Описание работы сервера устройств на основе Ардуино

Сервер может использоваться для управления устройствами умного дома и т.п.

Сервер состоит из:

1. Платы Ардуино Мега, Уно, Мини или Нано;

2. Wi-Fi адаптера для подключения к домашней сети;

3. GSM модуля для подключения к сотовой сети;

4. Платы питания конвертирующей 12 вольт в 5В и в 3,3В;

5. Платы конвертера логических уровней из 3.3В в 5.5В для управления GSM модулем;

6. Блока питания с питанием от 220В с выходным напряжение 12В.

Взаимодействие с сервером идет через протокол HTTP. Первоначально на сервер устройств отправляется карта портов, которая описывает, что именно подключено к каждому порту Ардуино. В соответствие с картой управляющая программа будет управлять устройствами, подключенными к портам.

Получив карту портов, сервер настраивает порты на работу по соответствующим протоколам связи.

Карта портов может отправляться не целиком для всех портов, а непосредственно вместе с командой управления конкретным портом или с командой получения состояния порта. Это позволит не хранить карту портов и позволит легко переподключать устройства на разные порты ардуины.

На сервер могут быть отправлены команда:

- [запроса состояния всех портов],

- запрос состояние конкретного порта,

- команда установки значения порта вывода в соответствующее состояние.

При запросе состояния портов сервер собирает информация со всех подключенных устройств (по протоколам, описанным в карте портов) и отправляет ее на HTTP-клиент.

При приходе команды установки порта в нужное состояние

Порт сервера может состоять как из одного, так и из нескольких физических портом Ардуино.

Например, порт сервера для подключения драйвера управления шаговым двигателем (порт линейного сервопривода) состоит из следующих физических портов Ардуино:

1. Порт ШИМ (step),

2. Порт DIR

3. Порт левого концевого датчика

4. Порт правого концевого датчика

Таким образом порт управления двигателем занимает 4 физических порта Ардуины.

Порт управления ультразвуковым датчиком расстояния состоит из портов необходимых для работы протокола I2C, по которому идет управления датчиком расстояния.

Порт управления реле состоит только из одного порта ардуины

Порт концевого датчика состоит из одного порта ардуины.

Порт переменного резистора состоит из одного аналогового порта.

Таким образом порт сервера не тождественен порту ардуины и управляется более сложно на программном уровне.

Карта портов сервера описывает какие именно порты ардуины входят в состав порта сервера.

Сервер устройств имеет следующие типы портов:

1. цифровой порт, работающий на ввод, состоит из одного порта цифрового порта ардуины, используется, например, для:

- получения информации о состоянии концевого датчика;

- получения информации о состоянии датчика препятствия;

- получения информации о состоянии датчика уровня воды;

- получения информации о состоянии датчика потока газа/жидкости.

2. цифровой порт, работающий на вывод, состоит из одного порта цифрового порта ардуины используется, например, для:

- управления электромагнитным или электронным реле;

- управления светодиодной индикацией из одного диода;

- управления устройствами, управляемыми логическими сигналами.

3. аналоговый порт, работающий на ввод, состоит из одного порта аналогового порта ардуины, используется, например, для:

- получения информации о состоянии переменного резистора;

- получения информации о состоянии терморезистора;

- получения информации о состоянии датчика температуры;

- получения информации о состоянии датчика расстояния;

- получения информации о состоянии фоторезистора;

- получения информации о состоянии датчика давления.

4. I2C-порт, состоит из двух аналоговых портов ардуины, используется для управления устройствами по протоколу I2C.

5. PSI-порт, состоит из трех цифровых портов ардуины, используется для управления устройствами по протоколу SPI.

6. 1-WIRE-порт, состоит из одного цифрового порта ардуины, используется для управления устройствами по протоколу 1-WIRE.

