

Протокол взаимодействия вычислительных устройств через каналы передачи текстовой информации, организованные по принципу точка-точка.

Автор: Беспалов Денис (master1312@yandex.ru)

Оглавление

<u>Общее описание.....</u>	<u>3</u>
<u>Описание протокола.....</u>	<u>3</u>
<u>Базовые типы сообщений.....</u>	<u>3</u>
<u>Зарезервированные названия команд.....</u>	<u>4</u>
<u>Формат описания датчиков (xml или json).....</u>	<u>5</u>
<u>XML.....</u>	<u>5</u>
<u>Схема.....</u>	<u>5</u>
<u>Пример.....</u>	<u>6</u>
<u>JSON.....</u>	<u>6</u>
<u>Схема.....</u>	<u>6</u>
<u>Пример.....</u>	<u>6</u>
<u>Форматы данных сенсоров.....</u>	<u>7</u>
<u>Формат описания интерфейса.....</u>	<u>7</u>
<u>XML.....</u>	<u>7</u>
<u>Схема.....</u>	<u>7</u>
<u>Пример.....</u>	<u>9</u>
<u>JSON.....</u>	<u>9</u>
<u>Схема.....</u>	<u>9</u>
<u>Пример.....</u>	<u>10</u>
<u>Типы параметров контроля и возможные ограничения.....</u>	<u>11</u>
<u>Состояние устройства.....</u>	<u>11</u>

Общее описание

Документ описывает упрощенный протокол взаимодействия вычислительных устройств, предназначенный главным образом для взаимодействия между устройствами со слабыми ресурсами между собой с использованием любых каналов связи, способных передавать текстовую информацию (СOM, TCP/IP и т. д.). В рамках протокола предполагается коммуникация между устройствами по принципу точка-точка, одно из устройств считается управляющим, другое управляемым, однако различия между ними не велики. Так же в рамках описания форматов сообщений указываются передающее и принимающее устройства (передающее устройство инициирует процесс обмена сообщениями, при этом им может быть как управляющее, так и управляемое устройство в зависимости от ситуации). Управляемое устройство может быть хабом, при этом оно объединяет несколько других устройств.

Описание протокола

1. Устройства взаимодействуют между собой путем двухстороннего обмена текстовыми сообщениями. Каждое сообщение — строка символов в кодировке UTF-8. Сообщения разделяются символом перевода строки с кодом 10 (\n).
2. Сообщения состоят из нескольких элементов, первый из которых называется заголовком, остальные аргументами. Элементы сообщения разделяются символом с кодом 124 «|».
Пример сообщения:
info|Argument 1|Argument 2|Argument 3\n
3. Сами элементы сообщений не должны содержать символов «\n» и «|».
4. Для выполнения ряда наиболее распространенных действий зарезервирован ряд заголовков, назначение которых описано ниже. Остальные заголовки могут использоваться в зависимости от необходимости.

Базовые сообщения

1. Информационное сообщение. Заголовок — **info**. Предназначено для передачи информации, не требующей специальной обработки (например, отладочная информация) и предназначенной для передачи ее человеку (разработчику или оператору). Сообщение не требует ответа от принимающего устройства. Аргументы сообщения — текстовые строки, содержащие информацию, предназначенную для прочтения человеком.
2. Сообщение готовности к работе. Заголовок — **ready**. Может использоваться устройством для уведомления другого устройства о готовности к работе, например, после окончания загрузки устройства. Аргументы отсутствуют.
3. Запрос идентификации устройства. Заголовок — **identify**. Аргументы отсутствуют. Принимающее устройство должно в течении 5 секунд вернуть ответное сообщение с заголовком **deviceinfo** со следующими аргументами:
 1. Уникальный идентификатор устройства в виде текстового шестнадцатеричного представления UUID "{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}" или "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx".
 2. Человеко-читаемое название.
4. Запрос идентификации устройства-хаба. Заголовок — **identify**. Аргументы отсутствуют. Принимающее устройство должно в течении 5 секунд вернуть ответное

сообщение с заголовком **deviceinfo** со следующими аргументами:

