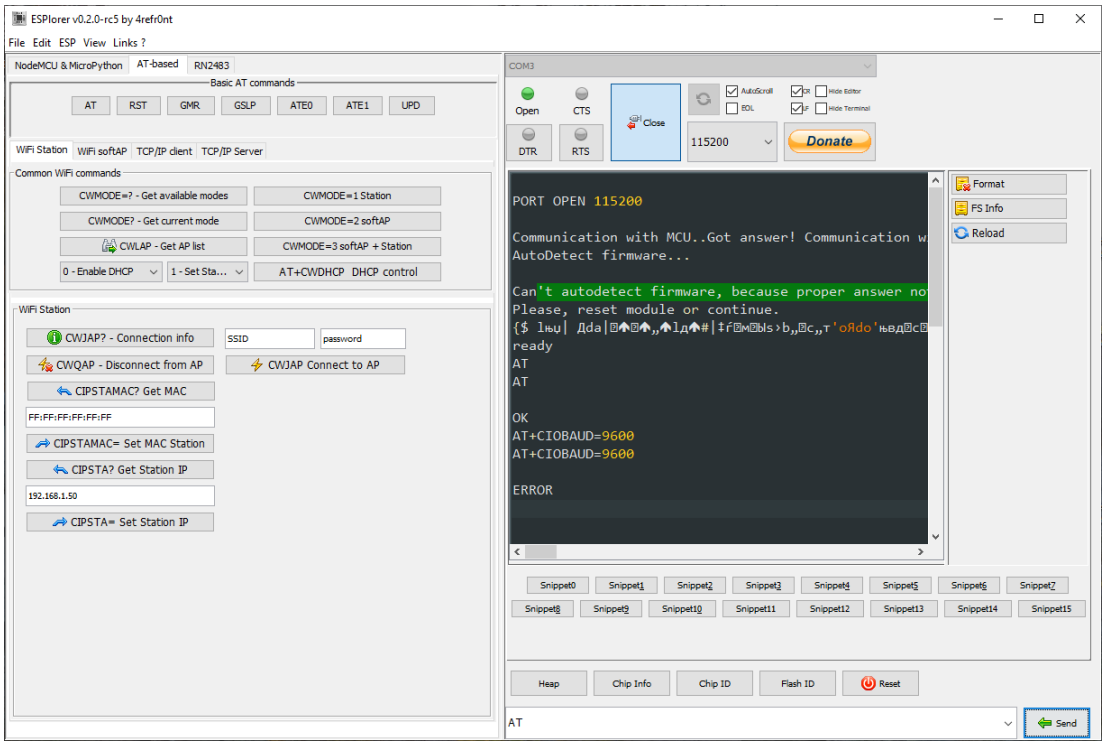


Рис. 5.41. Программа ESPlorer

Я не думаю, что разобрался бы с проблемой, перебирая разные программы, но они есть, и упомянуть их, думаю, стоило. Хотя достаточно и программы Arduino. Решение пришлось искать в Интернете, где в одной из статей я наткнулся упоминание другой команды для смены скорости последовательного порта: `AT+IPR=9600` (**рис. 5.42**). Реакции на эту команду нет, но переход на скорость 9600 в мониторе порта Arduino показывает, что скорость изменилась. Я использовал эту команду с WiFi модулем, который ранее подключил по ошибке к питанию 5 В, решив, что терять нечего.

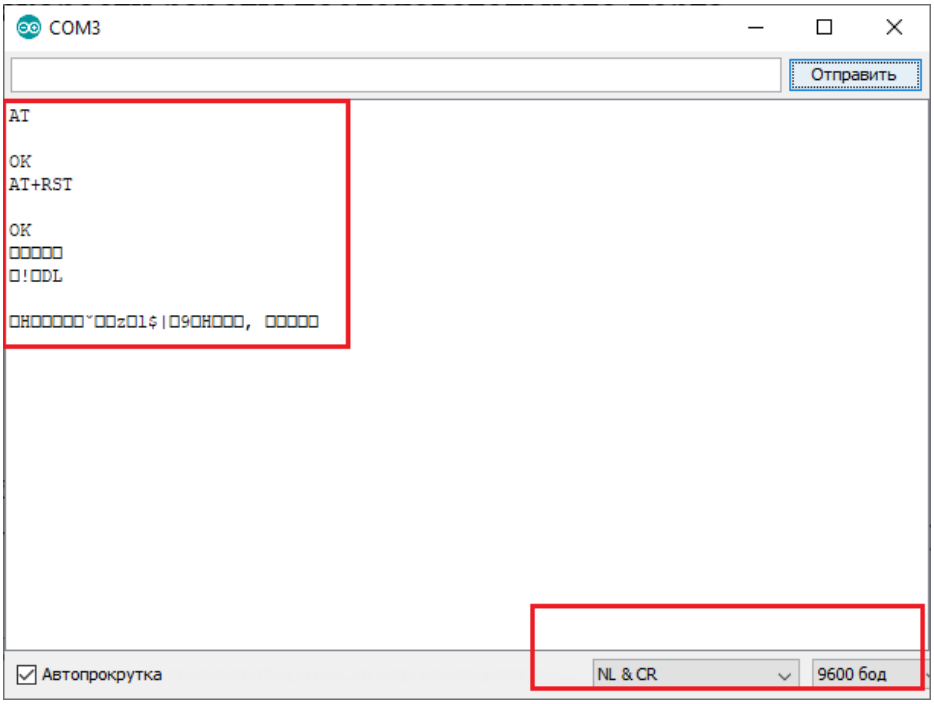


Рис. 5.42. Удачная смена скорости работы последовательного порта

Теперь можно вернуть WiFi модуль на плату и проверить, будет ли собранный модуль ESP8266 с реле отвечать на AT команды (**рис. 5.43**).

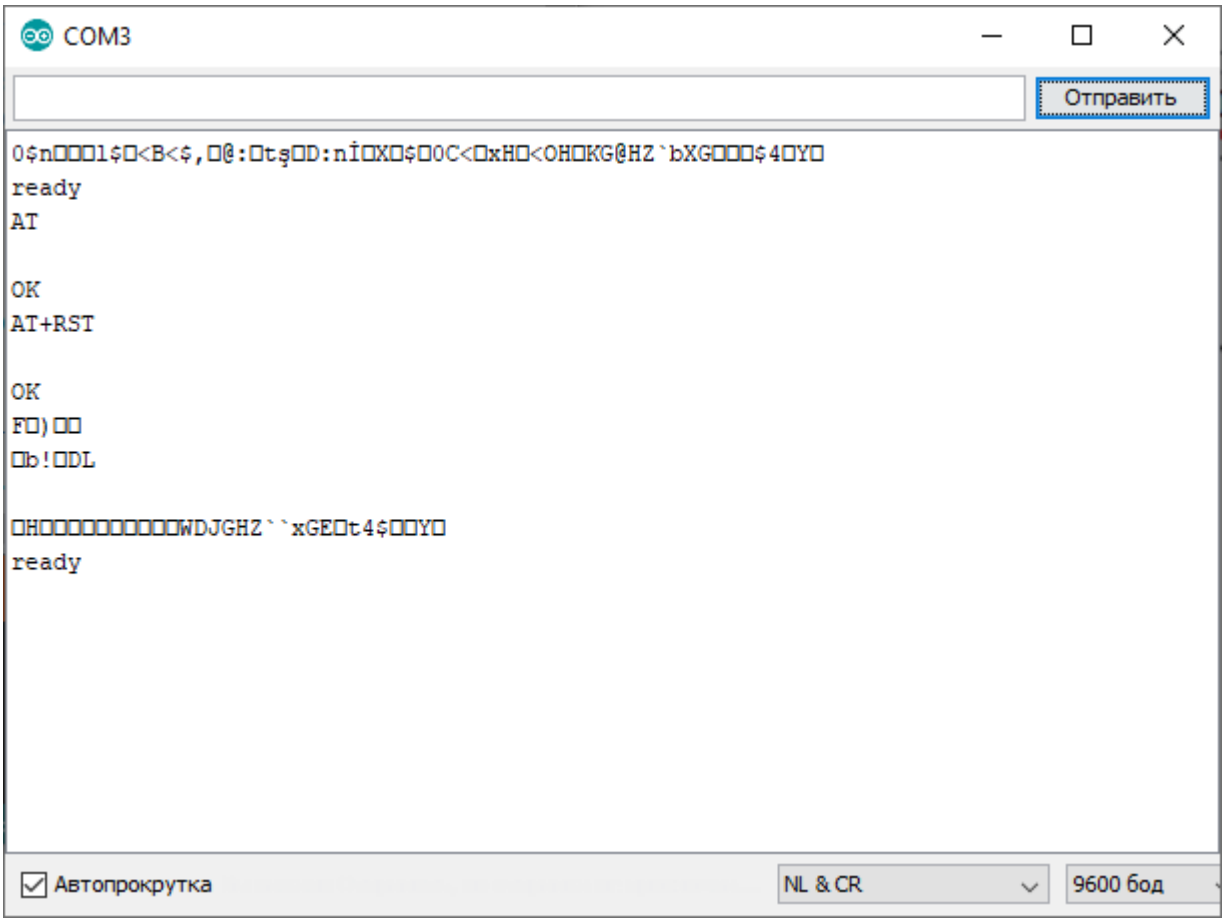


Рис. 5.43. Работа собранного модуля с программой Arduino

Конец приключениям? Но без приключений скучно!

У меня два одинаковых модуля. Все описанные выше опыты я проделывал с одним из них. Но сейчас решил повторить смену скорости работы последовательного порта у второго WiFi модуля (так мне было удобнее) командой `AT+IPR=9600`, и что? После соединения двух плат светодиод возле антенны на плате WiFI горит постоянно, а AT команды, которые я пытаюсь отправить, игнорируются. Модуль так ушел в себя, что ему не до моих команд.

Я подозреваю, что в своих экспериментах что-то сделал не должным образом. Я не знаю, как отвлечь модуль от его текущей работы, поэтому вновь отправлюсь на поиски по страницам Интернета.

Многие советуют в данном случае «перепрошить» ESP8266 WiFi модуль. Найденная мною статья **[12]** позволяет загрузить все необходимое для этой операции, в которой участвует программа FLASH_DOWNLOAD_TOOLS и набор необходимых файлов для загрузки ESP_SDK_v2.0.0. Правда, как и автор статьи, я использовал более позднюю версию инструмента загрузки, что, полагаю, не обязательно.

Я не использовал кнопки, как справедливо советует автор, а использовал провода для манипуляций с выводами сброса и программирования. Чтобы не оставлять вас в неведении, что это за выводы, я исправил **рис. 5.34**, добавив выводы для программирования. Последовательность операций такова:

- подключаем питание 3.3V к модулю;
- запускаем FLASH_DOWNLOAD;
- выбираем в диалоге программы кнопку с модулем ESP8266 (**рис. 5.44**);

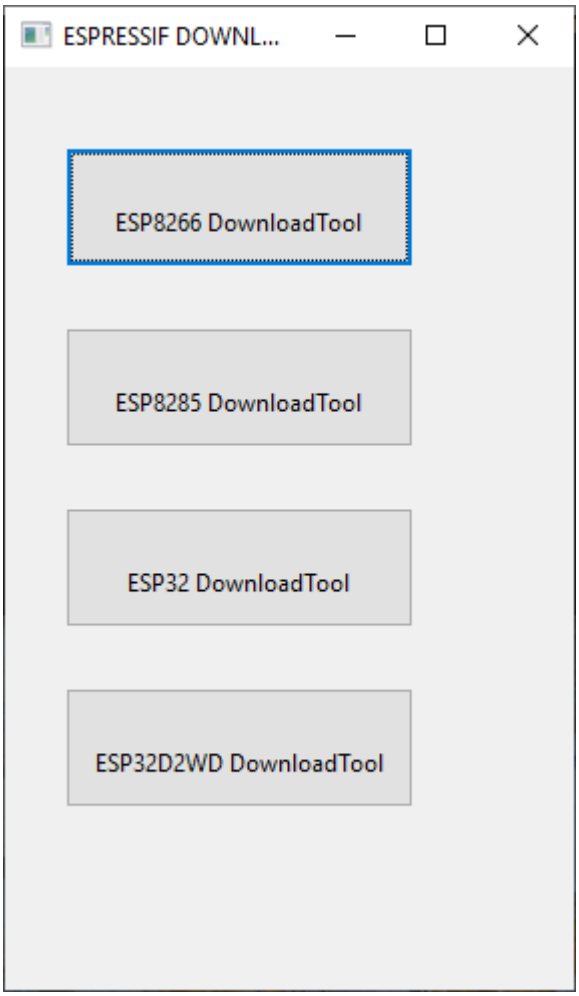


Рис. 5.44. Выбор своего модуля

- в появившемся окне (**рис. 5.45**) проверяем соответствие SPI SPEED и SPI MODE значениям 40 МГц и QIO; все галочки выше снимаем;

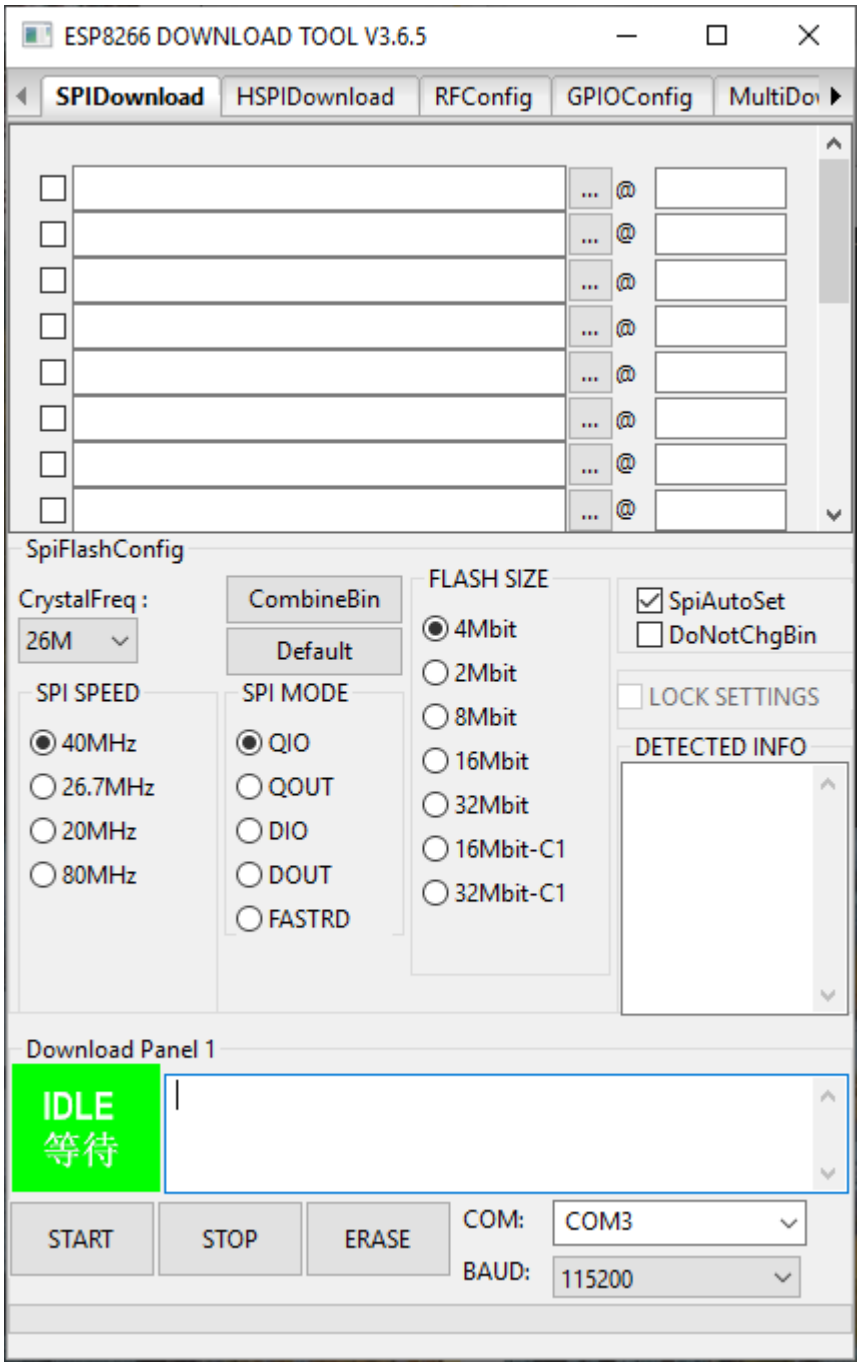


Рис. 5.45. Первичная настройка

- проверяем, чтобы COM-порт и скорость были выбраны правильно;
- подключаем вывод СБРОС к земле;
- подключаем вывод ПРОГРАММ. к земле;
- снимаем вывод СБРОС с земли;
- снимаем вывод ПРОГРАММ. с земли;
- нажимаем кнопку START на панели диалога программатора.

Результатом этих операций должно стать определение настроек модуля (**рис. 5.46**). Обратите внимание на окошко DETECTED INFO и на FLASH SIZE.

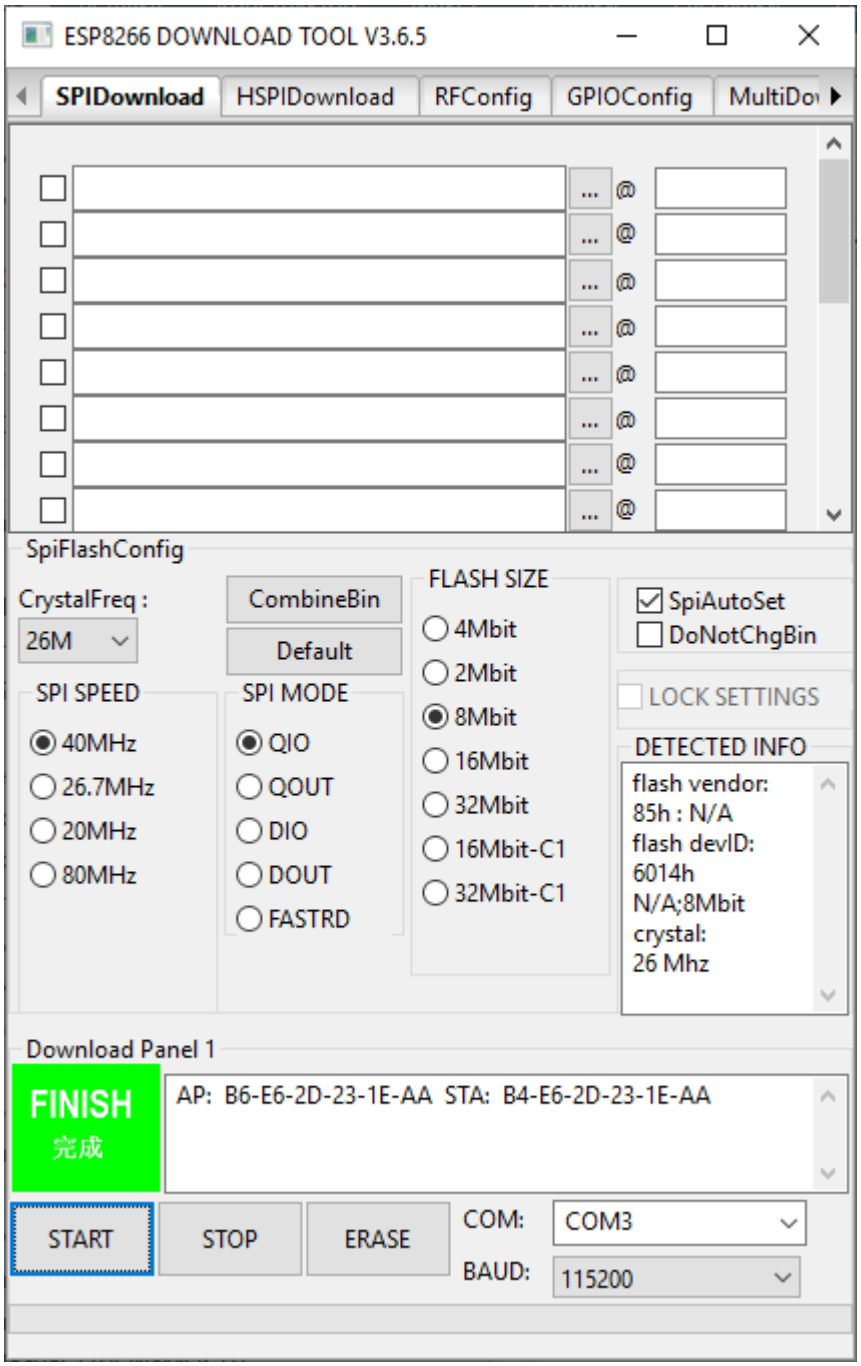


Рис. 5.46. Продолжение настроек

Теперь следует указать нужные файлы и адреса в окнах верхней части из загруженного набора файлов ESP_SDK_v2.0.0. Для этого используйте кнопки справа (с многоточием), а адреса и конкретные файлы возьмите из **таблицы 5.1**.

Таблица 5.1.

ESP8266_NONOS_SDK\bin\blank.bin	0xFB000
ESP8266_NONOS_SDK\bin\esp_init_data_default.bin	0xFC000
ESP8266_NONOS_SDK\bin\blank.bin	0x7E000
ESP8266_NONOS_SDK\bin\blank.bin	0xFE000
ESP8266_NONOS_SDK\bin\boot_v1.6.bin	0x00000
ESP8266_NONOS_SDK\bin\at\512+512\user1.1024.new.2.bin	0x01000

Заполнив данными все окна, отметьте их галочками и повторите процедуру подготовки к программированию:

- подключите вывод СБРОС к земле;
- подключите вывод ПРОГРАММ. к земле;
- снимите вывод СБРОС с земли;
- снимите вывод ПРОГРАММ. с земли;
- нажмите кнопку START на панели программатора.

Начинается процесс записи (**рис. 5.47**).

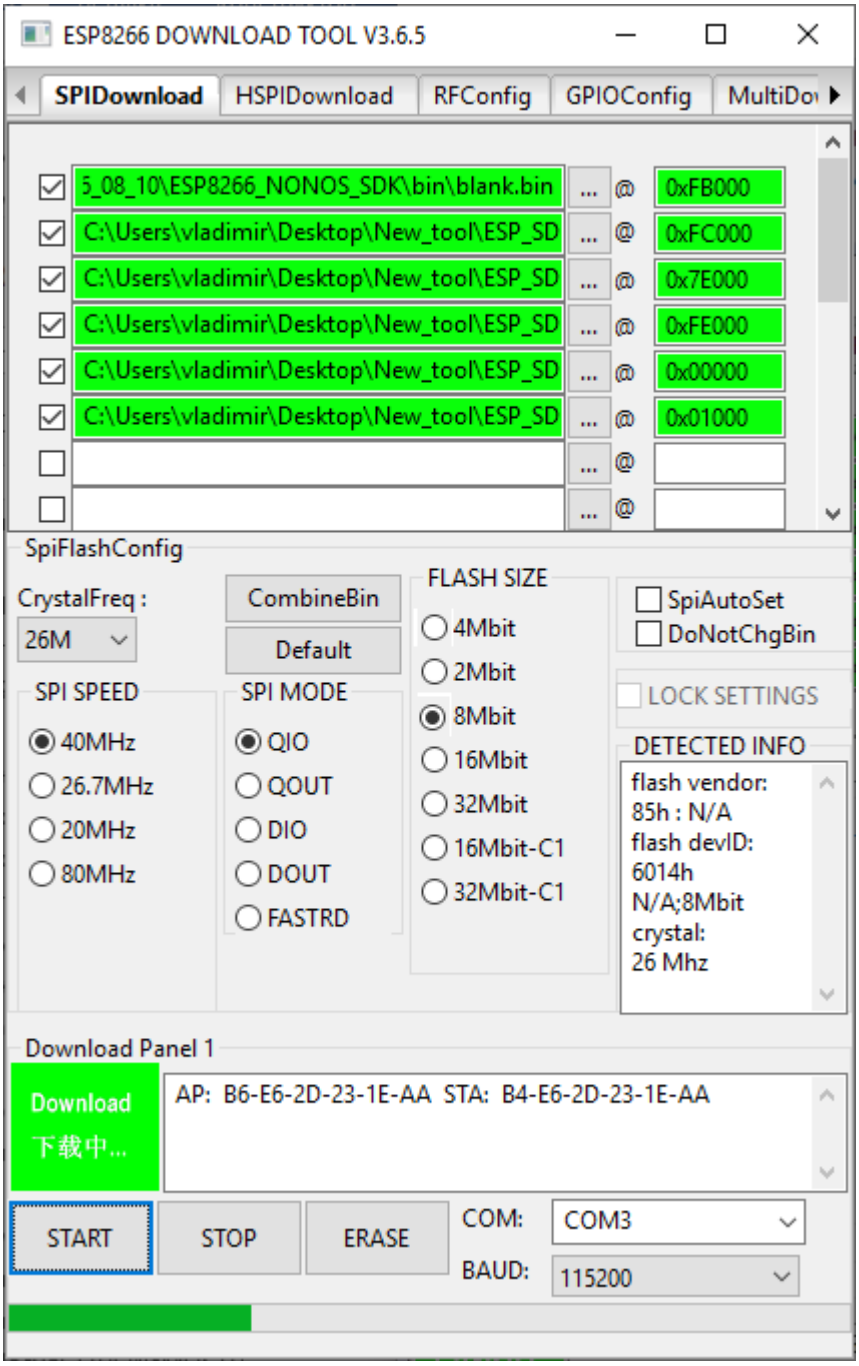


Рис. 5.47. Процесс «перепрошивки» модуля ESP8266

Завершается этот процесс заполнением строки состояния процесса и появлением надписи FINISH над кнопкой **START.** Теперь следует подключить вывод СБРОС к земле и отключить его.

Наверное, можно попробовать прошить самую последнюю версию, если это зачем-то нужно.

После обновления прошивки WiFi модуль перестал «капризничать», стал откликаться на АТ-команды. Однако попытка выполнить штатную команду смены скорости последовательного порта не работает, а команда `AT+IPR=9600` вновь погружает его в «сумеречное состояние души» – светодиод постоянно мигает, так что кажется, что он горит, а на команды WiFi модуль не реагирует. Приходится вновь «перепрошивать» его.

Для работающего модуля осталось выполнить рекомендованные для релейной сборки команды (на всякий случай):

- `AT+CWMODE=2`, выбор АР режима;
- `AT+RST`, сброс;
- `AT+CIPMUX=1`, открыть множественные соединения;
- `AT+CIPSERVER=1,8080`, конфигурируем TCP сервер, задаем номер порта;
- `AT+CIFSR`, проверяем IP адреса в режиме АР такие как APIP, “192.168.4.1";

И релейный модуль, что полностью у меня заработал, мне хотелось бы проверить при работе со смартфоном. Программа EasyTCP_20 на смартфоне установлена, с WiFi модулем она работает, это я проверил, но работает ли она после воссоединения устройства?

Вдобавок, программа на смартфоне тоже требует некоторых пояснений.

После ее установки, после выбора подключения по WiFi к релейному модулю, следует нажать и удержать одну из кнопок закладки *SWITCH*.

Первый раз мне удалось переместиться к этой закладке обычным для сенсорного экрана способом, но в следующий раз, когда я решил повторить все операции, сколько я не возил пальцем по экрану, переместиться не получалось. Помогло следующее: если использовать настройки программы, есть значок с рядом вертикальных точек, то можно выбрать раздел *Preferences*, предпочтительные настройки. В этом разделе можно указать, что при включении программы следует открывать закладу *SWITCH*.

На закладке ряд кнопок (**рис. 5.48**).

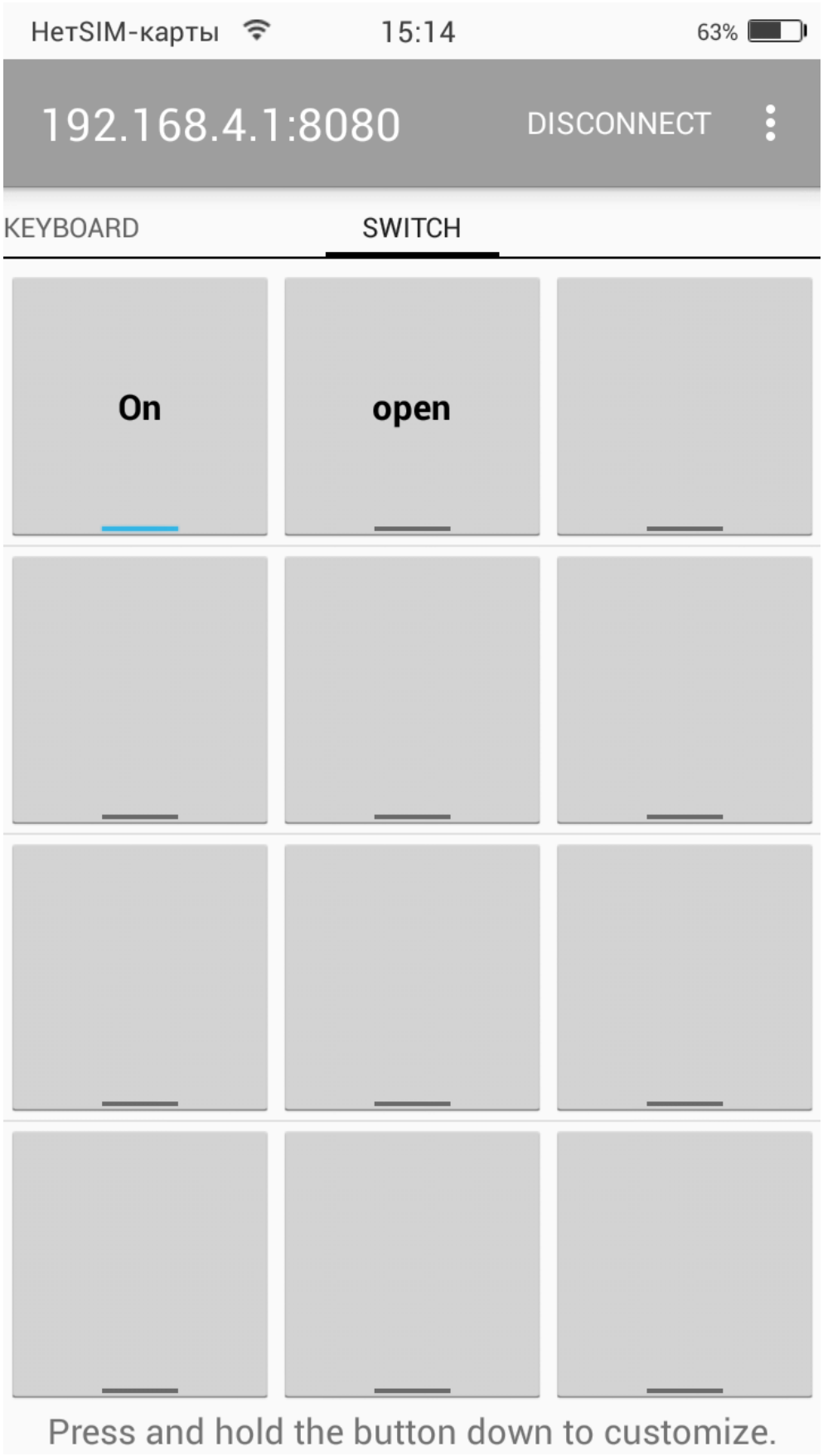


Рис. 5.48. Окно программы на смартфоне

После того, как вы нажмете и удержите одну из кнопок, появится диалог настройки кнопки (**рис. 5.49**).

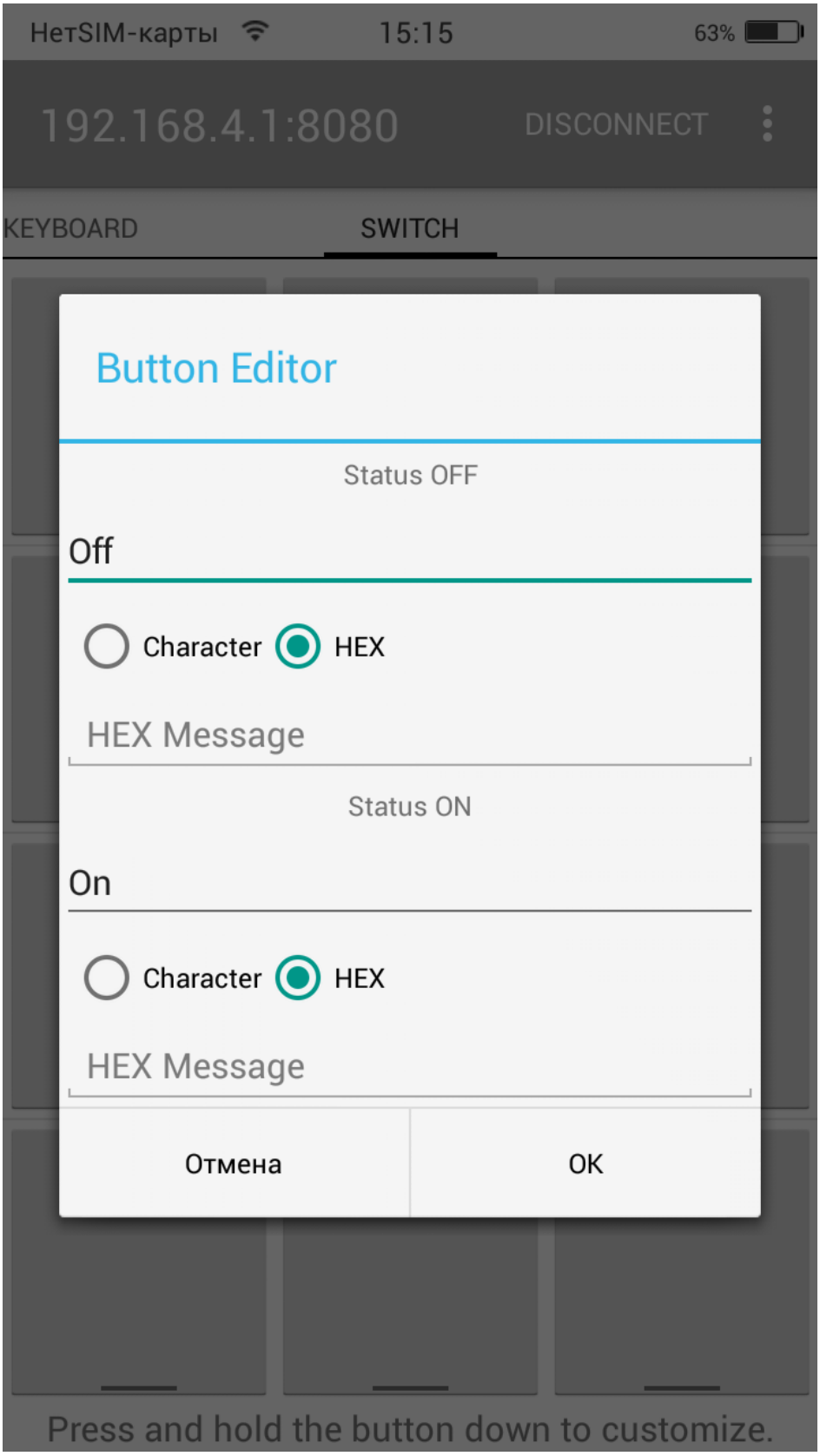


Рис. 5.49. Диалог настройки кнопок

В первую очередь следует выбрать то, что сообщения будут отправлены в шестнадцатеричном виде (HEX). Затем следует как-то назвать кнопки, названия, похоже, произвольные. Любой ввод – это нажатие пальцем в область ввода, после чего появляется клавиатура, где можно осуществить ввод текста или цифр.

Для сообщений (Message) нужно ввести такой соответствующий код: A0 01 01 A2 open relay, A0 01 00 A1 closed.

Что означает открыть реле, а что закрыть, я не разобрался, но решил, что проще будет поменять имена, если введены неверные коды. При вводе шестнадцатеричных кодов нет необходимости переключать регистр, он сам переключается (**рис. 5.50**). Дополнительно, после ввода двух знаков программа вводит пробел, таким образом, самостоятельно пробел вводить нет нужды, более того, если это сделать, то программа «бастует», не принимая такое решение.

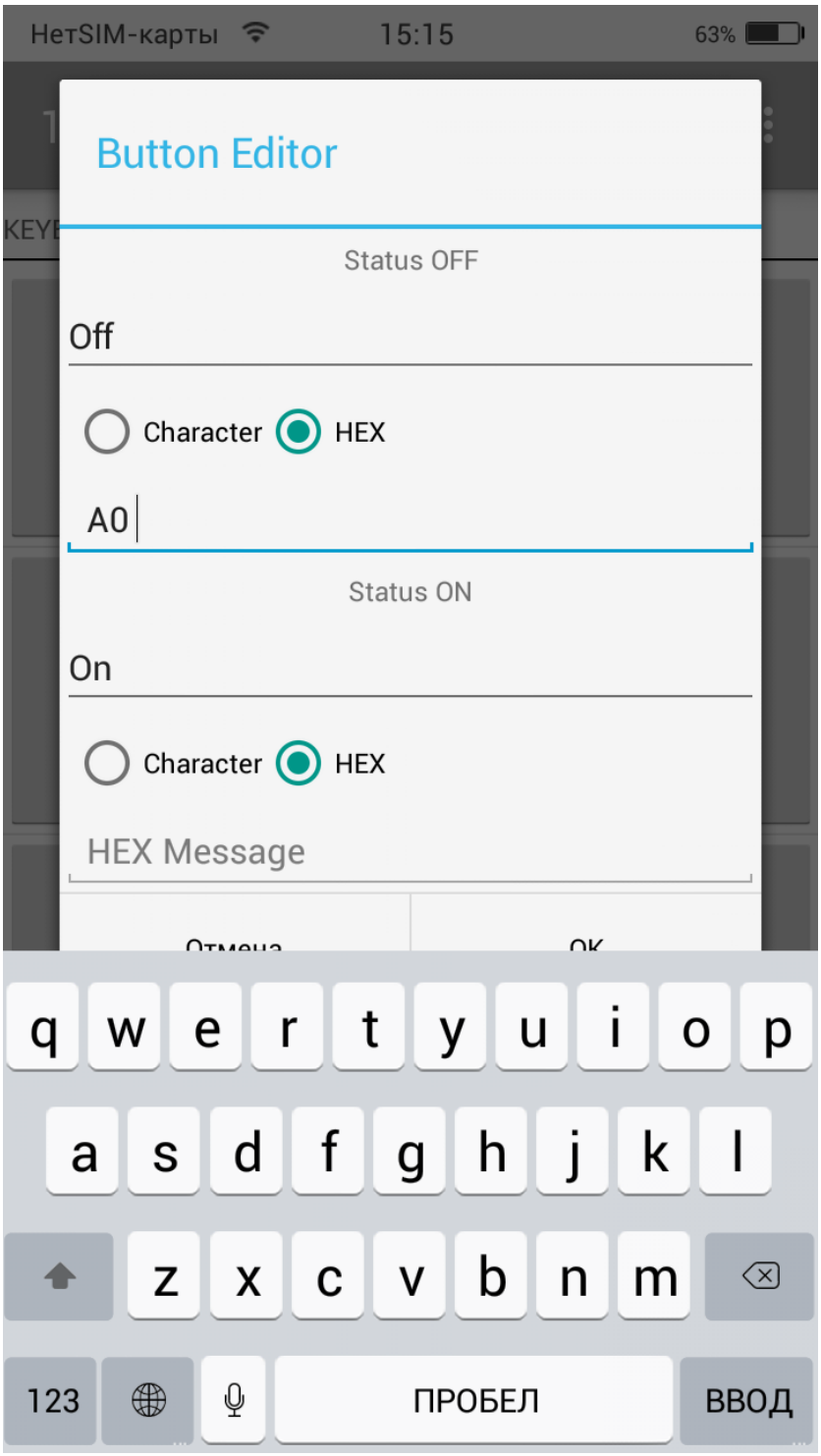


Рис. 5.50. Ввод необходимых данных для работы с ESP8266 на смартфоне

Завершив эти операции, нажатием кнопки **ОК** можно перейти к испытаниям. При первом опыте потребуется ввести ip-адрес и порт, в последующем этот выбор появится при нажатии кнопки **CONNECT**. Дальнейшая работа с кнопками программы возможна только при подключении к релейному модулю.

Впрочем, эти особенности не входили в круг моих интересов. Я проверил, что при нажатой кнопке реле щелкает (но не включается, нужен внешний источник), на том и завершив эту часть экспериментов.

Это важно!

Если отключить питание релейныого модуля, то для восстановления работы следует повторить команды:
- AT+CIPMUX=1, открыть множественные соединения;
- AT+CIPSERVER=1,8080, конфигурируем TCP сервер, задаем номер порта.

Удивительным для меня оказалось столь разное поведение двух одинаковых WiFi модулей. При этом «правильнее», с моей точки зрения, вел себя тот, что «попал под напряжение 5 В», тот, который я счел безнадежно испорченным.

Причина может быть в разных версиях прошивки модулей, кроме того, при подключении одного из модулей к напряжению 5 вольт что-то в нем могло испортиться.

Со вторым WiFi модулем проблема тоже была решена использованием команды: `AT+UART=9600,8,1,0,0`.

Кроме того, используя команду `AT+CWJAP="SSID","password"`, модуль можно подключить к домашней сети, если роутер использует WiFi. При отключении питания релейного модуля он сохраняет команду подключения, чтобы при следующем включении подключиться к домашней сети.

О том, как работать с ESP-модулями в MajorDoMo есть ряд статей, а я завершаю эту часть опытов, признав, экспериментатор пока из меня получается неважный.

Глава 6. Несколько опытов с модулями в сети

Опыт с Модулем ESP8266, оснащенным реле

Этот модуль доставил мне более всего хлопот. Он исполнительный, но безынициативный, если сказать честно, довольно ограниченный, хотя и услужливый. В хороших руках «мажордома» ему цены не будет.

Мы проводили опыт с выключателем, управляя работой светодиода. Попробуем повторить его, но с другим составом. Смысл простой – с помощью MajorDoMo мы будем включать и выключать свет у входа. Выполнение этого действия очень подходит по характеру модулю ESP8266 с реле. Контакты реле прекрасно должны справиться с задачей. Но прежде, чем начать эксперимент, модуль следует включить в сетевую группу. Иными словами, его нужно подключить к домашней сети.

О том, как это сделать, было написано в конце предыдущего раздела. Но в первую очередь, наверное, следует проверить в каком режиме модуль ESP8266. Дело в том, что для подключения к роутеру домашней сети ESP8266 должен находиться в режиме станции (см. Приложение А). Проверка – это команда AT+CWMODE? Отклик модуля выглядит так:

`+CWMODE: 1` (или 3, если режим смешанный)

Единица свидетельствует, что модуль в режиме станции, а 3 – это режим и клиента, и станции.

Если режим станции не включен, то его нужно включить. Для этого используется команда:

`AT+CWMODE_DEF=1` (или 3, если нужен смешанный режим)

Затем следует решить, подключаться к роутеру на постоянной основе, используя команду:

```
AT+CWJAP_DEF = <ssid>,< pwd >
```

Тогда при каждом включении питания модуля он автоматически войдет в домашнюю сеть. Или использовать другую команду:

```
AT+CWJAP_CUR = <ssid>,< pwd >
```

После отключения питания модуль «забудет» о домашней сети. Обе команды используют два элемента, где `ssid` – это имя вашего роутера, `pwd` – это пароль для подключения. Оба параметра берутся в кавычки, например:

```
AT+CWJAP_CUR = "BEELINE","12345"
```

Если модуль подключается на постоянной основе, то при включении питания вы получите отклик (**рис. 6.1**).

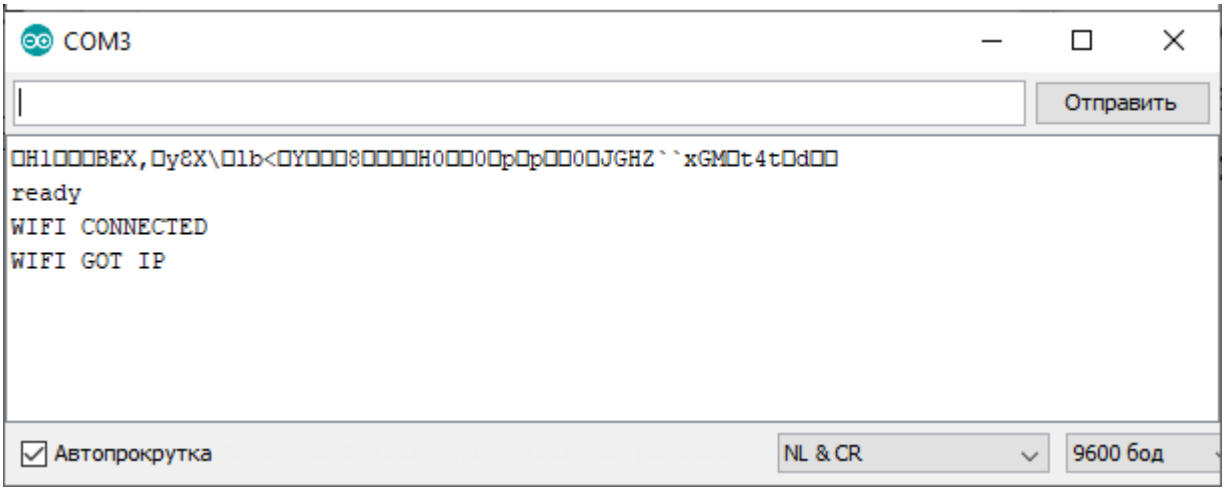


Рис. 6.1. Отклик модуля ESP8266 с реле при включении питания

Но по прошествии некоторого времени бездействия соединение будет разорвано. Правда, затем оно восстановится. Команда `AT+CIFSR` позволит вам узнать ip-адрес, присвоенный модулю роутером (**рис. 6.2**).