7. UART-порт, состоит из двух цифровых портов ардуины, используется для управления устройствами по протоколу UART.

8. Порт линейного сервопривода с шаговым, состоит из

1. Порт ШИМ (step), 1 цифровой порт ардуины работающий на вывод;

2. Порт DIR, 1 цифровой порт ардуины работающий на вывод;

3. Порт левого концевого датчика, 1 цифровой порт ардуины работающий на ввод;

4. Порт правого концевого датчика, 1 цифровой порт ардуины работающий на ввод;

9. Порт линейного сервопривода с коллекторным двигателем, состоит из

1. Порт START, 1 цифровой порт ардуины работающий на вывод;

2. Порт DIR, 1 цифровой порт ардуины работающий на вывод;

3. Порт левого концевого датчика, 1 цифровой порт ардуины работающий на ввод;

4. Порт правого концевого датчика, 1 цифровой порт ардуины работающий на ввод;

10. Порт углового сервопривода с коллекторным двигателем, состоит из

1. Порт START, 1 цифровой порт ардуины работающий на вывод;

2. Порт DIR, 1 цифровой порт ардуины работающий на вывод;

3. Порт угла поворота, 1 аналоговый порт ардуины работающий на ввод;

11. Порт углового сервопривода с шаговым двигателем, состоит из

1. Порт ШИМ (step), 1 цифровой порт ардуины работающий на вывод;

2. Порт DIR, 1 цифровой порт ардуины работающий на вывод;

3. Порт угла поворота, 1 аналоговый порт ардуины работающий на ввод.

12. Порт сервопривода шагового двигателя, с временной задержкой включения, состоит из

1. Порт ШИМ (step), 1 цифровой порт ардуины работающий на вывод;

2. Порт DIR, 1 цифровой порт ардуины работающий на вывод.

13. Порт сервопривода коллекторного двигателя, с временной задержкой включения, состоит из

1. Порт START, 1 цифровой порт ардуины работающий на вывод;

2. Порт DIR, 1 цифровой порт ардуины работающий на вывод.

14. Порт сервопривода коллекторного двигателя, с релейным управлением, состоит из

1. Порт ВПРАВО, 1 цифровой порт ардуины работающий на вывод;

2. Порт ВЛЕВО, 1 цифровой порт ардуины работающий на вывод.

3. Порт левого концевого датчика, 1 цифровой порт ардуины работающий на ввод;

4. Порт правого концевого датчика, 1 цифровой порт ардуины работающий на ввод;

15. Порт управления SGM модулем, состоит из двух цифровых портов ардуины.

16. Порт ШИМ, состоит из одного цифрового порта ардуины, используется для управления устройствами с помощью ШИМ.

Порт для управления Wi-Fi передатчиком отдельно не задается, а имеет жестко установленные порты ардуины: 1 и 2 цифровые порты.

XML-Структура, описывающая порт выглядит следующим образом:

Карта портов

Тип порта (номера от 1 до 15)

Порядковый номер (от 1 до максимально возможного для этого типа порта)

Назначение порта=номер порта ардуины

Назначение порта=номер порта ардуины

Назначение порта=номер порта ардуины

Порядковый номер

Тип порта

Конец карты портов

Например, структура для порта сервопривода и аналогового порта ввода выглядит так:

Карта портов

Тип порта 8

Порядковый номер 1

STEP = 3

DIR = 4

LEFT\_END=5

RIGHT\_END=6

Порядковый номер 1

Порядковый номер 2

STEP = 3

DIR = 4

LEFT\_END=5

RIGHT\_END=6

Порядковый номер 2

Тип порта 8

Тип порта 3

PORT1=2

PORT2=4

Тип порта 3

Конец карты портов

Указанная карта портов дана лишь для ознакомления, на самом деле, каждый порт настраивается отдельно, отдельным GET-запросом.

Обмен информацией между сервером устройств и HTTP-клиентом (браузером или веб-сервером) идет через GET-запросы протокола HTTP.