1. Специальный идентификатор "#hub".
 2. Уникальный идентификатор устройства в виде текстового шестнадцатеричного представления UUID "{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}" или "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx".
 3. Человеко-читаемое название.
5. Перечисление подключенных к хабу устройств. Заголовок — **identify_hub**. В ответ передаются сообщения **device_info** со следующими аргументами:
1. Уникальный идентификатор устройства в виде текстового шестнадцатеричного представления UUID "{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}" или "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx".
 2. Человеко-читаемое название.
6. Сообщение синхронизации. Заголовок — **sync**. Предназначено для уведомления принимающего устройства о том, что передающее устройство находится в рабочем состоянии. Используется для безопасного синхронного вызова функций, в остальное время возможно использование разработчиком для собственных целей. Аргументы отсутствуют.
7. Команда. Заголовок — **call**. Первый аргумент — название команды (любое кроме зарезервированных, описанных в разделе ["Зарезервированные названия команд"](#)). Остальные аргументы сообщения — аргументы команды. В процессе работы может передаваться регулярное сообщение **sync** (рекомендуется для нестабильных каналов связи). Принимающее устройство по окончании выполнения должно вернуть сообщение с заголовком **ok** в случае успешного завершения (аргументы содержат возвращаемые значения) или **err** в случае неудачи (аргументы на усмотрение разработчика).
8. Измерение. Заголовок — **meas**. Аргументы сообщения (в порядке следования):
- 1 — название датчика.
 - 2 и далее — значение. (см. ["Форматы данных сенсоров"](#))
9. Сообщение об изменении состояния устройства. Сообщение — **statechanged**. Передается устройством, когда происходит изменение его состояния. Аргументы:
1. Название команды или "#" (см. раздел ["Состояние устройства"](#)).
 2. Номер аргумента команды (начиная с 1) или название дополнительного параметра устройства.
 3. Новое значение.

Отправка сообщений устройству, находящемуся за хабом, происходит аналогично отправке сообщений самому устройству, но к каждому сообщению добавляется в начало два аргумента: специальный идентификатор "#hub" и идентификатор целевого устройства в виде текстового шестнадцатеричного представления UUID "{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}". Аналогичным образом приходят ответные сообщения.

Зарезервированные названия команд

Название	Параметры	Описание и возвращаемые значения
#sensors	отсутствуют	Запрос списка датчиков. Возвращаемое значение — список датчиков (формат описан ниже в разделе "Формат описания датчиков")
#controls	отсутствуют	Запрос описания интерфейса управления. Возвращаемое значение — описание интерфейса управления (формат описан

		ниже в разделе "Формат описания интерфейса")
#state	отсутствуют	Запрос состояния устройства (см. раздел "Состояние устройства")

При расширении списка зарезервированных команд в дальнейшем их названию будут начинаться с "#".

Формат описания датчиков (xml или json)

XML

Схема

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="ru">
      Sensors definition schema for ArduinoRpc library.
    </xs:documentation>
  </xs:annotation>
  <xs:element name="sensors">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="sensor" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="constraints" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:anyAttribute namespace="##local" processContents="lax"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string"/>
            <xs:attribute name="type">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="single"/>
                  <xs:enumeration value="single_lt"/>
                  <xs:enumeration value="single_gt"/>
                  <xs:enumeration value="text"/>
                  <xs:enumeration value="packet"/>
                  <xs:enumeration value="packet_lt"/>
                  <xs:enumeration value="packet_gt"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример

```
<sensors>
  <sensor name="$name" type="$type">
    <constraints key1="$value1" key2="$value2"/>
  </sensor>
  <sensor><!--sensor definition--></sensor>
</sensors>
```

JSON

Схема

```
{ "$schema": "http://json-schema.org/draft-04/schema#", "type": "object", "properties": { "sensors":
{ "type": "array", "items": { "type": "object", "properties": { "name": { "type": "string"}, "type":
{ "type": "string", "enum":
[ "single", "single_lt", "single_gt", "text", "packet", "packet_lt", "packet_gt" ] }, "constraints":
{ "type": "object", "properties": { "*": { "type": "string" } } }, "required": [ "name", "type" ] } }, "required":
[ "sensors" ] }
```

Пример

```
{ "sensors" : [
  {
    "name" : "$name",
    "type" : "$type",
    "constraints" :
    {
      "$key1" : "$value1",
      "$key2" : "$value2"
    }
  },
  {
    //sensor definition
  }
]}
```

Имя сенсора (\$name) уникально в пределах устройства. Атрибут сенсора "type" описывает формат передаваемого значения в сообщении. Форматы передаваемых значений описаны ниже в разделе [Форматы данных сенсоров](#).

Форматы данных сенсоров

В различных форматах могут использоваться метки локального и глобального времени. Локальное время — время в миллисекундах, измеряемое на устройстве от какой-то фиксированной точки (но не известной). Например — от момента включения устройства. Глобальное время — количество миллисекунд, прошедшее с 01.01.1970г. Использование

глобального времени возможно при наличии на устройстве часов реального времени.

Каждый формат данных описывает структуру самих данных и способ передачи в сообщении "meas".