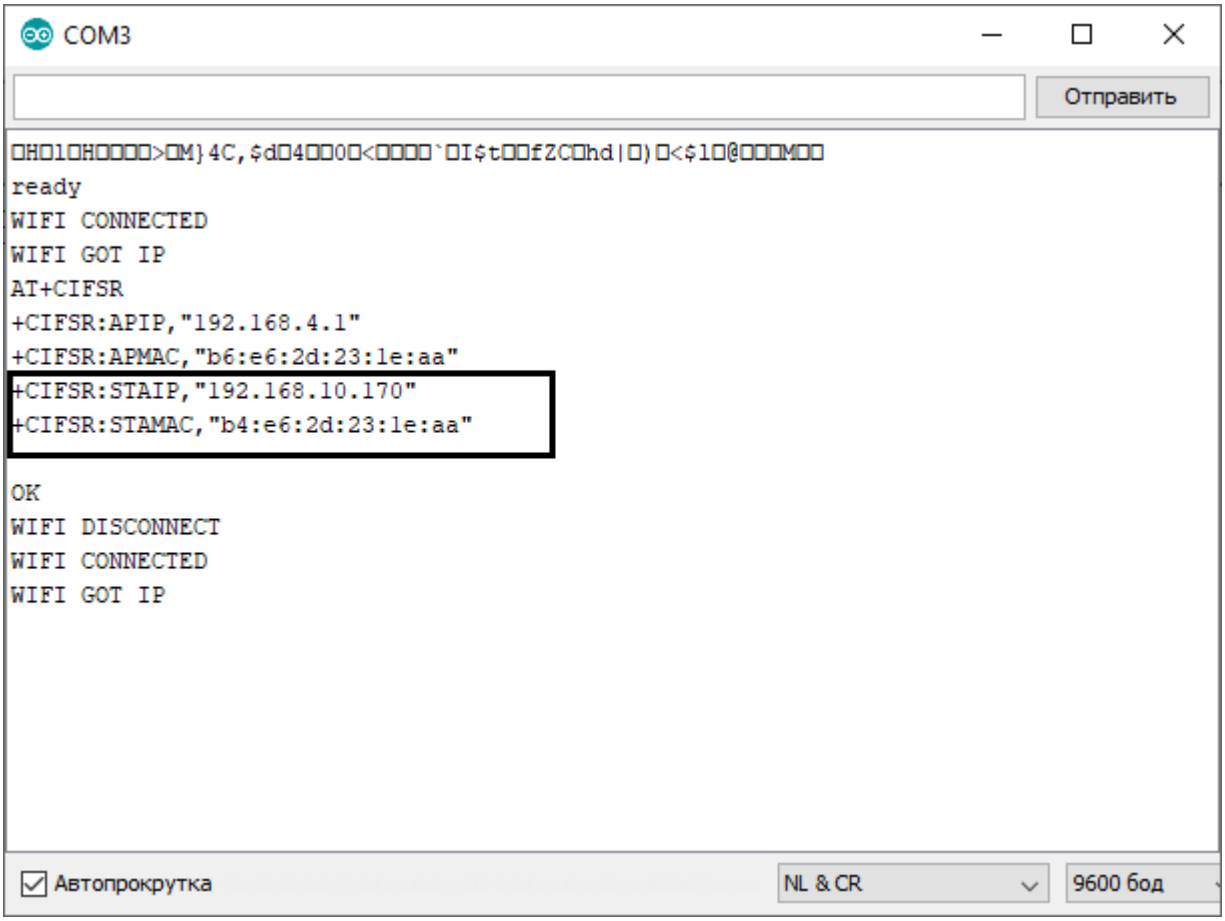


Рис. 6.2. Выяснение ip-адреса модуля ESP8266 в домашней сети

В руководстве к релейному модулю написано, что он может работать и в составе домашней сети. Поэтому одну из кнопок в программе EasyTCP_20 я настраиваю на работу в домашней сети, чтобы выяснить… что ничего у меня не получилось – не работает включение реле. После некоторых размышлений (не скажу веселых) у меня складывается впечатление, что что-то я не додаю модулю ESP8266 с реле.

В настройках модуля, если читать руководство (а как оказалось, читать руководства полезно), указано несколько АТ-команд. Часть из них связано с включением модуля в домашнюю сеть, а в домашнюю сеть модуль входит автоматически. Но есть две команды: `AT+CIPMUX=1` и `AT+CIPSERVER=1,8080`, которые следует попробовать выполнить. Можно проверить, что режим `CIPMUX` не единица. Более того, если попытаться запустить сервер без установки режима множественных соединений, то результат будет отрицательным (**рис. 6.3**).

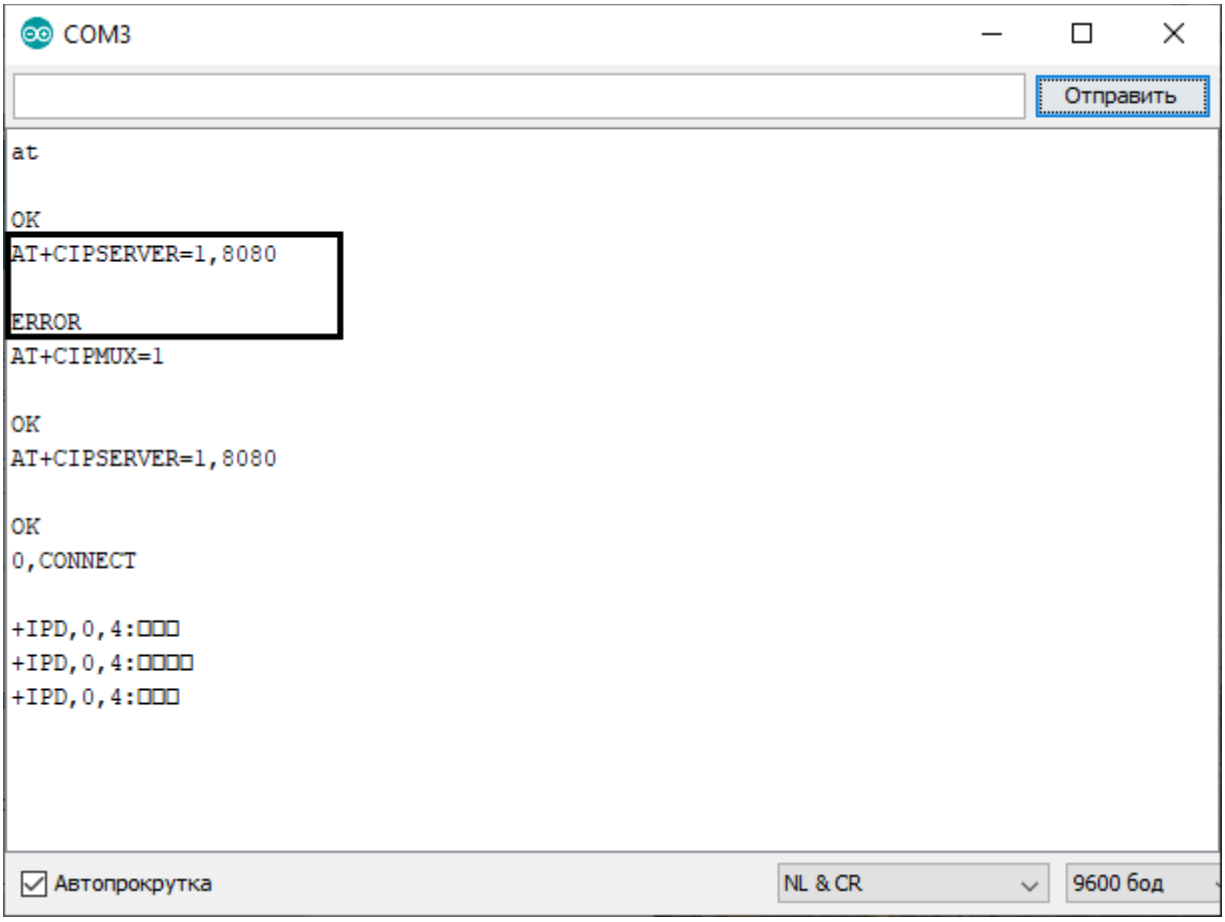


Рис. 6.3. Выполнение команд для подготовки модуля

Примечание.

После выключения питания модуль возвращается к первоначальному режиму.

Выполнив команды в должной последовательности: множественные соединения и запуск сервера, - можно убедиться, что модуль подключается к смартфону в домашней сети, а реле включается. Для экспериментов этого достаточно, но при использовании модуля в реальной конструкции придется что-то делать, может быть, добавить батарейку для удержания настроек при отключении сетевого питания.

Хотелось бы думать, что на этом приключения закончились. Но они только начались.

В MajorDoMo, прочитав несколько статей, я попытался следовать рекомендациям, но быстро запутался, решив, что самый простой путь – это то, что мне надо. Для включения и выключения реле, используя панель управления MajorDoMo, в разделе *Объекты* (и

подразделе *Объекты*) выберем кнопку **Добавить новый объект**. В диалоге задания нового объекта я называю его *Крыльцо* из класса *SControllers* (**рис. 6.4**).

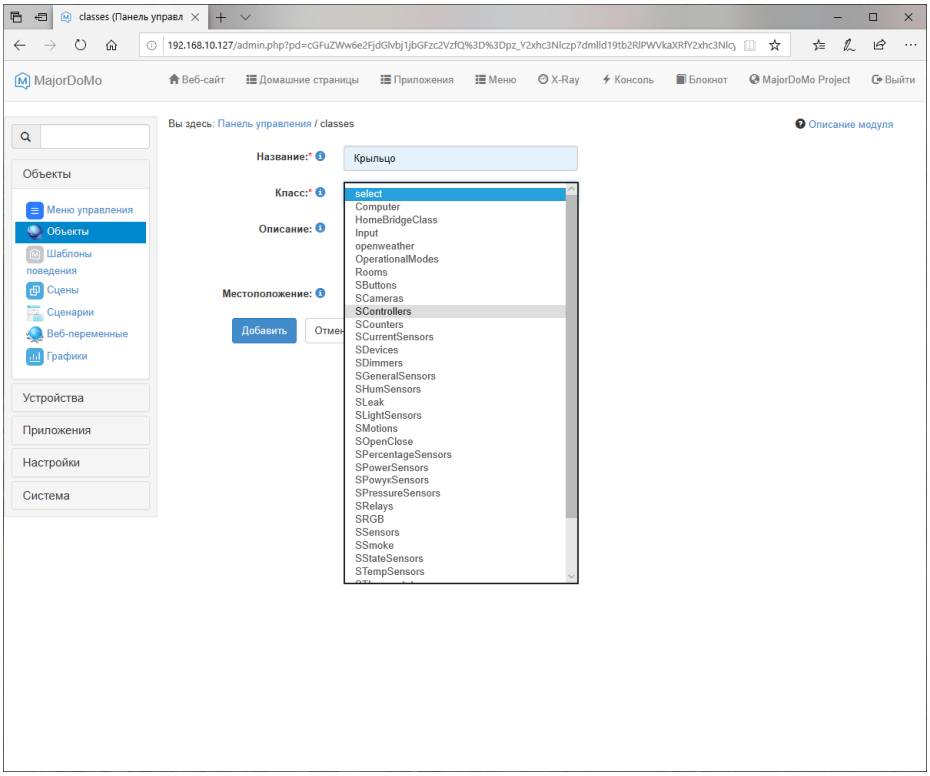


Рис. 6.4. Создание объекта для ESP с реле

Завершается эта часть работы с помощью кнопки **Добавить**.

Выберем раздел сцен, выберем *Scene 2*, как раньше назвали новую сцену. После выбора закладки *Элементы* используем кнопку **Добавить новый элемент** (**рис. 6.5**).

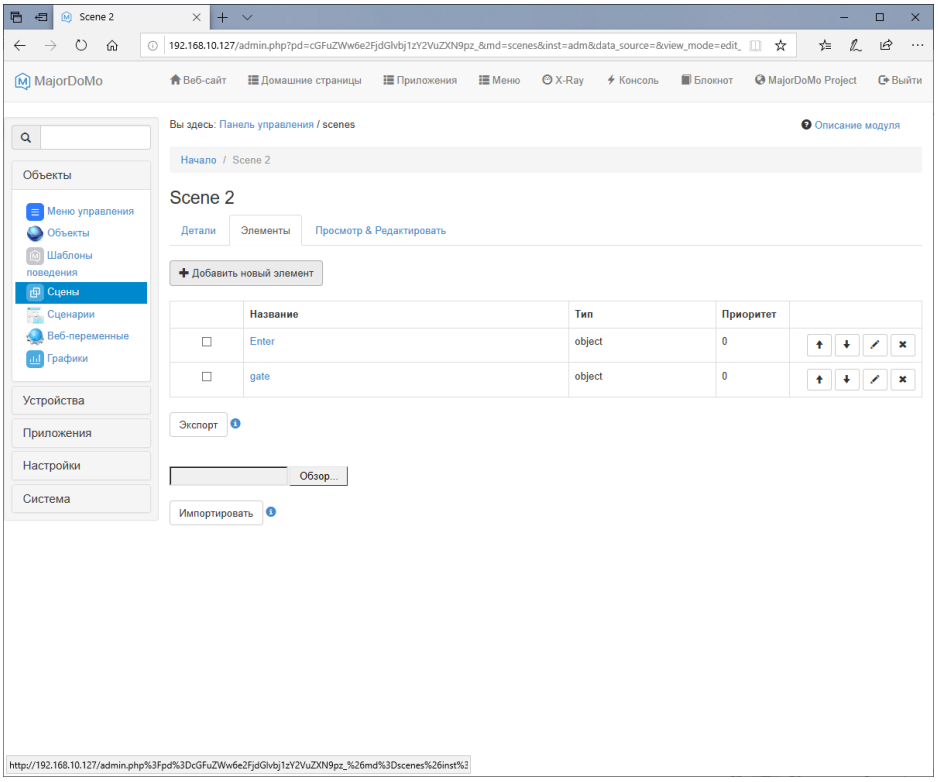


Рис. 6.5. Добавление нового элемента к сцене

В диалоговом окне заполним все нужные поля (**рис. 6.6**), чтобы этот элемент появился на сцене. Пока это только заготовка, но и с ее созданием следовало разобраться. Завершаем процесс с помощью кнопки **Сохранить**.

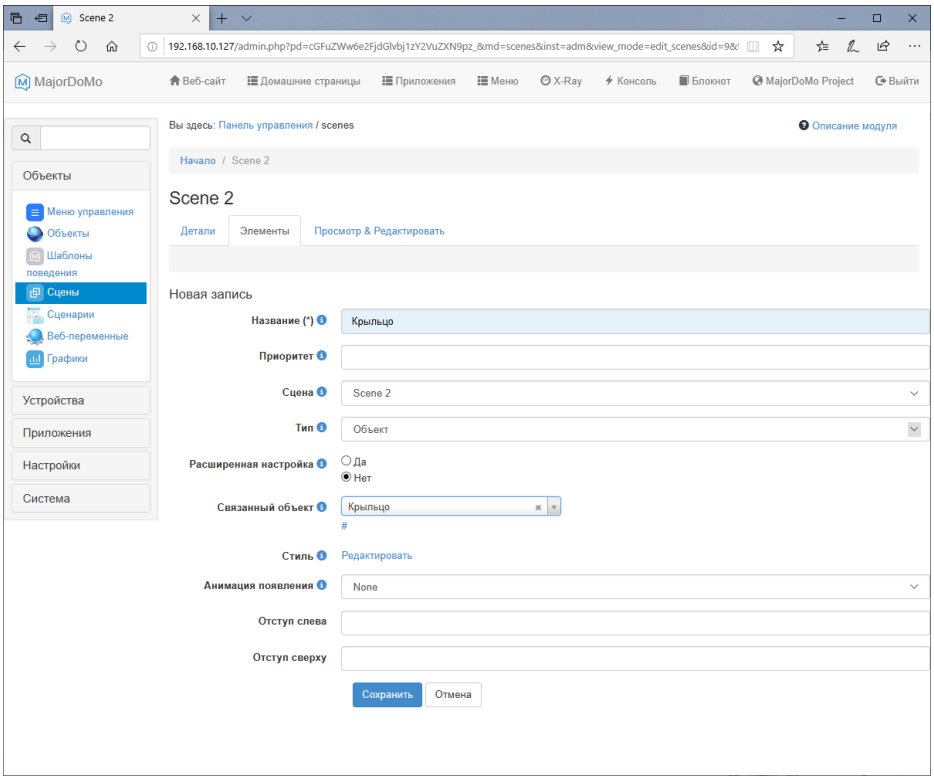


Рис. 6.6. Завершение создания нового элемента для сцены

Появление этого элемента на сцене можно проверить с помощью закладки *Просмотр&Редактировать* (**рис. 6.7**). Проверить следует, чтобы убедиться в возможности двигаться дальше.

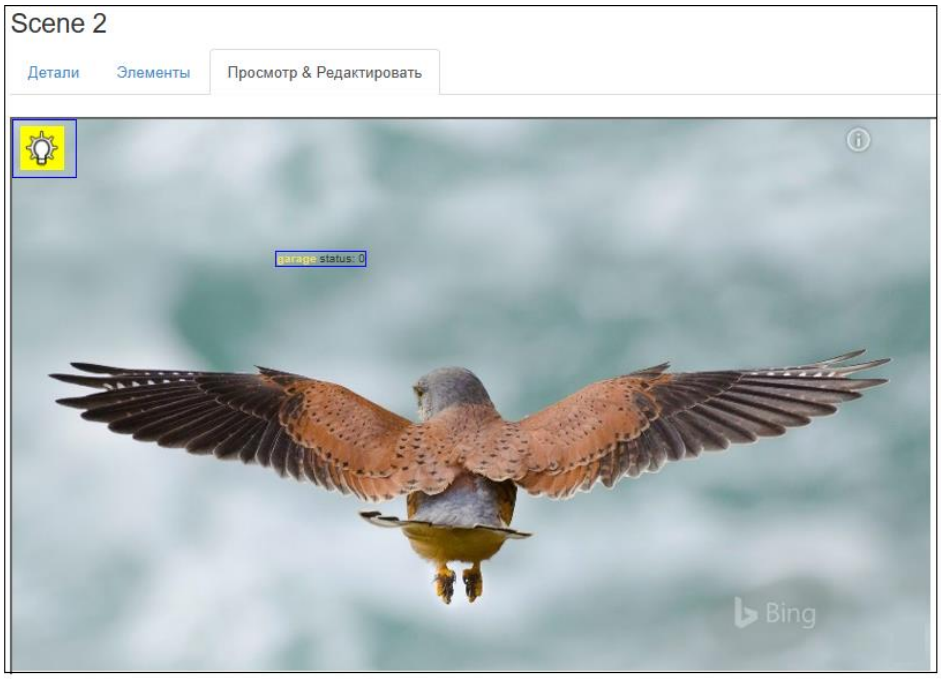


Рис. 6.7. Появление нового элемента на сцене

По новому элементу сцены можно щелкать мышкой, что, впрочем, не производит никакого впечатления на модуль ESP8266. И это правильно. Откуда ему знать, что мы обращаемся к нему, а не к кому-то еще. Выберем *Объекты→Объекты*. Выберем закладку *Методы*. Предустановленные методы – это *turnOn* и *turnOff*, которые еще предстоит настроить, выбрав кнопку **Настроить** (**рис. 6.8**).

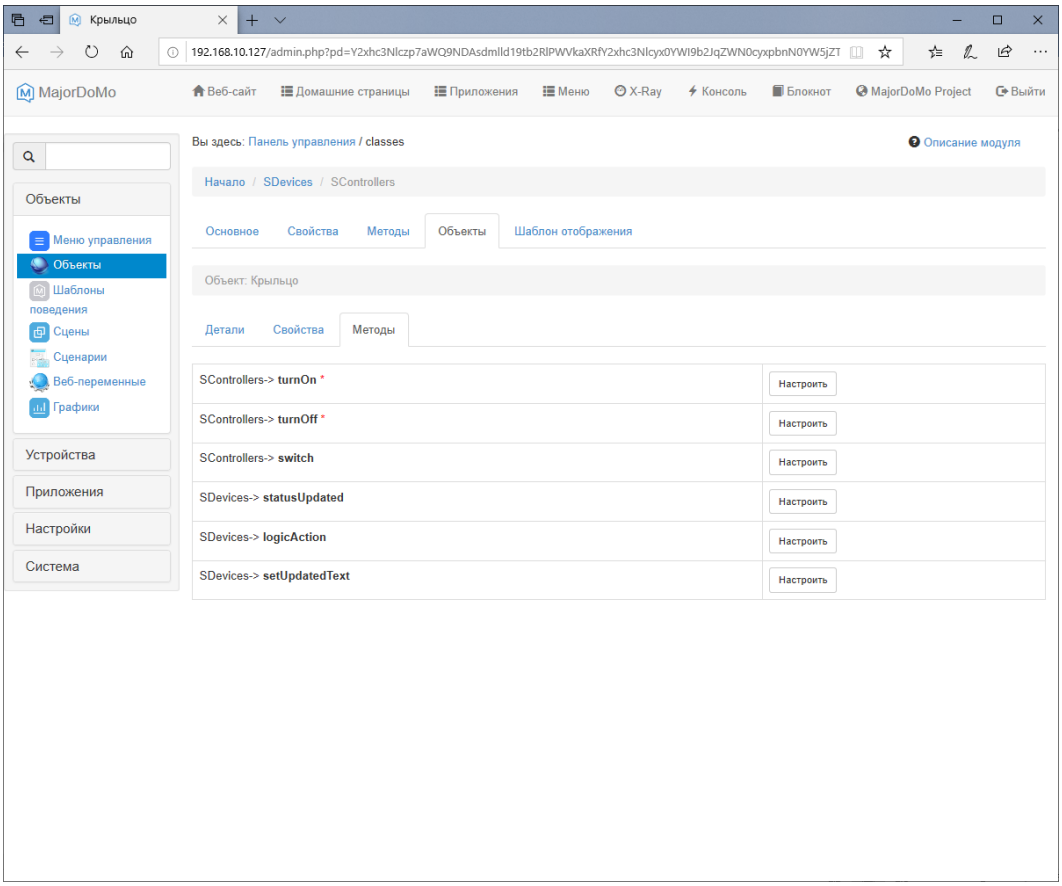


Рис. 6.8. Диалог перехода к настройке методов

И в новом диалоге, отметив, что нужно выполнить код, в окне ввода кода следует ввести код PHP.

Я прочитал более десятка рекомендации по написанию нужного мне кода. Я охотно соглашусь, что не знаю языка PHP. Но кому от этого легче? Для модуля я использую подключение через Arduino в качестве переходника USB-TTL. А для отображения результатов моих попыток и настройки ESP8266 с реле использую монитор порта. Максимум того, что мне удается получить, это вывод команды, напомню, для включения это шестнадцатеричные числа A0 01 01 A2, что можно увидеть на **рис. 6.9**.

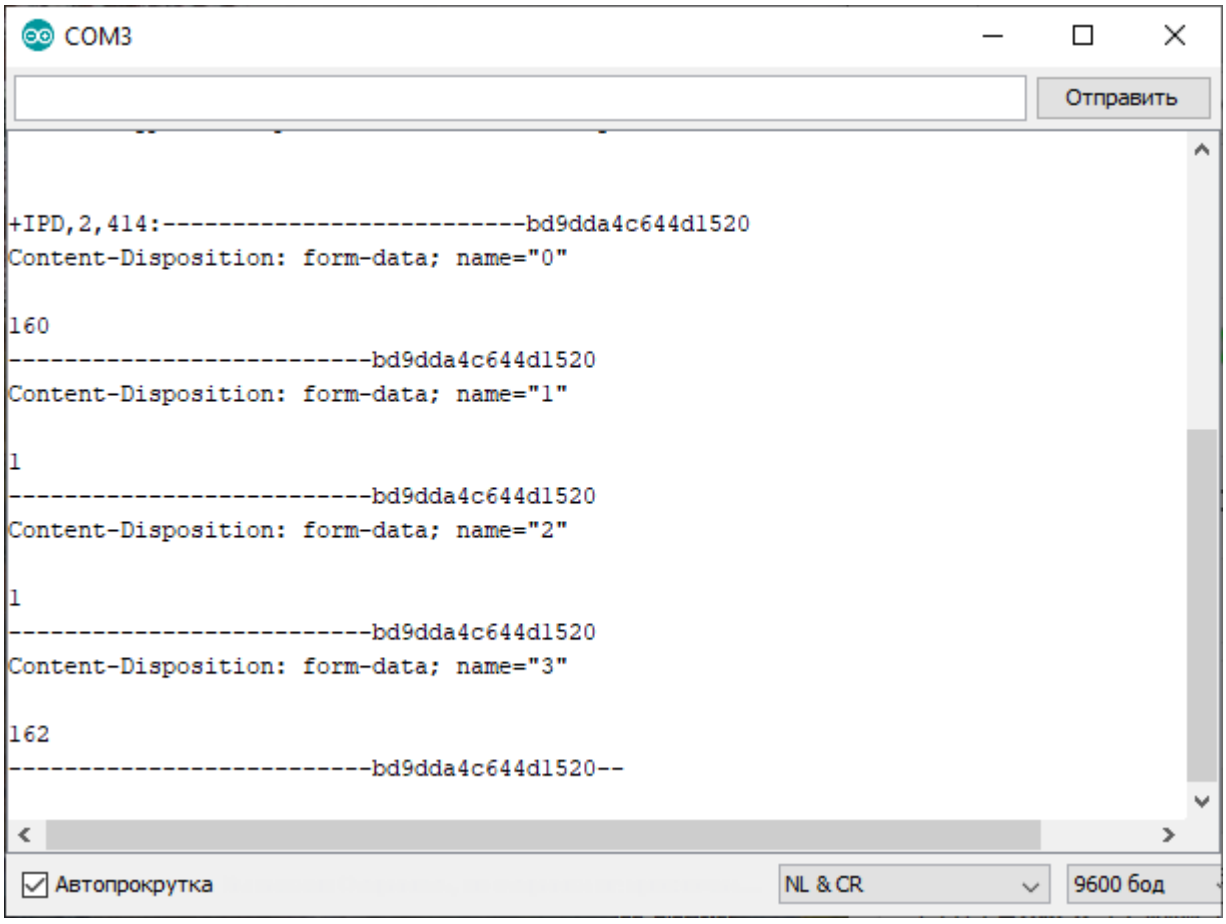


Рис. 6.9. Отклик модуля ESP8266 на щелчок по элементу сцены

При подключении ESP8266 выводится страница, затем команда переводится в символьный вид чисел команды, что никак не вызывает желаемого включения реле. Я честно перепробовал много вариантов, предлагаемых более опытными пользователями PHP. Вдобавок, что тоже важно, я не знаю, а работает ли вообще команда? Да, если ее отправлять со смартфона работает, но без смартфона?

Вспомнилась программа USR-TCP232, с которой у меня при первом знакомстве не сложились отношения. Пришлось исправить это – не вводить же АТ команды через монитор порта Arduino, чтобы затем выключить программу Arduino и запустить программу USR-TCP232.

В прошлый раз я был неправ при отправке АТ команд: недостаточно ввести команду в окно команд и нажать кнопку **Send**, нужно перед этим нажать клавишу **Enter** на клавиатуре (**рис. 6.10**). Все так просто (когда узнаешь, читая руководство).

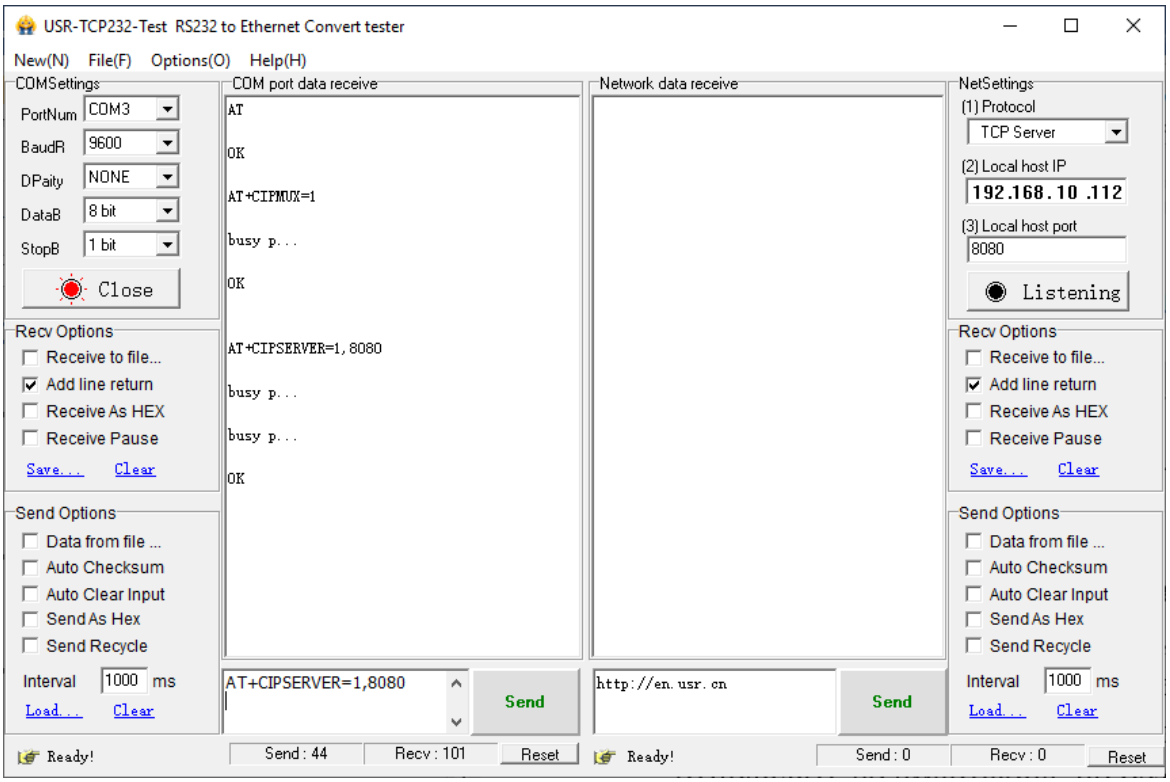


Рис. 6.10. Ввод AT команд в программе USR-TCP232

Для решения возникших проблем я решил искать решение по шагам. Правое окно программы можно настроить для соединения с модулем ESP8266: в верхнем окне выбрать TCP Client, а ниже ввести ip-адрес и номер порта. Теперь с помощью кнопки **Connect** можно подключиться к модулю и отключиться от него (**рис. 6.11**).

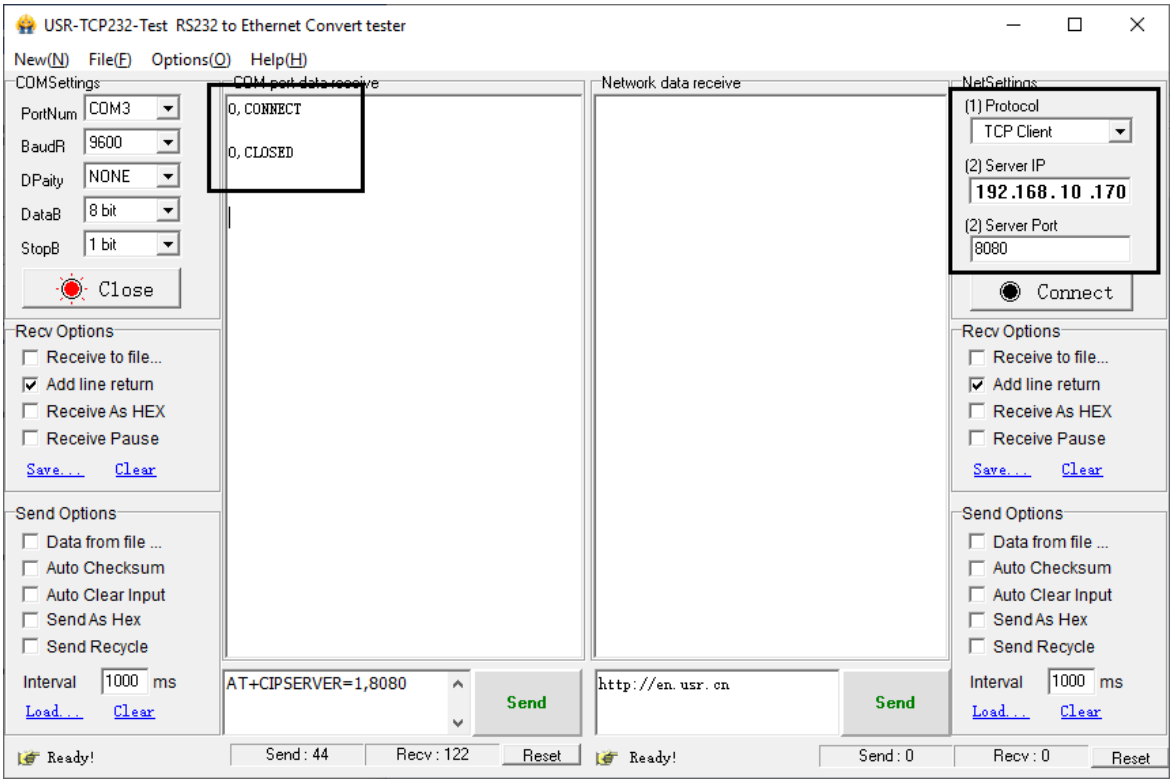


Рис. 6.11. Настройка программы для подключения к модулю ECP8266

При подключении в окне COM-порта отображается только сообщение о подключении и отключении, страница модуля не выводится.

Хотел бы сказать, что мне хватило нескольких минут, чтобы добиться такого же результата. Но врать не хочу. С неделю я пытался понять, что не так в командах PUT, POST и GET, пытался понять, чем плоха функция getUrl(). К сожалению, поиск в Интернете по запросу *PHP send* приводит к появлению ссылок на статьи запросов к базе данных, что меня никак не устаивало. Наконец, к моему удовлетворению, нашлась функция *fsockopen('ip адрес устройства', 8080).* При ее использовании подключение происходит точно так, как в программе USR-TCP232.

Второй шаг – поиск строки, а функция fputs(), она же fwrite(), использует строку при отправке. В программе USR-TCP232 для команды используется отправка строки как шестнадцатеричных чисел.

Примечание.

После ряда экспериментов левое окно программы заполнятся полностью. Чтобы удалить текст, можно выделить его мышкой и нажать клавишу ***Delete*** *на клавиатуре.*

Но можно использовать выпадающее меню, которое появляется после нажатия правой клавиши мышки в окне. Выглядит оно несколько странно, ориентироваться следует на буквы в скобках. Я не исключаю, что можно изменить это в настройках, но пока не хотелось бы экспериментировать (**рис. 6.12**).

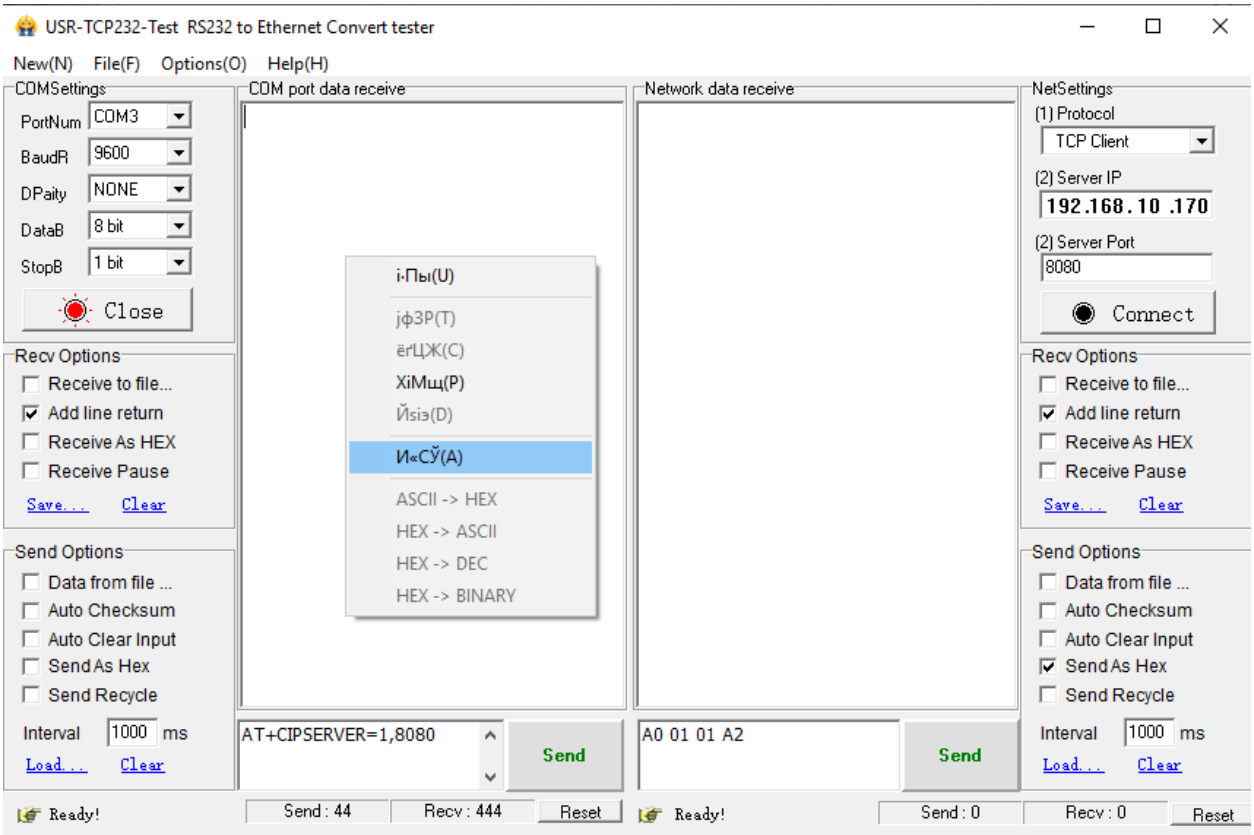


Рис. 6.12. Выпадающее меню в программе USR-TCP232

Команда включения и выключения реле в программпе USR-TCP232 выглядит просто (**рис. 6.13**). При этом, обратите внимание, следует в поле записи вводить числа с пробелами, отметив предварительно *Send As Hex*.

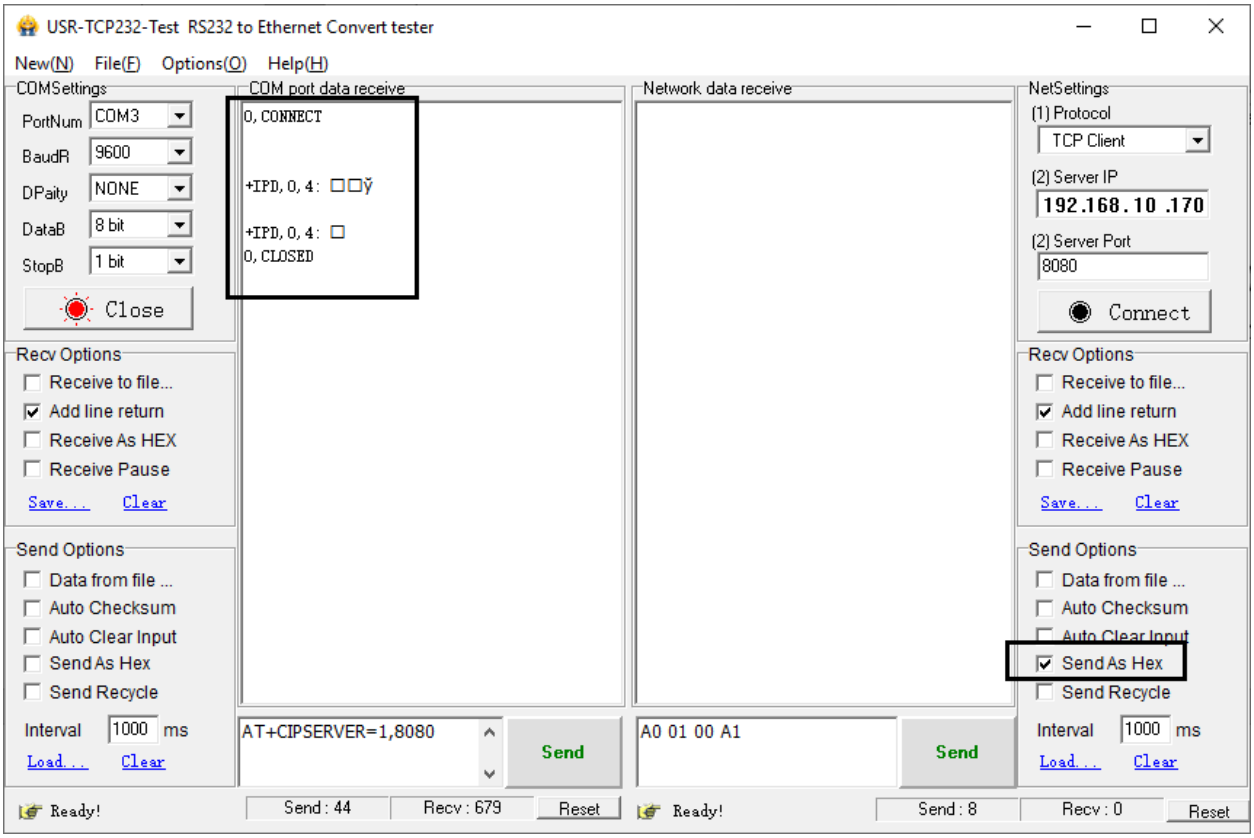


Рис. 6.13. Отправка команд в программе USR-TCP232

Поиск нужной формы строки для команды занял некоторое время, а завершился тем, что в раздел метода для *turnOn* вписался такой текст:

```
  $link = '192.168.10.170'; // Строка ip-адреса сервера
  $fp = fsockopen($link, 8080); // Открыть файл на сервере
  sleep(1);
  fputs($fp,chr(160).chr(1).chr(1).chr(162)); // Отправить
строку "A0 1 1 A2"
  sleep(1);
  fclose($fp); // Закрыть файл
```

А в раздел метода *turnOff*:

```
  $link = '192.168.10.170'; // Строка ip-адреса сервера
  $fp = fsockopen($link, 8080); // Открыть файл на сервере
  sleep(1);
  fputs($fp,chr(160).chr(1).chr(0).chr(161)); // Отправить
строку "A0 1 0 A1"
  sleep(1);
  fclose($fp); // Закрыть файл
```

Строка команды chr(160).chr(1).chr(0).chr(161) выглядит несколько необычно, в ней chr(160) – это значит вывести символ с номером 160 (или 0xA0) по таблице ASCII; значения соединяются в строку точкой между символами. Теперь появилась возможность включать и выключать реле (**рис. 6.14**).