**GET-запросы:**

Запрос на настройку порта сервера

?SETUP\_PORT\_TYPE=тип порта, параметры порта,…

Например:

?SETUP\_PORT\_TYPE=8, ORDER\_NUMBER=2, STEP=3, DIR=4, LEFT\_END=5, RIGHT\_END=6

?SETUP\_PORT\_TYPE=3, ORDER\_NUMBER=2, PORT=4

Запрос на получение состояния порта

?GET\_PORT\_TYPE=8, ORDER\_NUMBER=2

Запрос на изменение состояния порта вывода

?SET\_PORT\_TYPE=2, ORDER\_NUMBER=1, VALUE=TRUE

?SET\_PORT\_TYPE=2, ORDER\_NUMBER=1, VALUE=FALSE

**HTTP команды управления устройствами сервера устройств**

**Управления одним низковольтным реле и т.п. устройствами мокрым контактом**

Команда:

?PORT=WET\_CONTACT, PIN=[10], VALUE=1|0

Реле № - PIN=№

1. - 31

2. - 28

3. - 29

4. - 26

5. - 27

6. - 24

7. - 25

8. – 22

Управления высоковольтными реле

PIN=23

**Управления запоминающими низковольтными реле и т.п. устройствами мокрым контактом**

Для управления некоторыми устройствами необходимо подать импульс тока строго определенной длины, например, для управления запоминающими реле.

При получении команды пин контроллера PIN устанавливается в состояние VALUE и через время TIME переводится в инверсное состояние.

Команда:

?PORT=WET\_TEMP, PIN=[10], VALUE=1|0, TIME=1000, INFO\_MODE=1

**Управления множеством низковольтных реле и т.п. устройствами мокрым контактом одной командой с логическими операциями над текущим состоянием портов контроллера**

Используется для управления несколькими устройствами одной командой. При получении команды к значениям указанных цифровых пинов контроллера применяется указанная логическая операция, после чего получившееся значение снова записывается в порт контроллера. Команда для конкретного пина состоит из 4х символов. Первые два – это номер пина, с незначащим нулем спереди при необходимости. Третий символ (строчная буква) – это тип булевой операции, которая должна применяться к текущему значению пина (x-XOR, o-OR, a-AND). Четвертый символ – это двоичная цифра, использующаяся как второй аргумент при выполнении логической операции.

Тетрады команд объединяются в одну строку без разделителей и передаются в параметре PINS.

Команда:

?PORT=WET\_MULTI, PINS=**03a1**24x0**25o1**08a1

**Чтение состояния сухих контактов**

Команда:

?PORT=DRY\_CONTACT, PIN=[10] , PULLUP=[1|0]

Кнопка (желтый и оранжевый провод), PIN=51

Кнопка (серый и фиолетовый провод), PIN=50

Кнопка (красный и коричневый провод), PIN=49

Кнопка (зеленый и синий провод), PIN=48

Датчик наличия воды в баке, PIN=10

**Чтение аналогового порта контролера**

на МЕГА аналоговые пины имеют номера 54-69

(использование псевдонимов вида А1, А2..., запрещено!).

Команда:

?PORT=ANALOG\_INPUT, PIN=[10], READ\_COUNT=[5]

**Чтение датчика уровня воды 40 мм (красный датчик)**

Команда:

?PORT=WATER\_SENSOR\_40, PIN=[10]

Первый датчик PIN=54 (A0)

Второй датчик PIN=55 (A1)

**Чтение датчика температуры TMP36**

Команда:

?PORT=TMP36\_SENSOR, PIN=[10]

Первый датчик PIN=56 (A2)

Второй датчик PIN=57 (A3)

**Чтение датчика температуры и влажности DHT22**

Команда:

?PORT=DHT\_MODULE, VALUE=[11], PIN=[10]

Первый датчик (зеленый провод) PIN=11, VALUE=11

Второй датчик (красный провод) PIN=12, VALUE=11

**Чтение ультразвукового датчика расстояния HC\_SR04**

Команда:

?PORT=HC\_SR04, ECHO\_PIN=[11], TRIG\_PIN=[10]

Первый датчик

(фиолетовый провод) ECHO\_PIN =18

(синий провод) TRIG \_PIN =19

Второй датчик

(зеленый провод) ECHO\_PIN =20

(желтый провод) TRIG \_PIN =21

**Управление сервоприводом MG90**

Команда:

?PORT=MG90, PIN=[10], MIN\_VALUE=[0...180], MAX\_VALUE=[0...180], SPEED=[0...65535], TIME=[0...65535], INFO\_MODE=[0|1]

Первый сервопривод

(белый провод) PIN =7

Второй сервопривод

(синий провод) PIN =8

Третий сервопривод

(коричневый провод) PIN =9

**Управление драйвером двигателя постоянного тока с помощью ШИМ.**

Управление двигателем постоянного тока через Н-мост осуществляется с использованием двух программных портов, один из которых выключается (скорость ШИМ=0), при смене направления вращения двигателя.

К H-мосту можно подключать любые устройства, а не только двигатели.

Вращение влево (всегда сначала следует выключать предыдущее вращение т.е.):

Команда 1 (направление): ?PORT=PWM\_OUTPUT, PIN=5, VALUE=0

Команда 2 (скорость): ?PORT=PWM\_OUTPUT, PIN=6, VALUE=100

Вращение вправо (всегда сначала следует выключать предыдущее вращение т.е.):

Команда 1 (скорость): ?PORT=PWM\_OUTPUT, PIN=6, VALUE=0

Команда 2 (направление): ?PORT=PWM\_OUTPUT, PIN=5, VALUE=100

**Управление линейным сервоприводом на основе шагового двигателя с двумя концевиками**

**(управление идет через драйвер А4988 или аналогичный)**

Команда:

?PORT=STEP\_LINE\_SERVO, STEP\_PIN=44, DIR\_PIN=33, OFF\_PIN=53, LEFT\_CONTACT=51, RIGHT\_CONTACT=48, ENCODER=500, FAST\_SPEED=2000, INFO\_MODE=1, LOW\_SPEED=1000, WAIT\_TIME=1, CONTACT\_STATUS=LEFT

Команда:

?PORT=STEP\_LINE\_SERVO, STEP\_PIN=46, DIR\_PIN=30, OFF\_PIN=52, LEFT\_CONTACT=51, RIGHT\_CONTACT=48, ENCODER=500, FAST\_SPEED=2000, INFO\_MODE=1, LOW\_SPEED=1000, WAIT\_TIME=1, CONTACT\_STATUS=LEFT

**Управление сервоприводом с переменным резистором (потенциометром)**

Включение двигателя привода выполняется путем подачи единицы на указанные пины (т.е. через реле или драйвер двигателя). Переменный резистор состояния (угла поворота или координаты) привода подключается к аналоговому порту. На МЕГА аналоговые пины имеют номера 54-69 (использование псевдонимов вида А1, А2..., запрещено!).

При получении команды состояние привода приводится в соответствующее с командой состояние, если текущее состояние не укладывается в дельту.

Состояние резистора может быть указано в диапазоне 0 – 1023 (т.к. используется резистор 10КОм)

Команда:

?PORT=RESISTOR\_SERVO, MIN\_PIN=51, MAX\_PIN =48, RESISTOR\_PIN =54, RESISTOR=500, DELTA=30, INFO\_MODE=1, WORK\_TIME=10

**! Для справки:**

*Если после получения команды состояние привода укладывается в дельту, то никаких действий выполнено не будет. Если состояние не укладывается в дельту, то привод будет переведен в состояние, максимально точно соответствующее состоянию, переданному в команде, без учета дельты.*

**Чтение микроволнового датчика движения RCWL-0516**

Датчик при срабатывании выдает логическую единицу, поэтому для ее обнаружения нужно создать программный порт чтения сухого контакта, без подтягивающего резистора (т.к. встроенный подтягивающий резистор подтягивает порт только к питанию, что не позволит зарегистрировать пришедшую логическую единицу с датчика).