Тип сенсора	Описание
single	<p>Набор числовых значений с плавающей точкой, передающихся текстом, представляющих собой один многомерный отсчет. В сообщении каждое отдельное число передается как один аргумент сообщения. В описании сенсора в "constraints" может присутствовать параметр "dims", в котором указывается размерность значений (≥ 1, по-умолчанию 1).</p> <p>Пример — передача координаты (x,y,z)</p> <p>meas coords 12.0 16.3 67.9</p>
single_lt	<p>Метка локального времени и набор числовых значений с плавающей точкой, передающихся текстом, представляющих собой один многомерный отсчет. В остальном тип аналогичен типу "single".</p> <p>Пример — передача координаты (x,y,z)</p> <p>meas coords_lt 123456 12.0 16.3 67.9</p> <p>123456 — метка времени, далее 3 координаты</p>
single_gt	<p>Метка глобального времени и набор числовых значений с плавающей точкой, передающихся текстом, представляющих собой один многомерный отсчет. В остальном тип аналогичен типу "single".</p> <p>Пример — аналогично single_lt</p>
text	1 или более аргументов сообщения, содержащих текст
packet	<p>Массив одиночных значений типа "бинарный формат с плавающей запятой одинарной точности" по стандарту IEEE 754 (в языке C — тип float), закодированный в base64. В описании сенсора в "constraints" может присутствовать параметр "dims", в котором указывается размерность значений (≥ 1, по-умолчанию 1). В описании сенсора в "constraints" может присутствовать параметр "fixed_size", в котором указывается количество значений в пакете. При этом "количество элементов в пакете" = "количество значений" * "размерность". В противном случае размер пакета может меняться в процессе работы устройства, однако количество элементов в массиве должно быть кратно размерности значений. При этом "количество значений" = "количество элементов" / "размерность".</p> <p>Пример — передача координат (x,y,z)</p> <p>meas coords AABAQWZmgkHNzIdCAABQQc3MNEHNzKxB</p> <p>2 значения по 3 координаты (12.0;16.3;67.9), (13.0;11.3;21.6)</p>

packet_lt	<p>Метка локального времени и массив одиночных значений типа "бинарный формат с плавающей запятой одинарной точности" по стандарту IEEE 754 (в языке C — тип float), закодированный в base64. В остальном тип аналогичен типу "packet".</p> <p>Пример — передача координат (x,y,z)</p> <p>meas coords 123456 AABAQWZmgkHNzIdCAABQQc3MNEHNzKxB</p> <p>123456 — метка времени, далее 2 значения по 3 координаты (12.0;16.3;67.9), (13.0;11.3;21.6)</p>
packet_gt	<p>Метка глобального времени и массив одиночных значений типа "бинарный формат с плавающей запятой одинарной точности" по стандарту IEEE 754 (в языке C — тип float), закодированный в base64. В остальном тип аналогичен типу "packet".</p> <p>Пример — аналогично packet_lt</p>

Формат описания интерфейса

XML

Схема

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="ru">
      Controls definition schema for ArduinoRpc library.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleType name="LayoutType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="v"/>
      <xs:enumeration value="h"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BoolType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="0"/>
      <xs:enumeration value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="controls">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="group" minOccurs="0" maxOccurs="unbounded" type="GroupType"/>

```

```

</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="GroupType">
<xs:sequence>
<xs:element name="group" type="GroupType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="control" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="param" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="constraints" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:anyAttribute namespace="##local" processContents="lax"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="title" type="xs:string"/>
<xs:attribute name="type">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="checkbox"/>
<xs:enumeration value="text_edit"/>
<xs:enumeration value="select"/>
<xs:enumeration value="slider"/>
<xs:enumeration value="dial"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="title" type="xs:string"/>
<xs:attribute name="command" type="xs:string"/>
<xs:attribute name="layout" type="LayoutType"/>
<xs:attribute name="sync" type="BoolType"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="title" type="xs:string"/>
<xs:attribute name="layout" type="LayoutType"/>
</xs:complexType>
</xs:schema>

```

Пример

```
<controls>
  <group title="$title" layout="v|h">
    <group title="$title" layout="v|h">
      <control title="$title" command="$cmd" layout="v|h" sync="1|0">
        <param title="$title" type="$type">
          <constraints key1="$value1" key2="$value2"/>
        </param>
        <param><!--param definition--></param>
      </control>
      <control><!-- control definition--></control>
    </group>
    <control><!--command definition--></control>
    <group><!--group definition--></group>
  </group>
</controls>
```

JSON

Схема