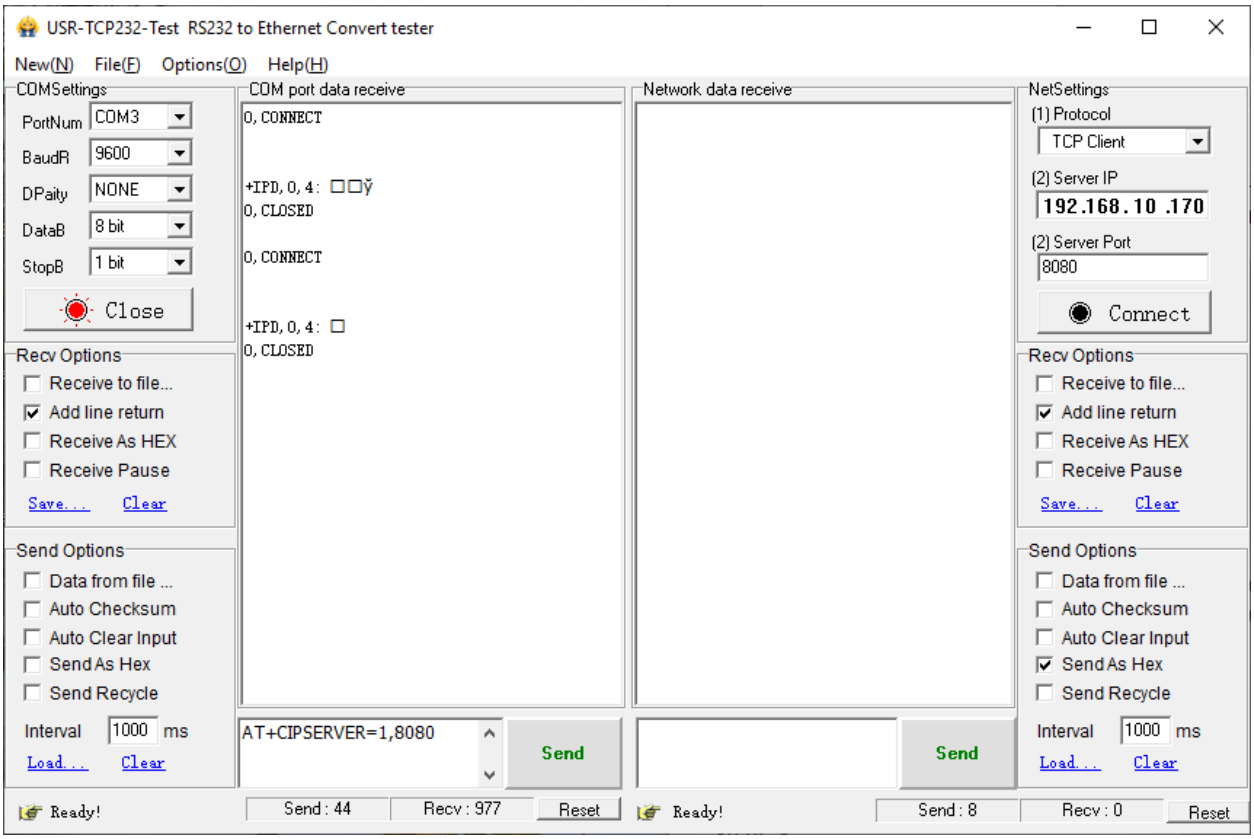


Рис. 6.14. Управление реле модуля ESP8266 из MajorDoMo

Две паузы `sleep(1)` появились позже, поскольку без них включить удалось только один раз, последующие попытки не давали эффекта. Ранее мне пришло в голову изменить время *timeout* командой `AT+CIPSTO=5`, поскольку по умолчанию это 180 секунд. Команда срабатывает, похоже, только после настройки модуля, то есть, выполнения двух команд, о которых речь шла ранее. Я проверил настройки в программе USR-TCP232, она вела себя так же, как и MajorDoMo.

Можно добавить еще несколько замечаний. На **рис. 6.7** отображен результат нескольких экспериментов. Если выполнить настройки так, как описано в этом разделе выше, то иконка отобразит лампочку, которая будет реагировать на нажатие, но не будет отображать состояние объекта. Чтобы получить такой вид иконки, который отображает состояние, следует при создании элемента для сцены выбрать в качестве типа не объект, а выключатель (**рис. 6.15**).

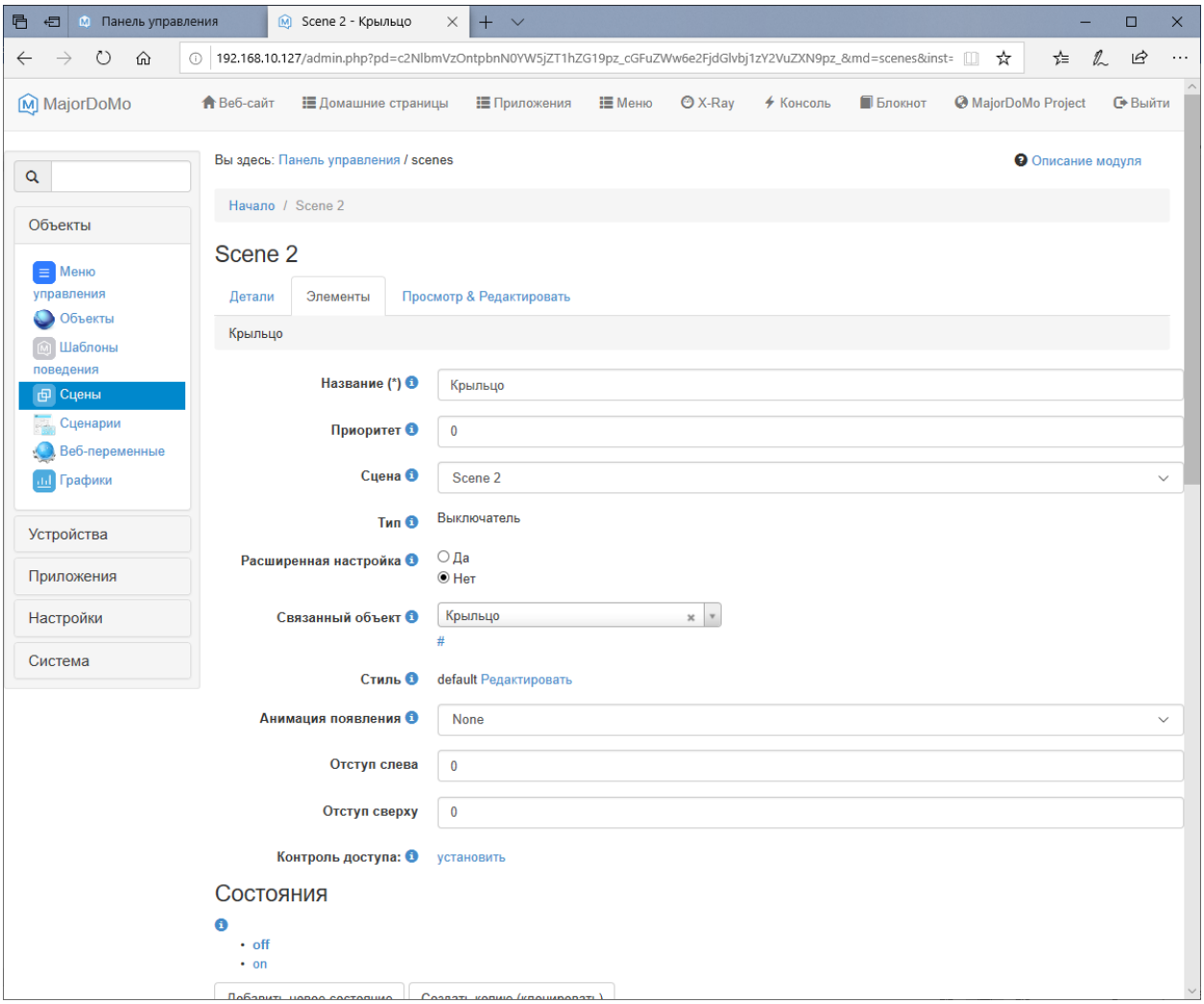


Рис. 6.15. Выбор другого типа добавленного элемента

Теперь вид добавленного компонента будет отвечать желаниям, при выключении иконка «гаснет», а включенная «загорается» (**рис. 6.16**). В остальном все остается по-прежнему: необходимо после включения модуля выполнить три команды настройки, то есть:

`AT+CIPMUX=1`, `AT+CIPSERVER=1,8080` и `AT+CIPSTO=5`.

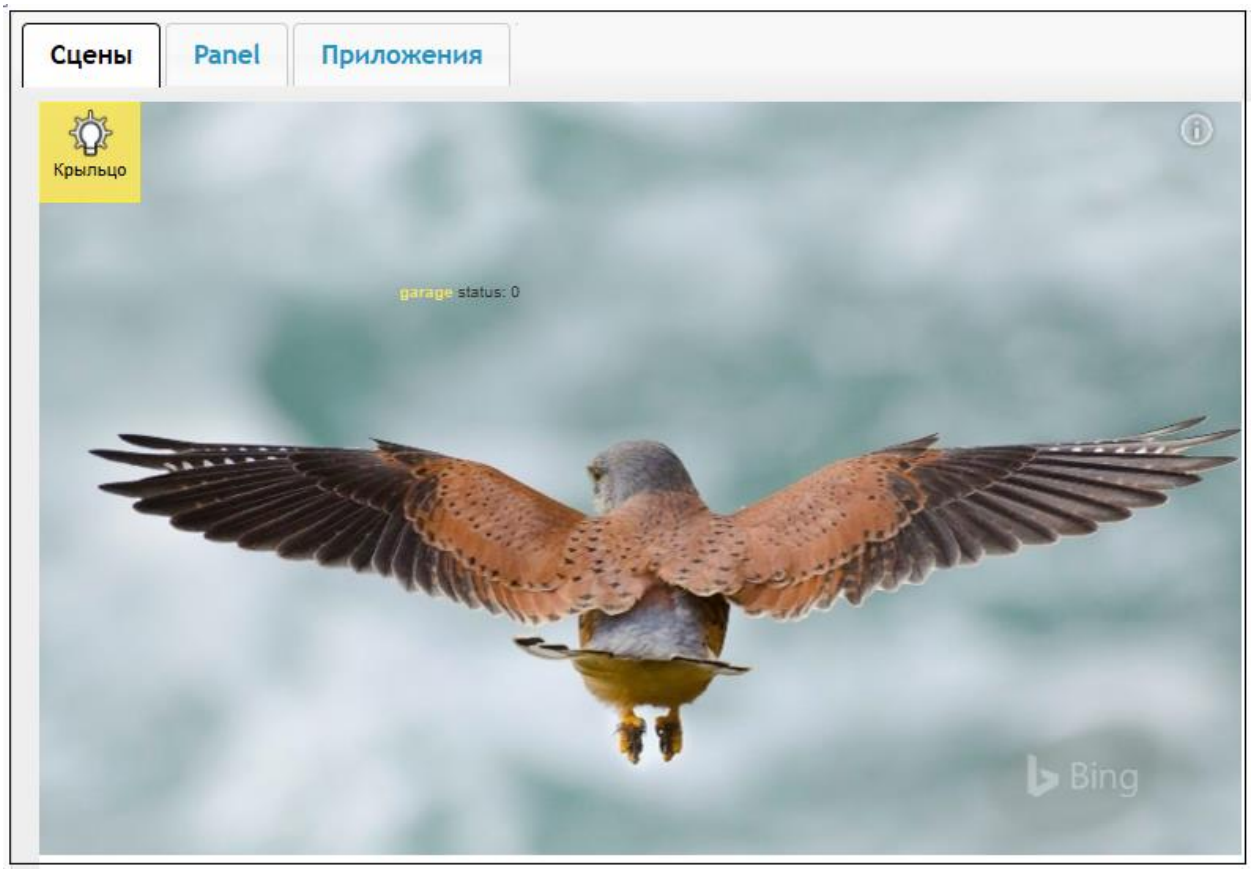


Рис. 6.16. Изменение поведения иконки нового объекта

Опыт с модулями ESP8266 и OTA WeMOS D1

Мы провели опыт по общению модуля OTA WeMOS D1 с системой MajorDoMo и опыт по управлению модулем ESP8266 с сервера MajorDoMo. Теперь было бы интересно провести опыт по общению двух модулей без «мажордома». Положим, что модуль OTA WeMOS D1 будет обслуживать освещение вокруг дома, командуя модулем ESP8266 с реле.

Начнем с команд:

`AT+CWMODE_DEF=1`, если нужный режим не установлен и

`AT+CWJAP_DEF = <ssid>,< pwd >,` если эта команда еще не использовалась.

Последняя команда выглядит так:

```
AT+CWJAP_DEF = “ваш провайдер”,”ваш пароль”
```

Когда модуль будет автоматически входить в домашнюю сеть, его следует настроить на работу в сети, выполнив команды:

```
AT+CIPMUX=1
AT+CIPSERVER=1,8080
AT+CIPSTO=6
```

Программу для модуля OTA WeMOS D1 загрузим такую:

```
  #include <ESP8266WiFi.h>
  WiFiClient client;
  const char* ssid = "BEELINE"; // Имя вашего роутера
  const char* password = "12345"; // Пароль на подключение к
роутеру
  String host = "192.168.10.119"; // Строка ip-адреса модуля
  int curIn=0; // Переменная для текущего состояния
  int old=0; // Переменная для предыдущего состояния
  char cmndOn[4]={160,1,1,162}; // Массив для A0 1 1 A2
  char cmndOff[4]={160,1,0,161}; // Массив для A0 1 0 A1

  void setup() {
    Serial.begin(115200); // Настройка последовательного порта
    delay(10);
    pinMode(4, INPUT); // Вывод 4 на вход для датчика

    // Подключаемся к сети по WiFi
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid); // Имя вашего роутер
    WiFi.begin(ssid, password); // Пароль на подключение к
роутеру

    while (WiFi.status() != WL_CONNECTED) { // Пока не
подключено
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");

  if (!client.connect(host, 8080)) { // Если нет подключения
    Serial.println("connection failed");
    return;
    }
  }

  void loop() {
    curIn = digitalRead(4); // Читаем состояние датчика
    if(old!=curIn) { // Если оно изменилось
    if (curIn==1) { // И состояние единица
      client.connect(host, 8080); // подключаемся к модулю
      delay(1000);
   client.write(cmndOn,4); // Отправляем команду включения
```

```
delay(1000);
client.stop(); // Останавливаем клиента
 }
 else { // Иначе, если состояние ноль
   client.connect(host, 8080); // Подключаемся к модулю
   delay(1000);
  client.write(cmndOff,4); // Отправляем команду выключения
  delay(1000);
  client.stop();
 }
 delay(4000);
 }
 old=curIn; // Теперь значение состояния «устарело»
}
```

Программа модифицирована на основании ранее использованной для первого опыта с модулем OTA WeMOS D1 и опытов с модулем ESP8266, которые проводились в предыдущем разделе этой главы.

COM-порты модулей различные, поэтому без каких-либо сомнений можно подключить OTA WeMOS D1 к монитору порта Arduino, а модуль ESP8266 к программе USR-TCP232, которая очень удачно подходит для отладки.

При подключении монитора порта можно увидеть отклик (**рис. 6.17**).

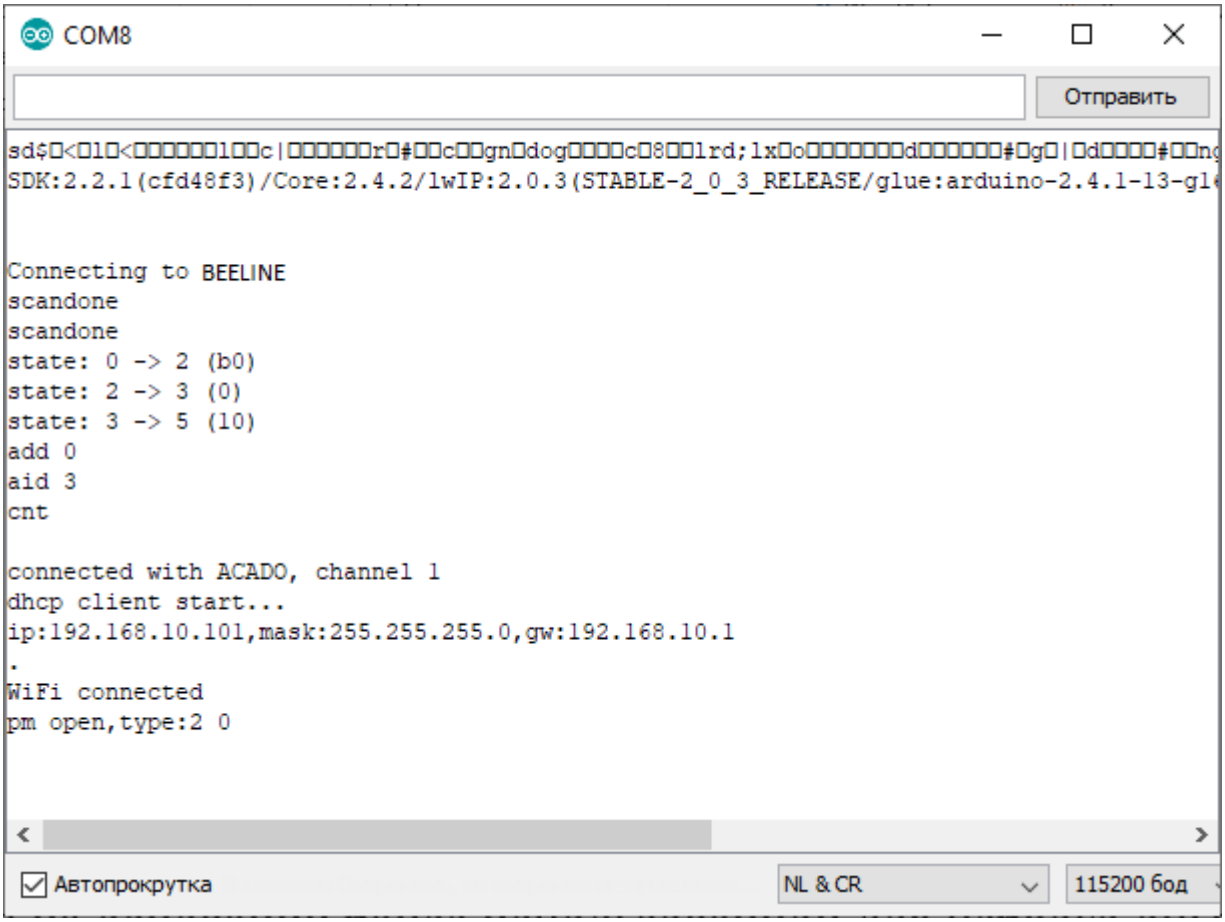


Рис. 6.17. Монитор порта, подключенный к OTA WeMOS D1

А программа USR-TCP232 показывает выполнение команд, когда меняется состояние вывода 4 WeMOS D1 (**рис. 6.18**). При этом реле при включении щелкает, как ему и положено.

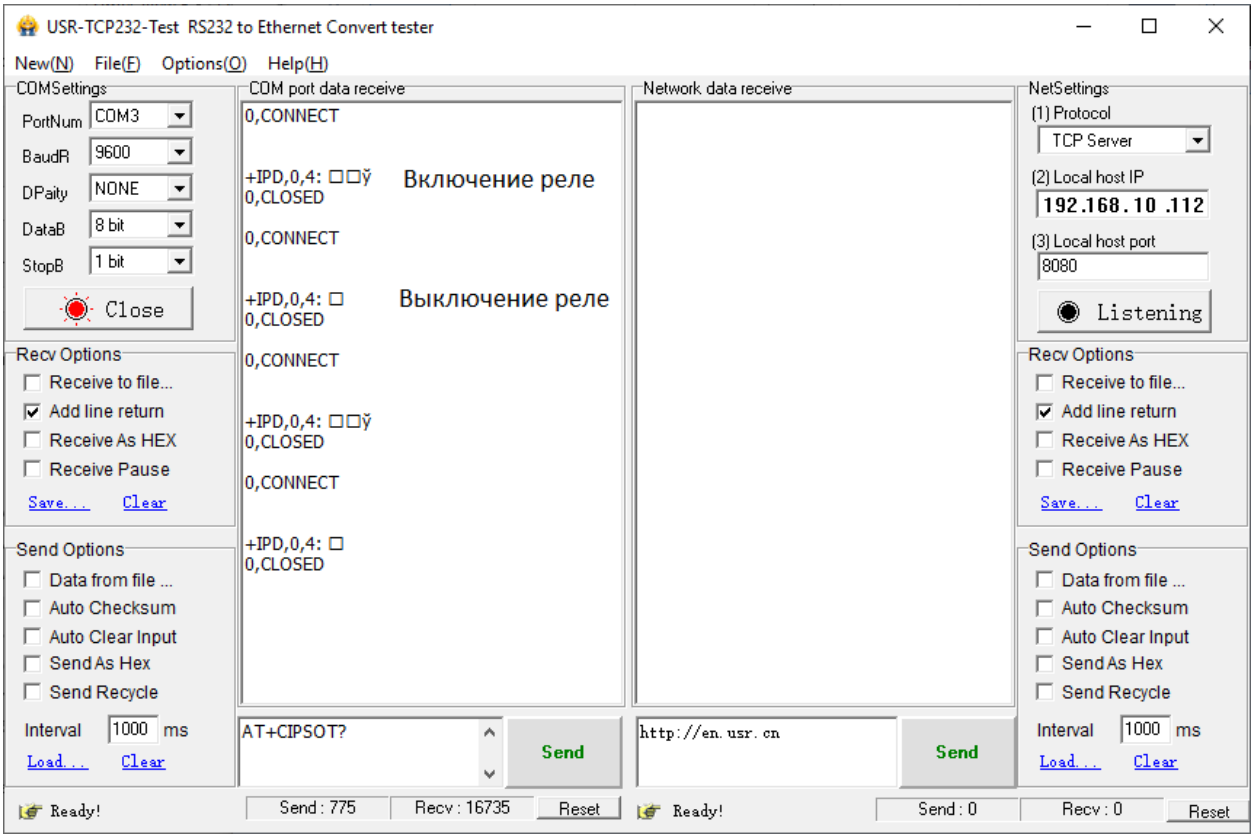


Рис. 6.18. Наблюдение за выполнениями команд

Освещение нужно включать только в темное время, а для этого нужно снабдить OTA WeMOS D1 средством для определения освещенности сада. Проще всего в качестве такого средства использовать фоторезистор, напряжение на котором будет меняться в зависимости от освещенности (нужен еще резистор и питающее напряжение!). С наступление темноты модуль OTA WeMOS D1 отдаст команду включить освещение, а утром отдаст команду выключить освещение.

Последовательно включенные фоторезистор и обычный резистор позволят с помощью АЦП модуля OTA WeMOS D1 определить изменение освещенности (см. Приложение Б, **рис. Б.3**). Выводы АЦП можно подключить и к фоторезистору, и к добавочному резистору. Проведем опыт с фотоприемником, подключив его к земле и выводу A0 модуля, последовательно с которым включен резистор 10 кОм. Питающее напряжение 3,3 В. Программа пока простая:

```
  int sensorPin = A0; // переменная для номера вывода на чтение напряжения
  int sensorValue = 0; // переменная для сохранения значения от датчика

  void setup() {
```

```
    Serial.begin(9600); // Скорость работы последовательного
порта
  }

  void loop() {
    // читаем значение напряжения от датчик:
    sensorValue = analogRead(sensorPin);
    // отправляем на монитор значение
   Serial.println(sensorValue);
    delay(1000);
  }
```

В зависимости от освещенности значение, возвращаемое ОТА WeMOS D1, меняется, как показано на **рис. 6.19**.

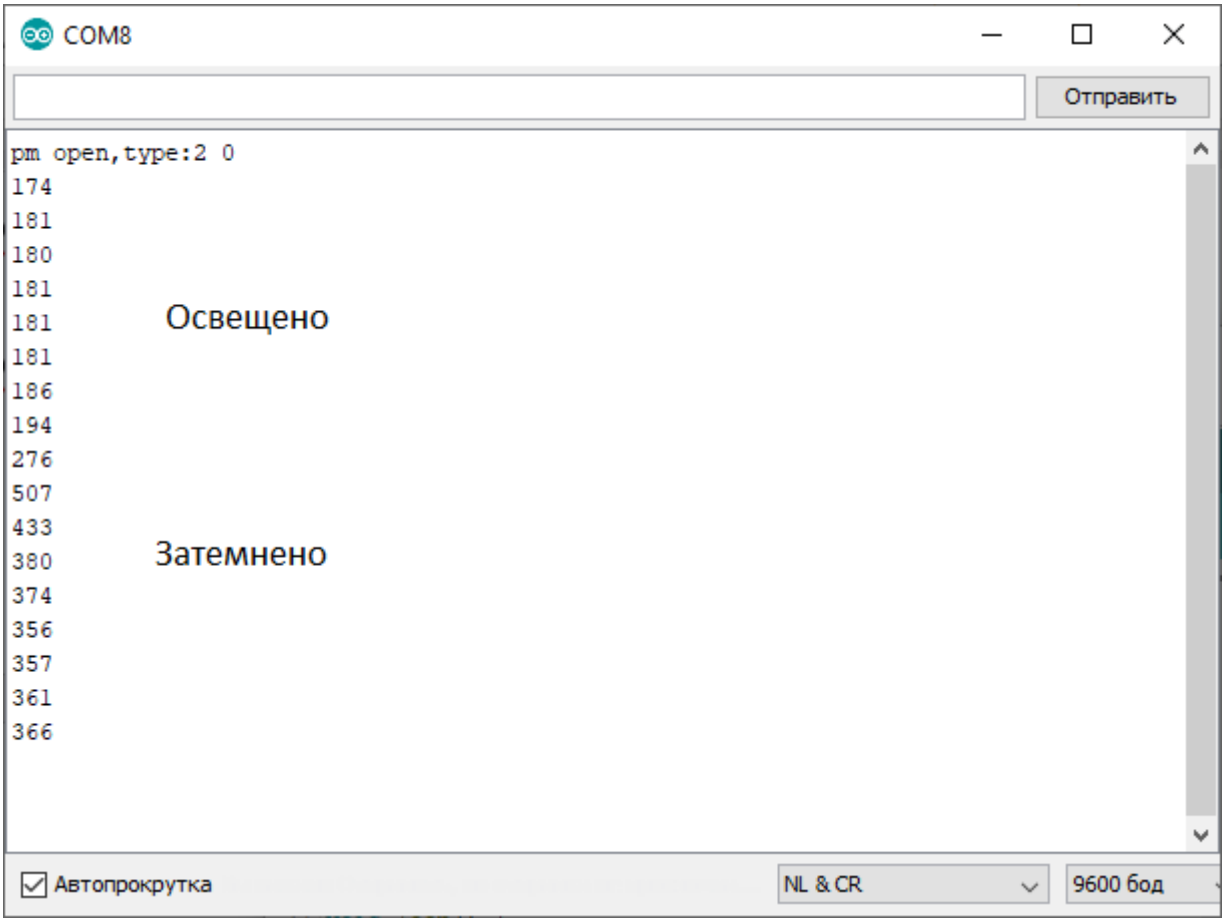


Рис. 6.19. Изменение показаний АЦП при изменении освещенности

Эту часть программы можно вставить в программу управления реле, получив такой вариант для проверки и отладки:

```
#include <ESP8266WiFi.h>
WiFiClient client;
const char* ssid = "BEELINE"; // Имя вашего роутера
const char* password = "12345678"; // Пароль на подключение
```

```
  String host = "192.168.10.119"; // ip-адрес модуля
  char cmndOn[4]={160,1,1,162}; // Массив строки включения
  char cmndOff[4]={160,1,0,161}; // Массив строки выключения
  int sensorPin = A0;    // Переменная аналогового входа
  int sensorValue = 0;  // Переменная для сохранения значения
напряжения от датчика
  int old=0; // Переменная для старого значения флага stat
  int stat=0; // Переменая для флага stat

  void setup() {
    Serial.begin(115200); // Скорость работы последовательного
порта
    delay(10);
    // Подключение к сети по WiFi
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid); // Имя вашего роутера
    WiFi.begin(ssid, password); // Пароль для подключения

    while (WiFi.status() != WL_CONNECTED) { // Пока нет
подключения
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
  if (!client.connect(host, 8080)) { // Если подключения нет
    Serial.println("connection failed");
    return;
    }
  }

  void loop() {
    while (old==stat) { // Пока значение не меняется
       sensorValue = analogRead(sensorPin); // Читаем значение
напряжения от датчика
       Serial.println(sensorValue); // Выводим его на монитор
порта
       if(sensorValue>200) { stat = 1; // Если значение больше
200, устанавливаем флаг stat
       }
       else { // Иначе сбрасываем флаг stat
       stat = 0;
      }
      Serial.println(stat); // Выводим значение флага на монитор
```

```
    Serial.println(old); // и выводим прежнее значение
    delay(2000);
  }

  if (stat == 1) { // Если флаг stat установлен
    client.connect(host, 8080); // Подключаемся к модулю
    delay(1000);
 client.write(cmndOn,4); // Отправляем команду включения
 delay(1000);
 client.stop(); // Останавливаем клиента
  }
  else { // Иначе, если флаг сброшен
    client.connect(host, 8080); // Подключаеся к модулю
    delay(1000);
   client.write(cmndOff,4); // Отправляем команду выключения
   delay(1000);
   client.stop(); // Останавливаем клиента
  }
  old=stat; // Состояние флага «устарело»
  delay(4000);
 }
```

Монитор порта при отладке показывает изменение освещенности и изменение переменных (**рис. 6.20**).

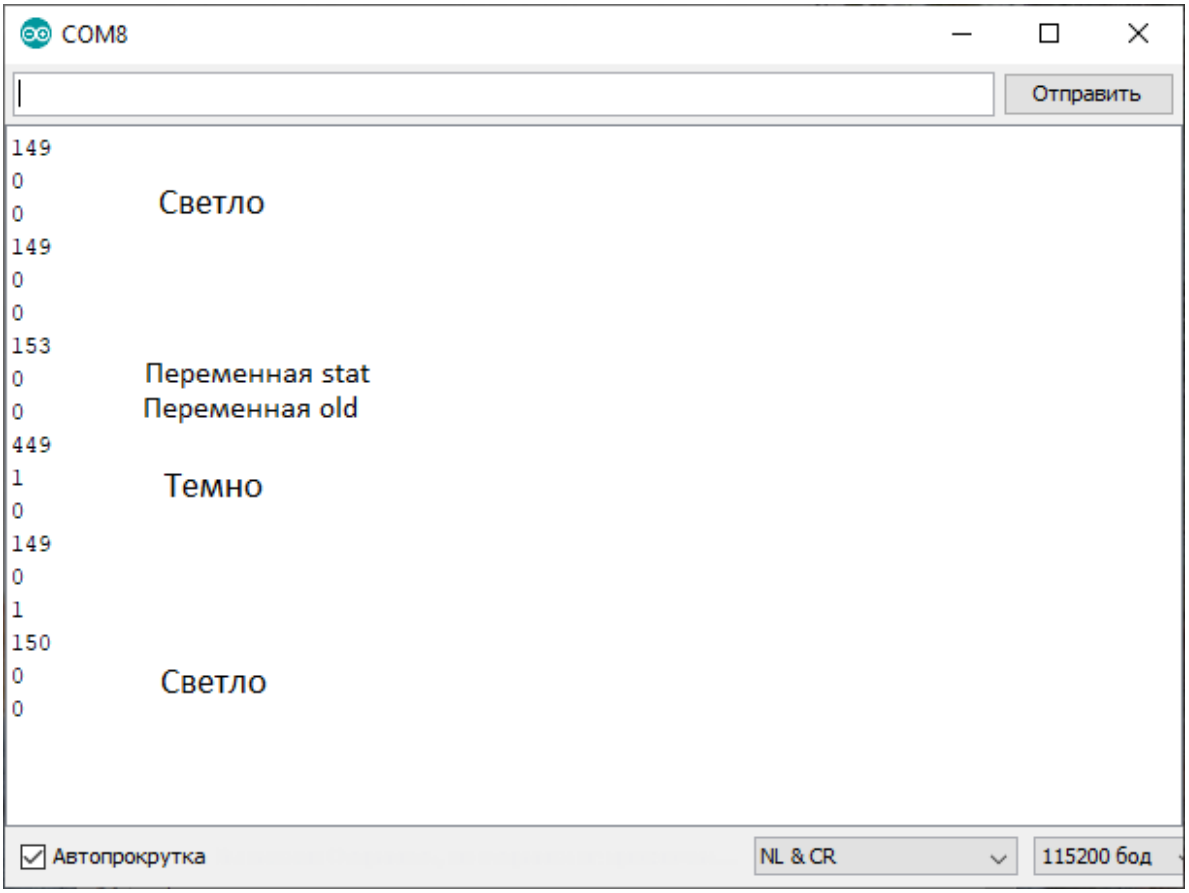


Рис. 6.20. Наблюдение за освещенностью и переменными

Используя программу USR-TCP232 можно наблюдать за поведением реле модуля ESP8266 (**рис. 6.21**).

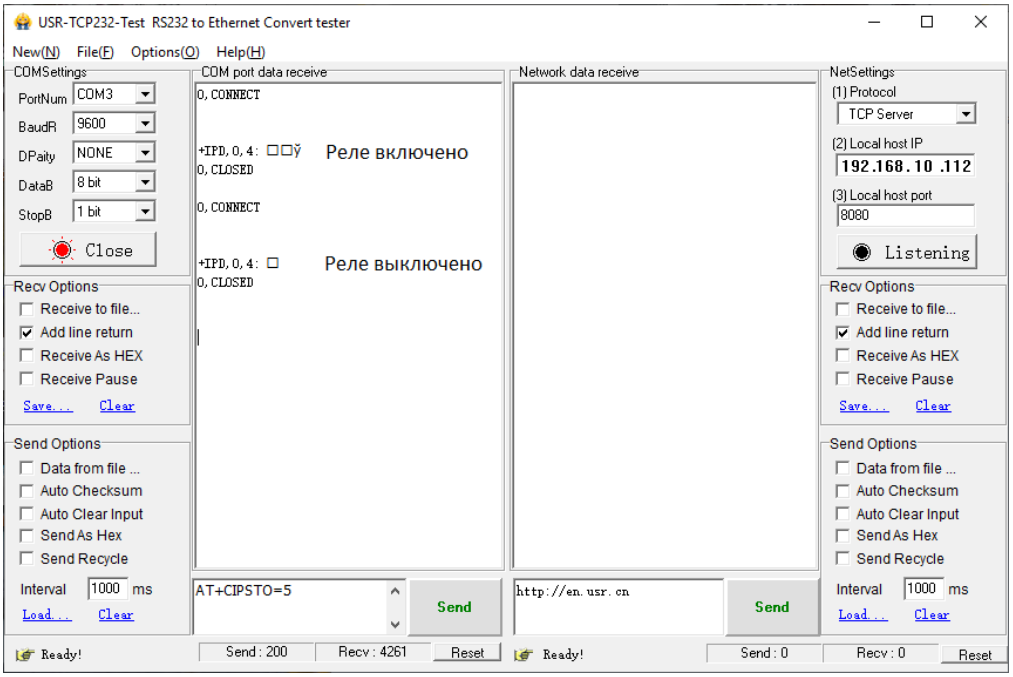


Рис. 6.21. Наблюдение за командами модулю ESP8266

Мы убедились, что модуль OTA WeMOS D1 может общаться с ESP8266 без посредника «мажордома», но сообщить «мажордому» о том, что свет в саду включен, наверное, следовало бы?

Пока мы экспериментируем, можно воспользоваться предыдущей заготовкой из раздела о модуле WeMOS D1 главы 5, добавив ее к работающей программе с ESP82266. Итог эксперимента: изменение статуса отображается в системе MajorDoMo, реле щелкает при затемнении фоторезистора (**рис. 6.22**).

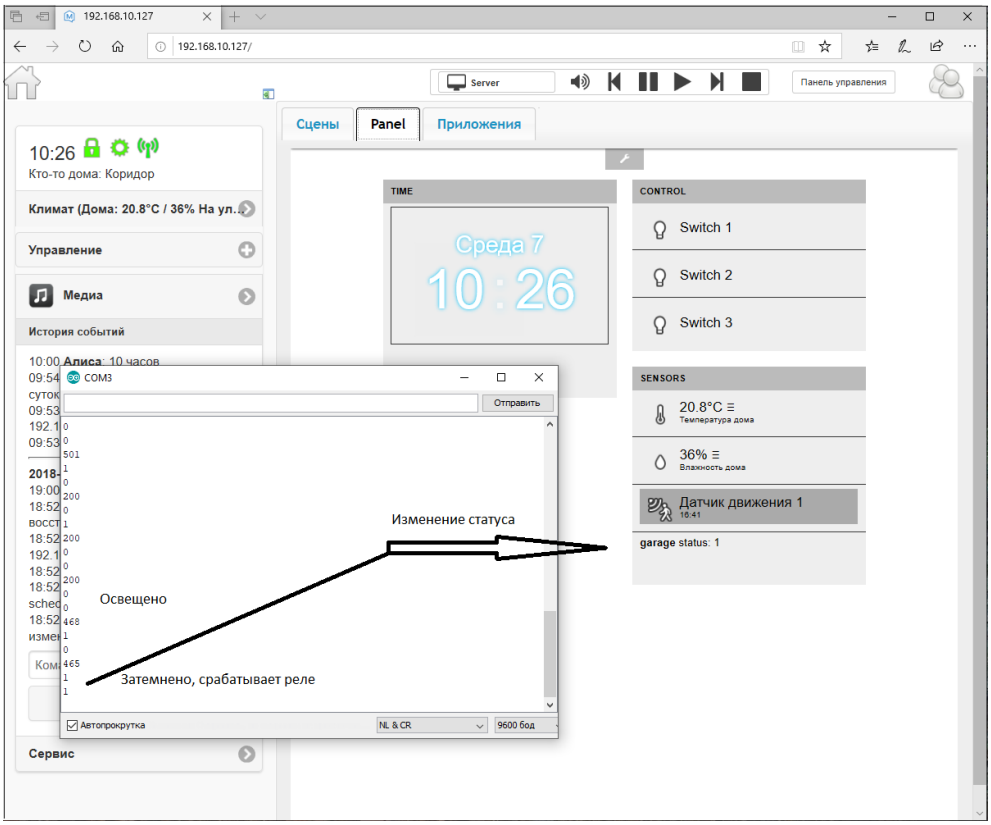


Рис. 6.22. Заключительный эксперимент этого раздела

Итак, два модуля OTA WeMOS D1 и ESP8266 общаются между собой, а модуль OTA WeMOS D1 общается с системой MajorDoMo. Программа выглядит так:

```
  #include <ESP8266WiFi.h>
  WiFiClient client;
  const char* ssid = "your-ssid"; // Имя вашего роутера
  const char* password = "pasword"; // Пароль на подключение к
сети
  String host1 = "192.168.10.119"; // ip-адрес первого модуля
  String host2 = "192.168.10.127"; // ip-адрес второго модуля
(Raspberry Pi)
  char cmndOn[4]={160,1,1,162}; // Массив для команды включения
реле
  char cmndOff[4]={160,1,0,161}; // Массив команды для
выключения реле
  int sensorPin = A0;    // Вывод для аналогового входа
  int sensorValue = 0;  // Переменная для считанного значения
  int old=0; // Старое значение флага
  int stat=0; // Текущее значение флага

  void setup() {
    Serial.begin(9600); // Скорость работы последовательного
порта
    delay(10);
```

```
    // Подключение к сети по WiFi
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid); // Имя роутера
    WiFi.begin(ssid, password); // Пароль для подключения к сети

    while (WiFi.status() != WL_CONNECTED) { // Пока нет
подключения к сети
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
  client.connect(host1, 8080); // Подключаем первый модуль
  if (!client.connect(host1, 8080)) { // Если нет подключения
    Serial.println("connection failed");
    return;
    }
  }

  void loop() {
    while (old==stat) { // Пока состояние флага не меняется
       sensorValue = analogRead(sensorPin); // Читаем напряжение
от датчика
       Serial.println(sensorValue); // Выводим на монитор порта
       if(sensorValue>200) { stat = 1; // Если значение больше
200, устанавливаем флаг stat
       }
       else { // Иначе сбрасываем флаг stat
       stat = 0;
      }
      Serial.println(stat); // Выводим на монитор значения
      Serial.println(old); // и новое и старое
      delay(2000);
    }

    if (stat == 1) { // Если флаг установлен
      client.connect(host1, 8080); // Подключаемся к модулю 1
      delay(1000);
   client.write(cmndOn,4); // Отправляем команду включения
   delay(1000);
   client.stop(); // Останавливаем клиента
       client.connect(host2, 80); // Подключаемся к модулю 2
(Raspberry Pi)
       delay(1000);
```

```
  // Отправляем строку запроса на установку состояния в единицу
       client.print(String("GET
/objects/?object=garage&op=m&m=status_d_input&status=1") + "
HTTP/1.1\r\n" + "Host: " + "192.168.10.127" + "\r\n" +
"Connection: close\r\n\r\n");
       delay(1000);
     }
     else { // Иначе
      client.connect(host1, 8080); // Подключаемся к модулю 1
       delay(1000);
      client.write(cmndOff,4); // Отправляем команду на
выключение реле
      delay(1000);
      client.stop(); // Останавливаем клиента
        client.connect(host2, 80); // Подключаемся к модулю 2
        delay(1000);
  // Отправляем строку запроса на установку состояния в ноль
       client.print(String("GET
/objects/?object=garage&op=m&m=status_d_input&status=0") + "
HTTP/1.1\r\n" + "Host: " + "192.168.10.127" + "\r\n" +
"Connection: close\r\n\r\n");
       delay(1000);
     }
     old=stat; //
     delay(4000);
  }
```

Я не буду настаивать на безоговорочной корректности программы, при реализации проекта, возможно, предстоит программу переписать более изящно, но для реализации следует учесть еще несколько факторов.

Во-первых, для приведения модуля ESP8266 к рабочему состоянию с помощью АТ команд его подключают к программатору (у меня это Arduino), но затем его следует подключить к внешнему питанию. Я пытался при программировании подключить только внешнее питание без питания от Arduino, но АТ команды модуль почему-то не принимал.

Пришлось при программировании питать от его от Arduino, затем подключить внешнее питание и отключить питание Arduino. Хотелось бы выполнять это как-то удачнее.

Во-вторых, фоторезистор следует при реализации располагать так, чтобы включенное освещение его не засветило. Более того, следует что-то придумать, если вы не хотите, чтобы свет в саду оставался на всю ночь. Это все решаемо, но не следует об этом забывать.

Примечание.

Экспериментируя с фоторезистором, я подумал о еще одном его применении.

Многие системы умного дома используют ИК-коды включения и выключения, например, телевизора. Но с пультом вы это делаете, когда видите, включен телевизор или нет. А при удаленном управлении хотелось бы знать это до отправки ИК-кода, иначе может статься, что кто-то из домашних смотрит телевизор, а вы запустили ИК-код, выключив телевизор на самом интересном месте.

Фоторезистор, направленный на экран выключенного телевизора, имеет сопротивление порядка 10 кОм, а при включенном телевизоре это сопротивление 5 кОм. Некоторые телевизоры имеют светодиоды, которые включены, когда телевизор выключен, и выключаются при его включении. Фоторезистор имеет сопротивление порядка 5 кОм при включенном телевизоре и 500 Ом при выключении, когда светодиод горит.

Опыт с Arduino Uno + Ethernet и BMP180

Раньше многие вешали один термометр за окном и второй в комнате. Сегодня есть устройство, которое совмещает измерение этих температур, показывая и температуру на улице, и температуру на подоконнике.

Измерение температуры имеет отношение к климат-контролю в системах умного дома, но этим не исчерпывается. Измерение температуры с помощью модуля Arduino и датчика DS18B20 давно освоено, но есть модуль BMP180, который измеряет и температуру, и атмосферное давление.

Модуль BMP180 работает с использованием интерфейса I^2C.

Напомню, что этот интерфейс использует один провод для передачи данных, SDA, а по второму передаются тактовые импульсы, SCL (относительно общего провода, земли). Поскольку для передачит данных в обе стороны используется одна линия, то, во избежание конфликтов, одно устройство всегда ведущее (Master), а второе – или другие, их может быть несколько – ведомое (Slave).

Общение начинает ведущий. Начинает он его с перевода линии SDA из высокого состояния в низкое в тот момент, когда линия SCL находится в высоком состоянии. Завершает работу на линии тоже ведущий, переводя линию SDA из низкого состояния в высокое при высоком уровне на лини SCL.

Передача данных может ограничиваться одним байтом, но может осуществляться в пакетном режиме. Тактовые сигналы SCL всегда формирует ведущий. Когда ведущий формирует сигнал «Старт», остальные устройства ожидают передачи адреса от главного. Все изменения на линии данных возможны только тогда, когда линия синхроимпульсов SCL находится в состоянии низкого уровня.

Первым на шине данных выставляется старший бит данных, а после передачи восьми битов ведущий ожидает подтверждения от ведомого, который должен выставить низкий уровень на линии SDA при высоком тактовом уровне, а ведущий, следуя протоколу обмена через интерфейс I2C, освобождает линию данных.

Если ведомое устройство занято другой работой, линия SDA остается в высоком состоянии, а ведущий может отправить сигнал «Стоп» для завершения работы. Но ведомое устройство может удержать линию SCL в низком состоянии, заставляя ведущего ожидать, когда ведомое устройство будет готово к общению.

У модуля Arduino Uno выводы SDA и SCL выведены отдельно, но с насадкой Ethernet, где эти выводы отсутствуют, их использовать неудобно. Решает эту проблему

существующий на плате вывод этих же линий на контакты A4 и A5 соответственно (см. Приложение Б, **рис. Б.4**).

Это важно!

Питающее напряжение для модуля BMP180, показанного на ***рис. 1.9****, 3,3 В, будьте внимательны!!*

Первый опыт мы проведем с модулем без участия «мажордома». Если библиотека для этого модуля у вас не добавлена к библиотекам Arduino, то можно воспользоваться **[13]** предлагаемым вариантом. После добавления библиотеки вы найдете среди примеров: *Файл→Примеры→Adafruit BMP085 Library→BMP085Test*. Программа после загрузки в модуль Arduino выведет такие данные (**рис. 6.23**).