Команда:

?PORT=DRY\_CONTACT, PIN=2 , PULLUP=0

**Запись и чтение данных из EEPROM микроконтроллера**

Данные для записи в EEPROM передаются в виде строки символов, разделенных символом-разделителем «;». Память может использоваться для хранения состояний объектов управления или объектов графического интерфейса пользователя. Так как память имеет ресурс 100 000 записей (без ограничения чтения), следует передавать строку данных с минимальными изменениями и с постоянной длиной разделенных данных, например так: «100;020;003;255;101;001;010». При таком подходе в память будут перезаписываться только измененные символы, а не вся строка, что увеличит ресурс работы ячеек памяти. Длина строки данных не должна быть больше 255 символов.

Команды:

?PORT=READ\_EEPROM, ADDRESS=2, LENGHT=255

?PORT=WRITE\_EEPROM, ADDRESS=2, VALUE=100;020;003;255;101;001;010

**Запись и чтение данных из RAM микроконтроллера**

RAM может использоваться для хранения данных любого типа, но в настоящем модуле чтение и запись идет в строковую переменную типа String в которой находятся данные, например в формате JSON, но не обязательно.

Память может использоваться для хранения быстро меняющихся состояний объектов управления или объектов графического интерфейса пользователя (т.е. переменных).

Команды:

?PORT=READ\_RAM, ADDRESS=[2]

?PORT=WRITE\_RAM, DATA="{100,1000,200,300}", ADDRESS=[2]

**Чтение данных о состоянии портов контроллера**

Так как выяснить текущее состояние портов микроконтроллера сложно все манипуляции с портами сохраняются в глобальном массиве PINS\_STATUS\_ARRAY, который может быть запрошен и отослан по протоколу HTTP по команде PINS\_STATUS.

Глобальный массив состояний отражает состояние следующих параметров порта:

- бит №1: способ управления портом (сторонняя библиотека[0]/внутренняя функция сервера устройств[1]),

- бит №2: режим работы порта (ввод[0]/вывод[1]),

- бит №3: состояние встроенного подтягивающего резистора (не подтянут к питанию[0]/подтянут к питанию[1]),

- бит №4: уровень напряжения на порте (низкий[0]/высокий[1]) !!! уровень напряжения можно читать не зависимо от того в каком режиме (ввод или вывод) работает порт

за состояние каждого параметра отвечает отдельный бит, т.о. состояние всего порта описывается четырьмя битами, поэтому для их хранения достаточно одного байта на порт

! состояние пинов, которыми управляют сторонние библиотеки, в массиве не отражаются, т.к. выяснить их состояние и режим работы не возможно,

! поэтому изначально элементы массива содержат 0 в первом бите (справа), который говорит о том, что данный пин управляется сторонней библиотекой и его состояние не известно

! если пин управляется функцией сервера устройств, то соответствующий элемент массива содержит трехзначное двоичное число, отражающее состояние параметров порта

Массив может использоваться как массив состояний объектов графического интерфейса пользователя, состояние которых отражает состояние портов контроллера.

Чтение данных из PINS\_STATUS\_ARRAY, чтение массива выполняется от элемента массива с присланным в параметре FROM номером до элемента массива с присланным в параметре TO адресом (нумерация элементов массива идет от 0!)