```
{ "$schema": "http://json-schema.org/draft-04/schema#", "definitions": { "group": {
  "type": "object", "properties": { "element_type": { "type": "string", "enum": ["group"] }, "layout": {
    "type": "string", "enum": ["v", "h"] }, "title": { "type": "string", "minLength": 1 }, "elements": {
    "type": "array", "items": { "anyOf": [ { "$ref": "#/definitions/group" }, {
      "$ref": "#/definitions/control" } ] } }, "required": [
    "element_type", "title", "additionalProperties": false }, "control": { "type": "object", "properties": {
    "element_type": { "type": "string", "enum": ["control"] }, "layout": { "type": "string", "enum": [
    "v", "h"] }, "title": { "type": "string", "minLength": 1 }, "command": {
    "type": "string", "minLength": 1 }, "sync": { "type": "string", "enum": ["0", "1"] }, "params": {
    "type": "array", "items": { "type": "object", "properties": { "title": {
    "type": "string", "minLength": 1 }, "type": { "type": "string", "enum": [
    "checkbox", "text_edit", "select", "slider", "dial" ] }, "constraints": { "type": "object", "properties": {
    }, "additionalProperties": true } }, "required": [
    "title", "type" ], "additionalProperties": false } } }, "required": [
    "element_type", "title", "command" ], "additionalProperties": false } }, "type": "object", "properties": {
    "controls": { "$ref": "#/definitions/group" } }, "required": ["controls"], "additionalProperties": false }
```

Пример

```
{ "controls" : {
  "element_type" : "group",
  "title" : "$title",
  "layout" : "v|h",
  "elements" :
```

```

[
  {
    "element_type" : "group"
    //group definition
  },
  {
    "element_type" : "control",
    "title" : "$title",
    "layout" : "v|h",
    "command" : "$command",
    "sync": true|false
    "params" : [
      {
        "title" : "$title",
        "type" : "$type",
        "constraints" :
        {
          "$key1" : "$value1",
          "$key2" : "$value2"
        }
      }
    ]
  }
]
}
}
}

```

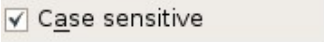
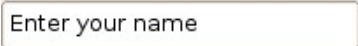
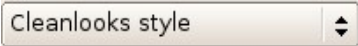
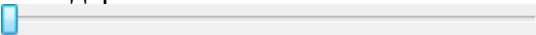

Группы предназначены для группировки контролов и дочерних групп. Размещение задается параметром "layout", принимающим значение "v" (по вертикали) или "h" (по горизонтали). Если не задано — по умолчанию по вертикали.

Команда (\$cmd) для контрола уникальна в пределах интерфейса устройства и фактически может являться идентификатором контрола.

Атрибут "sync" указывает, нужно ли в процессе команды передавать ежесекундное сообщение "sync" для синхронизации (0 — не нужно, 1 — нужно, 1 по-умолчанию). Рекомендуется передавать сообщение в случае, если канал связи между устройствами не надежен и не имеет собственных средств поддержки соединения.

Если параметры для контрола не заданы, отображается кнопка, при нажатии отправляется команда без параметров. Если задан один параметр, генерируется один элемент UI в зависимости от типа параметра, при взаимодействии с которым отправляется команда. Если параметров несколько, генерируется набор элементов UI согласно заданному размещению, соответствующих типам параметров, и кнопка, при нажатии на которую отправляется команда. Параметры команды добавляются в порядке следования в описании.

Типы параметров контроля и возможные ограничения

Тип параметра	Возможные ограничения	Элемент UI
checkbox	onValue — значение, передаваемое, когда чекбокс включен (по умолчанию "1") offValue — значение, передаваемое, когда чекбокс выключен (по умолчанию "0")	Чекбокс, checkable кнопка 
text_edit		Поле ввода (с кнопкой отправки справа, если параметр у контроля один) 
select	values — список значений, разделенных символом ";" (если отсутствует, всегда отправляется "0")	Выпадающий список 
slider	min — минимальное значение (по умолчанию 0) max — максимальное значение (по умолчанию 1023) step — величина шаг (по умолчанию 1) layout — вид слайдера ("h" — горизонтальный, "v" - вертикальный)	Слайдер 
dial	min — минимальное значение (по умолчанию 0) max — максимальное значение (по умолчанию 1023) step — величина шаг (по умолчанию 1)	"Крутилка" 

Состояние устройства

Состояние управляющего интерфейса устройства описывает текущие значения параметров команд устройства и значения дополнительных параметров, не связанных с интерфейсом управления. При изменении состояния устройство оповещает об этом подключенного клиента при помощи специального сообщения "statechanged". Аргументы сообщения сгруппированы по 3. Первым аргументом в группе идет команда, параметр которой изменился, или "#", если изменился дополнительный параметр, не связанный ни с какой командой. В первом случае вторым аргументом в группе идет номер аргумента команды (начиная с 1), во втором — название дополнительного параметра. Третий аргумент в группе — значение параметра. При запросе всего состояния устройства при помощи зарезервированной команды "#state" возвращается все состояние устройства в одном сообщении с заголовком "ok" и аргументами, сгруппированными по 3 по аналогии с

сообщением "statechanged".