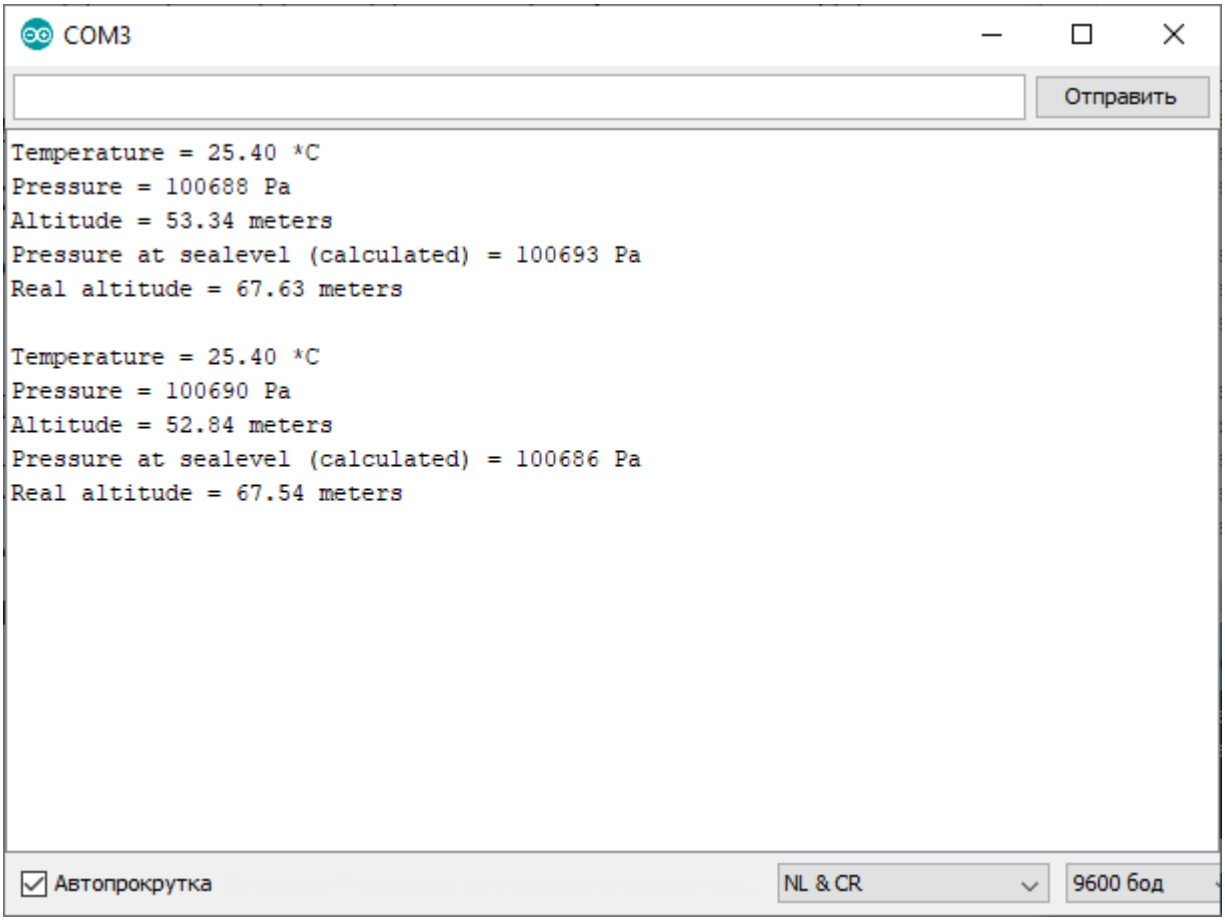


Рис. 6.23. Вывод данных на монитор порта тестовой программы

Нас в тестовой программе сейчас интересует температура и давление. И давление, мне так привычнее, хотелось бы видеть в миллиметрах ртутного столба.

Поищем соответствие.

Поиск дает такой результат: 1 па = 0075 мм.рт.ст. После умножения получается, что сейчас атмосферное давление 755 мм. ртутного столба. И это правильно (в коридоре висит обычный барометр). Переделаем программу так, чтобы можно было продолжить работу с ней:

```
#include <Wire.h>
```

```
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;
float temper = 0.0; // Переменная для хранения значения температуры
в десятичном виде
int pres = 0; // Переменная для хранения значения атмосферного
давления

void setup() {
  Serial.begin(9600); // Скорость работы порта последовательного
обменна данными
  bmp.begin(); // Начинаем работу датчика температуры и давления
}

void loop() {
    Serial.print("Temperature = ");
temper = bmp.readTemperature(); // Прочитываем значение температуры
    Serial.print(temper); // Выводим значение на монитор порта
    Serial.println(" *C");

    Serial.print("Pressure = ");
pres = bmp.readPressure()*0.0075; // Прочитываем значение
атмосферного давления и перводим из паскалей в мм. рт. столба
    Serial.print(pres); // Выводим давление на монитор порта
    Serial.println(" mm");

    Serial.println();
    delay(5000);
}
```

Теперь наша задача – отправить эти данные мажордому, используя ранее полученный вариант (глава 5) подключения Arduino к домашней сети.

```
#include <Ethernet.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;
EthernetClient rclient;

// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 120 }; // ip-адрес модуля
byte subnet[] = { 255, 255, 255, 0 }; // Маска подсети
byte gateway[] = { 192, 168, 10, 1 }; // ip-адрес шлюза
byte dns_server[] = { 192, 168, 10, 1 }; // ip-адрес DNS

// ip-адрес удалённого сервера (Raspberry Pi)
```

```
byte server[] = { 192, 168, 10, 127 };
float temper = 0.0; // Переменная для считывания температуры
int pres = 0; // Переменная для считывания атмосферного давления

// Инициализация начальных значений
char buf[80]; // Массив для строки запроса
char ipbuff[16]; // Массив для ip-адреса

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf); // Вывод буфера
  if (rclient.connect(server, 80)) { // Подключение к серверу
   Serial.println("OK");
   rclient.print(buf); // Отправка строки запроса
   rclient.println(" HTTP/1.0");
   rclient.print("Host: ");
// Перевод массива ip-адреса в строку
   sprintf(ipbuff, "%u.%u.%u.%u", ip[0], ip[1], ip[2], ip[3]);
   rclient.println(ipbuff); // Отправка ip-адреса в текстовом виде
   rclient.print("Content-Type: text/html\n");
   rclient.println("Connection: close\n");
   delay(2000);

   rclient.stop(); // Остановка клиента
  } else { // Иначе
   Serial.println("FAILED");
  }
}

void setup()
{
  // Вывод отладочных сообщений будет на монитор порта
  Serial.begin(9600); // Скорость работы порта
  Serial.println("Start");
// Инициализируем Ethernet Shield
 Ethernet.begin(mac, ip, dns_server, gateway, subnet);
  bmp.begin(); // Включаем работу датчика
}

void loop()
{
temper = bmp.readTemperature(); // Прочитываем температуру
pres = bmp.readPressure()*0.0075; // Прочитываем давление
Serial.println(temper); // Выводим температуру и давление
Serial.println(pres); // на монитор порта
// Строка запроса на изменение состояния датчика
    sprintf(buf, "GET
/objects/?object=garage&op=m&m=status_d_input&status=%i",
(int)current_garage);
    sendHTTPRequest(); // Отправляем запрос
```

```
    delay(1000);
        sprintf(buf, "GET
/objects/?object=garage&op=m&m=status_d_input&status=%i",
(int)current_garage);
        sendHTTPRequest();
    delay(1000);
    }
```

Пока это заготовка, пока это еще не программа, поскольку строки, выделенные в заготовке, предстоит исправить. Их следует привести в соответствие с настройками MajorDoMo. Заглянем на предустановленную сцену (**рис. 6.24**).

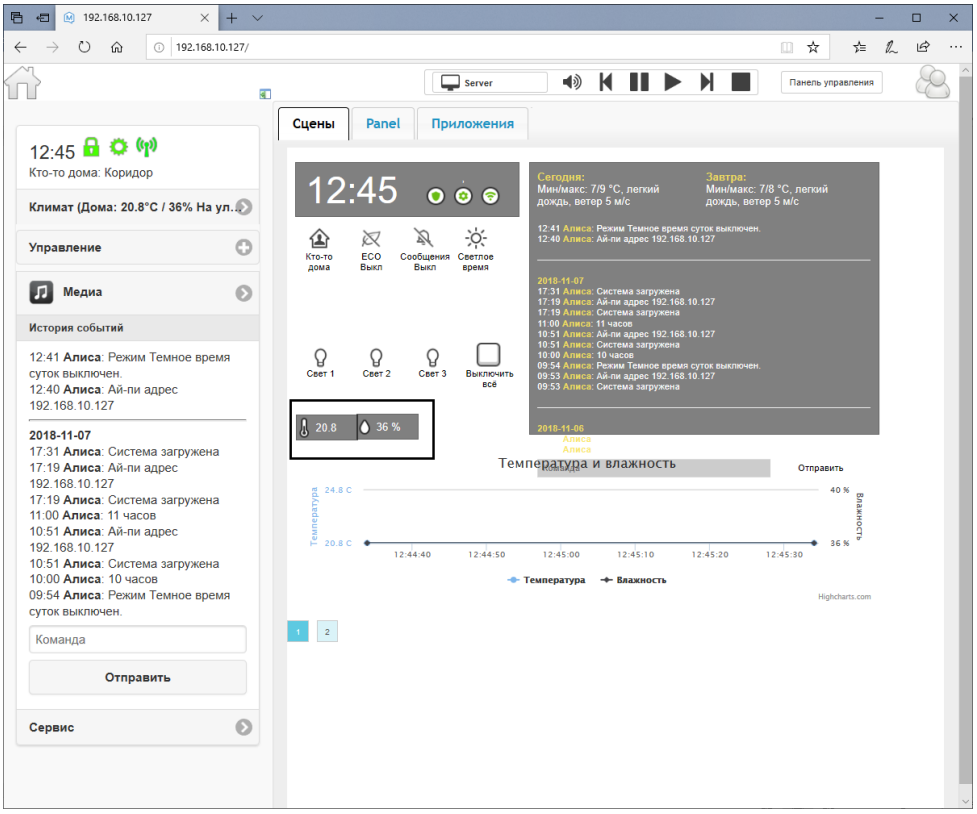


Рис. 6.24. Демонстрационная сцена MajorDoMo

Для этой сцены уже сделаны настройки для вывода температуры. Хотелось бы это использовать, попробуем. Заглянем в раздел объектов через панель управления (*Объекты→SDevices→SSensors*). Оказывается, кроме датчика температуры есть и заготовка для датчика давления (**рис. 6.25**).

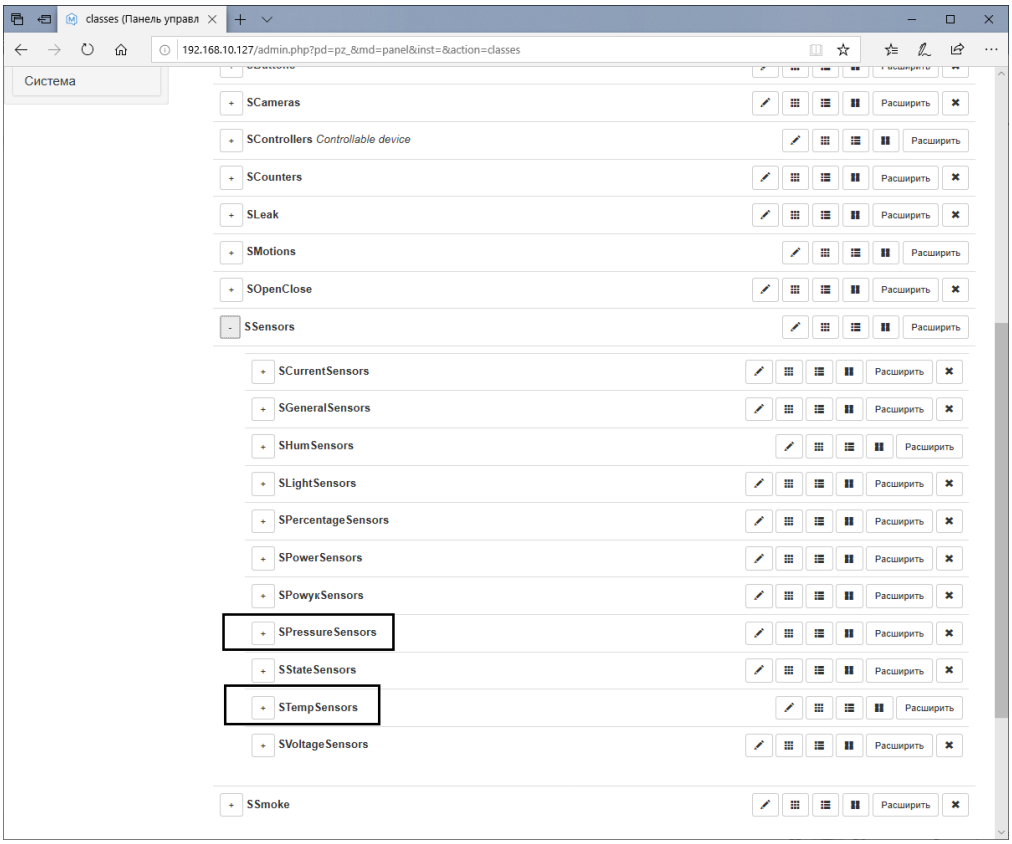


Рис. 6.25. Страница датчиков MajorDoMo

Казалось бы, что не надо ничего делать. Но ничего не делать, ничего и не выйдет!

Обрадованный тем, что заготовки в MajorDoMo для эксперимента есть, я попытался их использовать, начав с датчика температуры. Однако предыдущий удачный эксперимент использования функции GET в программе для Arduino не сработал. Температура не менялась. Несколько попыток что-то сделать привели к тому, что я в очередной раз, правда, не поняв, что я сделал не так, испортил систему MajorDoMo. Зайти на Raspberry Pi я мог, а попытка обратиться из браузера к MajorDoMo повисала на загрузке базы данных.

Возможно, проблема была бы решена, если бы я сделал резервную копию (*Панель управления→Система→Проверка обновлений*), **рис. 6.26**.

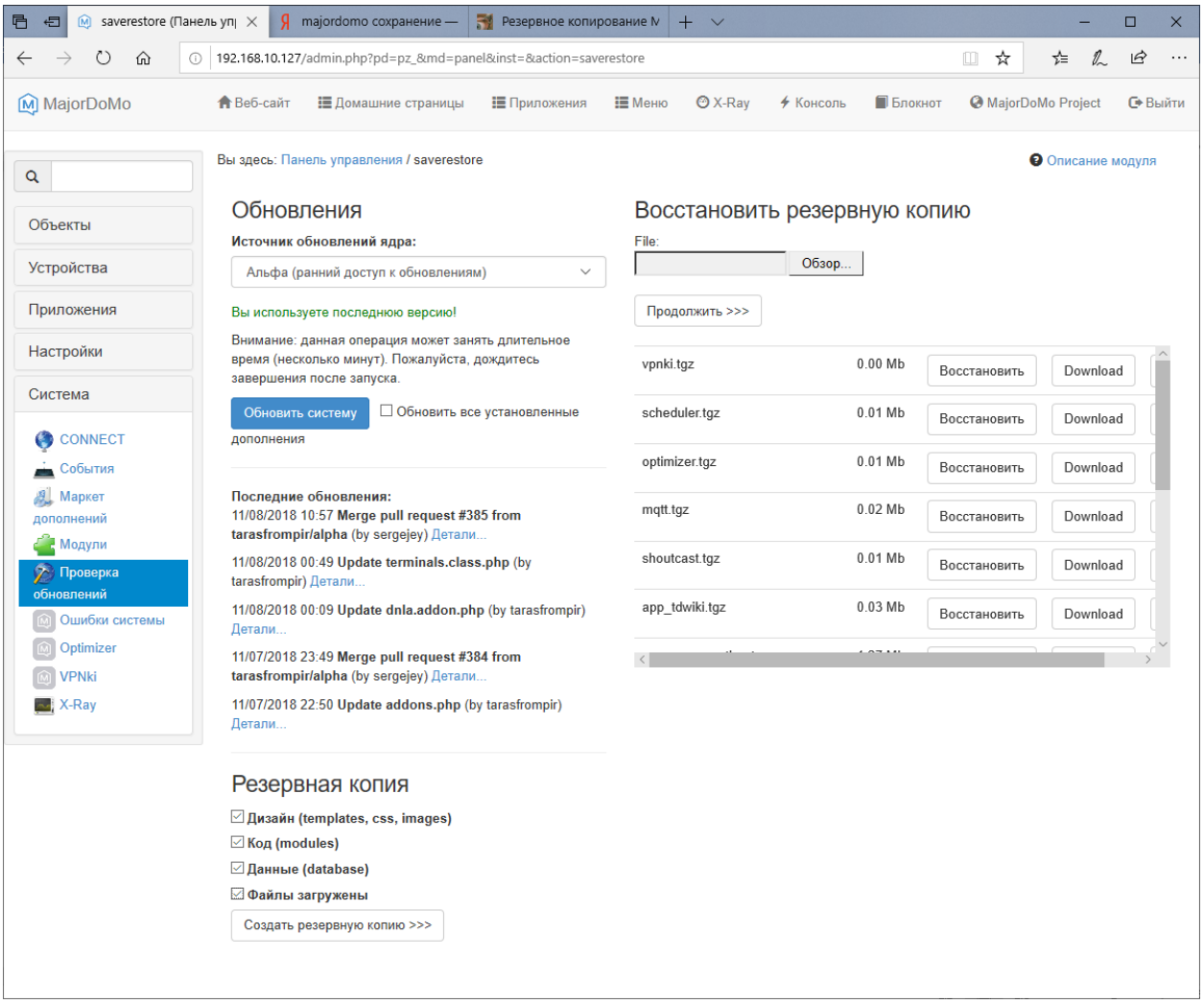


Рис. 6.26. Страница сохранения резервной копии в MajorDoMo

Хранится эта резервная копия на Raspberry Pi по адресу: */var/www/html/cms/saverestore*. Хорошо бы, конечно, попробовать восстановление, но резервную копию я не сделал, поэтому пришлось заново устанавливать MajorDoMo, освободив SD-карту от предыдущей версии.

Это важно!

Будьте умнее, сделайте резервную копию состояния проекта заранее!

Разбираясь с проблемой датчика температуры, я обратил внимание на то, что в настройках есть ссылка (**рис. 6.27**).

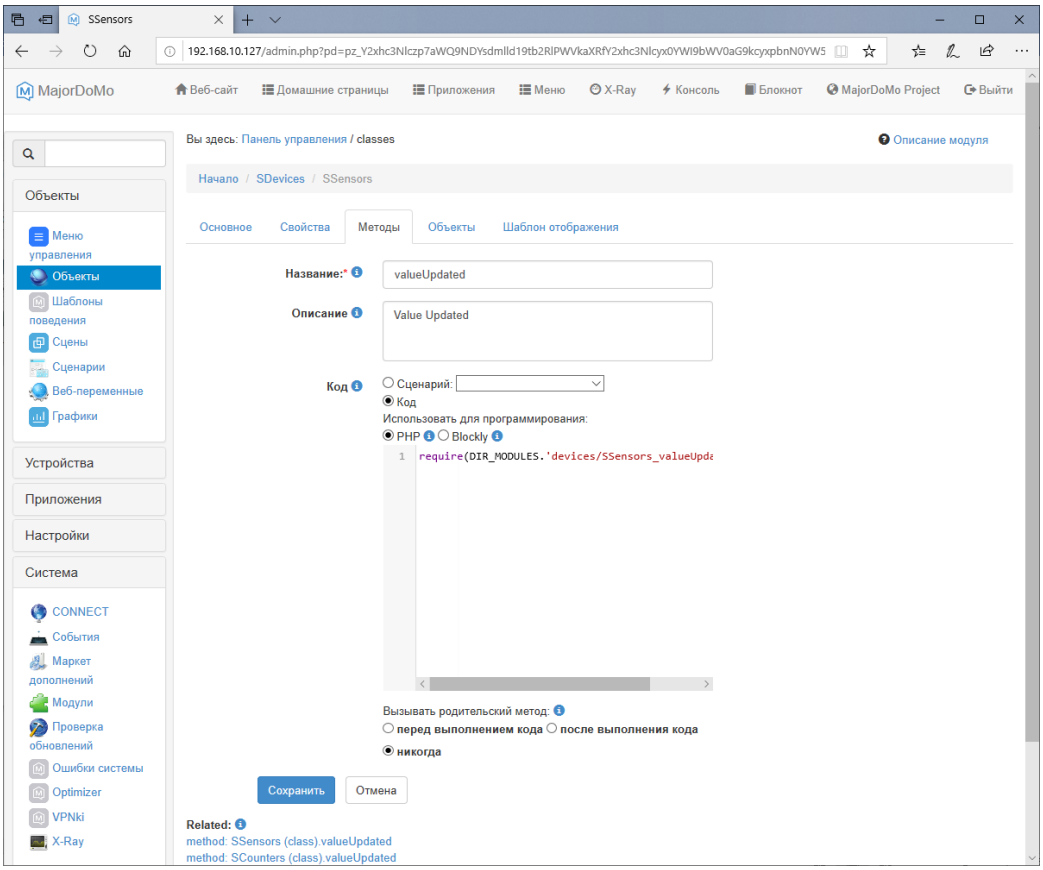


Рис. 6.27. Ссылка в настройках датчика температуры

Если вы хорошо разбираетесь в языке PHP, то можете заглянуть в этот файл, который найдете здесь (**рис. 6.28**).

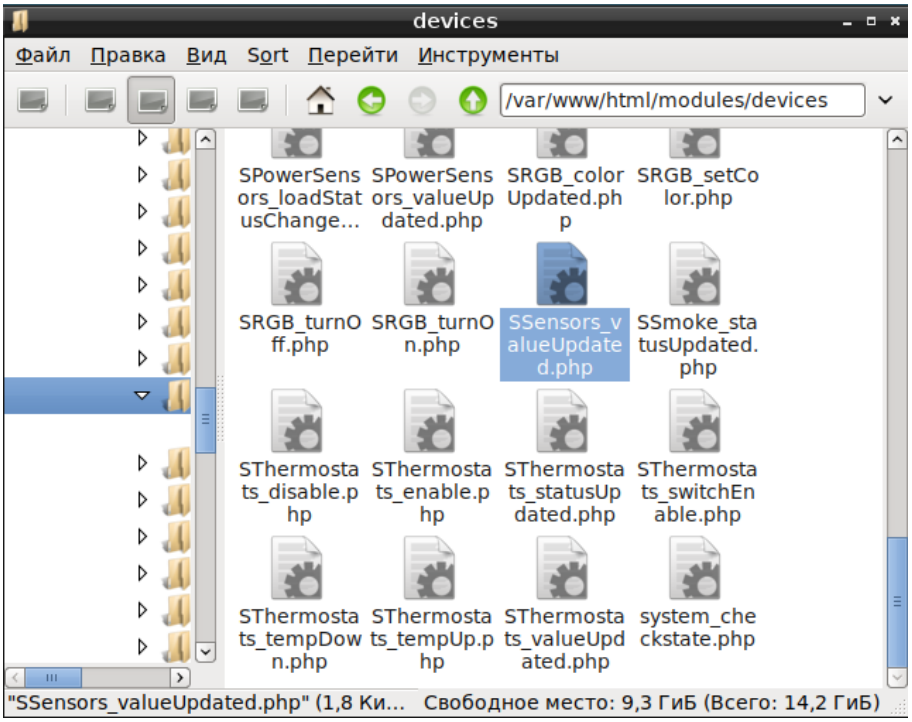


Рис. 6.28. Расположение файла ссылки

Меня этот файл заинтересовал тем, что, пытаясь получить отображение текущей температуры, я не добился результата, и мне хотелось бы понять, почему это произошло?

Я попытаюсь описать текст файла так, как я его понимаю:

```
<?php

    $this->callMethod('statusUpdated'); // Вызов метода изменения
состояния
    //$this->callMethod('logicAction'); // Вызов метода
logicAction

    $ot=$this->object_title; // Название объекта
    $description = $this->description; // Описание объекта
    if (!$description) { // Если нет описания
     $description = $ot; // Добавить описание
    }
    $linked_room=$this->getProperty('linkedRoom'); // Комната с
датчиком
    $value=(float)$this->getProperty('value'); // Читаем значение
    $minValue=(float)$this->getProperty('minValue'); //
Минимальное значени
    $maxValue=(float)$this->getProperty('maxValue'); //
Максимальное значение
    $is_normal=(int)$this->getProperty('normalValue'); //
Нормальное значение
    $directionTimeout=(int)$this-
>getProperty('directionTimeout'); // Время обновления
направления
    if (!$directionTimeout) { // Если нет времени обновления
     $directionTimeout=1*60*60;
    }
   // Если занчения нулевые и не равны нормальному
    if ($maxValue==0 && $minValue==0 && !$is_normal) {
     $this->setProperty('normalValue', 1);
   // Иначе, если значение больше максимального и меньше
минимально установить в ноль нормальное значение
    } elseif (($value>$maxValue || $value<$minValue) &&
$is_normal) {
     $this->setProperty('normalValue', 0);
   // Если есть уведомление
     if ($this->getProperty('notify')) {
      //уведомление о выходе за пределы
      say(LANG_DEVICES_NOTIFY_OUTOFRANGE. ' ('.$description.'
'.$value.')', 2);
     }
```

```
   // Иначе, если значение меньше максимального и больше
минимального, но не нормальное
    } elseif (($value<=$maxValue && $value>=$minValue) &&
!$is_normal) {
     $this->setProperty('normalValue', 1);
     if ($this->getProperty('notify')) {
      //возвращение к нормальному уведомлению
      say(LANG_DEVICES_NOTIFY_BACKTONORMAL. ' ('.$description.'
'.$value.')', 2);
     }
    }
// Получение предыдущего значения
    $data1 = getHistoryValue($this->object_title.".value",
time()-$directionTimeout);
    $direction = 0;
    if ($data1>$value) { // Если предыдущее значение было больше
     $direction=-1; // вывести направление -1
    } elseif ($data1<$value) { // Если предыдущее значение было
меньше текущего
     $direction=1; // вывести направление 1
    }
    $currentDirection = $this->getProperty('direction');
   // Если текущее направление не равно предыдущему
   if ($currentDirection != $direction) {
     $this->setProperty('direction',$direction); // изменить
    }
   // Если это основной датчик, то произвести изменения
    if ($linked_room && $this->getProperty('mainSensor')) {
     if ($this->class_title=='STempSensors') {
      sg($linked_room.'.temperature',$value);
     } elseif ($this->class_title=='SHumSensors') {
      sg($linked_room.'.humidity',$value);
     }
    }

   /*
   include_once(DIR_MODULES.'devices/devices.class.php');
   $dv=new devices();
   $dv->checkLinkedDevicesAction($this->object_title, $value);
   */
```

Я попробовал изменить заданное начально значение температуры в свойствах термометра (**рис. 6.29**), проделав этот несколько раз, и получил странные для себя результаты.

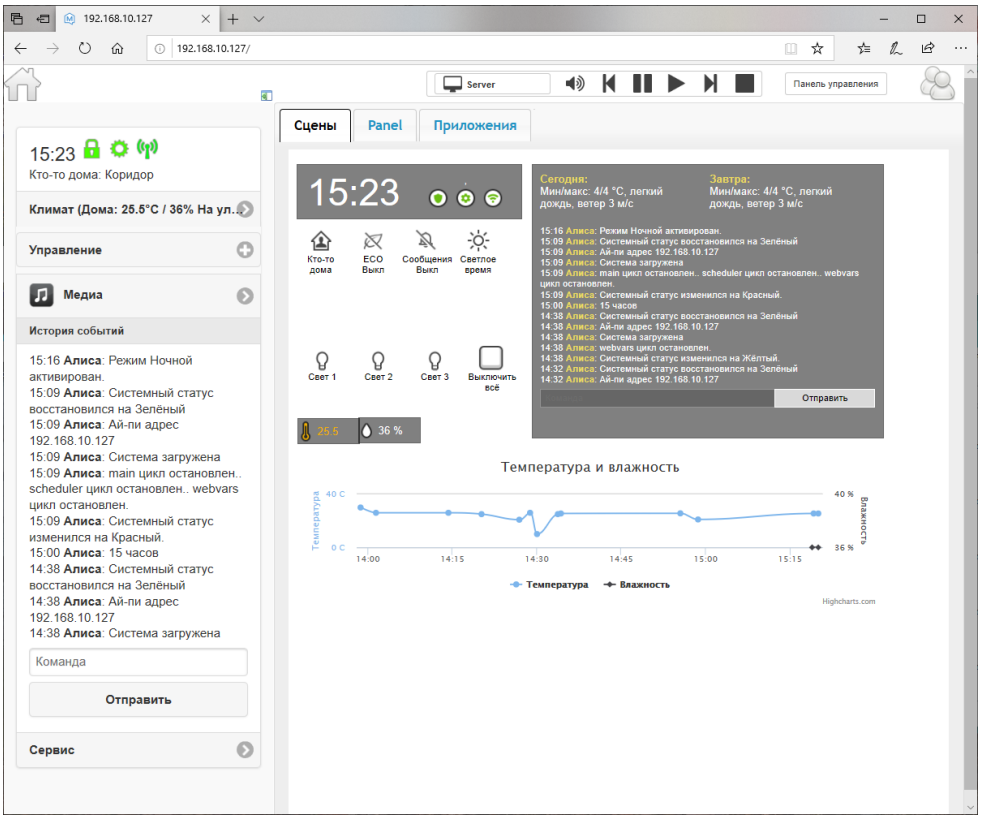


Рис. 6.29. Изменение значения температуры в доме

Неожиданно заработал график температуры, и при этом на панели MajorDoMo, что я, признаюсь, заметил случайно, внесенные мной изменения отображаются в виде значка направления изменений (**рис. 6.30**).

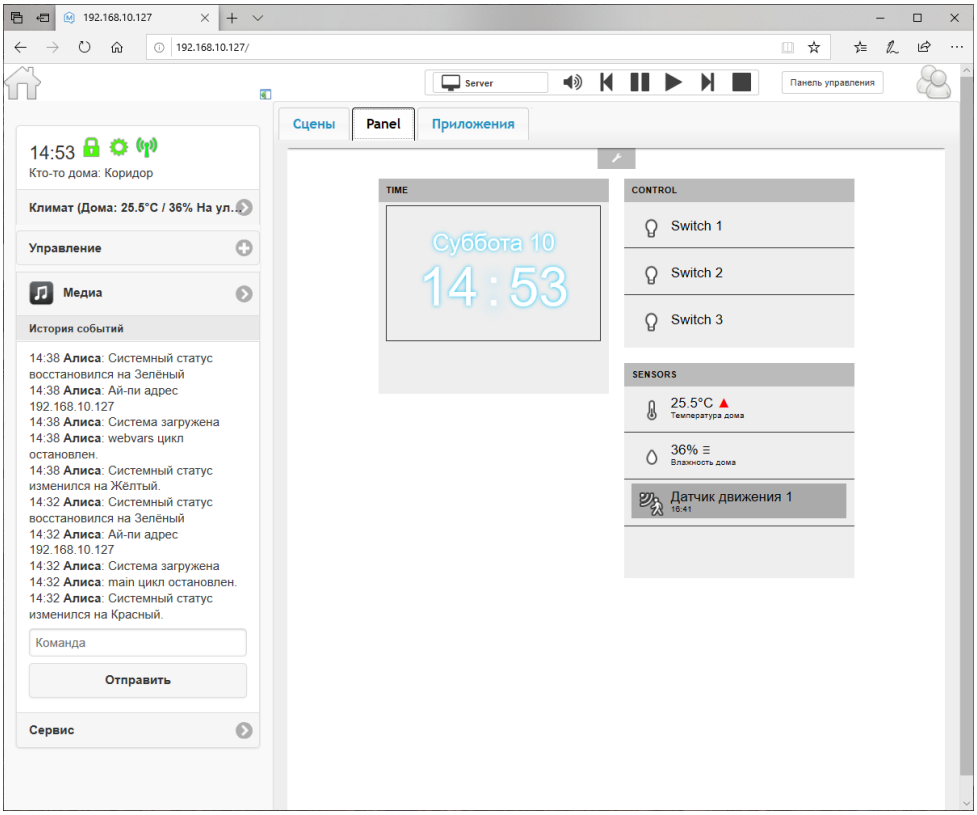


Рис. 6.30. Температура в доме на панели MajorDoMo

А, главное, результат, полученный мной от наблюдения за монитором порта Arduino, тоже не показался мне ясным и понятным (**рис. 6.31**).

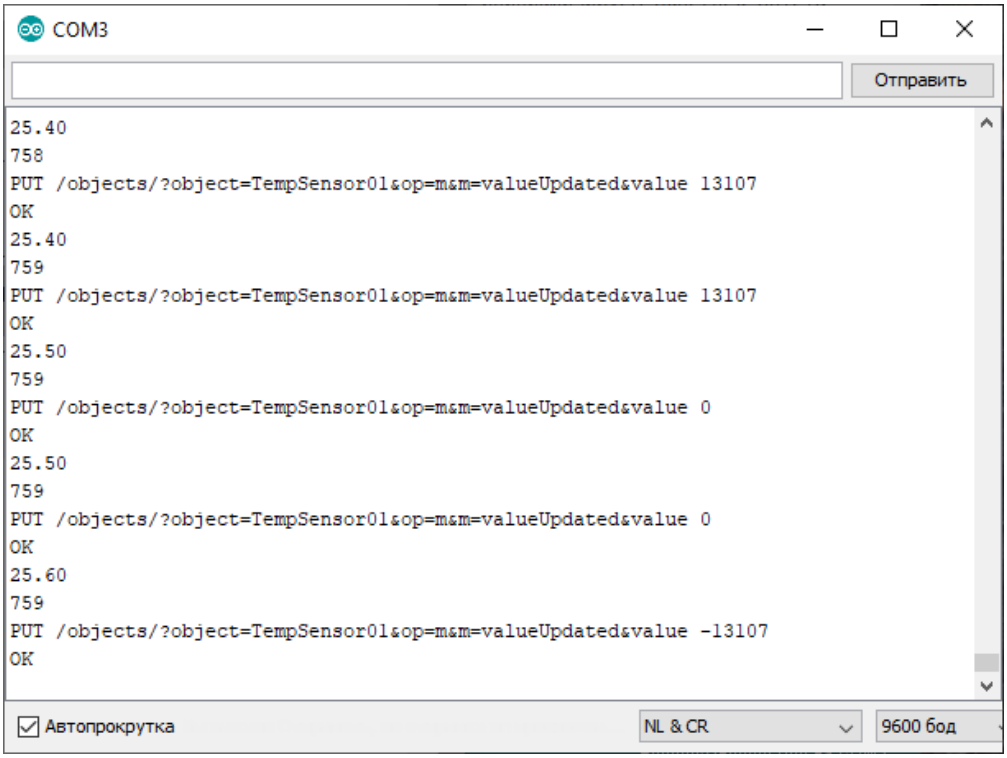


Рис. 6.31. Отображение ситуации на мониторе порта Arduino

Похоже, предустановленные датчики предназначены не совсем для тех целей, что нужны мне сейчас. Хорошо бы узнать о них у авторов программы, но я не хочу отвлекать их от работы, поэтому оставлю разборку с этим для другого раза.

Мне не хотелось бы, исправляя уже готовые варианты по своему усмотрению, еще раз испортить MajorDoMo – я только-только заново все установил. Попробуем повторить удачный опыт с этим модулем, создав класс *Input* (после последнего краха этот класс исчез). Я не хочу выдумывать что-то новое, поэтому повторяю все шаги, описанные в разделе «Arduino с Ethernet-шилдом» главы 5.

Создаю новый объект этого класса, который называю *store* (кладовка), создаю свойство *temper* (температура) и метод *changeTemper* (изменение температуры), который тоже копирую из предыдущего опыта:

```
$this->setProperty('temper', $params['temper']);
```

Чтобы увидеть результат, я добавляю панель для *Scene 1*, где будет выводиться температура в кладовке.

Так я думал, что будет выводиться, поскольку использовал уже проверенную программу для Arduino, использовал уже работавшие настройки MajorDoMo. Но, если я предполагал, то не я располагал.

После некоторой моей «мышиной» возни, заработала строка:

http://192.168.10.127:80/objects/?object=store&op=m&m=changeTmper&temper=22.0

Но заработала она только в браузере. Отправляя разные значения температуры, я мог увидеть изменения на панели. А отправка значения из Arduino как не работала, так и не заработала. Пришлось еще раз разобраться с функцией `sendHTTPRequest()` в программе для Arduino. В итоге программа приобрела вид:

```
#include <Ethernet.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;
EthernetClient rclient;

// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 120 }; // ip-адрес модуля
byte subnet[] = { 255, 255, 255, 0 }; // Маска подсети
byte gateway[] = { 192, 168, 10, 1 }; // ip-адрес шлюза
byte dns_server[] = { 192, 168, 10, 1 }; // ip DNS сервера

// ip-адрес удалённого сервера
byte server[] = { 192, 168, 10, 127 };
int temper = 0; // Переменная для температуры как целого
int pres = 0; // Переменная для атмосферного давления
char buf[80]; // Массив для строки запроса

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
  if (rclient.connect(server, 80)) { // Подключение к серверу
   Serial.println("OK");
   rclient.print(buf); // Отправка строки запроса
   rclient.println(" HTTP/1.0");
   rclient.print("Host: ");
   rclient.println("192.168.10.127"); // Строка ip-адреса
сервера (Raspberry Pi)
   rclient.println("Connection: close");
   rclient.println();
    rclient.println();
   delay(2000);
   rclient.stop(); // Останавливаем клиента
   rclient.flush(); // Очищаем
  } else {
   Serial.println("FAILED");
  }
```

```
}

void setup()
{
    Serial.begin(9600); // Скорость работы порта
  Serial.println("Start");
// Инициализируем Ethernet Shield
Ethernet.begin(mac, ip, dns_server, gateway, subnet);
  bmp.begin(); // Включаем датчик температуры и давления
}

void loop()
{
temper = bmp.readTemperature(); // Читаем температуру
pres = bmp.readPressure()*0.0075; // Читаем и приводим
давление к мм.рт.столба
Serial.println(temper);
// Строка для отправки значения температуры на сервер
    sprintf(buf, "GET
/objects/?object=store&op=m&m=changeTmper&temper=%i",
(int)temper);
    sendHTTPRequest(); // Отправляем запрос на сервер
delay(2000);
}
```

Теперь все заработало, но, если из командной строки я отправлял число с плавающей точкой, то из Arduino пришлось отправлять целое число. Вероятно, это поправимо, но мне пока этого не надо (**рис. 6.32**).

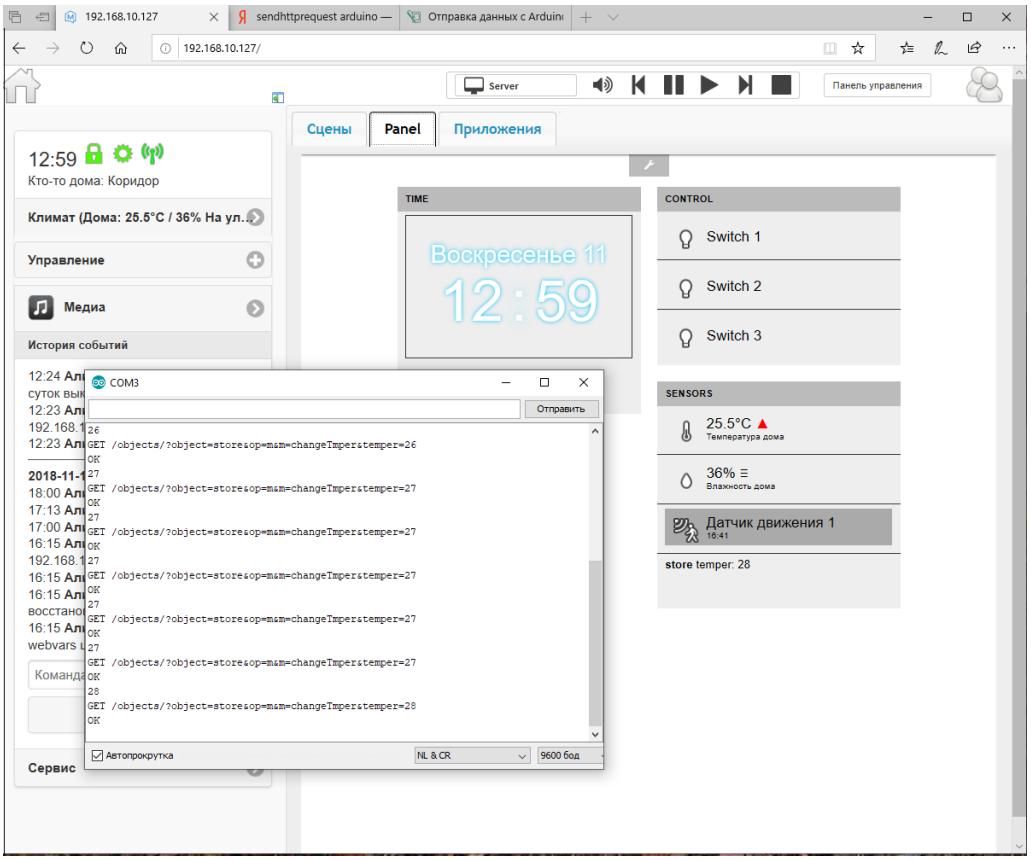


Рис. 6.32. Удачное отображение температуры от датчика

Осталось добавить вывод атмосферного давления, но я хотел обойтись небольшой доработкой. Но попытка добавить новый объект со своими свойствами и методами в класс *Input* оказалась неудачной. Отчего-то новый и старый объекты перемешались, а значения температуры и давления выводить не захотели. Пришлось создать новый класс *Pressure* по образу и подобию старого, чтобы добавить еще один объект *home* с его свойством атмосферного давления, *press*, и методом *changePress*. После добавления этого объекта на панель, после правки программы для Arduino, которая приобрела вид:

```
#include <Ethernet.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;
EthernetClient rclient;

// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 120 }; // ip-адрес модуля
byte subnet[] = { 255, 255, 255, 0 }; // Маска подсети
byte gateway[] = { 192, 168, 10, 1 }; // ip-адрес шлюза
byte dns_server[] = { 192, 168, 10, 1 }; // ip-адрес DNS
```

```
  byte server[] = { 192, 168, 10, 127 }; // ip-адрес удалённого
сервера

  int temper = 0; // Переменная для температуры
  int pres = 0; // Переменная для давления
  char buf[80]; // Массив для строки запроса

  // Функция отправки HTTP-запроса на сервер
  void sendHTTPRequest() {
    Serial.println(buf);
    if (rclient.connect(server, 80)) { // Подключение к серверу
     Serial.println("OK");
     rclient.print(buf);
     rclient.println(" HTTP/1.0");
     rclient.print("Host: ");

  rclient.println("192.168.10.127"); // Строка адреса сервера
     rclient.println("Connection: close");
     rclient.println();
      rclient.println();
     delay(2000);
     rclient.stop(); // Остановка клиента
     rclient.flush(); // Очистка
    } else {
     Serial.println("FAILED");
    }
  }

  void setup()
  {
    Serial.begin(9600); // Скорость работы последовательного
порта
    Serial.println("Start");
  // Инициализируем Ethernet Shield
  Ethernet.begin(mac, ip, dns_server, gateway, subnet);
    bmp.begin(); // Включаем датчик BMP180
  }

  void loop()
  {
  temper = bmp.readTemperature(); // Читаем температуру
  pres = bmp.readPressure()*0.0075; // Читаем давление
  // Строка для отправки запроса касательно температуры
  sprintf(buf, "GET
/objects/?object=store&op=m&m=changeTmper&temper=%i",
(int)temper);
```

```
    sendHTTPRequest();  // Отправляем запрос
  delay(1000);
  // Строка для отправки запроса по атмосферному давлению
    sprintf(buf, "GET
/objects/?object=home&op=m&m=changePress&press=%i", (int)pres);
    sendHTTPRequest();
  delay(2000);
  }
```

После этих операций я, наконец, увидел то, что хотел (**рис. 6.33**).