параметры URL должны выглядеть так: ?PORT=PINS\_STATUS, FROM=[2], TO=[10]

ответ клиенту: если FROM и ТО заданы верно возвращаем прочитанную часть массива как строку, иначе ADDRESS\_ERROR (если адрес указан неверно) или ADDRESS\_NOT\_FOUND\_ERROR (если параметр FROM или ТО не указан)

Команды:

?PORT=PINS\_STATUS, FROM=[2], TO=[10]

**Чтение цифровых датчиков температуры DS18B20, находящихся на шине 1-wire**

Ответ клиенту: если PIN задан верно - значение температуры в градусах Цельсия всех датчиков на шине, разделенные символом "|", иначе PIN\_NUMBER\_ERROR

Команда:

?PORT= DS18B20, PIN=[11]

**Чтение датчика атмосферного давления и температуры BMP080/BMP180, протокол связи I2C (используется аппаратная реализация шины I2C, пины 20-SDA и 21-SCL, на шине может быть только один датчик)**

Ответ клиенту: - значение температуры С и давления Па в формате JSON

Команда:

?PORT=BMP180

**! для справки:**

*Библиотеки iarduino могут использовать как аппаратную, так и программную реализацию шины I2C. При этом выбранный тип реализации шины I2C не влияет на синтаксис функций и методов библиотек iarduino. При работе с аппаратной шиной I2C, датчики подключаются к заранее указанным на плате (или в документации) выводам, так как прием/передача данных по шине осуществляется на физическом уровне (внутренними блоками микроконтроллера).*

*При работе с программной шиной I2C, датчики можно подключать к любым выводам.*

*Однако номера пинов нужно определять до подключения библиотеки, поэтому пины нужно определять два раза, до подключения библиотеки, и при создании программного порта.*

*На одной шине может быть только один датчик.*

*Так что в библиотеке iarduino не предусмотрено указание адреса.*

*Библиотеку можно использовать со стандартной библиотекой I2C для аппаратного порта, поэтому можно создать один аппаратный и один программный порт I2C*

**Чтение датчика освещенности ВН1750 (модули GY-30, GY302 и т.п.), протокол связи I2C (используется аппаратная реализация шины I2C, пины 20-SDA и 21-SCL, на шине может быть только два датчика т.к. в датчик аппаратно записаны только адреса 0x5C и 0x23)**

Для смены адреса у датчика есть пин, который можно замыкать с питанием или землей.

Ответ клиенту: - значение освещенности в люксах в десятичном формате #.##

Команда:

?PORT= ВН1750, ADDRESS=[0x23]

**Чтение лазерного датчика расстояния VL53L0X, протокол связи I2C (используется аппаратная реализация шины I2C, пины 20-SDA и 21-SCL, на шине может быть только один датчик т.к. в датчик аппаратно записан адрес 0x52)**

Датчик работает в режиме максимальной дальности и максимальной точности.

Ответ клиенту: - расстояния до объекта в миллиметрах, на который направлен датчик (максимальное 2 метра, минимальное 30 мм)

Команда:

?PORT= VL53L0X

**! для справки:**

*Адрес датчика все же может быть изменен, но для этого требуется задействовать цифровые пины контроллера для выбора конкретного датчика для изменения адреса.*

**Дозатор жидкости (и других субстанций: сыпучих и летучих) или предметов (деталей на конвейере) на основе расходомера (типа yf-s401) и электронно-управляемого клапана (задвижки) с управлением одним цифровым портом. РЕАЛИЗОВАНО, НО НЕ ПРОВЕРЕНО!**

Команда:

?PORT=DISPENSER, PIN\_ON=[20], PIN\_COUNT=[2], PULSE=[10000], DIV=[10], WAIT\_TIME=[10000], INFO\_MODE=1

Где:

PIN\_ON – номер пина для управления клапаном

PIN\_COUNT – номер пина, на который приходят импульсы со счетчика (счетчика субстанций или предметов).

PULSE – количество импульсов, которое должно прейти для отключения клапана.

DIV – делитель количество отсчитанных импульсов, используется когда количество импульсов должно быть очень большим и точность не имеет большого значения.

WAIT\_TIME – время (в миллисекундах) в течение которого ожидается очередной импульс от счетчика, если импульс не получен в течение указанного времени, выполнение команды прекращается, пин PIN\_ON переводится в состояние «0».