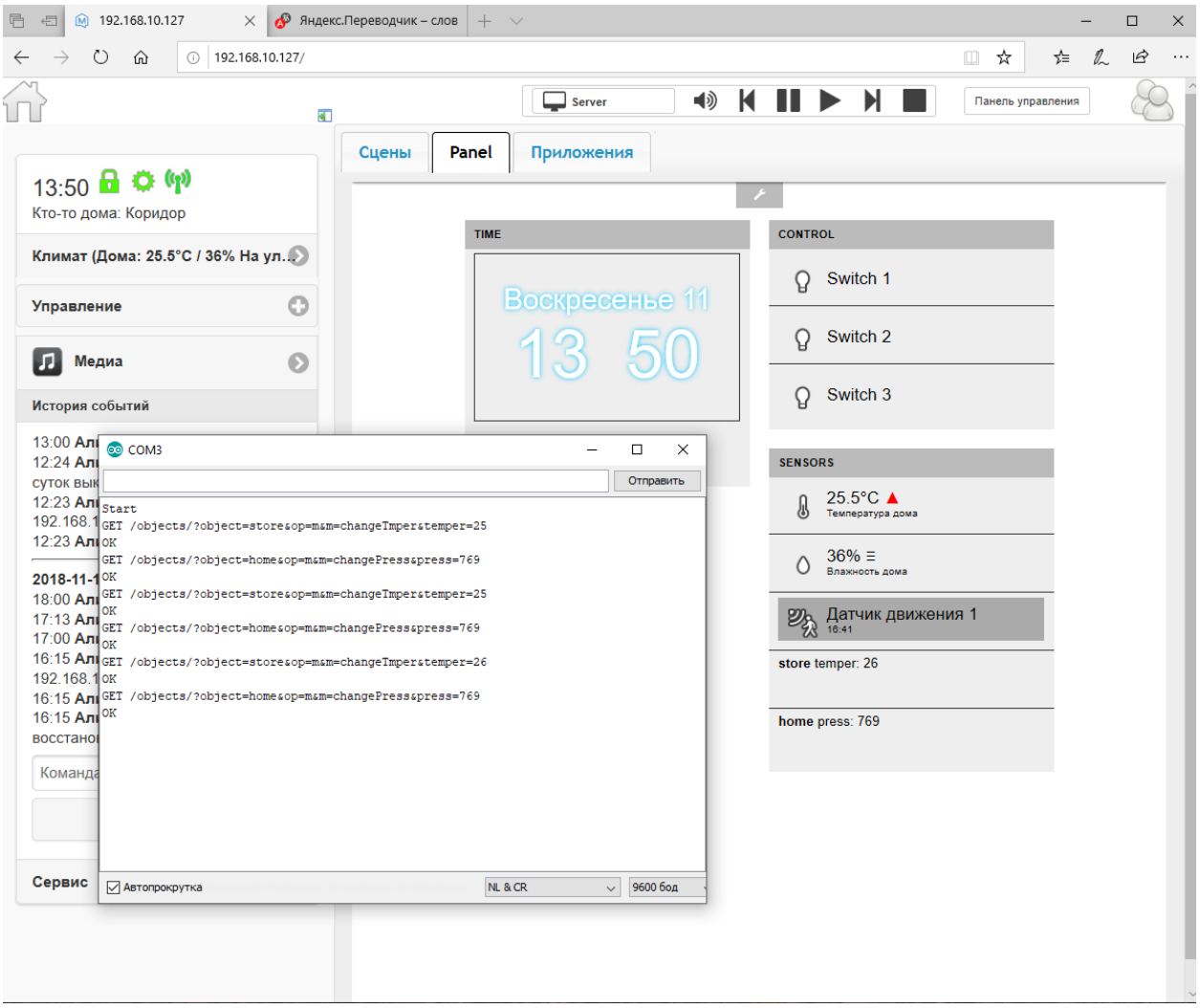


Рис. 6.33. Вывод температуры и давления от датчика BMP180

Пусть вас не удивляет давление 769 мм.рт.ст., в этот день такое давление и было.

Глава 7. О сценарии в MajorDoMo и не только о нем

Несколько слов о сценарии в системе MajorDoMo

Любой сценарии подразумевает описание действий объектов, как и в случае с MajorDoMo. Я плохо знаю язык PHP, что вы уже заметили, поэтому просто приведу образец сценария – срабатывание датчика движения на видеокамере:

```
   DebMes("Motion detected: ".serialize($params));
   callMethod('MotionSensorCam.motionDetected');
   setTimeOut('motionDetectedTimer','runScript("camImagesProcess"
);',10);
   if (getGlobal('ThisComputer'.'.'.'WebCamMotionAuto')) {
     setTimeOut('stopWebCamDetection', "
runScript('manageWebCamMotion', array('stop'=>'1'));",
(int)('60'));
   }
```

Таким образом, сценарий в MajorDoMo – это подпрограмма на языке PHP, которая выполняется при появлении некоторого события. Система MajorDoMo для пользователей при создании сценариев предлагает ряд встроенных функций, упрощающих, подобно любым библиотечным функциям, написание кода.

Разработчики системы не остановились на достигнутом, добавив возможность «визуального программирования» с помощью Google Blockly – проекта с открытым кодом. Посмотрим, как это выглядит на примере создания своего сценария, хотя вы можете увидеть Blockly на многих страницах, где вводите PHP код.

Итак: *Панель управления→Сценарии*, кнопка **Добавить новый сценарий**, назовем его *TestVisualProgr*. После сохранения сценария кнопкой **Сохранить** мы попадаем на страницу создания кода, где предусмотрительно выбран способ *Blockly*. (**рис. 7.1**). Если нажать на ссылку *[Редактировать]*, то мы попадем в мир визуального программирования (**рис. 7.2**).

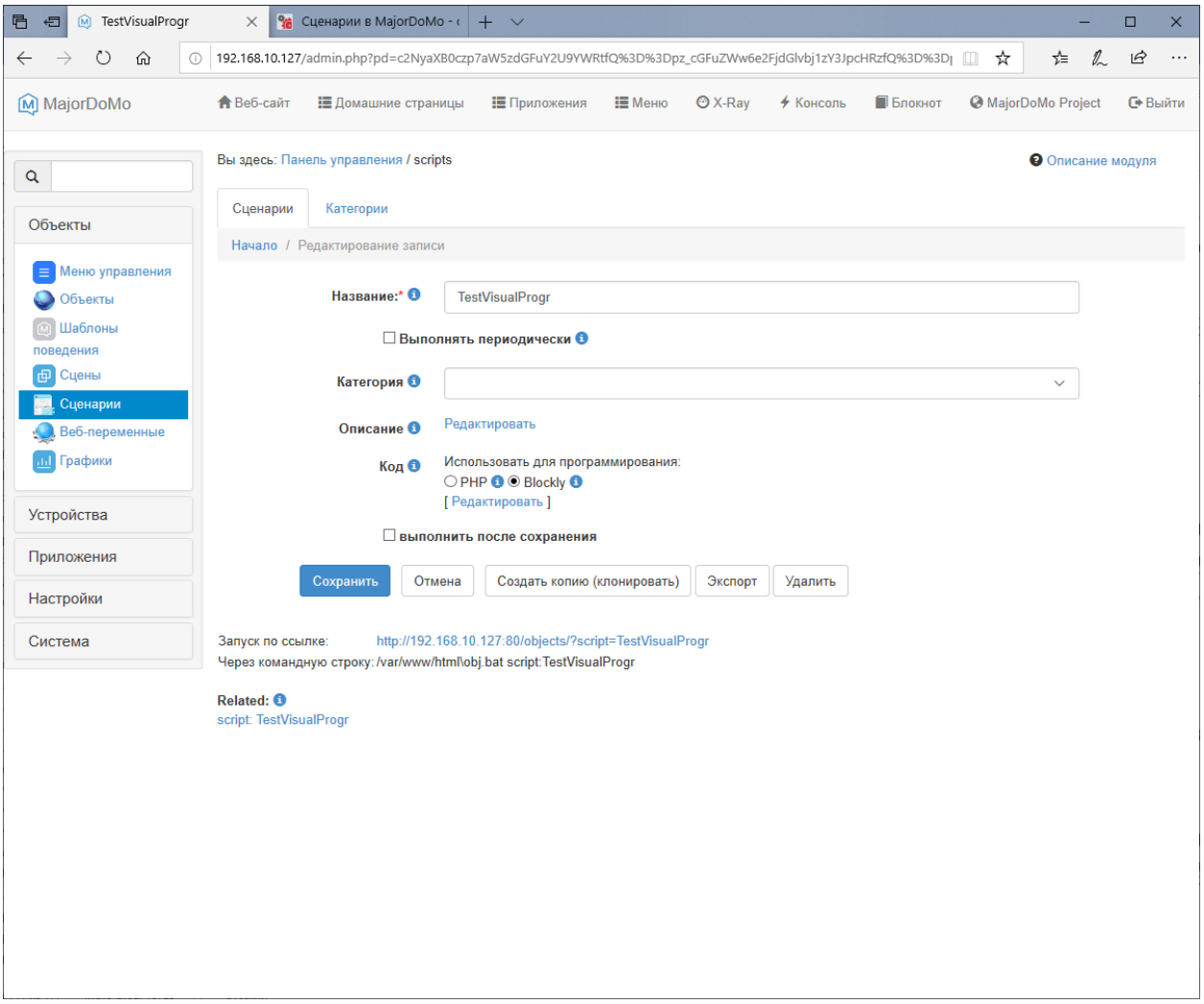


Рис. 7.1. Страница создания сценария

Рис. 7.2. Страница заготовок визуального программирования

Для тех, кто хотел бы поэкспериментировать с автоматизацией своего дома, но плохо, как и я, знает язык PHP, такое программирование существенно облегчит опыты. Хотя, я неоднократно это говорил, если вы хотите превратить свой дом в умный, лучше обратиться к специалистам.

Элементы визуального программирования вы просто «цепляете» мышкой и переносите в рабочее поле.

Вот, что сказано в базе знаний MajorDoMo о категориях, используемых для визуального программирования:

- Общее -- некие общие функции системы MajorDoMo (произнести, проиграть звуковой файл и т.п.)

- Объекты, блоки работы с объектами, их методами и свойствами.

- Время, блоки работы с текущим временем и таймерами.

- Логика, блоки условий и логических операций.

- Циклы, блоки повторов и циклов.

- Математика, математические операции.

- Текст, блоки работы с текстовыми строками.

- Списки, работа со списками (массивами данных).

- Цвет, работа с цветом.

- Переменные, возможность создания и повторного использования переменных.

- Функции, блоки создания собственных функций и их применение в программе.

- Простые устройства, блоки операций со всеми добавленными в систему устройствами и их данными.

- Сценарии, созданные пользователем сценарии (каждый из сценариев автоматически становится блоком, который можно использовать в другом сценарии).

Мне жаль, что я не догадался освоить и использовать эту возможность раньше.

Использование USB-камеры

К миникомпьютеру Raspberry Pi можно подключить USB-видеокамеру.

Поскольку сам я этим раньше не занимался, сошлюсь на статью из Интернета **[14]**. Я хочу повторить все, что предлагает автор. На Raspberry Pi можно зайти либо по SSH через программу putty.exe, либо используя ОС, если вы установили дополнения, о которых написано в главе 3.

В терминале (или putty) выполняем установку программы motion, командой:

```
sudo apt-get install motion
```

Далее, следуя указаниям автора, настроим конфигурацию программы:

```
sudo nano /etc/motion/motion.conf
```

Открывается редактор *nano* с указанным файлом конфигурации (**рис. 7.4**).

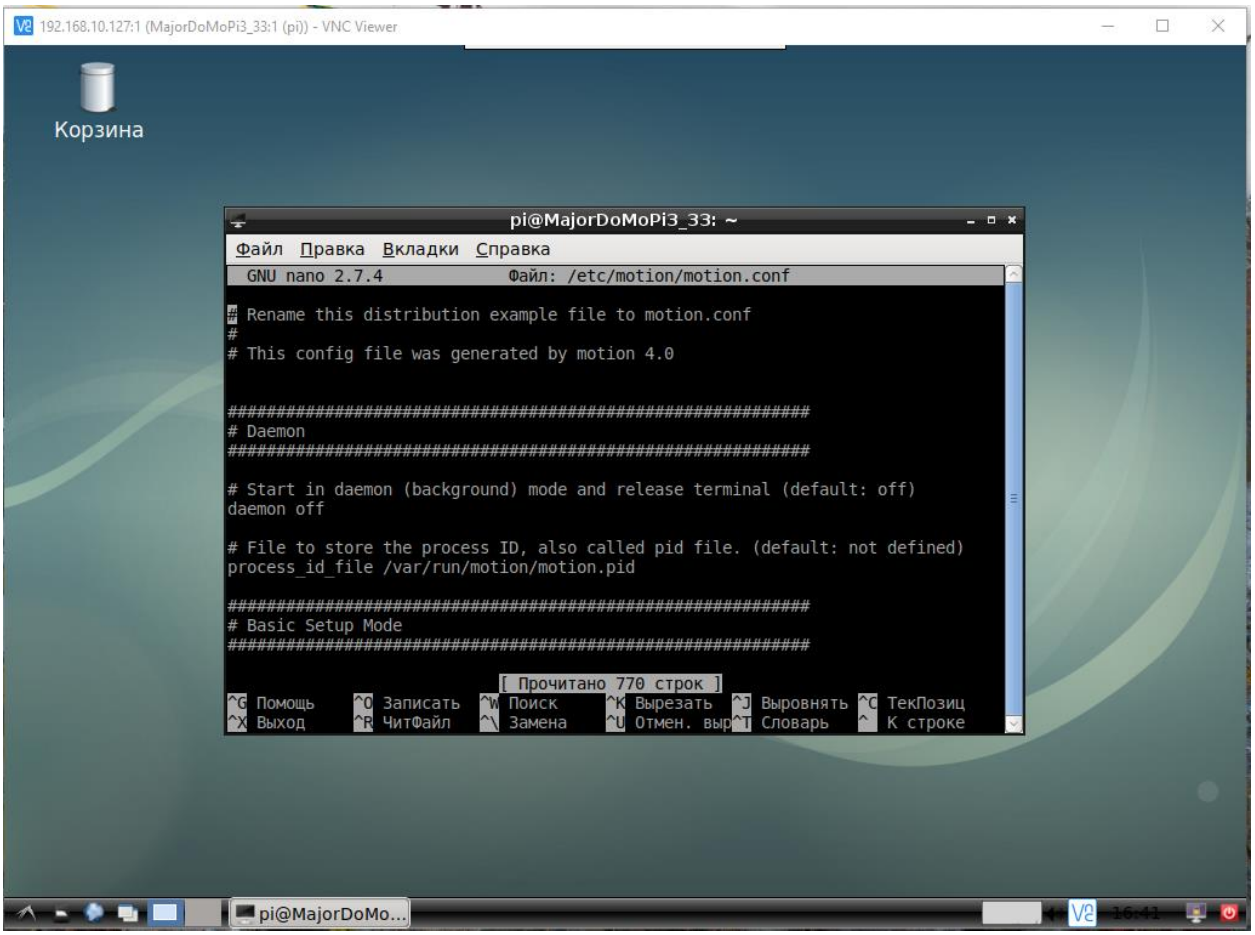


Рис. 7.4. Правка файла в редакторе nano

В файле следует исправить две строки:

`daemon off` (меняем на on)
`webcontrol_localhost on` (меняем на off)

Закрывается редактор с сохранением изменений клавишами ctrl+o, Enter, crtl+x. Следующая правка:

```
sudo nano /etc/default/motion
```

Здесь в файле одна строка, где:

`start_motion_daemon=no` (меняем на yes)

Мне потребовалось еще заменить порт 8080 на 8081 в файле конфигурации. Теперь, запустив программу: `sudo service motion start`, я могу зайти на Raspberry Pi из браузера основного компьютера (**рис. 7.5**).

Рис. 7.5. Браузер связывается с USB-камерой на Raspberry Pi

Дальше автор советует:

Остановить трансляцию:
sudo service motion stop
Для того, чтобы поток транслировался автоматически после выключения или перезагрузки Raspberry Pi набираем:

```
sudo nano /etc/rc.local
```

и в конце над строкой exit 0 добавляем motion. Примерно вид файла rc.local такой:

```
#!/bin/sh -e
# rc.local
# Этот сценарий выполняется в конце каждого
многопользовательского уровня выполнения.
# Убедитесь, что скрипт будет "exit 0" при успехе
# или будет другим значением при ошибке.
# В плане разрешения или запрета этот скрипт только
# биты выполнения.
# По умолчанию этот скрипт ничего не делает.
# Вывести IP-адрес.
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
printf "My IP address is %s\n" "$_IP"
fi
# Start motion
motion
exit 0
```

Теперь наш поток доступен по локальной сети по адресу http://IP_адрес_вашей_raspberri_pi:8081.
:8081 – это порт для трансляции видео.
Если необходимо транслировать видео через интернет, предварительно надо открыть данный порт для внутреннего IP адреса вашей Raspbery Pi на роутере. И обращаться к видео потоку из интернета через http://IP_адрес_вашего_роутера:8081.

К сожалению, я так и не увидел долгожданной картинки. Все, что удалось получить, используя web-браузер Raspberry Pi, никак нельзя назвать «правильной» картинкой (**рис. 7.6**).

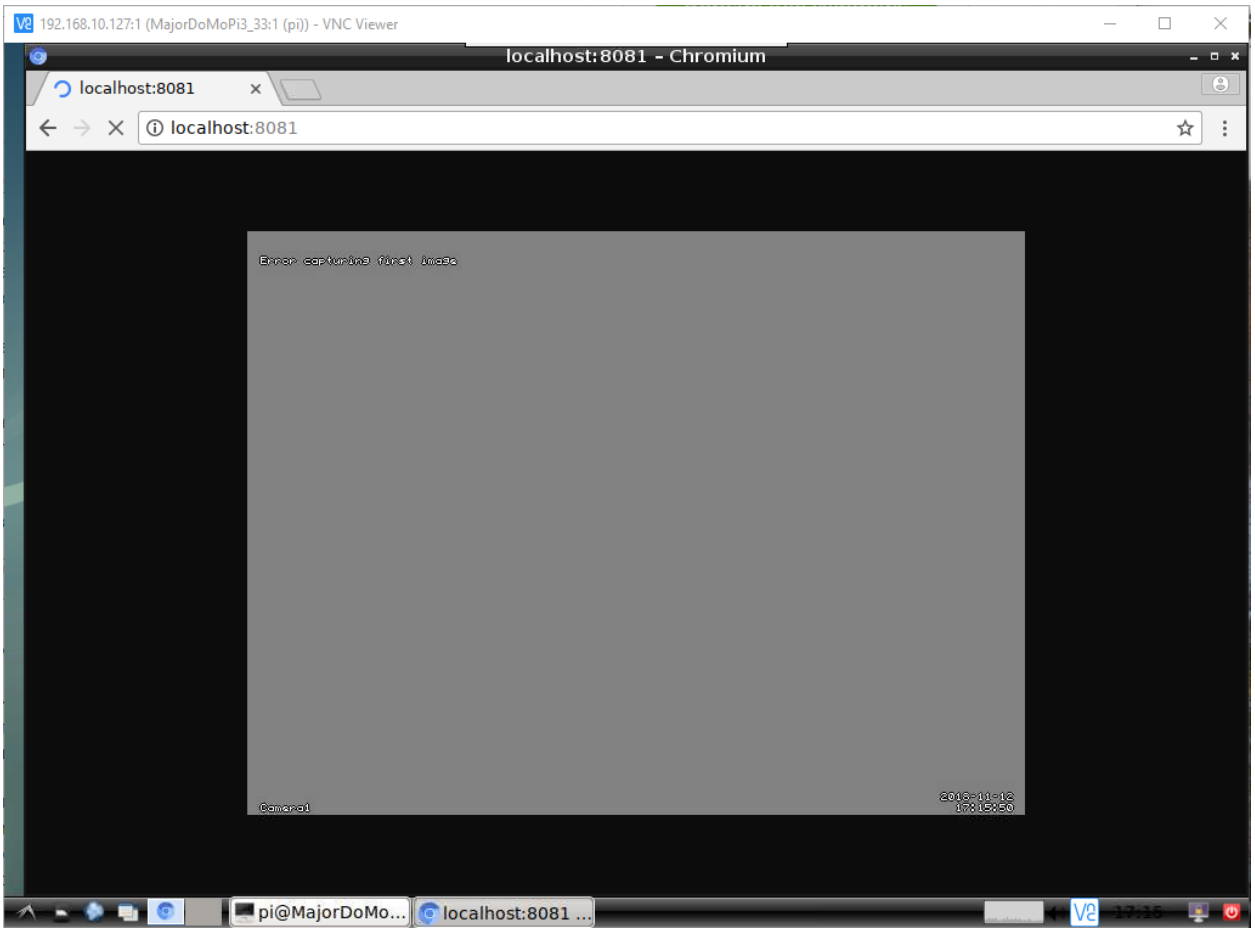


Рис. 7.6. Пустой экран приложения motion

Надпись меняется время от времени, а причина отсутствия изображения кроется в сообщении, которое трудно разобрать на рисунке, но это: *Error capturing first image*.

Не знаю, нужно ли еще что-то, но, чтобы не упрекать себя в лености, я перебрал ряд приложений. Я не добился большего, чем можно увидеть на **рис. 7.6**. Не берусь утверждать, но, видимо, не подходит формат моей USB-камеры, оказавшейся под рукой, ни одному приложению, которые я пробовал. Поэтому, если вы захотите экспериментировать с USB-камерой, постарайтесь выяснить, подходит ли она для работы с приложениями на Raspberry Pi. Или, это, наверное, даже лучше, купите видеокамеру, которая подключается к Raspberry Pi в предназначенное для нее гнездо (**рис. 7.7**).

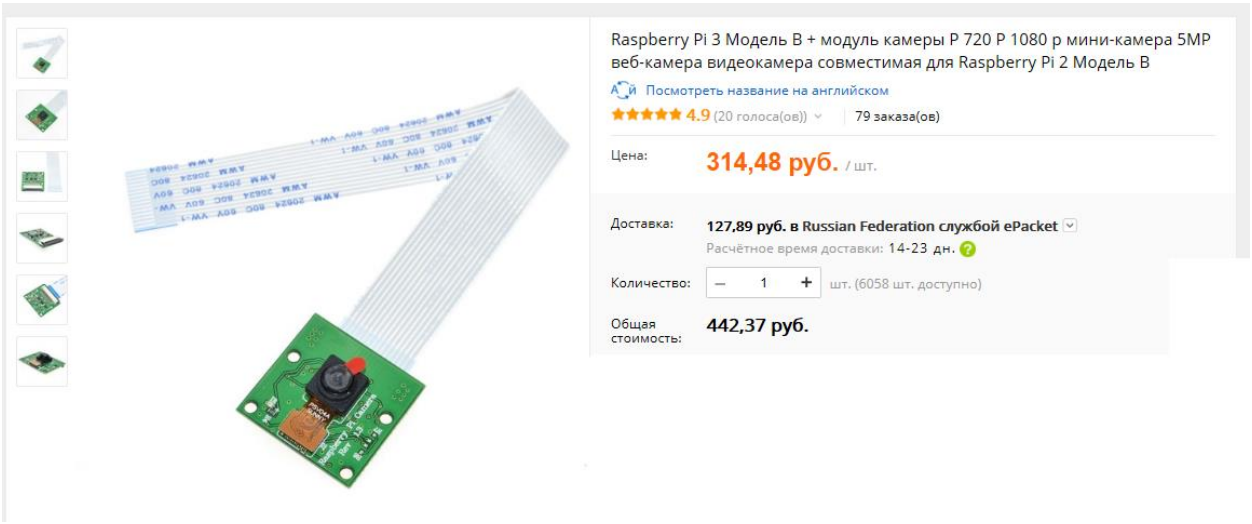


Рис. 7.7. Видеокамера для Raspberry Pi

Впрочем, я сам эту камеру не покупал, не пробовал включать, не мне и советовать.

В Интернете есть список USB-видеокамер, которые могут или нет работать с Raspberry Pi. Я приведу таблицу, как она есть.

Таблица 7.1.

Этот список не совсем надежен, работа не обязательно означает работу без ошибок. Пожалуйста, поделиться своими впечатлениями!

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
Canyon		CNR-FWC113	0c45:6340	raspbian/wheezy	2013-04-11	640x480	Хорошо работает «из коробки». Нет некоторых удобств, работать лучше в помещении.
Canyon		CNR-FWC120H		raspbian/wheezy	2013-07-26	640x480	Работает сразу, проверено с guvcview и mjpg-streamer. Для работы вне дома потребует доработки.
Canyon		CNR-WCAM820		raspbian/wheezy		1280x1024	Работает с fswebcam и v4l4j на Raspbian Wheezy armhf; с разрешением 1280x1024 и ниже работает хорошо.
CBR		CW 835M Black		raspbian/wheezy	2013-02-12		Хорошо работает без концентратора.
Creative	Live!	VF0470		ArchLinux			Работает сразу на ArchLinux
Creative	Live! Cam Socialize HD	VF0610	041e:4080	raspbian/wheezy	2012-11-26	960x544	С разрешением 1280x720 еще работает в fswebcam с некоторыми ошибками, используя MJPEG, с YUYV не работает. При 960x544 работает и с MJPEG, и с YUYV.

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
Creative	Live! Cam Sync HD		041e: 4095	raspbian/ wheezy	2013-04-11	1280x720	Работает сразу. Автоматическая выдержка хорошо работает и внутри, и вне помещения. С fswebcam нужно несколько снизить уровень "sharpness"
Creative	Go	PD00040		raspbian/ wheezy	2013-04-11		**Не работает совсем. Испробовано с fswebcam/motion. Запорчены данные JPEG: преждевременное завершение сегмента данных.**
Creative	Live! Cam Vista IM	VF0640		raspbian/ wheezy			Работает на Raspbian при 320x240, 15fps
Creative	Live! Cam Socialize	VF0640		raspbian/ wheezy			Работает на Raspbian при 320x240, 15fps
Creative	Webcam Noteboo k	PD1170					Обнаруживается, не проверена.
Creative	Webcam Pro	PD1030					**Драйвер ov519 «вылетает» почти сразу: ("gspca: ISOC data error: [0] len=0, status=-4004")**
Eminent		EM1089		raspbian/ wheezy	2014-05-06	640x480	Работает хорошо без концентратора.
GE	MiniCam Pro	98756	0ac8: 3420	raspbian/ wheezy	2014-01-07	640x480, 352x288, 320x240, 176x144, 160x120	Проверена и работает с motion. Работает с моделью B без концентратора (потребляет 100 mA).
Hercules	Webcam Deluxe		05a9: 4519	raspbian/ wheezy + Arch	2013-02-09		**Драйвер ov519 "Corrupt JPEG data: premature end of data segment", дает испорченное изображение с motion и fswebcam.**
HP	Webcam -2100	2100		Raspbian	3.18.7	640x480	Нужно пропускать кадры при использовании fswebcam пока не рассчитаны настройки экспозиции.
HP	Webcam HD-2200	HD-2200		raspbian/ Jessie	4.4.50 March 2017	1280x720	Работает без концентратора.
HP	Webcam HD-2300	HD-2300		raspbian/ wheezy	2013-08-28	1280x720	Работает хорошо без концентратора.
HP	Webcam HP-3100	HP-3100					UVCVideo /dev/video0 Нужно chmod до 666 для работы. Без концентратора будет

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
							работать только с одним устройством, подключенным к USB-порту. Работает сразу и с Arch, и с Wheezy.
Kodak	Webcam S101	Kodak S101	0979:0206	raspbian/wheezy	2014-02-09	640x480, 320x240	/dev/video0 Работает сразу с guvcview. Проверено и работает с motion. Проверено на модели B без концентратора.
Logitech	Webcam C100	V-U0013		raspbian/wheezy	2012-08-16		Работает хорошо без концентратора.
Logitech	Webcam C110		046d:0829	raspbian/wheezy	2014-04-04		Работает хорошо без концентратора. Однако пробуйте fswebcam -p YUYV test.jpeg
Logitech	Webcam C160	V-U0011		Raspbian	2015-03-04	640x480, 320x240	Работает хорошо без концентратора.
Logitech	Webcam C170			raspbian/wheezy		1024x760	Работает хорошо без концентратора, качество image/video плохое.
Logitech	Webcam C200		046d:0802				Работает хорошо без концентратора.
Logitech	Webcam C210		046d:0819	Raspbian/wheezy	2012-12-16	320x240, 640x480	Работает хорошо без концентратора.
Logitech	Webcam C270		046d:0825	Raspbian/wheezy		1280x720	Работает хорошо с внешним питанием, image/video четкое. Осторожно: Pi может зависнуть (по крайней мере, потерять удаленный доступ) при использовании UVC video kernel module. Исправление: загрузите модуль, используя следующее «магическое заклинание»: `modprobe uvcvideo nodrop=1 timeout=5000 quirks=0x80` . (2014-06-01, Raspbian Linux 3.12.20+). Без внешнего питания (камера подключена напрямую к RPi) RPi работает периодически. Работает, скажем, раз 10, затем отказывается работать. Но потом

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
							начинает работать вновь. Попытка использовать «магическое заклинание» не помогла.
Logitech	Webcam C300	V-U0004	046d:0805	Raspbian /Wheezy	2013-02-09	320x240, 640x480, 1280x1024	Работает сразу. Не было указаний на концентратор.
Logitech	Webcam C310						Не нужно концентратора для snapshots.
Logitech	Webcam C510		046d:081d	Raspbian /Wheezy	2013-08-30	320x240, 640x480	Захват изображения работает без концентратора. Другие разрешения не испытывались.
Logitech	Webcam C525		046d:0826			1920x1080	Работает хорошо без концентратора.
Logitech	Webcam C615	V-U0027					Работает хорошо без концентратора.
Logitech	Webcam C905		046d:080a	Raspbian /Wheezy + occidentalis	v0.2	1600x1200	Работает хорошо без концентратора, определяется как устройство Video0 V4L device (uvcvideo module). 1600x1200 на маленькой скорости, но работает (проверено с motion, uv4l_uvc).
Logitech	Webcam C910						С внешним питанием, uncvideo. 320x240 работает напрямую с Raspberry Pi.
Logitech	Webcam C920			raspbian/ wheezy		1920x1080	С концентратором определяестя стразу как Video0 V4L устройство. Работает сразу на модели B+ без концентратора.
Logitech	Webcam C922		046d:085c	raspbian/ jessie		1920x1080	Работает сразу на модели RPi 3 без концентратора. Работает также как /dev/video0 (V4L) сразу, проверено с VLC.
Logitech	QuickCam E2500		046d:089d	Raspbian /Wheezy	3.12	320x240	Захват изображения и motion работают без концентратора. Другие разрешения не испытывались.
Logitech	QuickCam Orbit/Sphere						Работает с внешним питанием.

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
Logitech	QuickCam Express	861037-0000 V-UB2	046d:0840	raspbian/ wheezy	2014-02-09	320x240	/dev/video0 Работает сразу с моделью B+ без концентратора. Проверено с fswebcam.
Logitech	QuickCam Messenger	V-UM14	046d:08f0	raspbian/ wheezy + Arch	2013-02-09		**Не работает, STV06xx driver "ioctl (VIDIOCGCAP): Inappropriate ioctl for device", Supported palettes: GRBG, дает испорченное изображение в fswebcam.**
Logitech	QuickCam Communicate STX			raspbian/ wheezy	2013-09-25		**Не работает. Есть видео изображение, но испорченное. Проверялось и с концентратором, и без него.**
Logitech	QuickCam Communicate STX			Raspbian	2015-11-17		Повреждение нашло решение для скудных 320x240 4fps после добавления **options usbcore autosuspend=-1** в /etc/modprobe.d/disable-usb-autosuspend.conf с последующей перезагрузкой. Более высокое разрешение и частота кадров все еще остаются испорчены.
Logitech	QuickCam Pro 5000						Работает прекрасно (не испытано без концентратора).
Logitech	QuickCam Pro 9000	V-UBM46	046d:0990	3.10.25-1-ARCH			Прекрасно работает без концентратора.
Logitech	QuickCam Pro 9000			raspbian/ wheezy			Питание от RasPi.
Logitech	QuickCam Pro for Notebooks	960-000047	046d:0991	Raspbian Wheezy	2012-12-16	160x120 320x240 640x480	С guvcview показывает около 4fps при 160x120, и около 1fps при 640x480. GUVCViewer Controls доступны для фокусироваки и частоты кадров.
Logitech	QuickCam Ultra Vision			raspbian/ wheezy			Питание от RasPi.
Logitech	Webcam Pro 4000						**Использует pwc драйвер, который не работает. Возможно,**

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
							из-за главной ошибки с Raspberry Pi USB.
Logitech	Webcam Pro 9000		046d: 0809	Arch Linux			Питается от RasPi, потребляет ~120 mA, зафиксировано при ~5fps. Имеет проблемы захвата изображений с разрешением выше, чем по умолчанию (использовано с motion - Arch и Debian).
Mannhattan	MINI CAM			raspbian/ stretch	2018-01-03		**Не работает совсем. Испытано с разными приложениями (fswebcam) – не работает. Проблема с не совместимым форматом (Palette).**
Medion		MD86511		raspbian/ wheezy	2012-07-15		Питание от RasPi.
Mexxcom		M-104		raspbian/ wheezy	2012-12-16		Питание от USB концентратора.
Microsoft	LifeCam Cinemap 720p USB HD Webcam	H5D-00001		raspbian/ wheezy			
Microsoft	LiveCam HD-3000	HD-3000	045e: 0779	Archlinux	2013-02-06	160x120	Работает сразу с проверенным разрешением. Может питаться непосредственно от Raspberry Pi и работает хорошо с wifi подключенным к другому USB порту.
Microsoft	LiveCam HD-3000	HD-3000	045e: 0779	raspbian/ wheezy	2013-04-11	1280x720	Отбросьте первые пару кадров после активации, в противном случае изображение распадается и экспозиция плохая. Пример команды: fswebcam -S 5 -r 1280x720 tmp.jpg
Microsoft	LiveCam HD-3000	HD-3000	045e: 0810	raspbian/ stretch	2018-01-03	1280x720	Прекрасно работает с RP3b непосредстенным питанием (JPG и видео). Не было проблем с первыми кадрами для картинок, но нужно

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
							некоторое время для посдстройки видео.
Microsoft	LifeCam HD-5000	HD-5000		raspbian/ wheezy	21-02-2014		Картинка распадается внизу. Я недавно приобрел Pi и была LifeCam HD-5000. Картинка НЕ распадается внизу, но YMMV. 21-2-14.
Microsoft	LifeCam HD-6000	HD-6000		raspbian/ wheezy		1280x720	Питание от USB концентратора непосредственно от Pi.
Microsoft	LifeCam	NX-3000	045e: 0721	raspbian/ wheezy	2014-01-07	640*480	Питается от Raspberry.
Microsoft	LifeCam	NX-6000		raspbian/ wheezy		1280x720	Питается от USB концентратора.
Microsoft	LifeCam	VX-7000		raspbian/ wheezy			Питается от USB концентратора.
Microsoft	LifeCam	VX-3000		raspbian/ wheezy			Похоже, есть некоторые проблемы с качеством изображения и получением частичных кадров с fswebcam
Microsoft	LifeCam	VX-500		raspbian/ wheezy			640x480. Проверено с непосредственым подключением к Raspberry Pi.
Microsoft	LifeCam	VX-1000		raspbian/ wheezy			**Не похоже, чтобы это сработало. Испытавлось с концентратором.**
Microsoft	LifeCam	VX-1000		Octopi (Raspbian tweaked)			Это работает после нескольких тестов. fswebcam на raspberry pi 2, и теперь получаются 320x240 pixel фото .
Microsoft	LifeCam	VX-2000	045e: 0761	raspbian/ wheezy	2013-12-20	320x240, 640x480	Работает без концентратора. Испытано с motion и fswebcam, обе работают гладко.
Microsoft	LifeCam	NX-6000		raspbian/ wheezy			Питается от USB концентратора.
Microsoft	LifeCam	VX-800		raspbian/ wheezy		352x288	Не работает с полным разрешением.
Microsoft	LifeCam Studio/Cinema						Имеет UVC проблемы. Проблемы с горизонтальными линиями. Проблемы со стабильностью.
Microsoft	LifeCam Studio	1080p HD	045e: 0772	Raspbian Wheezy	2013-04-12	1280x720	Работает без проблем с концентратором (Belkin).

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
Microsoft	Xbox Live Vision	X806235	045e: 0294	Arch Linux/Raspbian Wheezy		960x720	Питается от RasPi.
MSI	MSI StarCam 370i	370i (snake)		Raspbian Wheezy		352 x 288	Работает с питанием от RasPi или USB концентратора с Motion при 352 x 288 – работает хорошо. Имеет ИК-светодиоды, но включаемые программно на Windows, так что, не работают на Pi, но нет проблем с захватом.
Novatek	Webcam		0603: 8124	Arch Linux	2014-09-19	640x480	Картинка распадается внизу на Arch Linux, питание либо от USB концентратора (0424:9512), либо от Raspberry (не рекомендуется, максимальный ток 500mA).
Philips	Webcam	SPC 900NC	0471: 0329	Raspbian Stretch	2018-10-16, Kernel 4.14.74	640x480	(old: 2012-12-16 Raspbian Wheezy) Распознается как USB устройство ID 0471:0329 Philips (или NXP) SPC 900NC PC Camera / ORITE CCD Webcam(PC370R). Работает с guvcview, но не с luvcview. Так же работает с командой: ~$ fswebcam -r 160x120 -d v4l2:/dev/video0 test.jpg (new: 2018-10-16). Подключается через USB концентратор. Снимки экрана только с: fswebcam -r 640x480 image.jpg
Philips	toucam	Philips 720K/40 webcam	0471: 0313	Raspbian Wheezy	2013-04-03	320x240	Распознается как lsusb ID 0471:0329 Philips (or NXP). Работает с $> guvcview -s 320x240 -f yv12 -t 5 -n rec5sec.mkv --exit_on_close --no_display, но не скомандой $> fswebcam -r 160x120 -d v4l2:/dev/video0 test.jpg

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
Realtek	Generic Camera	2SF022	0bda: 5801	Raspbian Wheezy			Когда запускается с luvcview на 15fps и 320x240, похоже, это дает частоту кадров около 1 в сеунду.
Silicon Motion	SM731 Camera		090c: 71b3	Raspbian Wheezy			Требует UVCVideo драйвера – работает сразу. Проверено для 320x240 с использованием motion & camorama для картинок, потокового вещания.
Sony	Playstation Eye for PS3	SLEH-00448		jessie-raspbian	2016-12-21	640x480 320x240	Работает сразу при прямом подключении к raspberry pi. Никаких проблем. Проверено с motion. Dec'16 проверено и с fswebcam.
Sony	EyeToy for PS2						**Случайный искаженный кадр при прямом подключении к Rev 2 Raspberry Pi**
Sony	EyeToy for PS2	SLEH 00030		Arch Linux			**(OV519 camera). Картинка постоянно рассыпается на xawtv и wxcam под Arch Linux. Замечено, есть ISOC ошибка данных len=0 status=-4004 в dmesg. Так получалось при питании от Raspberry Pi и от Pluscom USB концентратора. Arch обновлено 17 июля 2012.**
T'nB	Minipix 100K pixels	IMWB032 992	1e4e: 0100	raspbian/ wheezy	2012-12-16		**RasPi виснет (нужна перезагрузка) после нескольких минут использования Motion с потоковым видео (проверено с внешним питанием).**
Trust	2 MP Auto Focus Webcam			Arch Linux		1600x1200	Прекрасно работает без концентратора. 160x120 - 1600x1200. 5 дней тестирования без проблем.
Trust	SPACEC @M 200			Arch Linux			**(OV511 camera). Картинка замирает после нескольких секунд в xawtv под**

Фирма	Имя	Номер модели	ID	ОС	Версия	Разр.	Примечания
							Arch Linux, и xawtv сообщает об ошибке в libv4l2. Происходит при питании от Raspberry Pi и питании от Pluscom USB концентратора. Arch обновлено и 17 июля 2012.
Trust	SPYC@M 100		0553:0202	Raspbian /Wheezy	2013-08-22	352x288	Работает сразу. Проверено с концентратором, не проверялось при прямом подключении к Raspberry Pi. Могло иметь LD_PRELOAD=/usr/lib/arm-linux-gnueabihf/libv4l/v4l2convert.so для motion.
Trust	Spotlight		0c45:62c0	Raspbian /Wheezy		640x480	Работает сразу. Проверено с питанием прямо от Raspberry Pi, не проверено с концентратором.
Trust	WB-1400T			Raspbian /Wheezy			**Дешевая камера, распознается в 'lsusb', но не поддерживается.**
Trust	WB-1200p Mini Webcam		093a:2468	Raspbian /Wheezy	2013-12-12		**Распознается в 'lsusb' как Pixart Imaging, Inc. SoC PC-Camera. Не работает. gspca_main сообщает о постоянной ошибке: "ISOC data error".**
Vega	USB 2.0 Camera		0ac8:c302	Raspbian Wheezy	2014-01-01	640x480	Должна подклюдчаться к USB концентратору. Дешевая камера из Китая. Использует UVC драйвер.

ИК-коды управления телевизором

Современные телевизоры, возможно, позволяют управлять ими по WiFi, но далеко не все. Менять телевизор только для того, чтобы привязать его к Raspberry Pi – это не самое умное решение. Тем более, что любым телевизором можно управлять с помощью ИК-кодов. В прошлом рассказе про Raspberry Pi **[1]** я довольно подробно рассказывал о программе lirc.

Я не буду повторять тот рассказ, но использую эту программу для создания всего необходимого для воспроизведения кода включения телевизора, чтобы написать сценарий:

вы припарковали автомобиль в гараже, прошли в дом (герконовый датчик гаража сработал), и MajorDoMo, спустя некоторое время, включает телевизор в столовой.

Начнем с установки программы lirc. В операционной системе Raspbian была программа, которая занимается учетом всех установленных программ и позволяет найти и установить новые, которые вам потребовались в данный момент. Сейчас такой программы нет, хотя ее можно установить, поэтому используем возможности терминала для поиска и установки всех необходимых компонентов lirc (**рис. 7.8**). Используем команду:

```
sudo apt list lirc*
```

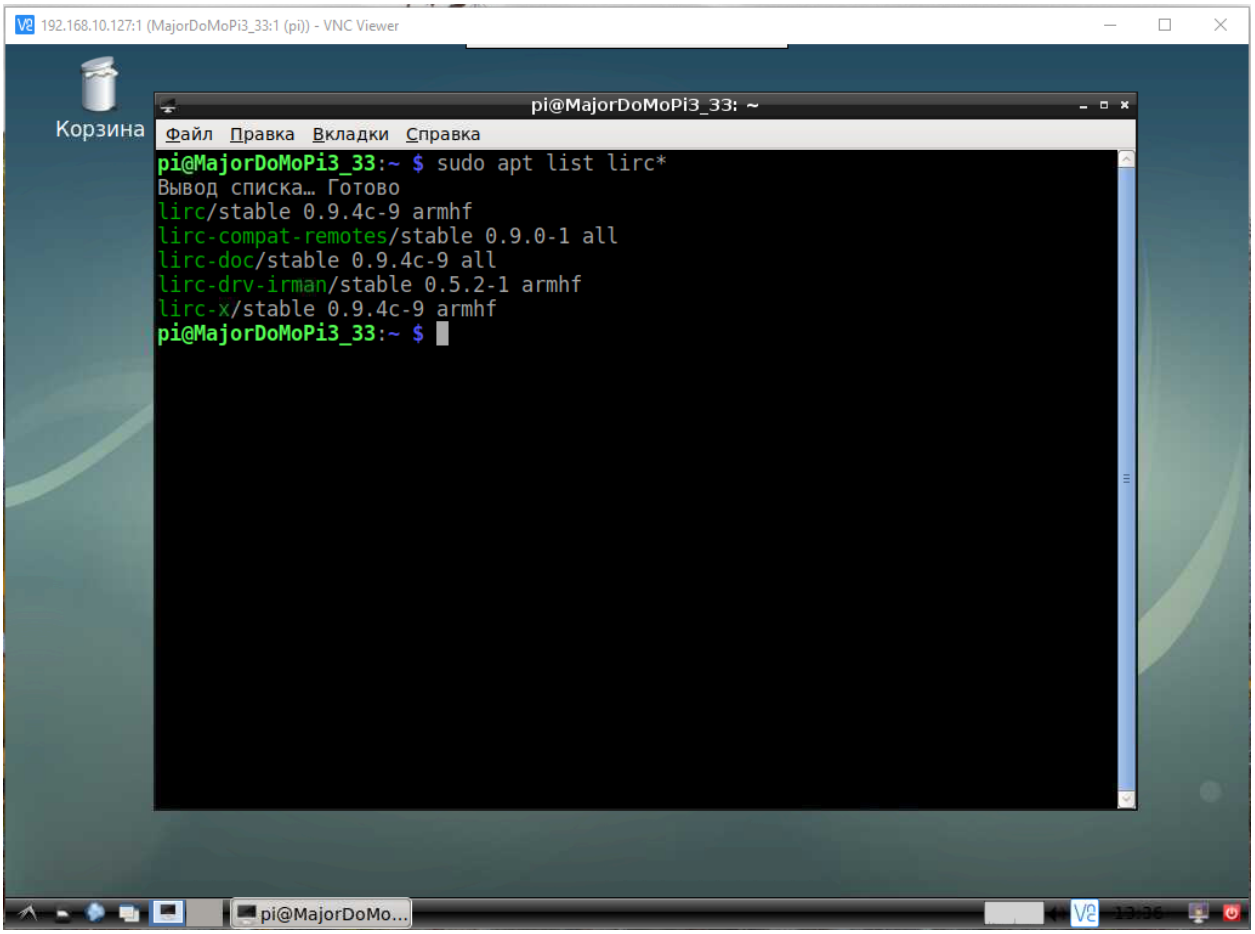


Рис. 7.8. Поиск программы lirc в терминале

Что из этого списка устанавливать обязательно, а что нет? Я пока не уверен в том, что является необходимыми составляющими, поэтому установлю все, удалить лишнее можно всегда.

Установка самой программы производится с помощью «волшебной» команды:

```
sudo apt-get install lirc
```

Работа в терминале кому-то нравится, кого-то пугает, а мне напоминает те времена, когда операционную систему приходилось загружать с большой дискеты путем ввода восьмеричного кода. Я немного ленив для повторения тех подвигов, вдобавок подслеповат; вводя длинную строку, могу забыть переключить клавиатуру, в конце ввода обнаружить несусветную строку команды, чтобы, ворча, удалить ее и ввести заново. По этой причине я,

например, выписываю, что несложно сделать копированием, все программы непосредственно из окна терминала на лист этого рассказа:

```
lirc/stable 0.9.4c-9 armhf
lirc-compat-remotes/stable 0.9.0-1 all
lirc-doc/stable 0.9.4c-9 all
lirc-drv-irman/stable 0.5.2-1 armhf
lirc-x/stable 0.9.4c-9 armhf
```

Затем ввожу «волшебную» команду, копируя ее из текста, и запускаю установку, меняя только названия программ, которые тоже копирую с листа.

Вероятно, можно при установке указать весь список, но для этого нужно быть уверенным в том, что никаких неожиданностей не будет.

Начнем использование lirc с команды: `sudo /etc/init.d/lircd stop`. Она должна остановить службу lirc, отвечая: *[ok] Stopping lircd (via systemctl): lircd.service*.

Стоп, не начнем, поскольку я забыл, что еще не настроил lirc и не оборудовал Raspberry Pi фотоприемникомTSOP. Сделаем «стоп» для себя, а не для службы lirc.

Итак. Вот описание процедуры, найденное в Интернете, которое я приведу из предыдущего рассказа о lirc, когда отыскивал файлы, где нужно было что-то изменить:

Хорошая новость в том, что, если вы установили Lirc впервые, текущая версия Raspberry Pi/Raspbian Stretch использует версию 0.9.4c, которая имеет ряд улучшений, облегчающих вам жизнь. Файлы, использовавшиеся в ранних версиях, теперь не используются, это /etc/modules и /etc/lirc/hardware.conf.

Следуя описанию на этом ресурсе, я повторяю предложенные там команды (часть из них уже выполнена, предваряя их `sudo`:

```
apt-get update \
&& apt-get upgrade -y \
&& apt-get install -y lirc \
&& rm -rf /var/lib/apt/lists/*
```

Далее следует добавить в файл /boot/config.txt следующую строку там, где есть соответствующее пояснение:

```
# Uncomment this to enable the lirc-rpi module
dtoverlay = lirc-rpi, gpio_out_pin=17,
gpio_in_pin=18,gpio_in_pull = up
```

Примечание.

Команда для работы с подобными файлами выглядит следующим образом: `sudo nano /boot/config.txt`. *В окне редактора nano после исправлений и добавлений следует выполнить последовательность команд (горячие клавиши клавиатуры):* ***ctrl+o***, ***Enter***, ***Ctrl+x (рис. 7.9)***.

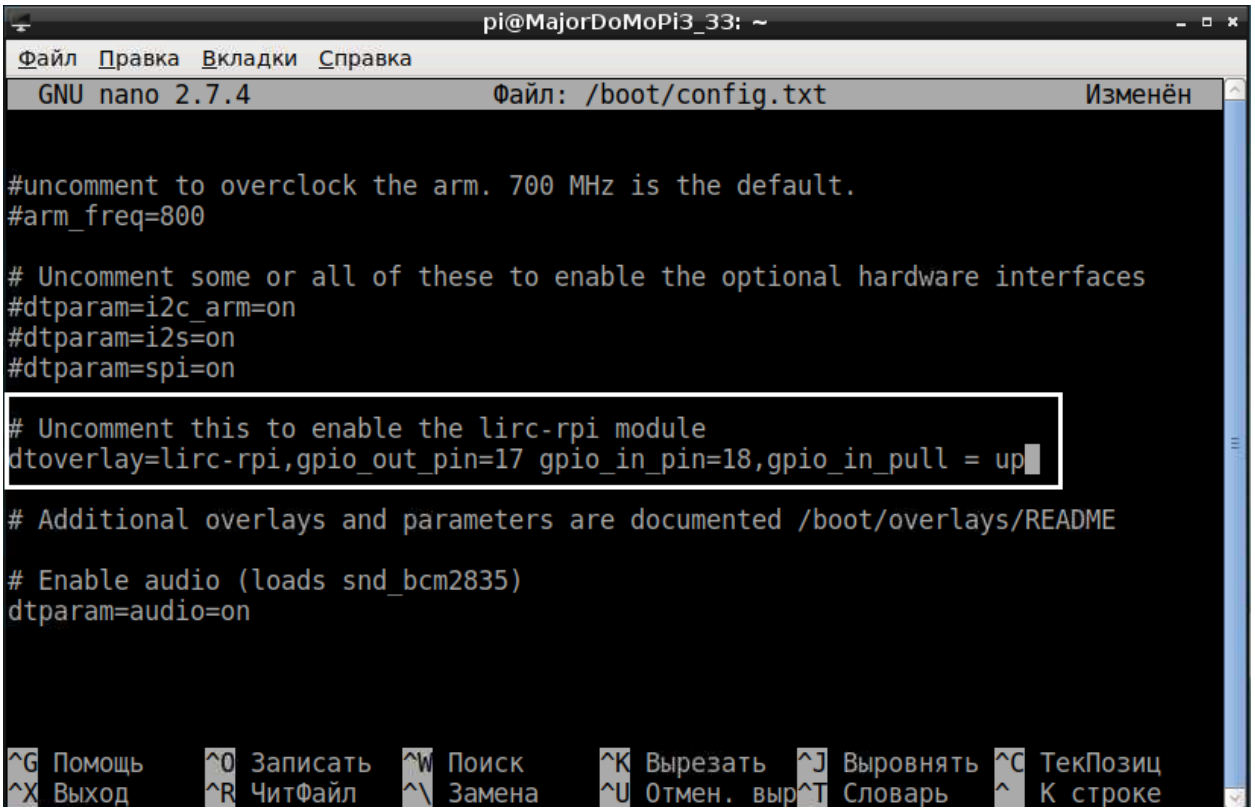


Рис. 7.9. Изменения в загрузочном файле для Raspberry Pi

Далее следует внести исправление в файл /etc/lirc/lirc_options.conf (команда `sudo nano /etc/lirc/lirc_options.conf`), **рис. 7.10**:

```
 [lircd]
driver  = devinput
device  = auto
на:
driver  = default
device  = /dev/lirc0
```

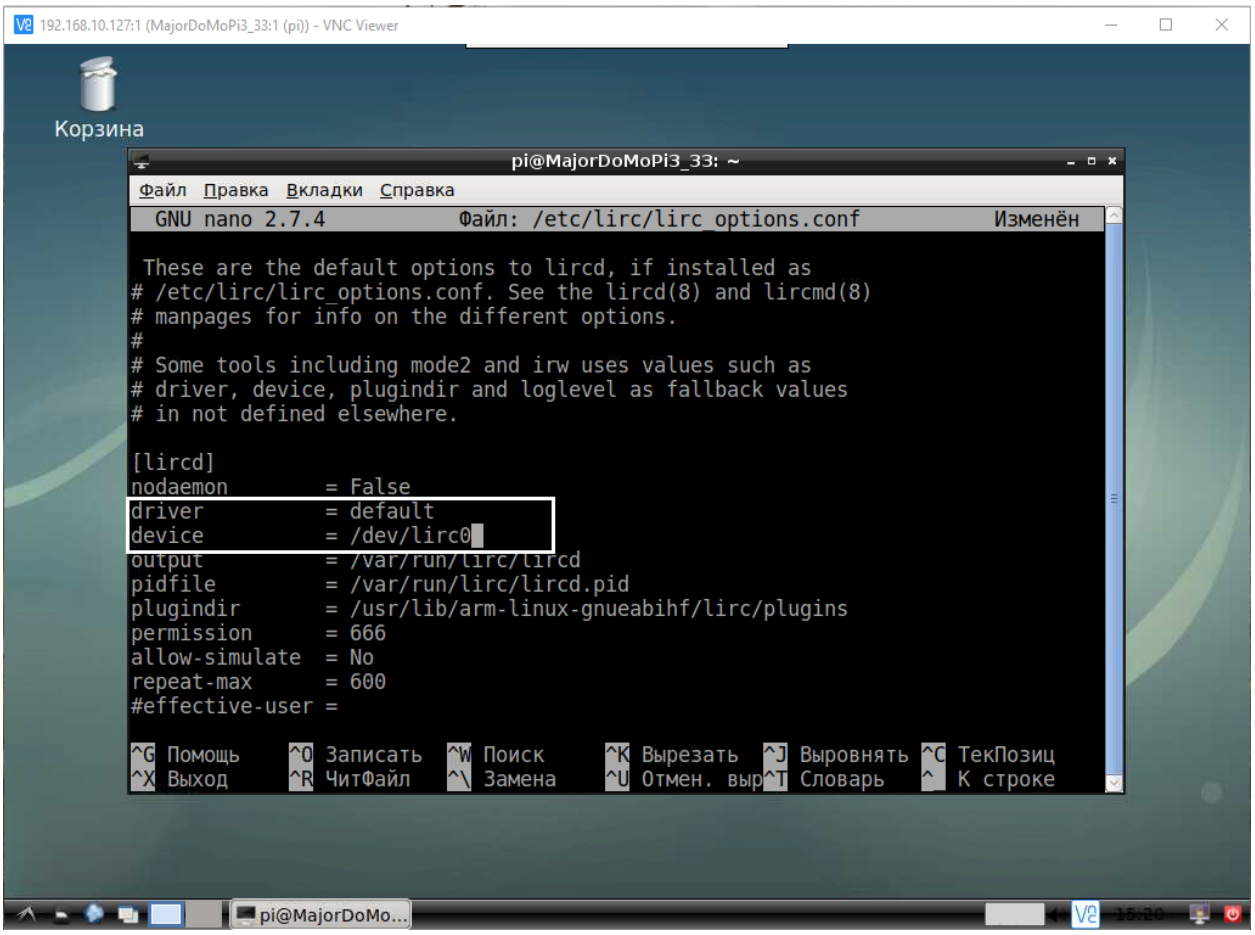


Рис. 7.10. Изменение файла конфигурации lirc

После этих исправлений следует перезагрузить Raspberry Pi, а затем можно произвести проверку, но главное, что мне очень помешало: в записи `gpio_in_pin=18` используется нумерация BCM, что соответствует физическому выводу 12!

На макетной плате я устанавливаю фотоприемник TSOP31438 (см. Приложение Б, **рис. Б.5**). Если смотреть на его лицевую сторону, то ножки слева-направо: CND, V+, out. Подключаются они к модулю Raspberry: GND к выводу 6, V+ к выводу 1, out к выводу 12 (все выводы физические).

Теперь можно в консоли дать команды:

```
sudo systemctl stop lircd.socket
sudo systemctl stop lircd.service
```

И следом даем команду:

```
mode2 --driver default --device /dev/lirc0
```

Теперь фотоприемник готов к приему кодов (**рис 7.11**). Если нажать кнопку пульта управления телевизора, то появится код этой кнопки в режиме временного представления этого кода.

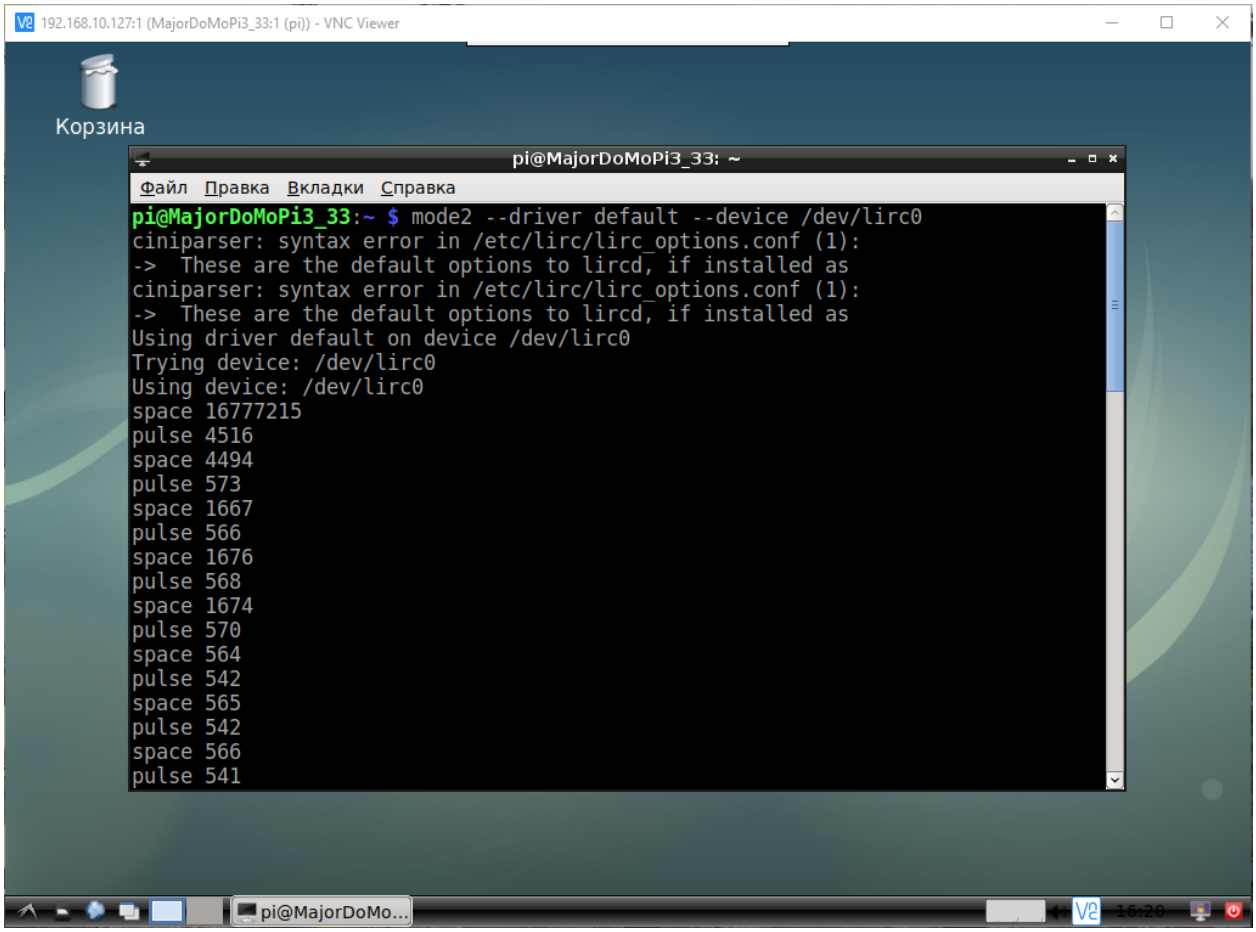


Рис. 7.11. Проверка работы приемника ИК-кода

Запишем этот код. Начинается процедура записи кодов с команды: `sudo /etc/init.d/lircd stop`. Следующая команда, которую нужно выполнить: `irrecord -d /dev/lirc0 --disable-namespace`.

Вы может узнать больше о командах, если введете: `irrecord --help`. Ключ `-d` предшествует вводу устройства, в данном случае `/dev/lirc0`. Завершает команду ключевое слово `--disable-namespace`. Оно указывает на то, что названия клавиш будут отличаться от тех, что прописаны по умолчанию. Нужно это в том случае, если вы не используется названия кнопок, заданных при установке программы lirc.

После запуска программы записи я советую внимательно знакомиться с текстом, который появляется в консоли. Все действия, как правило, подтверждаются нажатием клавиши Enter (Return). Например:

```
Press RETURN to continue.
Checking for ambient light  creating too much disturbances.
Please don't press any buttons, just wait a few seconds...
No significant noise (received 0 bytes)
```
Нажмите RETURN (ENTER) для продолжения.
Проверка рассеянного света, создающего слишком много помех.
Пожалуйста, не нажимайте кнопок, только подождите несколько секунд…
Нет существенного шума (получено 0 байт).

Первое, что вас попросят сделать – ввести имя устройства, я выбрал tvOnOff:

```
Enter name of remote (only ascii, no spaces): tvOnOff
Using tvOnOff.lircd.conf as output filename
Now start pressing buttons on your remote control.
```

Введите название пульта (только в коде ascii без пробелов): tvOnOff
Используется tvOnOff.lircd.conf в качестве имени выходного файла
Теперь начните нажимать кнопки на вашем пульте управления.

```
Now start pressing buttons on your remote control.
It is very important that you press many different buttons
randomly and hold them down for approximately one second. Each
button should generate at least one dot but never more than ten
dots of output.
Don't stop pressing buttons until two lines of dots (2x80)
have been generated.
Press RETURN now to start recording.
```

Теперь нажимайте кнопки на вашем пульте.
Очень важно, чтобы вы нажимали много разных кнопок случайным образом и удерживали их около одной секунды. Каждая кнопка должна вызвать появление хотя бы одной точки, но не более десяти точек при нажатии.
Не останавливайте процесс до появления двух линий точек (2x80).
Теперь нажмите RETURN (ENTER), чтобы начать запись.

```
Please keep on pressing buttons like described above.
```

Пожалуйста, продолжайте нажимать кнопки, как это описано выше.

Следуя инструкциям, нажимайте любую кнопку, пока вам не будет предложено ввести имя кнопки. Я использовал имена 0, 1, 2, 3. После ввода имени и нажатия на клавишу Enter, появится предложение: `Now hold down button "0".` Нажимаем кнопку 0 и ждем появления предложения ввести имя следующей кнопки. Процесс повторяем до записи всех нужных кнопок. Для завершения этой части записи нажимаем клавишу Enter.

Следующее предложение:

```
Checking for toggle bit mask.
Please press an arbitrary button repeatedly as fast as
possible. Make sure you keep pressing the SAME button and that
you DON'T HOLD the button down!.
If you can't see any dots appear, wait a bit between button
presses.
Press RETURN to continue.
```

Проверка битовой маски переключения.
Пожалуйста, нажимайте любую кнопку так быстро, как это возможно. Убедитесь, что нажимаете ОДНУ и ту же кнопку, и что вы не УДЕРЖИВАЕТЕ эту кнопку.

Если вы не видите появления точек, подождите немного между нажиманием на кнопку.

Нажмите RETURN (ENTER) для продолжения.

Нажимайте кнопки, наблюдайте за появлением точек, пока не появится завершающее сообщение (**рис. 7.12**).

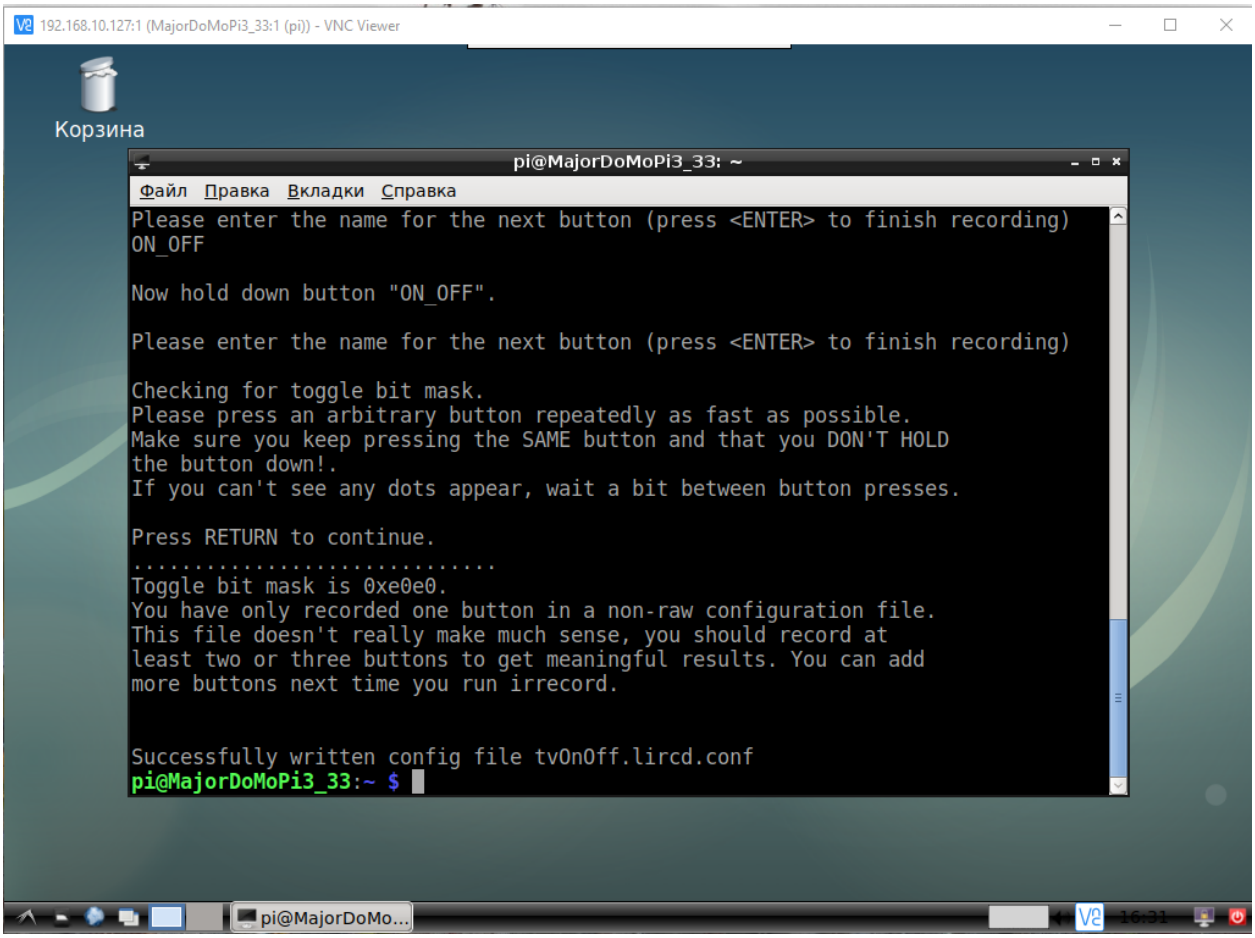


Рис. 7.12. Завершение записи кода кнопки

Последнее сообщение о том, что успешно записан файл tvOnOff.lircd.conf. В домашней папке это файл есть.

Если бы процесс был неудачен, появился бы пустой файл с расширением *tmp*. Его следует удалить. И после неудачи я советую перезагрузить Raspberry. Удачно записанный файл имеет такой текст:

```
# Please take the time to finish this file as described in
# https://sourceforge.net/p/lirc-remotes/wiki/Checklist/
# and make it available to others by sending it to
# <lirc@bartelmus.de>
#
# This config file was automatically generated
# using lirc-0.9.4c(default) on Tue Nov 13 16:31:18 2018
# Command line used: -d /dev/lirc0 --disable-namespace
```

```
# Kernel version (uname -r): 4.14.71-v7+
#
# Remote name (as of config file): tvOnOff
# Brand of remote device, the thing you hold in your hand:
# Remote device model nr:
# Remote device info url:
# Does remote device has a bundled capture device e. g., a
#     usb dongle? :
# For bundled USB devices: usb vendor id, product id
#     and device string (use dmesg or lsusb):
# Type of device controlled
#     (TV, VCR, Audio, DVD, Satellite, Cable, HTPC, ...) :
# Device(s) controlled by this remote:

begin remote

  name  tvOnOff
  bits           32
  flags SPACE_ENC|CONST_LENGTH
  eps            30
  aeps          100

  header       4523  4493
  one           568  1672
  zero          568   541
  ptrail        570
  gap          108146
  min_repeat      1
#  suppress_repeat 1
#  uncomment to suppress unwanted repeats
  toggle_bit_mask 0xE0E0
  frequency    38000

      begin codes
          ON_OFF                   0xE0E040BF
      end codes

end remote
```

Нам этот файл потребуется для воспроизведения кода включения телевизора. Воспроизводится наш код командой: `irsend SEND_ONCE tvOnOff ON_OFF`. Предварительно следует внести ряд изменений, а именно:

1. Скопируем полученный файл записи в нужное место командой

```
sudo cp tvOnOff.lircd.conf
/etc/lirc/lircd.conf.d/tvOnOff.lircd.conf
```

2. Перезагрузим Raspberry Pi (так проще).

3. Проверим, есть ли наш файл: `irsend list "" ""`. Должно появиться такое сообщение:

```
devinput
tvOnOff
devinput
```

4. Теперь можно воспроизвести код кнопки «ON_OFF» командой:

```
irsend SEND_ONCE tvOnOff ON_OFF
```

Если светодиод, а я использую обычный красный светодиод АЛ307Б (см. Приложение Б, **рис. Б.6**), подключен, то он загорится при подаче команды. Светодиод подключен анодом к выводу 11 разъема Raspberry Pi, к его катоду подключен резистор 220 Ом, второй конец которого соединен с выводом 9 Raspberry Pi.

Механизм записи и воспроизведения ИК-кода можно использовать для управления телевизорами, музыкальными центрами, домашними кинотеатрами, кондиционерами и т.д. В моем эксперименте был обычный индикаторный красный светодиод АЛ307Б с резистором 220 Ом. Его можно располагать возле фотоприемника устройства, которым вы предполагаете управлять, и все должно получиться. Но можно использовать специализированный ИК-светодиод для управления на расстоянии. В этом случае придется добавить транзисторный ключ.

На домашней странице lirc вы можете найти обширную базу готовых кодов для бытовых приборов. Если найдете нужный, вам не придется записывать, можно будет воспроизводить их сразу.

Но вводить терминальные команды для Raspberry при воспроизведения нескольких кодов – не самый удобный вариант. Что можно предпринять?

Начнем эксперименты с использования исполняемого текстового файла (скрипта).

```
#!/bin/bash
/usr/bin/irsend SEND_ONCE tvOnOff  ON_OFF
```

Эту запись можно сделать в текстовом редакторе. Файл сохраним под именем, например, *OnOff.sh*. Теперь, если щелкнуть по этому файлу левой кнопкой мышки дважды, так обычно запускаются все программы, то появится окно выбора, что нужно сделать с этим файлом. Но у меня этого не получилось, пришлось выполнить команду в терминале:

```
chmod +x /home/pi/OnOff.sh
```

После задания метода выполнения этого файла стало появляться окошко выбора, предлагающее выбрать, что нужно сделать с этим файлом. При выборе кнопки **Выполнить**, светодиод на макетной плате мигает, что пока и следовало проверить.

Теперь в текстовом редакторе напишем еще один файл, который сохраним под именем *OnOff.py*:

```
import subprocess
subprocess.call("/home/pi/OnOff.sh", shell=True)
```

Параметр `sheel=True` подтверждает выбор запуска скрипта *OnOff.sh* без обращения к диалогу выбора, что нужно сделать с этим файлом. И обратите внимание, что `True` следует написать с большой буквы, иначе Питон не поймет вас.

Теперь команда в консоли (можно и без `sudo`) `python OnOff.py` выполняется, светодиод мигает. А мы вспомним про MajorDoMo.

Сценарий для телевизора

Эту часть я хочу начать с выяснения, была ли ошибка при описании функции запроса? Вот первый вариант:

```
// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
  if (rclient.connect(server, 80)) {
   Serial.println("OK");
   rclient.print(buf);
   rclient.println(" HTTP/1.0");
   rclient.print("Host: ");
   sprintf(ipbuff, "%u.%u.%u.%u", ip[0], ip[1], ip[2], ip[3]);
   rclient.println(ipbuff); // ip адрес модуля
   rclient.print("Content-Type: text/html\n");
   rclient.println("Connection: close\n");
   delay(2000);

   rclient.stop();
  } else {
   Serial.println("FAILED");
  }
}
```

```
Переменные:
// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 119 };
byte subnet[] = { 255, 255, 255, 0 };
byte gateway[] = { 192, 168, 10, 1 };
byte dns_server[] = { 192, 168, 10, 1 };

// ip-адрес удалённого сервера
byte server[] = { 192, 168, 10, 172 };
char ipbuff[16];
```

В тот раз я решил, что это правильный, работающий вариант программы. Но позже, пытаясь передать температуру и давление, я обнаружил, что работает не этот, а другой вариант:

```
// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 120 };
byte subnet[] = { 255, 255, 255, 0 };
byte gateway[] = { 192, 168, 10, 1 };
byte dns_server[] = { 192, 168, 10, 1 };

// ip-адрес удалённого сервера
byte server[] = { 192, 168, 10, 127 };

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
  if (rclient.connect(server, 80)) {
   Serial.println("OK");
   rclient.print(buf);
   rclient.println(" HTTP/1.0");
   rclient.print("Host: ");
   rclient.println("192.168.10.127"); // ip-адрес сервера
   rclient.println("Connection: close");
   rclient.println();
    rclient.println();
   delay(2000);
   rclient.stop();
   rclient.flush();
  } else {
   Serial.println("FAILED");
  }
}
```

Код почти одинаковый, разница небольшая, но она есть. Функцию создавал не я, взял готовую, но я не понимаю, что следует отправлять в отмеченной строке – ip-адрес сервера или ip-адрес модуля Arduino? Поскольку я намерен использовать Arduino в этом эксперименте, я хочу убедиться, что ошибки нет, что я правильно понимаю функцию запроса.

По описанию главы 5 раздела «Arduino с Ethernet-шилдом» подготовим MajorDoMo.

Вероятно, не нужно создавать еще один класс, но мне проще его создать, чем разобраться с уже созданными. Новый класс я назову *Telly*, свойство оставлю *status*, а метод назову *irsender*. Последним я добавлю объект *garage*, где будет датчик двери гаража. Чтобы увидеть состояние датчика, я добавлю свойство *status* на панель *Scena 1* рядом с

температурой и давлением. И, пожалуй, запишу строку обращения к объекту для программы Arduino:

```
http://192.168.10.127:80/objects/?object=garage&op=m&m=irsender&
```

Теперь я скопирую программу из ранее работавшей (как мне казалось), чтобы проверить оба варианта функции sendHTTPRequest().

Первый вариант программы такой:

```
#include <SPI.h>
#include <Ethernet.h>

char buf[80]; // Массив для строки запроса
char ipbuff[16]; // Массив для строки ip-адреса
int old_garage=0; // Переменная состояния

// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 120 }; // ip-адрес модуля
byte subnet[] = { 255, 255, 255, 0 }; // Маска подсети
byte gateway[] = { 192, 168, 10, 1 }; // ip-адрес шлюза
byte dns_server[] = { 192, 168, 10, 1 }; // ip-сервера DNS

// ip-адрес удалённого сервера (Raspberry Pi)
byte server[] = { 192, 168, 10, 127 };

EthernetClient rclient;

// Функция отправки HTTP-запроса на сервер
void sendHTTPRequest() {
  Serial.println(buf);
  if (rclient.connect(server, 80)) { // Соединение с сервером
   Serial.println("OK");
     rclient.println(buf);
    rclient.println("HOST: ");
    sprintf(ipbuff, "%u.%u.%u.%u", ip[0], ip[1], ip[2],
ip[3]);
    rclient.println(ipbuff); // ip-адрес модуля
    rclient.println("Connection: clos\n");
    rclient.println();
    delay(2000);
    rclient.stop(); // Останавливаем клиента
  } else {
   Serial.println("FAILED");
  }
```

```
}

void setup()
{
  // Для отладки будем выводить сообщения на монитор порта
  Serial.begin(9600); // Скорость работы порта
  Serial.println("Start");
//Ethernet.init(10);  // Для большинства Arduino shields
// Инициализируем Ethernet Shield
Ethernet.begin(mac, ip, dns_server, gateway, subnet);
 pinMode(4, INPUT); // Вывод 4 на вход
 old_garage=digitalRead(4); // Читаем состояние датчика
}

void loop()
{
  //Датчик гаражной двени
  Serial.println("G");
  int current_garage=digitalRead(4); // Читаем текущее
состояние датчика гаражной двери
// Если текущее и предыдущее значения не совпадают
  if (current_garage!=(int)old_garage) {
    old_garage=(int)current_garage; // Значение устарело
// Строка для отправки запроса на изменение состояния
   sprintf(buf, "GET
/objects/?object=garage&op=m&m=irsender&status=%i",
(int)current_garage);
    sendHTTPRequest(); // Отправка запроса
  }
  delay(4000);
}
```

Открываем монитор порта и наблюдаем следующее (**рис. 7.13**).

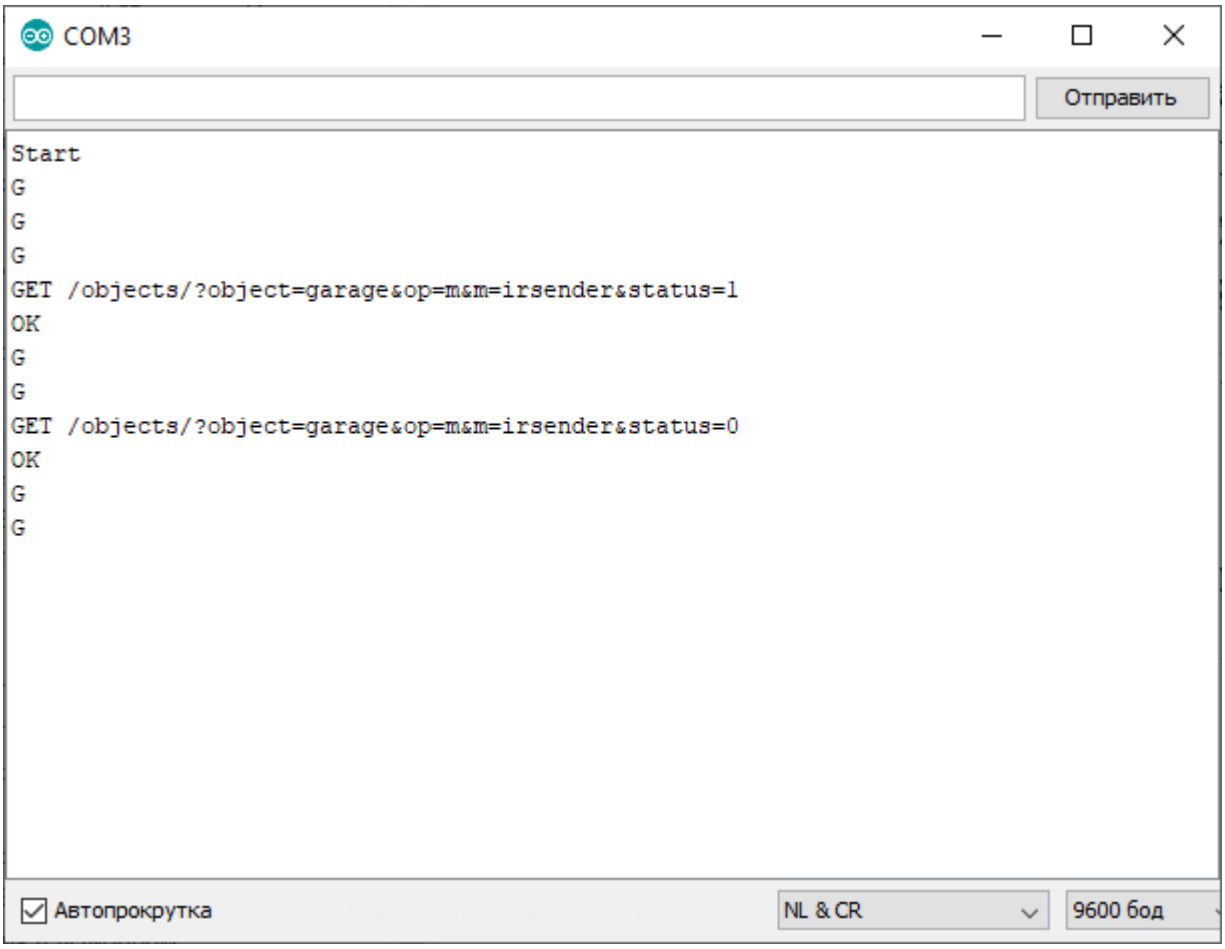


Рис. 7.13. Работа первого варианта программы

И в браузере, наблюдающем за MajorDoMo, состояние датчика двери меняется (**рис. 7.14**).

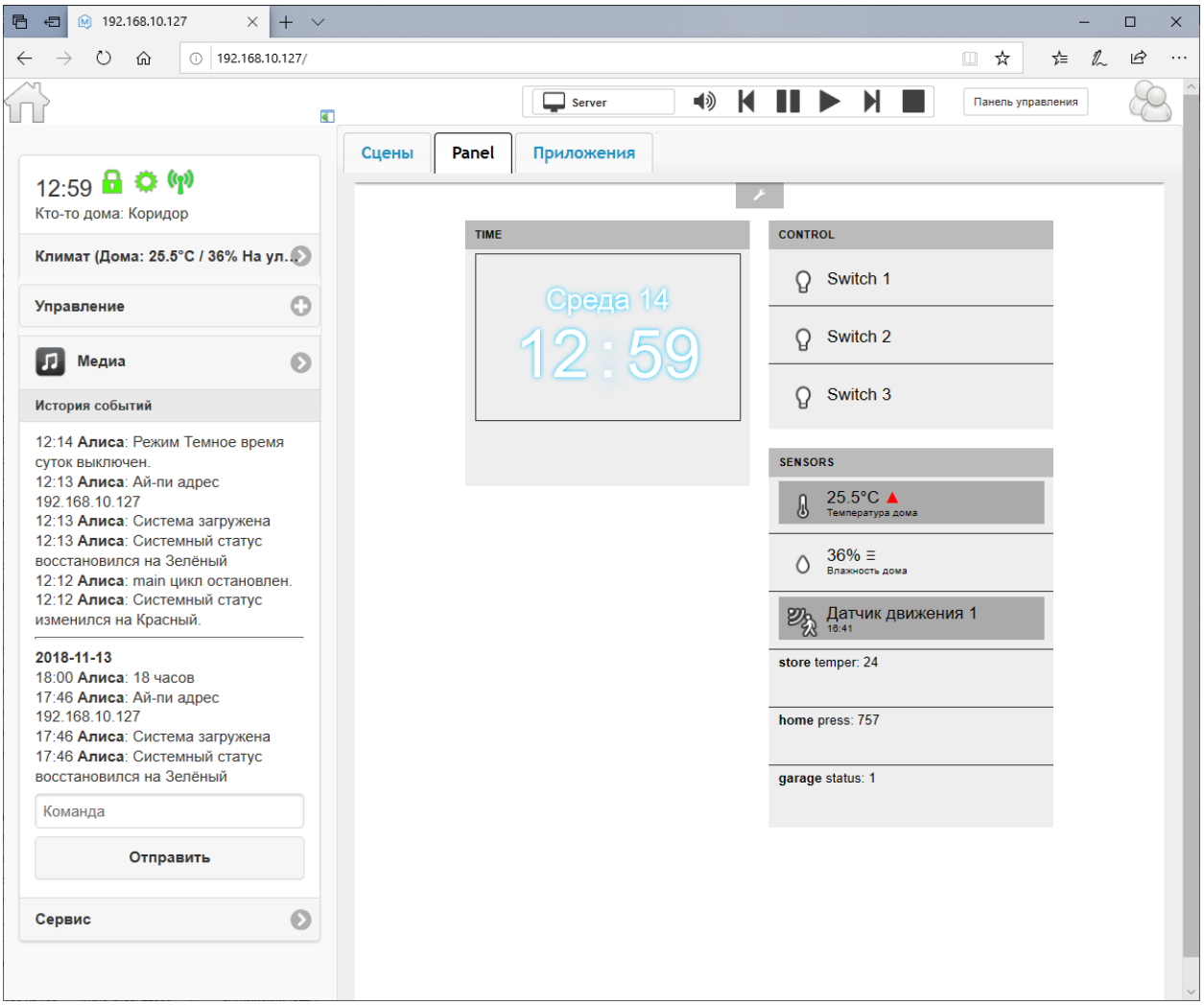


Рис. 7.14. Изменение состояния датчика в браузере

Перейдем ко второму варианту:

```
#include <SPI.h>
#include <Ethernet.h>

char buf[80]; // Массив для строки запроса
char ipbuff[16]; // Массив для строки адреса
int old_garage=0; // Переменная для старого значения состояния

// MAC-адрес нашего устройства
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 10, 120 }; // ip-адрес модуля
byte subnet[] = { 255, 255, 255, 0 }; // Маска подсети
byte gateway[] = { 192, 168, 10, 1 }; // ip-адрес шлюза
byte dns_server[] = { 192, 168, 10, 1 }; // ip-адрес сервера
DNS

// ip-адрес удалённого сервера (Raspberry Pi)
```

```
  byte server[] = { 192, 168, 10, 127 };

  EthernetClient rclient;

  // Функция отправки HTTP-запроса на сервер
  void sendHTTPRequest() {
    Serial.println(buf); // Строка запроса на монитор порта
    if (rclient.connect(server, 80)) { // Подключение к серверу
     Serial.println("OK");
       rclient.println(buf); // Строка запроса для сервера
      rclient.println("HOST: ");
      rclient.println("192.168.10.127"); // Адрес сервера
      rclient.println("Connection: clos\n");
      rclient.println();
     delay(2000);
     rclient.stop(); // Останавливаем клиента
    } else {
     Serial.println("FAILED");
    }
  }

  void setup()
  {
    // Для отладки сообщения на монитор порта
    Serial.begin(9600); // Скорость работы порта
    Serial.println("Start");
  //Ethernet.init(10);  // Для большинства Arduino shields
  // Инициализируем Ethernet Shield
  Ethernet.begin(mac, ip, dns_server, gateway, subnet);
   pinMode(4, INPUT); // Вывод 4 на вход
   old_garage=digitalRead(4); // Читаем значение состояния
  }

  void loop()
  {
    // Датчик гаражной двери
    Serial.println("G");
    int current_garage=digitalRead(4); // Текущее состояние
датчика гаражной двери
    // Если текущее и предыдущее состояния не совпадают
    if (current_garage!=(int)old_garage) {
      old_garage=(int)current_garage; // Устаревшее значение
  // Строка для отправки запроса
     sprintf(buf, "GET
/objects/?object=garage&op=m&m=irsender&status=%i",
(int)current_garage);
```

```
    sendHTTPRequest(); // Отправка запроса
  }
  delay(4000);
}
```

Монитор порта отображает изменение состояния датчика (**рис. 7.15**).

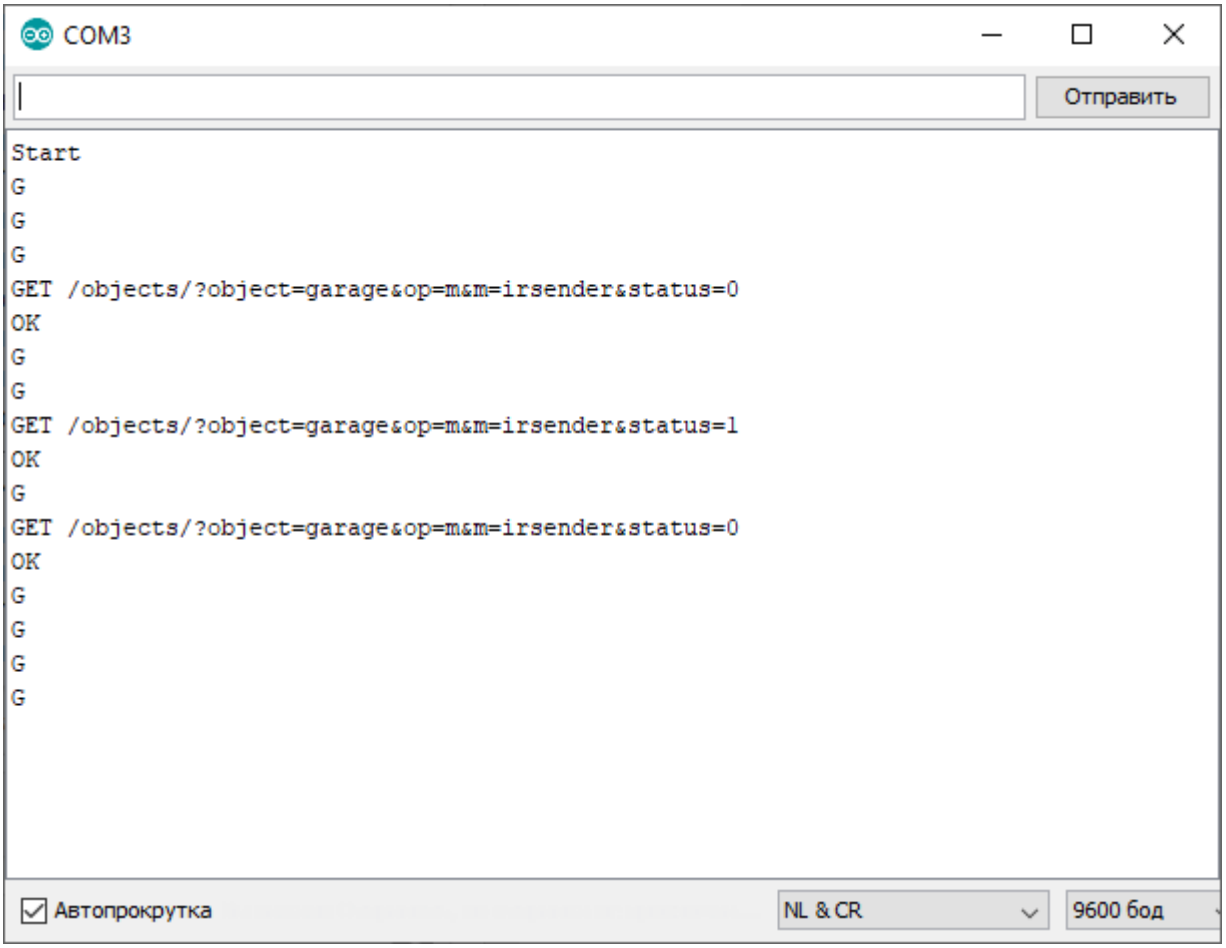


Рис. 7.15. Монитор порта, второй вариант программы

И вы можете не верить, но и MajorDoMo подтверждает – датчик меняет состояние. То есть, оба варианта работают, что мне не понятно, но, главное, работает второй вариант, который понадобится мне для дальнейшей работы.