INFO\_MODE - режим работы, блокирующий (ответ на сервер не отправляется пока команда не выполниться), не блокирующий (ответ на сервер отправляется сразу как будут верно распознаны все параметры и их значения)

**Отправка команды на радиоприемник через радиопередатчик.**

В качестве радиопередатчика могут использоваться передатчики, аналогичные WL102-341, и приемники аналогичные WL101-341. Следует помнить, что передатчик питается от 3,6 вольта и управляется логикой на 3.3. вольта, приемник питается от 5 вольт и управляется логикой на 5 вольт! **НЕ РЕАЛИЗОВАНО!**

!!! При получении данных от передатчика данные отправляются на разбор URL строки, т.о. передатчик приемник притворяется HTTP протоколом. Контроллер должен обрабатывать команды с Wi-Fi приемника и с передатчика одинаково. Это позволил легко отправлять команды на разные контроллеры, удаленные и находящиеся в разных Wi-Fi сетях, но имеющие кнопки ГИП с одинаковыми идентификаторами. Это позволит создавать клоны кнопок ШИП на разных контроллерах.

**Кнопка с обратной связью**

Механическая кнопка фиксируется вручную, но отключается сигналом с контроллера, после считывания контроллером состояния нажатия кнопки.

**НЕ РЕАЛИЗОВАНО!**

Команда:

?PORT=ARMBUTTON, ONPIN=[50], RESETPIN=[51], WAIT=500

Где:

ONPIN-порт контролера, по которому контроллер фиксирует состояние включения кнопки

RESETPIN – порт контроллера, на который контроллер выдает сигнал сброса (отжатия) кнопки, и держит его пока кнопка не отожмется или пока не кончится время таймаута отжатия в миллисекундах

WAIT – время по истечение которого контроллер пытается отжать кнопку сигналом на порт RESETPIN, по истечении которого контроллера перестает пытаться отжать кнопку.

Настоящий программный порт создается специальным роботом (воркером) на устройстве управления и выполняет создание порта как можно чаще, для оперативного опроса механической кнопки. Кнопка может быть выполнена в виде модуля сенсорной кнопки.

**Создание программного порта для запроса данных о Wi-Fi сети, к которой подключен контроллер устройств.**

Порт используется только для получения информации и никаких управляющих действий не выполняет

Команда:

?PORT=NETINFO, NAME=["Vovan"], TIME=[5000]

Где:

NAME – имя Wi-Fi -сети

TIME - времени ожидания ответа от Wi-Fi модуля (не может превышать 40 000 миллисекунд т.к. сработает таймер перезагрузки).

**! для справки:**

возврат данных выполняется в формате: AT+CWLAP:<шифрование>,<ssid>,<rssi>,<mac>,<channal>

где:

<шифрование>: 0 — без шифрования, 1 — WEP, 2 — WPA\_PSK, 3 — WPA2\_PSK, 4 — WPA\_WPA2\_PSK

<ssid> : SSID точки доступа

<rssi> : Сила сигнала

<mac> : MAC-адрес

<channal> : канал

параметры URL должны выглядеть так: ?PORT=NETINFO, NAME=["Vovan"], TIME=[5000]

*ответ клиенту:*

если параметр NAME задан верно - AT+CWLAP:<шифрование>,<ssid>,<rssi>,<mac>,<channal> , иначе NAME\_NOT\_FOUND\_ERROR, TIME\_NOT\_FOUND\_ERROR или TIME\_ERROR

! данные отправляются в том виде, в котором их прислал Wi-Fi модуль без парсинга и обработки

!!! функция полностью блокирует работу контроллера на время таймаута (TIME мс), в течение которого будет ожидаться ответ Wi-Fi модуля

!!! поэтому, частый вызов этой функции сильно затормозит работу контроллера устройств, вызывать функцию чаще 1 раза в 5 минут не рекомендуется!

!!! практика показывает, что Wi-Fi модулю нужно не менее 5000 миллисекунд, чтобы сформировать ответ!