В этом месте моего повествования компьютер Raspberry Pi обиделся на меня. Я выключил его, но не отключил от USB-порта, когда ушел в магазин за хлебом. И обиделся он так, что перестал отвечать на ip-адрес ни по WiFi, ни при подключении к роутеру кабелем. Пришлось перезаписать SD-карту, настроить все заново, заново восстановить графическую оболочку и т.д. Нужно было как-то загладить мою вину!

Это важно!

Если вы заинтересованы в использовании Raspberry Pi с системой MajorDoMo, то постарайтесь до закупки оборудования и выполнения большой работы проверить все на долговременную устойчивость. Поскольку даже один отказ без видимых причин должен настораживать. Проверьте обязательно!

Сделав попытку добавить на панель *status* объекта *garage*, я обнаруживаю, что панель перестала работать (**рис. 7.16**). То есть, она пытается загрузиться, но у нее ничего не получается. Все-таки, похоже, программа *lirc* зацепила что-то, что цеплять не следовало. Но это только мое предположение. В прошлый раз, когда я экспериментировал с Raspberry Pi, подключение дисплея и переключение на работу с портом HDMI сносило устройство *lirc0*, восстановить которое мне не удалось, поскольку переустановка программы *lirc* не помогала.

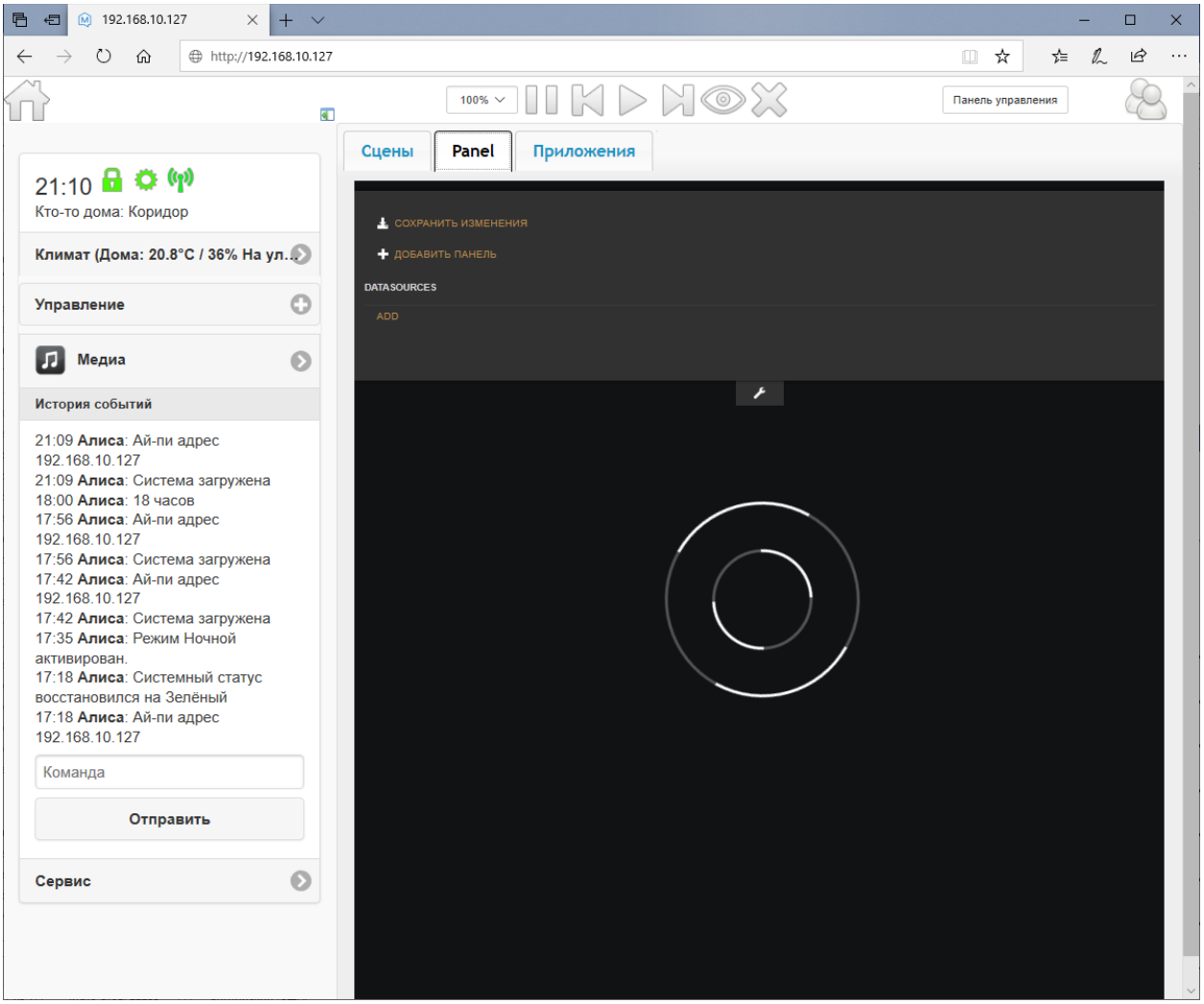


Рис. 7.16. Попытка перейти к панели MajorDoMo

Это было предположение, которое не оправдалось. Похоже, что причина вышеописанного явления в очередном обновлении Windows 10. На следующий день картина изменилась (**рис. 7.17**).

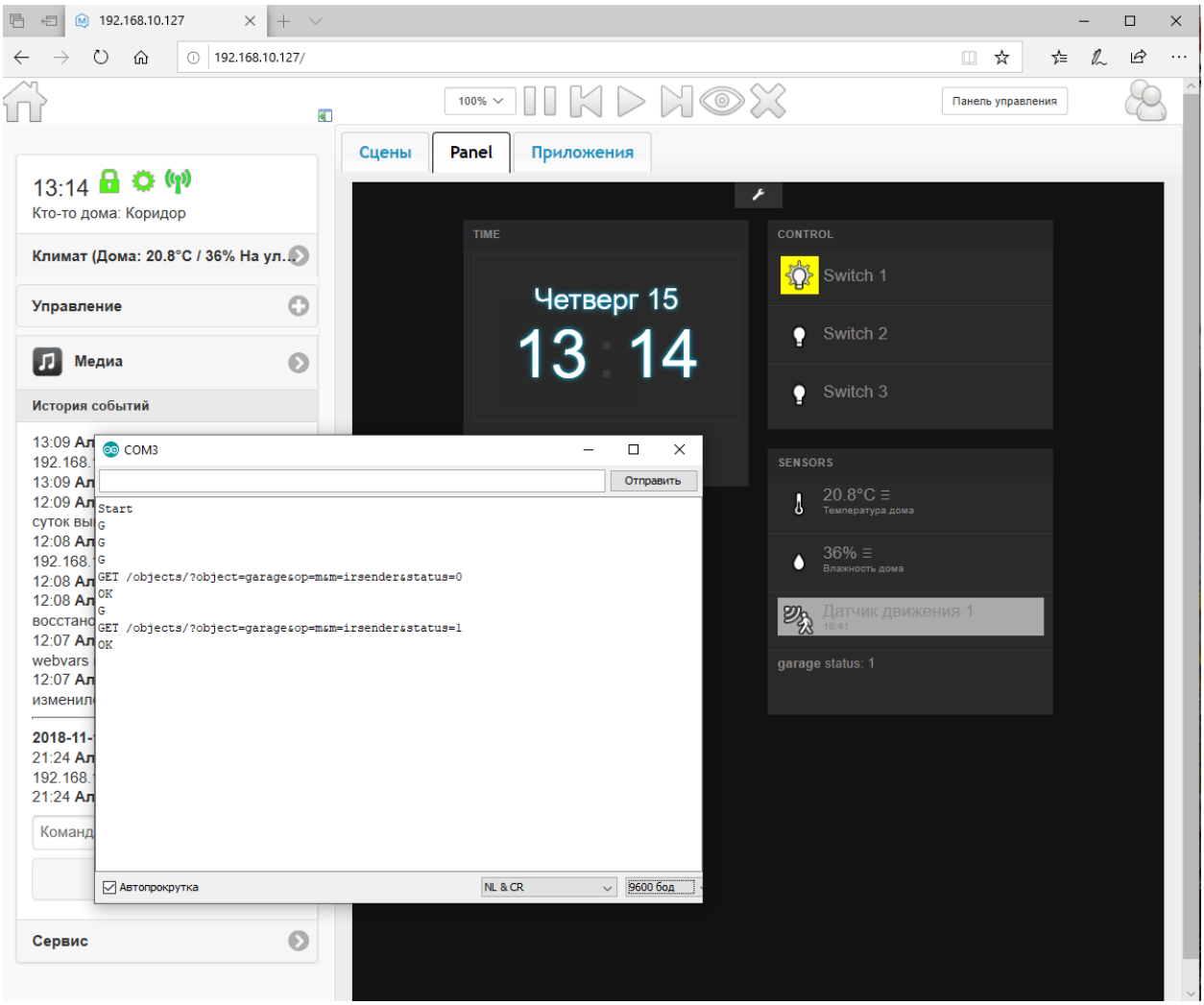


Рис. 7.17. Запуск MajorDoMo на следующий день

И состояние входа в гараж исправно менялось.

Теперь мы попробуем создать сценарий, выполняющий включение телевизора. Для этого выберем:

Панель упрвления→Объекты→Сценарии, кнопка **+Добавить новый сценарий**. Назовем новый сценарий *Television*, добавив его в категорию *События*. В раздел кода PHP я добавлю ту команду, которая работала у меня при прошлых опытах с Raspberry Pi (**рис. 7.18**):

```
exec("python /home/pi/OnOff.py");
```

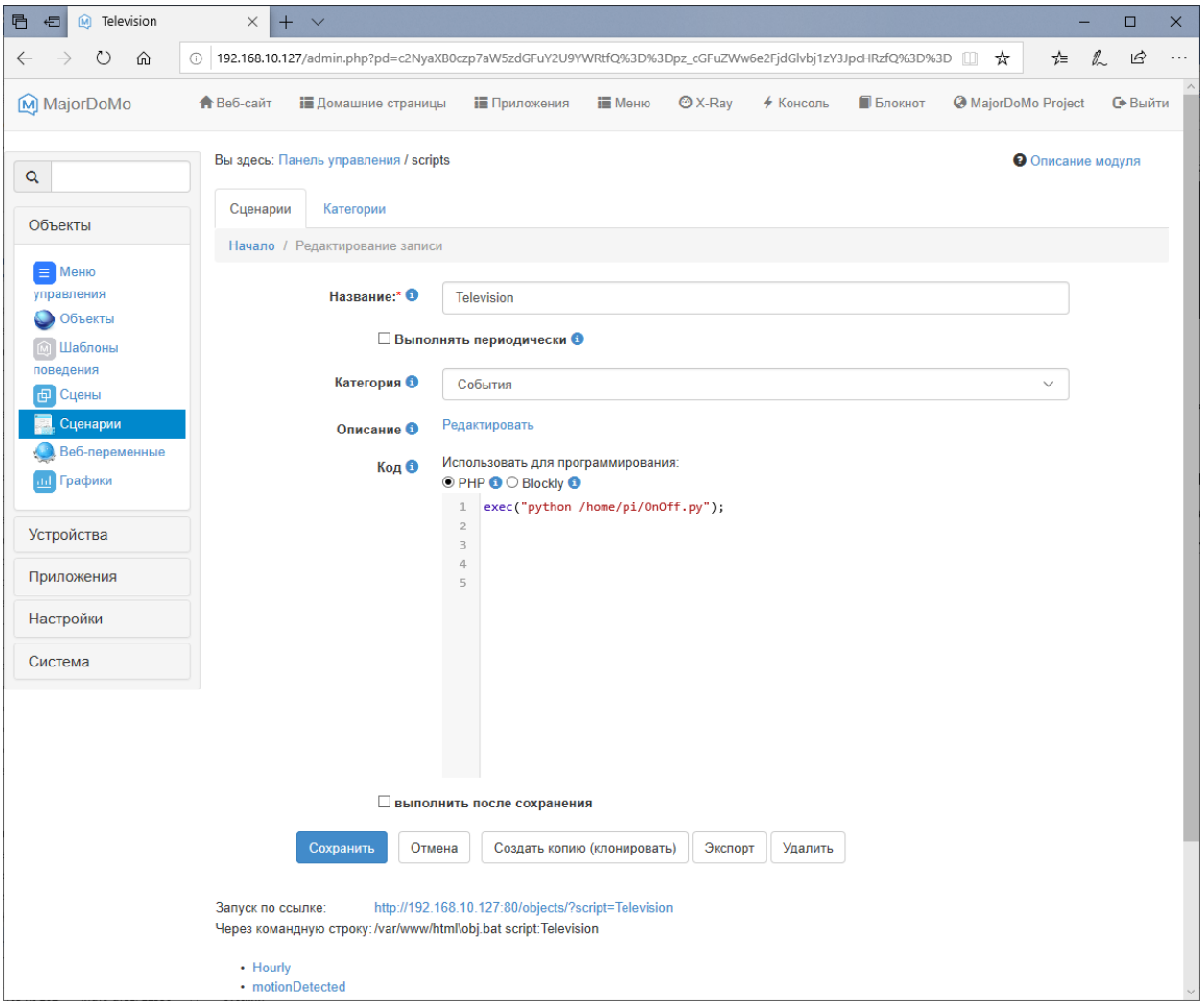


Рис. 7.18. Создание нового сценария

Этот опыт удался у меня не с первого раза. В процессе создания нужных кодов несколько раз получалось так, что состояние датчика на панели MajorDoMo не менялось, но и светодиод не реагировал на команду. Чтобы понять происходящее в других средах разработки я привык использовать отладчик, но есть ли отладчик в MajorDoMo, я не знаю. Зато в MajorDoMo есть вывод сообщений, которым можно воспользоваться.

Решив изменить метод, записанный для датчика гаража, я добавил функцию: `say("Хорошо")`. Появление сообщения помогло понять, работает ли изменение метода. Вот, как выглядит в окончательном виде метод для датчика гаража (**рис. 7.19**):

```
$this->setProperty('status',$params['status']);
if($params['status'] == 1) {
 say('{Хорошо}');
 rs('Television');
}
```

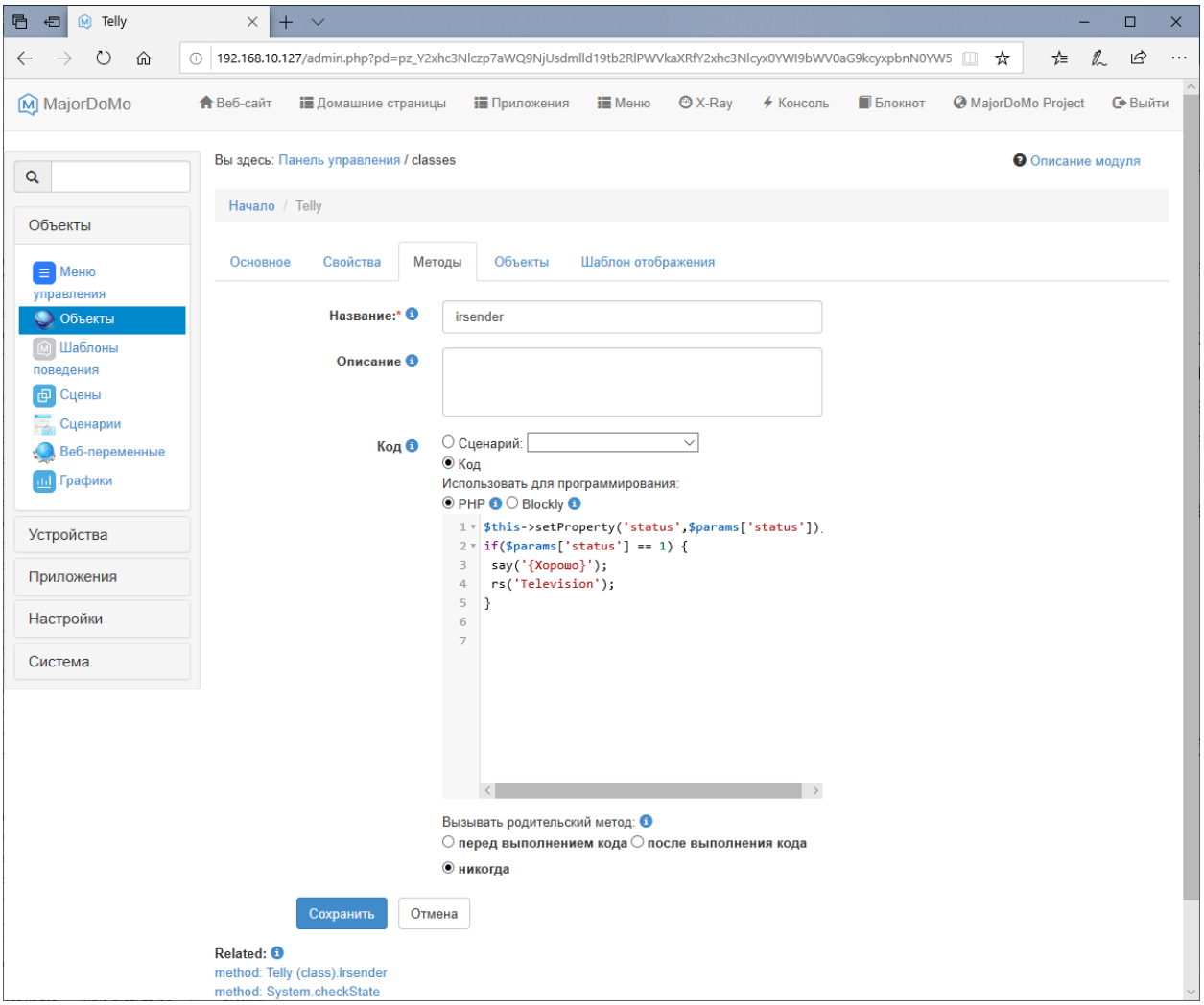


Рис. 7.19. Изменение метода для датчика гаража

Теперь состояние датчика меняется на панели, появляется сообщение от Алисы, а светодиод мигает (**рис. 7.20**).

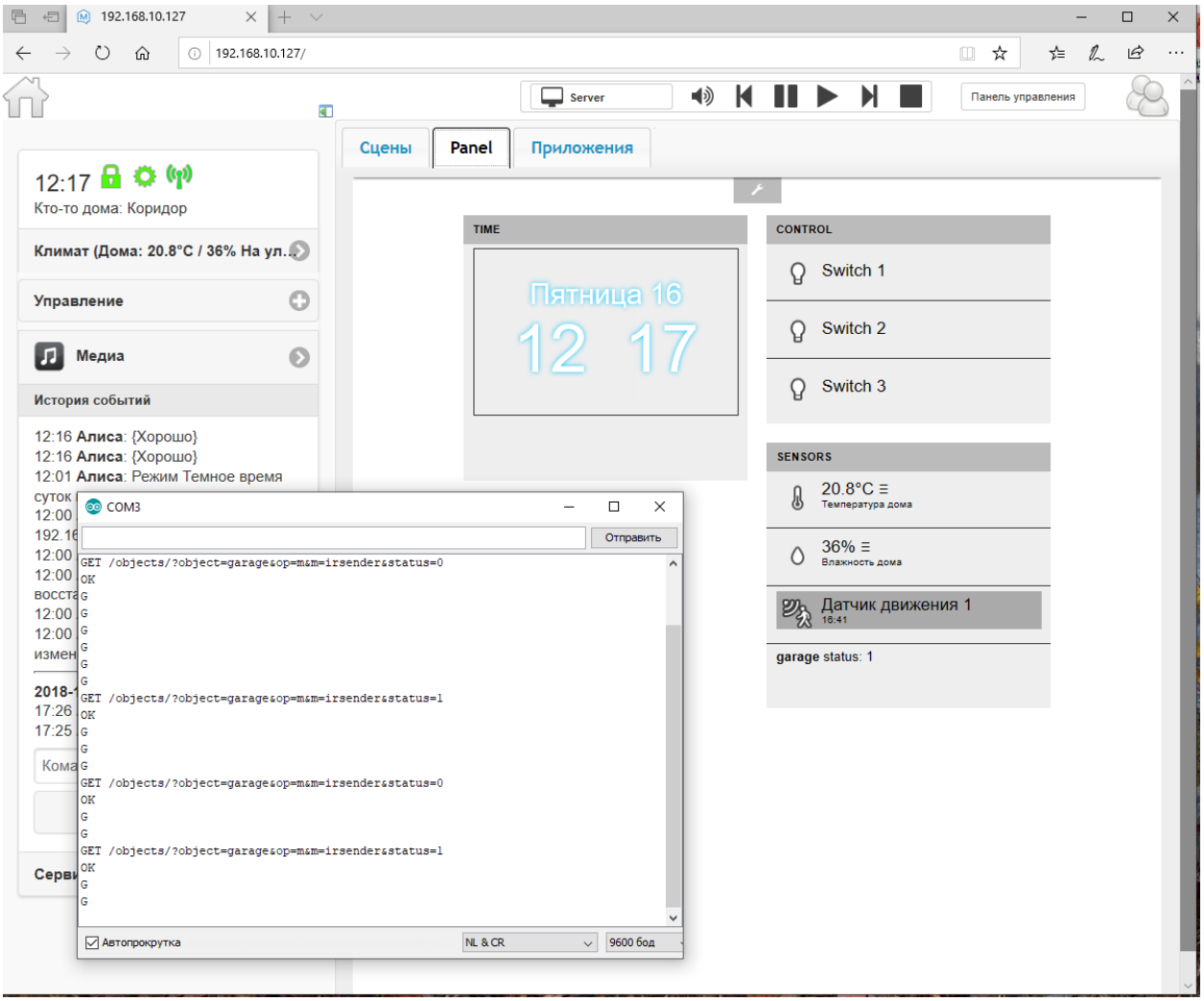


Рис. 7.20. Работа сценария

Можно удалить отладочное сообщение, а можно изменить фразу чем-то вроде: «Дверь гаража открыта». Конечно, для реального использования этого сценария мало – если вы живете не один, то перед выполнением ИК-кода следует проверить, не включен ли телевизор? Иначе вы его выключите. И это не все. Утром, отправляясь на работу, вы тоже откроете гараж, включите телевизор, но для кого? Все эти изменения предстоит внести в сценарий, а для этого предстоит лучше узнать систему MajorDoMo, язык PHP, провести еще не один эксперимент с оборудованием.

Или, что я советую, для обучения дома разным разностям обратитесь за помощью к специалистам.

Эпилог

Проделанные опыты с разными модулями, компьютером Raspberry Pi и системой MajorDoMo помогли нам убедиться в том, что модули могут общаться между собой в домашней сети, могут общаться с сервером. Но Интернет – это только расширение локальной сети. Перед модулями ставились простые задачи: что-то включить или выключить, определить состояние датчиков или прочитать данные о темереатуре и давлении.

Но эти задачи могут быть гораздо сложнее. Сегодня успешно проводятся опыты по созданию искусственного интелекта. Электронные устройства, обладающие искусственным интелектом, будут способны успешно решать, как выполинть те или действия, решить те или иные задачи, какие средства привлечь для достижения целей. Важно только, чтобы на пути достижения целей искусственного интелекта не оказалось таких помех, как обитатели дома. Особенно, если это будет наш общий дом – Земля!

Приложение А. Система АТ команд ESP8266 версия 0.2

Группа 1. Базовый набор команд

Команда	**Описание**
AT	Тестовая проверка при старте
AT+RST	Рестарт модуля
AT+GMR	Обзор информации о версии
AT+GSLP	Переход в режим глубокого сна
ATE	Будет или нет эхо АТ команд
AT+RESTORE	Сброс до фабричных настроек
AT+UART	UART конфигурация
AT+UART_CUR	UART текущая конфигурация (?)
AT+UART_DEF	UART предопределенная конфигурация, сохраняется во флэш

Группа 2. Функции WiFi

WiFi	
Команда	**Описание**
AT+CWMODE	WIFI режим（sta/AP/sta+AP）
AT+CWMODE_CUR	WIFI режим（sta/AP/sta+AP). Не будет храниться во флэш памяти.
AT+CWMODE_DEF	WIFI default mode（sta/AP/sta+AP). Сохранится во флэш памяти.
AT+CWJAP	Подключение к AP.
AT+CWJAP_CUR	Подключение к AP, не будет храниться во флэш памяти.
AT+CWJAP_DEF	Подключение к AP, будет храниться в флэш памяти (Flash).
AT+CWLAP	Список доступных AP.
AT+CWQAP	Отключение от AP.
AT+CWSAP	Задание конфигурации ESP8266, программный AP (softAP).
AT+CWSAP_CUR	Задание конфигурации ESP8266, программный AP. Не сохраняется во флэш.
AT+CWSAP_DEF	Задание конфигурации ESP8266, программный AP. Сохраняется во флэш.
AT+CWLIF	Получить ip станции, которая подключена к программному AP ESP8266.
AT+CWDHCP	Разрешить/Запретить DHCP.
AT+CWDHCP	Разрешить/Запретить DHCP, не будет храниться во флэш памяти.
AT+CWDHCP	Разрешить/Запретить DHCP, будет храниться во флэш памяти.
AT+CWAUTOCONN	Подключиться к AP автоматически, когда питание включено.
AT+CIPSTAMAC	Задать mac адрес ESP8266 станции.
AT+CIPSTAMAC_CUR	Задать mac адрес ESP8266 станции. Не будет храниться во флэш.
AT+CIPSTAMAC_DEF	Задать mac адрес ESP8266 станции. Будет храниться во флэш.
AT+CIPAPMAC	Задать mac адрес ESP8266, программный AP (softAP).
AT+CIPAPMAC_CUR	Задать mac адрес ESP8266, программный AP. Не будет храниться во флэш.
AT+CIPAPMAC_DEF	Задать mac адрес ESP8266, программный AP. Будет храниться во флэш.
AT+CIPSTA	Задать ip адрес ESP8266 станции.

AT+CIPSTA_CUR	Задать ip адрес ESP8266 станции. Не будет храниться во флэш.
AT+CIPSTA_DEF	Задать ip адрес ESP8266 станции. Будет храниться во флэш.
AT+CIPAP	Задать ip адрес ESP8266, программный AP (softAP).
AT+CIPAP_CUR	Задать ip адрес ESP8266, программный AP. Не будет храниться во флэш.
AT+CIPAP_DEF	Задать ip адрес ESP8266, программный AP. Будет храниться во флэш.

Группа 3. Команды, связанные с TCP/IP

TCP/IP	
Команда	**Описание**
AT+ CIPSTATUS	Получает статус соединения
AT+CIPSTART	Устанавливает TCP соединение или регистрирует UDP порт
AT+CIPSEND	Отправляет данные
AT+CIPCLOSE	Закрывает TCP/UDP соединение
AT+CIFSR	Получает локальный IP адрес
AT+CIPMUX	Устанавливает режим множественных соединений
AT+CIPSERVER	Конфигурирует как сервер
AT+CIPMODE	Задает режим передачи
AT+SAVETRANSLINK	Сохраняет сквозную связь передачи во флэш
AT+CIPSTO	Задает timeout, когда ESP8266 запускается как TCP сервер
AT+CIUPDATE	Обновляет firmware (базовый код) через сеть
AT+PING	Функция PING

Подробно о командах первой группы

AT – Тестовая команда	
AT	Отклик: OK
	Описание параметров: нет

AT+RST – Рестарт модуля	
AT+RST	Отклик: OK
	Описание параметров: нет
AT+GMR – Обзор информации о версии	
AT+GMR	Отклик: <номер> OK
	Описание параметров: <номер> информация о версии, длина: 8 байт
Примечание	Например, при отклике 0017xxxxxx 0017 означает версию AT.
AT+GSLP – Переход в режим глубокого сна	
AT+GSLP=<время>	Отклик: <время> OK
	Описание параметров: <время> ms , задает время сна ESP8266 в ms. ESP8266 «проснется» после X ms.
Примечание	Оборудование должно поддерживать пробуждение ото сна (XPD_DCDC объединяется с EXT_RSTB через 0R).

ATE – эхо AT команд	
ATE	Отклик: OK
	Описание параметров: ATE0 : Отменить эхо ATE1 : Применить эхо
AT+RESTORE – Сброс настроек до фабричных	
AT+RESTORE	Отклик: OK
Примечание	Сбрасывает конфигурацию до предустановленных фабричных настроек. Произойдет рестарт.
AT+UART – UART конфигурация	
Этот API устарел, лучше использовать AT+UART_CUR или AT+UART_DEF.	
AT + U A R T = <скорость> , <количество битов>,<количество стоповых битов>, <четность>,<контроль потока>	Отклик: OK
	Описание параметров: <скорость> UART baudrate <количество битов> data bits 5: 5 bits data 6: 6 bits data 7: 7 bits data 8: 8 bits data <количество стоповых битов> stop bits 1: 1 bit stop bit 2: 1.5 bit stop bit 3: 2 bit stop bit <четность> parity 0: None 1: Odd 2: EVEN <контроль потока> flow control 0: отмена контроля потока 1: разрешить RTS 2: разрешить CTS 3: разрешить и RTS, и CTS
Примечание	1. Эта конфигурация будет храниться во флэш области параметров пользователя. 2. Разрешение аппаратного контроля потока подразумевает его поддержку. MTCK - это UART0 CTS , MTDO - это UART0 RTS 3. Baudrate диапазон: 110~115200*40
Пример	AT+UART=115200,8,1,0,0

AT+UART_CUR – текущая UART конфигурация, не будет храниться во флэш	
AT+UART_CUR=<baudrate>, <databits>,<stopbits>, <parity>,<flow control>	Отклик: OK
	Описание параметров: <baudrate> UART скорость работы <databits> количество битов данных 5: 5 bits data 6: 6 bits data 7: 7 bits data 8: 8 bits data <stopbits> количество стоповых битов 1: 1 bit stop bit 2: 1.5 bit stop bit 3: 2 bit stop bit <parity> четность 0: None 1: Odd 2: EVEN <flow control> контроль потока 0: запретить flow control 1: разрешить RTS 2: разрешить CTS 3: разрешить и RTS, и CTS
Примечание	1. Эта конфигурация не будет храниться во флэш. 2. Чтобы использовать аппаратный flow control, оборудование должно поддерживать его. MTCK - этоUART0 CTS , MTDO - это UART0 RTS 3.Baudrate диапазон: 110~115200*40
Пример	AT+UART_CUR=115200,8,1,0,3

AT+UART_DEF – задает конфигурацию UART и сохраняет ее во флэш как предопределенное значение.	
AT+UART_DEF=<baudrate>, <databits>,<stopbits>, <parity>,<flow control>	Отклик: OK
	Описание параметров: <baudrate> UART baudrate <databits> data bits 5: 5 bits data 6: 6 bits data 7: 7 bits data 8: 8 bits data <stopbits> stop bits 1: 1 bit stop bit 2: 1.5 bit stop bit 3: 2 bit stop bit <parity> parity 0: None 1: Odd 2: EVEN <flow control> flow control 0: disable flow control 1: enable RTS 2: enable CTS 3: enable both RTS and CTS
Примечание	1. ТЭта конфигурация будет храниться во флэш в области параметров пользователя. 2. Для разрешения аппаратного контроля потока оборудование должно его поддерживать. MTCK - это UART0 CTS , MTDO - это UART0 RTS 3. Диапазон baudrate: 110~115200*40
Пример	AT+UART_DEF=115200,8,1,0,3

Подробно о командах второй группы

AT+CWMODE – режим WIFI (station/softAP/station+softAP)	
Используйте лучше AT+CWMODE_CUR или AT+CWMODE_DEF.	
Тестовая функция: Получает значение границ режима wifi **AT+CWMODE=?**	Отклик: +CWMODE:(значение границы <mode>) OK
	Описание параметров: <mode> 1 означает Station mode 2 означает AP mode 3 означает AP + Station mode
Функция запроса: Запрос текущего wifi режима ESP8266. **AT+CWMODE?**	Отклик: +CWMODE:<mode> OK
	Описание параметров: то же, что выше.
Функция задания: Задает ESP8266 wifi режим **AT+CWMODE=<mode>**	Отклик: OK
	Описание параметров: то же, что выше.
Примечание	Эта конфигурация будет храниться в области системных параметров во флэш.
Пример	AT+CWMODE=3

AT+CWMODE_CUR – задает режим WIFI (sta/AP/sta+AP), не будет сохраняться во флэш (Flash)	
Тестовая функция: Получает значение границ wifi режима. **AT+CWMODE_CUR=?**	Отклик: +CWMODE_CUR:(значение границ <mode>) OK
	Описание параметров: <mode> 1 означает Station mode 2 означает AP mode 3 означает AP + Station mode
Функция запроса: Запрос текущего wifi режима ESP8266. **AT+CWMODE_CUR?**	Отклик: +CWMODE_CUR:<mode> OK
	Описание параметров: то же, что выше.
Функция задания: Задает режим ESP8266 wifi (mode) **AT+CWMODE_CUR= <mode>**	Отклик: OK
	Описание параметров: то же, что выше.
Примечание	Эта конфигурация не сохраняется во флэш памяти.
Пример	AT+CWMODE_CUR=3

<table>
<tr><th colspan="2">AT+CWMODE_DEF - WIFI режим (sta/AP/sta+AP), сохраняется во флэш (Flash)</th></tr>
<tr><td rowspan="2">Тестовая функция:
Получает значение границ wifi режима.

AT+CWMODE_DEF=?</td><td>Отклик:
+CWMODE_DEF:(значение границ <mode>)

OK</td></tr>
<tr><td>Описание параметров:
<mode> 1 означает Station mode
2 означает AP mode
3 означает AP + Station mode</td></tr>
<tr><td rowspan="2">Функция запроса:
Запрашивает текущий wifi режим ESP8266.

AT+CWMODE_DEF?</td><td>Отклик:
+CWMODE_DEF:<mode>

OK</td></tr>
<tr><td>Описание параметров: то же, что выше.</td></tr>
<tr><td rowspan="2">Функция задания:
Задает wifi режим ESP8266

AT+CWMODE_DEF=
<mode></td><td>Отклик:

OK</td></tr>
<tr><td>Описание параметров: то же, что выше.</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация будет храниться в области системных параметров во флэш памяти.</td></tr>
<tr><td>Пример</td><td>AT+CWMODE_DEF=3</td></tr>
</table>

<table>
<tr><th colspan="2">AT+CWJAP – Подключение к AP</th></tr>
<tr><td colspan="2">Лучше использовать AT+CWJAP_CUR или AT+CWJAP_DEF.</td></tr>
<tr><td rowspan="2">Функция запроса:
Запрашивает информацию AP, что подключает ESP8266.

AT+ CWJAP?</td><td>Отклик:
+ CWJAP:<ssid>

OK</td></tr>
<tr><td>Описание параметров:
<ssid> строка, AP SSID</td></tr>
<tr><td rowspan="2">Функция задания:
Задает информацию AP, с которой будет подключение ESP8266.

AT+ CWJAP =
<ssid>,< pwd ></td><td>Отклик:

OK
ERROR</td></tr>
<tr><td>Описание параметров:
<ssid> строка, AP’s SSID
<pwd> строка пароля, MAX: 64 байта ASCII

Эта команда требует, чтобы был разрешен режим станции.
Символ escape нужен в тех случаях, когда "SSID" или "password" содержат какие-то специальные символы
(’,’、 ’“’and’\’)</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация будет сохраняться в области системных параметров во флэш (Flash).</td></tr>
<tr><td>Пример</td><td>AT+ CWJAP =“abc”, “0123456789”
Если SSID - это “ab\,c”
а пароль – это “0123456789”\”
AT+CWJAP =“ab\\\,c”, “0123456789\”\\”</td></tr>
</table>

<table>
<tr><th colspan="2">AT+CWJAP_CUR – подключение к AP, не будет храниться во флэш</th></tr>
<tr><td rowspan="2">Функция запроса:
Запрашивает информацию AP, что подключает ESP8266.

AT+CWJAP_CUR?</td><td>Отклик:
+ CWJAP_CUR:<ssid>

OK</td></tr>
<tr><td>Описание параметров:
<ssid> строка, AP’s SSID</td></tr>
<tr><td rowspan="2">Функция задания:
Задает информацию AP, с которой будет подключение ESP8266.

AT+CWJAP_CUR =
<ssid>,< pwd ></td><td>Отклик:

OK
ERROR</td></tr>
<tr><td>Описание параметров:
<ssid> строка, AP’s SSID
<pwd> строка пароля, MAX: 64 байта ASCII

Эта команда требует, чтобы был разрешен режим станции.
Символ escape нужен в тех случаях, когда "SSID" или "password" содержат какие-то специальные символы
(’,’、 ’“’and’\’)</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация не будет храниться во флэш.</td></tr>
<tr><td>Пример</td><td>AT+CWJAP_CUR =“abc”,“0123456789”
Если SSID - это “ab\,c”
а пароль “0123456789”\”
AT+CWJAP_CUR=“ab\\\,c”,“0123456789\”\\”</td></tr>
</table>

<table>
<tr><th colspan="2">AT+CWJAP_DEF – Подключение к AP, сохраняется как предопределенное</th></tr>
<tr><td rowspan="2">Функция запроса:
Запрашивает информацию AP, что подключает ESP8266.

AT+CWJAP_DEF?</td><td>Отклик:
+ CWJAP_DEF:<ssid>

OK</td></tr>
<tr><td>Описание параметров:
<ssid> строка, AP’s SSID</td></tr>
<tr><td rowspan="2">Функция задания:
Задает информацию AP, с которой будет подключение ESP8266.

AT+ CWJAP_DEF =
<ssid>,< pwd ></td><td>Отклик:

OK
ERROR</td></tr>
<tr><td>Param description ：
<ssid> строка, AP’s SSID
<pwd> строка пароля, MAX: 64 байта ASCII

Эта команда требует, чтобы был разрешен режим станции.
Символ escape нужен в тех случаях, когда "SSID" или "password" содержат какие-то специальные символы
(’,’、’“’and’\’)</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация будет храниться в области системных параметров во флэш.</td></tr>
<tr><td>Пример</td><td>AT+CWJAP_DEF =“abc”,“0123456789”
Если SSID - это “ab\,c”
а пароль “0123456789”\”
AT+CWJAP_DEF=“ab\\\,c”,“0123456789\”\\”</td></tr>
</table>

AT+CWLAP – список доступных AP	
Функция задания: Ищет доступные AP с заданными условиями. **AT+ CWLAP =** **<ssid>,< mac >,<ch>**	Отклик: + CWLAP: <ecn>,<ssid>,<rssi>,<mac>,<ch> OK ERROR
	Описание параметров: то же, что ниже.
Выполняемая функция: Список всех доступных AP. **AT+CWLAP**	Отклик: + CWLAP: <ecn>,<ssid>,<rssi>,<mac>,<ch> OK ERROR
	Описание параметров: < ecn > 0 OPEN 1. WEP 2. WPA_PSK 3. WPA2_PSK 4. WPA_WPA2_PSK <ssid> строка, SSID для AP <rssi> сила сигнала <mac> строка, MAC адрес
Пример	AT+CWLAP="wifi","ca:d7:19:d8:a6:44",6 Или находится AP со специфическим ssid: AT+CWLAP="wifi"," "

AT+CWQAP – Отключение от AP	
Тестовая функция: Только для проверки **AT+CWQAP=?**	Отклик: OK
	Описание параметров:
Выполняемая функция: Отключение от AP. **AT+ CWQAP**	Отклик: OK
	Описание параметров:

<table>
<tr><th colspan="2">AT+ CWSAP – Конфигурация программного softAP режима</th></tr>
<tr><td colspan="2">Лучше использовать AT+CWSAP_CUR или AT+CWSAP_DEF.</td></tr>
<tr><td rowspan="2">Функция запроса:
Запрашивается конфигурация softAP режима.

AT+ CWSAP?</td><td>Отклик:
+ CWSAP:<ssid>,<pwd>,<chl>,<ecn></td></tr>
<tr><td>Описание параметров: то же, что ниже.</td></tr>
<tr><td rowspan="2">Функция задания:
Задает конфигурацию softAP режима.

AT+ CWSAP=
<ssid>,<pwd>,<chl>,
<ecn></td><td>Отклик:

OK
ERROR</td></tr>
<tr><td>Примечание: эта команда доступна только тогда, когда разрешен режим softAP, нуждается в предварении AT+RST, чтобы могла работать.
Описание параметров:
<ssid> строка, ESP8266 softAP SSID
<pwd> строка пароля, MAX: 64 байт ASCII
<chl> id (идентификатор) канала
< ecn > 0 OPEN
2 WPA_PSK
3 WPA2_PSK
4 WPA_WPA2_PSK</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация сохраниться в области системных параметров во флэш памяти.</td></tr>
<tr><td>Пример</td><td>AT+CWSAP="ESP8266","1234567890",5,3</td></tr>
</table>

<table>
<tr><th colspan="2">AT+CWSAP_CUR – Текущая конфигурация softAP режима, не сохранится во флэш</th></tr>
<tr><td rowspan="2">Функция запроса:
Запрос конфигурации softAP режима.

AT+CWSAP_CUR?</td><td>Отклик:
+CWSAP_CUR:<ssid>,<pwd>,<chl>,<ecn></td></tr>
<tr><td>Описание параметров: то же, что ниже.</td></tr>
<tr><td rowspan="2">Функция задания:
Задает конфигурацию softAP режима.

AT+CWSAP_CUR=
<ssid>,<pwd>,<chl>,
<ecn></td><td>Отклик:

OK
ERROR</td></tr>
<tr><td>Примечание: эта команда доступна только тогда, когда разрешен режим softAP, нуждается в предварении AT+RST, чтобы могла работать.
Описание параметров:
<ssid> строка, ESP8266 softAP SSID
<pwd> строка пароля, MAX: 64 байт ASCII
<chl> id (идентификатор) канала
< ecn > 0 OPEN
2 WPA_PSK
3 WPA2_PSK
4 WPA_WPA2_PSK</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация не сохранится во флэш памяти.</td></tr>
<tr><td>Пример</td><td>AT+CWSAP_CUR="ESP8266","1234567890",5,3</td></tr>
</table>

AT+ CWSAP_DEF – Предопределенная конфигурация softAP режима, сохраняется во флэш	
Функция запроса: Запрашивает конфигурацию softAP режима. **AT+ CWSAP_DEF?**	Отклик: + CWSAP_DEF:<ssid>,<pwd>,<chl>,<ecn>
	Описание параметров: то же, что ниже.
Функция задания: Задает конфигурацию softAP режима. **AT+CWSAP_DEF= <ssid>,<pwd>,<chl>, <ecn>**	Отклик: OK ERROR
	Примечание: эта команда доступна только тогда, когда разрешен режим softAP, нуждается в предварении AT+RST, чтобы могла работать. Описание параметров: <ssid> строка, ESP8266 softAP SSID <pwd> строка пароля, MAX: 64 байт ASCII <chl> id (идентификатор) канала < ecn > 0 OPEN 2 WPA_PSK 3 WPA2_PSK 4 WPA_WPA2_PSK
Примечание	Эта конфигурация будет храниться в области системных параметров во флэш.
Пример	AT+CWSAP_DEF="ESP8266","1234567890",5,3

AT+ CWLIF– ip станции, которая подключена к ESP8266 softAP	
Исполняемая функция: Получает ip станции, что подключена к ESP8266 softAP **AT+CWLIF**	Отклик: <ip addr>,<mac> OK
	Описание параметров: <ip addr> ip адрес станции, подключенной к ESP8266 softAP <mac> mac адрес станции, подключенной к ESP8266 softAP

AT+ CWDHCP – Разрешить/Запретить DHCP	
Лучше использовать AT+CWDHCP_CUR или AT+CWDHCP_DEF.	
Запрос: **AT+CWDHCP?**	Отклик: DHCP запрещено или разрешено сейчас?
	Описание: Bit0 : 0 – программное (soft) ap dhcp запрещено 1 - softap dhcp разрешено bit1: 0 - станционное dhcp запрещено 1 - станционное dhcp разрешено
Функция установки: Разрешено/Запрещено DHCP. **AT+CWDHCP=<mode>,<en>**	Отклик: OK
	Описание параметров: <mode> 0 : задать ESP8266 softAP 1 : задать ESP8266 station 2 : задать и softAP, и station <en> 0 : запретить DHCP 1 : разрешить DHCP
Примечание	Эта конфигурация будет храниться в области пользовательских параметров во флэш.

AT+CWDHCP_CUR – Разрешить/Запретить DHCP, не будет храниться во флэш	
Функция установки: Разрешено/Запрещено DHCP. **AT+CWDHCP_CUR= <mode>,<en>**	Отклик: OK
	Описание параметров: <mode> 0 : установить ESP8266 softAP 1 : установить ESP8266 station 2 : установить и softAP, и station <en> 0 : запретить DHCP 1 : разрешить DHCP
Примечание	Эта конфигурация не будет храниться во флэш.
Пример	AT+CWDHCP_CUR=0,1

AT+CWDHCP_DEF – Разрешить/Запретить DHCP и сохранить во флэш	
Функция задания: Разрешено/Запрещено DHCP. **AT+CWDHCP_DEF=** **<mode>,<en>**	Отклик: OK Описание параметров: <mode> 0 : установить ESP8266 softAP 1 : установить ESP8266 station 2 : установить и softAP, и station <en> 0 : запретить DHCP 1 : разрешить DHCP
Примечание	Эта конфигурация будет храниться в области пользовательских параметров во флэш.
Пример	AT+CWDHCP_CUR=0,1

AT+CWAUTOCONN – подключение к AP автоматически или нет	
Функция задания: Подключаться к AP автоматически или нет. **AT+CWAUTOCONN=** **<enable>**	Отклик: OK Описание параметров: <enable> 0 : не подключаться автоматически к AP при включении 1 : подключаться к AP автоматически при включении По умолчанию - enable, ESP8266 станция будет подключаться к AP автоматически при включении питания.
Примечание	Эта конфигурация будет сохранена в области системных параметров во флэш памяти.
Пример	AT+CWAUTOCONN=1

AT+ CIPSTAMAC – Задать mac адрес ESP8266 станции	
Лучше использовать AT+CIPSTAMAC_CUR или AT+CIPSTAMAC_DEF.	
Функция запроса: Получает mac адрес ESP8266 станции. **AT+CIPSTAMAC?**	Отклик: +CIPSTAMAC:<mac> OK
	Описание параметров: <mac> строка, mac адрес ESP8266 станции
Функция задания: Задает mac адрес ESP8266 станции. **AT+CIPSTAMAC=<mac>**	Отклик: OK
	Описание параметров: <mac> строка, mac адрес ESP8266 станции
Примечание	Эта конфигурация будет храниться в области пользовательских параметров во флэш.
Пример	AT+CIPSTAMAC="18:fe:35:98:d3:7b"

AT+ CIPSTAMAC_CUR – Задает mac адрес ESP8266 станции, не будет храниться во флэш	
Функция запроса: Получает mac адрес ESP8266 станции. **AT+CIPSTAMAC_CUR?**	Отклик: +CIPSTAMAC_CUR:<mac> OK
	Описание параметров: <mac> строка, mac адрес ESP8266 станции
Функция задания: Задает mac адрес ESP8266 станции. **AT+CIPSTAMAC_CUR= <mac>**	Отклик: OK
	Описание параметров: <mac> строка, mac адрес ESP8266 станции
Примечание	Эта конфигурация не будет храниться во флэш.
Пример	AT+CIPSTAMAC_CUR="18:fe:35:98:d3:7b"

AT+ CIPSTAMAC_DEF – Задает mac адрес ESP8266 станции, сохраняет во флэш	
Функция запроса: Получает mac адрес ESP8266 станции. **AT+CIPSTAMAC_DEF?**	Отклик: +CIPSTAMAC_DEF:<mac> OK
	Описание параметров: <mac> строка, mac адрес ESP8266 станции
Функция задания: Задает mac адрес ESP8266 станции. **AT+CIPSTAMAC_DEF** **=<mac>**	Отклик: OK
	Описание параметров: <mac> строка, mac адрес ESP8266 станции
Примечание	Эта конфигурация будет храниться в области пользовательских параметров во флэш.
Пример	AT+CIPSTAMAC_DEF="18:fe:35:98:d3:7b"

AT+ CIPAPMAC – Задает mac адрес ESP8266 softAP	
Лучше использовать AT+CIPAPMAC_CUR или AT+CIPAPMAC_DEF.	
Функция запроса: Получает mac адрес ESP8266 softAP. **AT+CIPAPMAC?**	Response： +CIPAPMAC:<mac> OK
	Param description： <mac> string, mac address of ESP8266 softAP
Type: set Function: Set mac address of ESP8266 softAP. Command： **AT+CIPAPMAC=<mac>**	Response： OK
	Param description： <mac> string, mac address of ESP8266 softAP
Note	This configuration will store in Flash user parameter area.
Example	AT+CIPAPMAC="1a:fe:36:97:d5:7b"

AT+CIPAPMAC_CUR – Задает mac адрес ESP8266 softAP, не будет храниться во флэш	
Функция запроса: Получает mac адрес ESP8266 softAP. **AT+CIPAPMAC_CUR?**	Отклик: +CIPAPMAC_CUR:<mac> OK
	Описание параметров: <mac> строка, mac адрес ESP8266 softAP
Функция задания: Задает mac адрес ESP8266 softAP. **AT+CIPAPMAC_CUR= <mac>**	Отклик: OK
	Описание параметров: <mac> строка, mac адрес ESP8266 softAP
Примечание	Эта конфигурация не будет храниться в области пользовательских параметров во флэш.
Пример	AT+CIPAPMAC_CUR="1a:fe:36:97:d5:7b"

AT+ CIPAPMAC_DEF – Set mac address of ESP8266 softAP, save to Flash	
Функция запроса: Получает mac адрес ESP8266 softAP. **AT+CIPAPMAC_DEF?**	Отклик: +CIPAPMAC_DEF:<mac> OK
	Описание параметров: <mac> строка, mac адрес ESP8266 softAP
Функция задания: Задает mac адрес ESP8266 softAP. **AT+CIPAPMAC_DEF =<mac>**	Отклик: OK
	Описание параметров: <mac> строка, mac адрес ESP8266 softAP
Примечание	Эта конфигурация будет храниться в области пользовательских параметров во флэш.
Пример	AT+CIPAPMAC_DEF="1a:fe:36:97:d5:7b"

<table>
<tr><td colspan="2">AT+ CIPSTA – Задает ip адрес ESP8266 станции</td></tr>
<tr><td colspan="2">Лучше использовать AT+CIPSTA_CUR или AT+CIPSTA_DEF.</td></tr>
<tr><td rowspan="2">Функция запроса:
Получает ip адрес ESP8266 станции.

AT+CIPSTA?</td><td>Отклик:
+CIPSTA:<ip>

OK</td></tr>
<tr><td>Описание параметров:
<ip> строка, ip адрес ESP8266 станции</td></tr>
<tr><td rowspan="2">Функция задания:
Задает ip адрес ESP8266 станции.

AT+CIPSTA=<ip>
[,<gateway>,<netmask>]</td><td>Отклик:

OK</td></tr>
<tr><td>Описание параметров:
<ip> строка, ip адрес ESP8266 станции
[<gateway>] щлюз
[<netmask>] маска</td></tr>
<tr><td>Примечание</td><td>Эта конфигурация будет храниться в области пользовательских параметров во флэш.</td></tr>
<tr><td>Пример</td><td>AT+CIPSTA="192.168.6.100","192.168.6.1","255.255.255.0"</td></tr>
</table>

AT+CIPSTA_CUR – Задает ip адрес ESP8266 станции, не будет сохраняться во флэш	
Функция запроса: Получает ip адрес ESP8266 станции. **AT+CIPSTA_CUR?**	Отклик: +CIPSTA_CUR:<ip> OK
	Описание параметров: <ip> строка, ip адрес ESP8266 станции
Функция задания: Задает ip адрес ESP8266 станции. **AT+CIPSTA_CUR** **=<ip>[,<gateway>,<netmask>]**	Отклик: OK
	Описание параметров: <ip> строка, ip адрес ESP8266 станции [<gateway>] шлюз [<netmask>] маска
Примечание	Эта конфигурация не сохраняется в области пользовательских параметров во флэш.
Пример	AT+CIPSTA_CUR="192.168.6.100","192.168.6.1","255.255.255.0"

AT+CIPSTA_DEF – Задает ip адрес ESP8266 станции, сохраняется во флэш	
Функция запроса: Получает ip адрес ESP8266 станции. **AT+CIPSTA_DEF?**	Отклик: +CIPSTA:<ip> OK
	Описание параметров: <ip> строка, ip адрес ESP8266 станции
Функция задания: Задает ip адрес ESP8266 станции. **AT+CIPSTA_DEF** **=<ip>[,<gateway>,<netmask>]**	Отклик: OK
	Описание параметров: <ip> строка, ip адрес ESP8266 станции [<gateway>] шлюз [<netmask>] маска
Примечание	Этак конфигурация будет сохраняться в области пользовательских параметров во флэш.
Пример	AT+CIPSTA_DEF="192.168.6.100","192.168.6.1","255.255.255.0"

AT+ CIPAP – Задает ip адрес ESP8266 softAP	
Лучше использовать AT+CIPAP_CUR или AT+CIPAP_DEF.	
Функция запроса: Получает ip адрес ESP8266 softAP. **AT+CIPAP?**	Отклик: +CIPAP:<ip> OK
	Описание параметров: <ip> строка, ip адрес ESP8266 softAP
Функция задания: Задает ip адрес ESP8266 softAP. **AT+CIPAP=<ip>**	Отклик: OK
	Описание параметров: <ip> строка, ip адрес ESP8266 softAP
Примечание	Эта конфигурация не будет сохраняться в области пользовательских параметров во флэш.
Пример	AT+CIPAP="192.168.5.1"

AT+CIPAP_CUR – Задает ip адрес ESP8266 softAP, не будет сохраняться во флэш	
Функция запроса: Получает ip адрес ESP8266 softAP. **AT+CIPAP_CUR?**	Отклик: +CIPAP_CUR:<ip> OK
	Описание параметров: <ip> строка, ip адрес ESP8266 softAP
Функция задания: Задание ip адреса ESP8266 softAP. **AT+CIPAP_CUR** **=<ip>**	Отклик: OK
	Описание параметров: <ip> строка, ip адрес ESP8266 softAP
Примечание	Эта конфигурация не сохраняется во флэш.
Пример	AT+CIPAP_CUR="192.168.5.1"

AT+ CIPAP_DEF – Задает ip адрес ESP8266 softAP, сохраняется во флэш	
Функция запроса: Получает ip адрес ESP8266 softAP. **AT+CIPAP_DEF?**	Отклик: +CIPAP_DEF:<ip> OK
	Описание параметров: <ip> строка, ip адрес ESP8266 softAP
Функция задания: Задает ip адрес ESP8266 softAP. **AT+CIPAP_DEF** **=<ip>**	Отклик: OK
	Описание параметров: <ip> строка, ip адрес ESP8266 softAP
Примечание	Эта конфигурация будет сохраняться в области пользовательских параметров во флэш.
Пример	AT+CIPAP_DEF="192.168.5.1"

AT+CWSTARTSMART – Запускается SmartConfig	
Функция задания: Запускается SmartConfig. Command： **AT+CWSTARTSMART =<type>**	Отклик: OK
	Описание параметров: < type> тип протокола SmartConfig 1：ESP_TOUCH 2：AirKiss
Примечание	1. Вы можете воспользоваться многими документами про наш SmartConfig от Espressif. 2. ESP8266 станция должна быть разрешена 3. Сообщение “Smart get wifi info” означает, что Smart Config прошло успешно, затем вы можете использовать “AT+CIFSR”, чтобы проверить, получен ли ip от роутера или нет. 4. ESP8266 может ничего не делать в процессе SmartConfig, так что дождитесь успешного завершения или используйте команду “AT +CWSTOPSMART” для остановки SmartConfig.
Пример	AT+CWMODE=3 AT+CWSTARTSMART=1

AT+CWSTOPSMART остановка SmartConfig	
Исполняемая функция: останавливает SmartConfig. **AT+CWSTOPSMART**	Отклик: OK
Примечание	Неважно, успешно завершилась SmartConfig или нет, всегда вызывайте “AT +CWSTOPSMART” для освобождения занятого процессом буфера.
Пример	AT+CWSTOPSMART

Подробно о командах третьей группы

<table>
<tr><td colspan="2">AT+CIPSTATUS – Информация о соединении</td></tr>
<tr><td rowspan="2">Исполняемая функция:
Получает информацию о соединении.

AT+CIPSTATUS</td><td>Отклик:
STATUS:<stat>
+CIPSTATUS:<ID>,<type>,<remote_IP>,<remote_port> ,<local_ port>,<tetype></td></tr>
<tr><td>Описание параметров:
<stat>
2: Получение IP
3: Соединено
4: Разъединено
<id> id (идентификатор) соединения, (0~4) для множественных соединений
<type> строка, “TCP” или “UDP”
<remote_ip> строка, удаленный IP адрес.
<remote_port> номер удаленного порта.
<local_port> ESP8266 номер локального порта
<tetype>
0: ESP8266 работает как клиент
1: ESP8266 работает как сервер</td></tr>
</table>

<table>
<tr><th colspan="2">AT+CIPSTART – Установленное TCP соединение или регистрируемый UDP порт, запускается соединение</th></tr>
<tr><td rowspan="2">Тестовая функция:
Получение информации о параметрах.

AT+CIPSTART=?</td><td>Отклик:
1) Если AT+CIPMUX=0
+CIPSTART:(<type>),(<IP address>),(<port>)[,(<local port>), (<mode>)]
+CIPSTART:(<type>),(<domain name>),(<port>)[,(<local port>), (<mode>)]

OK
2) Если AT+CIPMUX=1
+CIPSTART:(id),(<type>),(<IP address>),(<port>)[,(<local port>), (<mode>)]
+CIPSTART: (id), (<type>),(<domain name>),(<port>)[,(<local port>), (<mode>)]</td></tr>
<tr><td>Описание параметров: нет</td></tr>
<tr><td rowspan="2">Функция задания:
Запускает соединение как клиент.

1) Единичное соединение
(+CIPMUX=0)
AT+CIPSTART=
<type>,<addr>,<port>
[,(<local port>),(<mode>)]

2) Множественное соединение (+CIPMUX=1)
AT+CIPSTART=
<id><type>,<addr>,<port>
[,(<local port>),(<mode>)]</td><td>Отклик:

OK
или
ERROR
Если соединение уже установлено, возвращает
ALREAY CONNECT</td></tr>
<tr><td>Описание параметров:
<id> 0-4 , id соединения
<type> строка, “TCP” или “UDP”
<addr> строка, удаленный ip
<port> строка, удаленный порт
[<local port>] для UDP только
[<mode>] для UDP только
0 : назначенный одноранговый объект UDP не будет меняться.
1 : назначенный одноранговый объект UDP может измениться один раз.
2 : назначенному одноранговому объекту UDP позволено меняться.

Примечание:
[<mode>] может быть использовано только тогда, когда [<local port>] задан.</td></tr>
<tr><td>Пример</td><td>AT+CIPSTART="TCP","192.168.101.110",1000

(Относится только к “Espressif AT Command Examples”)</td></tr>
</table>

<table>
<tr><th colspan="2">AT+CIPSEND – Отправка данных</th></tr>
<tr><td rowspan="2">Тестовая функция:
Только для тестирования.

AT+CIPSEND=?</td><td>Отклик:

OK</td></tr>
<tr><td>Описание параметров: нет</td></tr>
<tr><td rowspan="2">Функция задания:
Задает длину отправляемых данных. Для обычной отправки.

1) Для единичного соединения:
(+CIPMUX=0)
AT+CIPSEND=<length>

2) Для множественных соединений:
(+CIPMUX=1)
AT+CIPSEND=
<id>,<length>

Для UDP передачи, удаленный ip & порт могут быть заданы:
AT+CIPSEND=
[<id>,]<length>
[,<remote ip>,<remote port>]</td><td>Упаковка возвращает “>” после команды установки. Начинает получать последовательность данных, когда длина данных достигнута, начинается отправка данных.

Если соединение не может установиться или разрывается при передаче, возвращается
ERROR
Если данные переданы успешно, возвращается
SEND OK</td></tr>
<tr><td>Примечание: для этой команды
Описание параметров:
<id> ID no. для соединения передачи
<length> длина данных, MAX 2048 байт</td></tr>
<tr><td>Исполняемая функция:
Оправка данных. Для режима передачи без «красот».

AT+CIPSEND</td><td>Отклик:

Упаковка возвращает “>” после выполнения команды. Входит в простейшую передачу, 20 мс – интервал между пакетами, максимум 2048 байт на пакет. Когда единичный пакет, содержащий “+++”, получен, все возвращается в командный режим.

Эта команда может использоваться только в простейшем режиме передачи, для которого требуется единичный режим связи.</td></tr>
<tr><td>Пример</td><td>Обратитесь к “Espressif AT Command Examples”</td></tr>
</table>

AT+CIPCLOSE – Закрывает TCP или UDP соединение	
Тестовая функция: Только для тестирования. **AT+CIPCLOSE=?**	Отклик: OK
Функция задания: Закрывает TCP или UDP соединение. Для режима множественных соединений **AT+CIPCLOSE=<id>**	Отклик: Нет ошибок, возвращает OK Если <id> соединение разорвано, возвращает Link is not
	Описание параметров: <id> ID no. для закрывающегося соединения, когда id=5, все соединения будут закрыты. (id=5 не имеет эффекта в режиме сервера)
Исполняемая функция: Для режима единичного соединения **AT+CIPCLOSE**	Отклик: OK или если нет такого соединения, возвращает ERROR Выводится UNLINK, когда нет соединения

AT+CIFSR – Получает локальный IP адрес	
Тестовая функция: Только для тестирования. **AT+CIFSR=?**	Отклик: OK
Исполняемая функция: Получает локальный IP адрес. **AT+ CIFSR**	Отклик: + CIFSR:<IP address> + CIFSR:<IP address> OK ERROR
	Описание параметров: <IP address> IP адрес ESP8266 softAP IP адрес ESP8266 station

AT+ CIPMUX – Разрешает или нет множественное соединение	
Функция запроса: Получает параметры конфигурации. **AT+ CIPMUX?**	Отклик: + CIPMUX:<mode> OK
	Описание параметров: то же, что ниже.
Функция задания: Задает режим соединения. **AT+ CIPMUX=<mode>**	Отклик: OK Если уже соединено, возвращает Link is builded
	Описание параметров: <mode> 0 единичное соединение 1 множественное соединение
Примечание	1. “AT+CIPMUX=1” может устанавливаться только тогда, когда сквозная передача не разрешена (“AT+CIPMODE=0”). 2. Этот режим может измениться только после разрыва всех соединений. Если сервер работает, требуется перезагрузка.
Пример	AT+CIPMUX=1

AT+ CIPSERVER – Конфигурация TCP сервера	
Функция задания: Устанавливает TCP сервер. **AT+ CIPSERVER= <mode>[,<port>]**	Отклик: OK
	Описание параметров: <mode> 0 Удаляет сервер (необходимо выполнить перезагрузку) 1 Открывает сервер <port> номер порта, по умолчанию 333
Примечание	1. Сервер может быть открыт только тогда, когда AT+CIPMUX=1 2. Монитор сервера будет автоматически открыт, когда открыт сервер. 3. Когда клиент подключается к серверу, он получает одно соединение, получив id.
Пример	AT+ CIPMUX=1 AT+ CIPSERVER=1,1001

AT+ CIPMODE – Задание режима передачи	
Функция запроса: Запрос режима передачи. **AT+ CIPMODE?**	Отклик: + CIPMODE:<mode> OK
	Описание параметров: то же, что ниже.
Функция задания: Задает режим передачи. **AT+CIPMODE=<mode>**	Отклик: OK Если соединение уже есть, возвращает Link is builded
	Описание параметров: <mode> 0 обычный режим 1 простейший режим передачи, только для единичного TCP соединения
Примечание	Эта конфигурация не будет храниться во флэш.
Пример	AT+CIPMODE=1

AT+SAVETRANSLINK – Сохраняет ссылку на сквозную передачу во флэш	
Функция задания: Сохраняет ссылку на сквозную передачу во флэш. **AT+SAVETRANSLINK =<mode>,<ip>,<port>**	Отклик: OK или ERROR
	Описание параметров: <mode> 0 обычный режим 1 режим сквозной передачи <ip> удаленный ip <port> удаленный порт
Примечание	1. Эта команда будет сохранять режим сквозной передачи и ссылку на TCP в области пользовательских параметров. 2. Пока ip и нумерация порта соответствуют заданным, они будут храниться во флэш.
Пример	AT+SAVETRANSLINK=1,”192.168.6.110”,1002

AT+ CIPSTO – Задает timeout сервера TCP	
Функция запроса: Запрашивает timeout (время ожидания) сервера. **AT+CIPSTO?**	Отклик: + CIPSTO:<time> OK
	Описание параметров: то же, что и ниже.
Функция задания: Задает timeout сервера. **AT+CIPSTO=<time>**	Отклик: OK
	Описание параметров: < time> timeout TCP сервера, диапазон 0~7200 секунд
Примечание	ESP8266 как TCP сервер будет отключен от TCP клиента, который не связался с ним, даже при timeout. Если AT+CIPSTO=0, timeout не работает.
Пример	AT+ CIPMUX=1 AT+ CIPSERVER=1,1001 AT+CIPSTO=10

AT+ CIUPDATE – сетевое обновление	
Исполняемая функция: Запускается upgrade. **AT+ CIUPDATE**	Отклик: +CIPUPDATE:<n> OK
	Описание параметров: <n> 1 сервер найден 2 соединение с сервером 3 получение версии 4 запускается обновление
Примечание	Обновление прошивки зависит от условий сетевого соединения. Возвращает ERROR, если upgrade не состоялось, нужно подождать.

AT+PING – Функция Ping	
AT+PING=<ip>	Отклик: +<time> OK или ERROR // означает неудачный ping
	Описание параметров: <ip> : строка, host ip или domain name <time> : время отклика при ping
Пример	AT+PING=”192.168.1.1” AT+PING=”www.baidu.com”

+IPD – Получение данных сети	
1) Единичное соединение: **(+CIPMUX=0)** **+IPD,<len>:<data>** 2) Множественное соединение: **(+CIPMUX=1)** **+IPD,<id>,<len>:<data>**	Примечание: Когда модуль получает сетевые данные, он отправляет их через порт последовательного обмена, используя +IPD команду. Описание параметров: <id> id no. соединения <len> длина данных <data> полученные данные

ESP8266 AT команды, сохраняющие конфигурацию во флэш памяти

Команда	**Пример**
Сохраняется в области пользовательских параметров	
AT+UART_DEF	AT+UART_DEF=115200,8,1,0,3
AT+CWDHCP_DEF	AT+CWDHCP_DEF=1,1
AT+CIPSTAMAC_DEF	AT+CIPSTAMAC_DEF="18:fe:35:98:d3:7b"
AT+CIPAPMAC_DEF	AT+CIPAPMAC_DEF="1a:fe:36:97:d5:7b"
AT+CIPSTA_DEF	AT+CIPSTA_DEF="192.168.6.100"
AT+CIPAP_DEF	AT+CIPAP_DEF="192.168.5.1"

AT+SAVETRANSLINK	AT+SAVETRANSLINK =1,"192.168.6.10”,1001
Сохраняется в системной области параметров	
AT+CWMODE_DEF	AT+CWMODE_DEF=3
AT+CWJAP_DEF	AT+CWJAP_DEF=“abc”, “0123456789”
AT+CWSAP_DEF	AT+CWSAP_DEF="ESP8266","12345678",5,3
AT+CWAUTOCONN	AT+CWAUTOCONN=1

Примечания:

- Новые установки будут проверяться вначале с конфигурацией во флэш, только в том случае, если конфигурация изменилась, она будет записана.
- Для флэш объемом 512KB предопределенные установки:
  область пользовательских параметров 0x3C000 ~ 0x40000, 16KB; область системных параметров 0x7C000~0x80000, 16KB
- Для флэш объемом 1MB (или больше 1MB), предопределенные установки:
  область пользовательских параметров 0x7C000 ~ 0x80000, 16KB; область системных параметров последние 16KB флэш памяти.

Приложение Б. Схемы соединений, использованные в книге

Схема подключений для опыта с входом Arduino

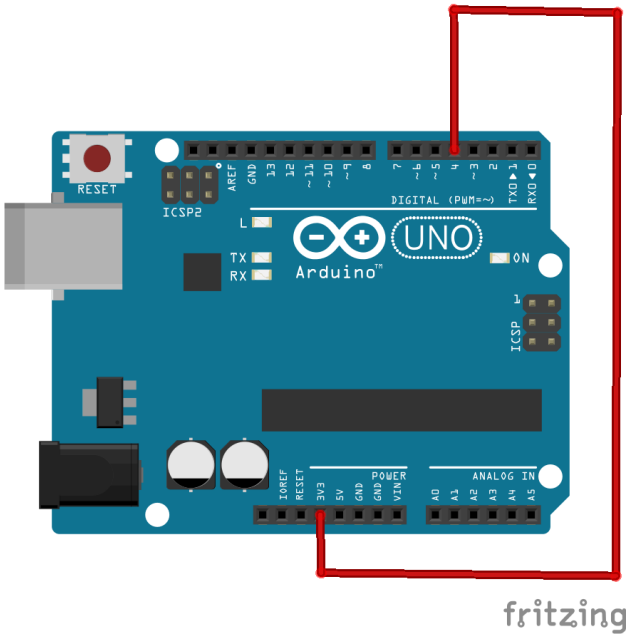


Рис. Б.1. Arduino в опыте главы 5, раздел 2

Схема включения Arduino в качестве переходника для ESP8266

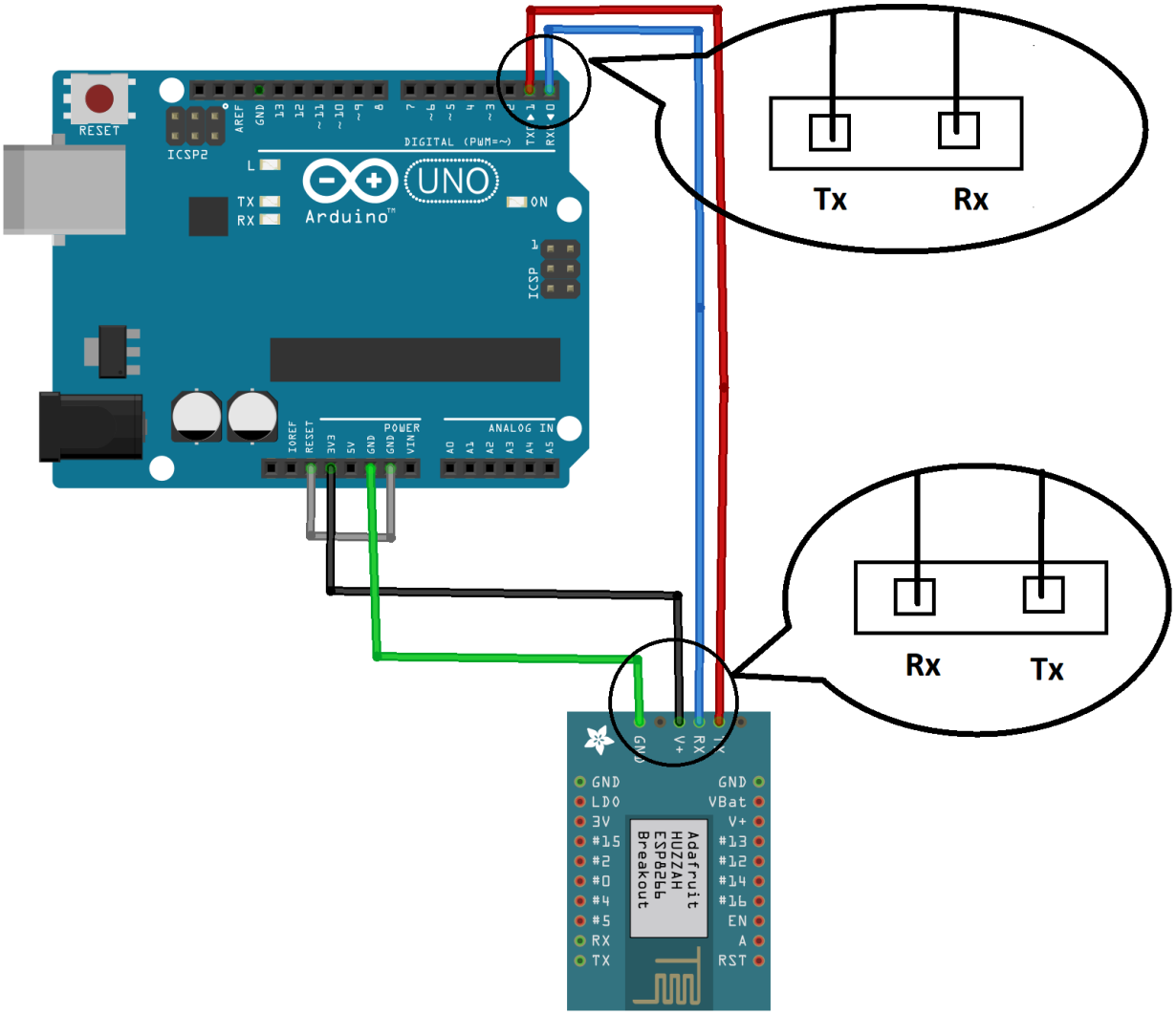
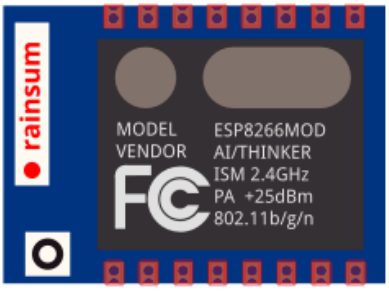


Рис. Б.2. Arduino в режиме USB-TTL переходника для ESP8266

Схема подключения фоторезистора к WeMOS для передачи данных ESP8266



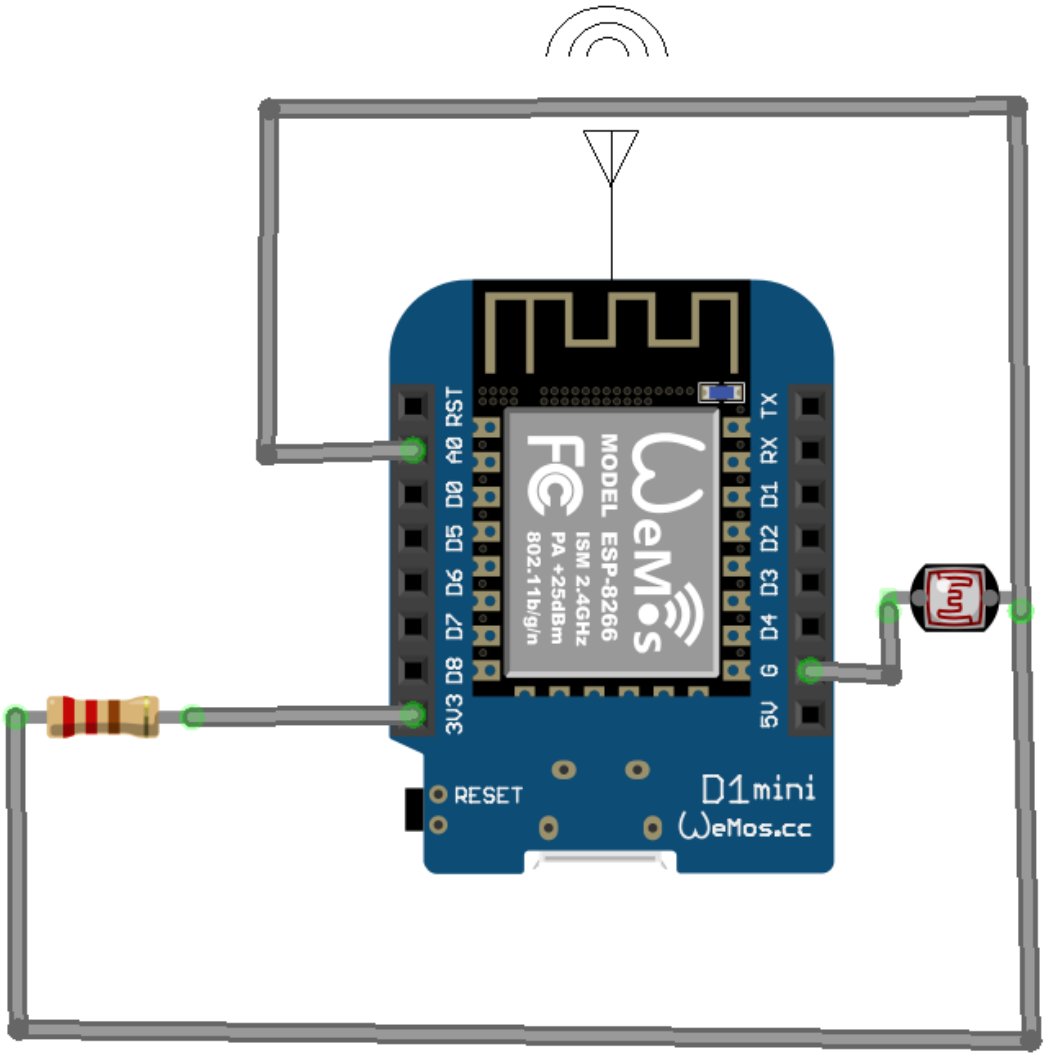


Рис. Б.3. Подключение фоторезистора к модулю WeMOS

Схема подключения датчика BMP180 к Arduino

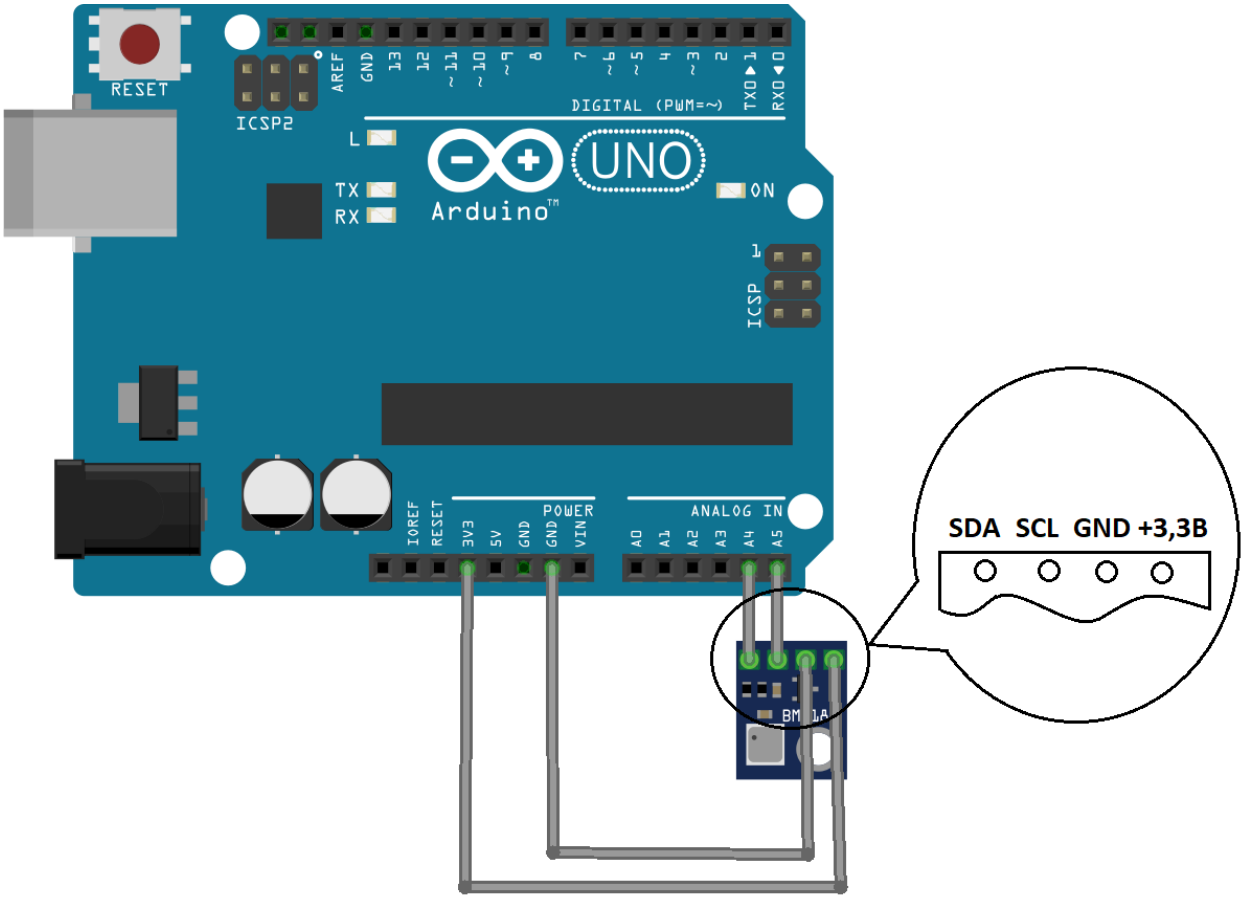


Рис. Б. 4. Подключение датчика BMP180 к Arduino Uno или OTA WeMOS D1

Схема подключения фотоприемника TSOP к Raspberry Pi

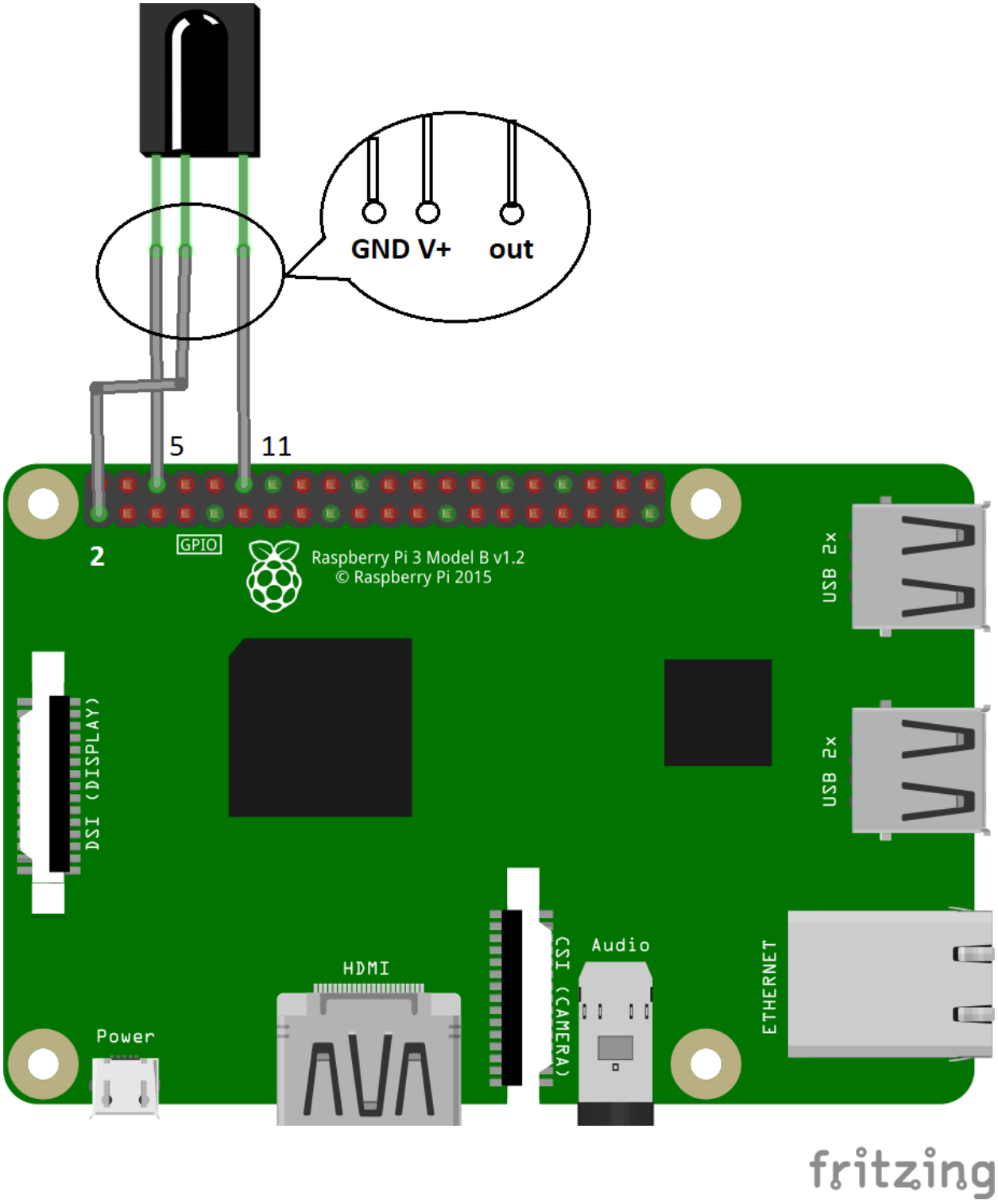


Рис. Б. 5. Подключение фотоприемника для работы с программой lirc

Добавление светодиода для излучения ИК-кода

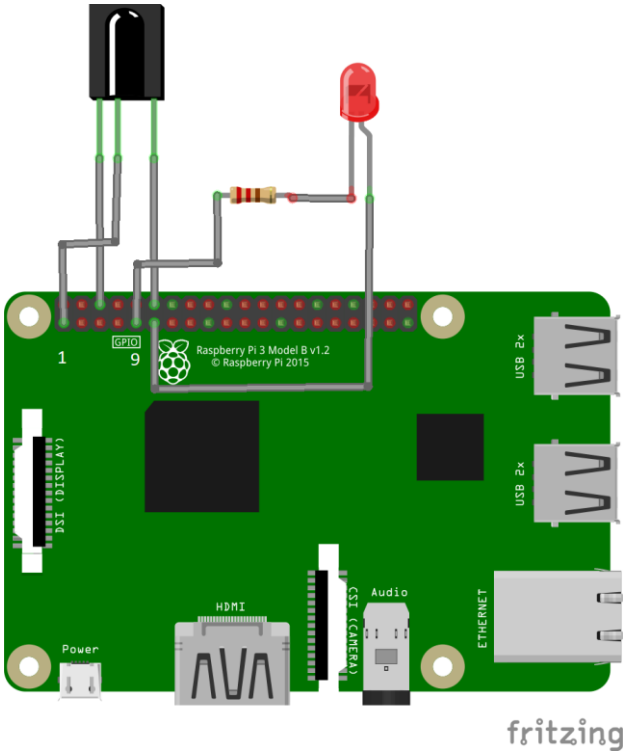


Рис. Б. 6. Светодиод для излучения ИК-кода

Использованная литература и Интернет-ресурсы:
1. В.Н. Гололобов «Raspberry Pi для любознательных», изд-во «Наука и техника», СПб, 2019г.
2. В.Н. Гололобов «Arduino для любознательных», изд-во «Наука и техника», СПб, 2017, ISBN 978-5-94387-879-4.
3. Образ диска MajorDoMo: https://connect.smartliving.ru/tasks/20.html
4. Что такое «Умный дом»: https://kb.smartliving.ru/chto-takoe-umnyi-dom-i-ego-struktura/
5. Arduino без Ethernet: https://kb.smartliving.ru/chema-ispolzovaniya-kontrollera-arduino/
6. Операции по добавлению объекта для работы с Arduino в Majordomo: http://majordomo.smartliving.ru/forum/viewtopic.php?f=8&t=498&start=10
7. Подключение флэшки: http://dmitrysnotes.ru/raspberry-pi-3-pravilnoe-montirovanie-usb-hdd-i-fleshek
8. Управление Arduino на языке PHP: http://www.useto.ru/index.php/project/252-arduino-linux-php-usb
9. Подключение WeMOS D1: https://diyprojects.io/esp8266-web-client-tcp-ip-communication-examples-esp8266wifi-esp866httpclient/#.W8M18fJuJzM
10. Статья про ESP8266: https://istarik.ru/blog/esp8266/28.html
11. Программа SocketTest: https://github.com/akshath/SocketTest
12. Статья про прошивку ESP8266: https://radioprog.ru/post/219
13. Библиотека для BMP180: https://github.com/adafruit/Adafruit-BMP085-Library
14. Статья про USB-камеру: http://academicfox.com/raspberry-pi-usb-web-kamera-potokovoe-vydeo-strym/