Для возможности интеграции стороннего оборудования в робота с поддержкой ПО "Вектор" необходимо обеспечить следующий функционал:

Общение между ПО "Вектор" и "сервер управления внешними устройствами" (далее сервер) через TCP запросы.

Разработать шаблон сервера для быстрой адаптации нового оборудования в окружение роботизированных платформ.

Запрос формируется из обязательного первого ([0]) байта который содержит код команды, и последующие структуру запроса.

**Основные варианты команд:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Команды управления сервером [0х0\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

0х01 – Пинг сервера

0х02 – Перезагрузить сервер

0х03 – Debaug сообщений

0х0..

0х0F –

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Команды настройки сервера [0х1\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

0х11 – Авто Пинг оборудования

0х12 – Авто перезагрузка сервера

0х13 – Логирование

0х1..

0х1Е – Сохранить настройки

0х1F – Сброс на заводские настройки сервера (по умолчанию)

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Команды управления оборудованием [0х2\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

0х20 – Перевести в режим сна (выключить) оборудования

0х21 – Вывести из режима сна (включить или просто пинг, состояние) оборудования

0х22 – Перезагрузить оборудование

0х23 – Запуск (старт / возобновить)

0х24 – Пауза оборудования

0х25 – Остановить оборудование

0х26 – Запись данных в файл

0х27 – Получить значения оборудования

0х2..

0х2F – Авто калибровка оборудования

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Команды настройки оборудования [0х3\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

0х30 – Задать комплект параметров настройки оборудования

0х31 – Получить комплект параметров настройки оборудования

0х3..

0х3D – Обновить параметры (проброс всех доступных настроек из файла)

0х3Е – Сохранить настройки

0х3F – Сброс на заводские настройки оборудования (по умолчанию)

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Общие команды резерв [0х4\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Общие команды резерв [0х5\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Общие команды резерв [0х6\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\* Общие команды резерв [0х7\*] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

Резерв под уникальные запросы оборудования не попадающие под критерии "основных" команд:

0х80 .. 0хАF – управление

0хВ0 .. 0хDF – настройка

0хЕ0 .. 0хFF – не использовать

**Ответ от сервера:**

Первый байт ([0]) код ошибки. Основной перечень кодов ошибок с расшифровкой:

0х00 – Ошибок нет

0х01 – Нет такой команды

0х02 – Неверный размер принятой структуры запроса

0х03 – Оборудование не поддерживает эту команду

0х04 – Нет связи с оборудованием

0х05 – Оборудование настроено неправильно

0х06 – Ошибка имени файла

0х07 – Ошибка при создании/записи файлов

0х08 – Не удалось задать новые параметры

0х7Е – Оборудование ответило ошибкой

0х7F – Что-то не так "Unknown Error"

Резерв под специфичные ошибки оборудования не попадающие под критерии "основных" ошибок: 0х80 .. 0хEF

Не использовать: 0хF0 .. 0хFF – резерв

Второй байт ([1]) "эхо запроса" (копируем из запроса, организовываем проверку – что отправили, то и получили). На запрос команды "0х01, ..." – ответ без ошибок должен быть "0х00, 0х01, ..." если в ответ пришло "0х00, 0х02, ..." значит повторить запрос.

Последний байт всех структур заканчивается постоянным значением char constant;

Для структур send constant = 0x0F, а для структур recv constant = 0xF0.

**Структуры по командам запроса и ответ на них:**

**0х01** – Пинг сервера (проверка работоспособности сервера, запуск самодиагностики):

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02 }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х02** – Перезагрузить сервер:

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02 }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х03** – Debug сообщения (включение/выключение вывода в консоль определенного типа):

tipe – тип сообщений:

0х00 – включить вывод всех debug сообщений

0х01 – включить вывод сообщений с ошибками (описание ошибки и значение переменной с ошибкой)

0х02 – включить вывод сообщений с важных уведомлений

0х03 – включить вывод информационных (отладочных) сообщений (старт функции, вывод переменных которые принимает функция, значение переменных изменившихся в функции и тд. Все что может быть полезно для отладки ПО)

0х04 – включить вывод сообщений относящихся к безопасности (ввод логина/пароля, удачно или нет, все что связанно с активацией и лицензированием...)

0х80 – отключить вывод всех debug сообщений

0х81 – отключить вывод сообщений с ошибками

0х82 – отключить вывод сообщений с важных уведомлений

0х83 – отключить вывод информационных (отладочных) сообщений

0х84 – отключить вывод сообщений относящихся к безопасности

struct SENDSTRUCT {

uint8\_t command;

uint8\_t tipe;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03 }

0х03 – если tipe отсутствует в перечне команд

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х11** – Авто Пинг оборудования (настройка автоматической проверки состояния оборудования, в случае отсутствия связи перезагрузка оборудования и проброс последних сохраненных настроек)

time – время между проверками в секундах (0 – отключить проверку, максимум 255 секунд)

struct SENDSTRUCT {

uint8\_t command;

uint8\_t time;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02 }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х12** – Авто перезагрузка сервера (настройка автоматической перезагрузки сервера по интервалу времени)

time – время работы между перезагрузками сервера в минутах (0 – отключить restart, максимум 255 минут)

struct SENDSTRUCT {

uint8\_t command;

uint8\_t time;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02 }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х13** – Логирование (включение/выключение записи логов определенного типа в определенные файлы):

tipe – тип логов:

0х00 – включить запись всех логов

0х01 – включить запись логов сервера

0х02 – включить запись логов оборудования

0х03 – включить запись логов ошибок (дата, время, отклонение от нормы в виде сообщения, значение)

0х04 – включить запись логов безопасности (дата, время, сообщение, удачно или нет)

0х80 – отключить запись всех логов

0х81 – отключить запись логов сервера

0х82 – отключить запись логов оборудования

0х83 – отключить запись логов ошибок

0х84 – отключить запись логов безопасности

struct SENDSTRUCT {

uint8\_t command;

uint8\_t tipe;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03 }

0х03 – если tipe отсутствует в перечне команд

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х1Е** – Сохранить настройки (относящиеся к серверу)

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02 }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х1F** – Сброс на заводские настройки сервера (по умолчанию, все логи вкл, весь дебаг, автопинг и авторестарт выкл)

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02 }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х20** – Перевести в режим сна (выключить) оборудования

time – время через которое оборудование должно проснуться в минутах (0 – если не нужно автовключение, максимум 255 минут. Продумать реализацию на стороне сервера)

struct SENDSTRUCT {

uint8\_t command;

uint8\_t time;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х21** – Вывести из режима сна (включить или просто пинг, состояние) оборудования

time – время через которое оборудование должно отключиться, перейти в спящий режим в минутах (0 – если не нужно автовыключение, максимум 255 минут. Продумать реализацию на стороне сервера)

struct SENDSTRUCT {

uint8\_t command;

uint8\_t time;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х22** – Перезагрузить оборудование

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х23** – Запуск (старт / возобновить)

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х24** – Пауза оборудования

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х25** – Остановить оборудование

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х26** – Запись данных в файл

file\_name – имя файла которое создаст сервер и запишет в него данные (файлы записываются в папки в соответствии с текущей датой). Длина имени файла ограничена 40 символами и всегда > 3 символов.

type – тип данных (номер набора определенных параметров) которые необходимо записать в файл. Всегда > 0.

struct SENDSTRUCT {

uint8\_t command;

string file\_name;

uint8\_t type;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х05, 0х06, 0х07, 0х7Е, 0х7F }

0х03 – если не предусмотрена запись данных в файл, если type не предусмотрен или равен 0

0х05 – если в файле нет данных для записи/ответа

struct RECVSTRUCT { // при type = 0х01

uint8\_t error;

uint8\_t command;

float variable1;

float variable2;

float variable3;

float variable4;

bool status1;

char constant;

};

struct RECVSTRUCT { // при type = 0х02

uint8\_t error;

uint8\_t command;

int32\_t variable5;

int32\_t variable6;

int32\_t variable7;

int32\_t variable8;

bool status2;

char constant;

};

struct RECVSTRUCT { // при type = 0х03

uint8\_t error;

uint8\_t command;

int32\_t variable9;

int32\_t variable10;

float variable11;

float variable12;

bool status3;

char constant;

};

struct RECVSTRUCT { // при type = 0х7F

uint8\_t error;

uint8\_t command;

char array[random];

char constant;

};

**0х27** – Получить значения оборудования

type – тип данных (номер набора определенных параметров) которые необходимо записать в файл. Всегда > 0.

struct SENDSTRUCT {

uint8\_t command;

uint8\_t type;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х05, 0х7Е, 0х7F }

Структуры ответа как для "0х26 – Запись данных в файл"

**0х2F** – Авто калибровка оборудования

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х30** – Задать комплект параметров настройки оборудования

type – номер комплекта параметров настройки. Всегда > 0.

struct SENDSTRUCT { // при type = 0х01

uint8\_t command;

uint8\_t type;

float variable1\_1;

float variable1\_2;

float variable1\_3;

float variable1\_4;

bool status1;

char constant;

};

struct SENDSTRUCT { // при type = 0х02

uint8\_t command;

uint8\_t type;

int32\_t variable2\_1;

int32\_t variable2\_2;

int32\_t variable2\_3;

int32\_t variable2\_4;

bool status2;

char constant;

};

struct SENDSTRUCT { // при type = 0х03

uint8\_t command;

uint8\_t type;

int32\_t variable3\_1;

int32\_t variable3\_2;

float variable3\_3;

float variable3\_4;

bool status3;

char constant;

};

struct SENDSTRUCT { // при type = 0х7F

uint8\_t command;

uint8\_t type;

char array[random];

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х08, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х31** – Получить комплект параметров настройки оборудования (возвращается установленные параметры настройки оборудования)

type – номер комплекта параметров настройки. Всегда > 0.

struct SENDSTRUCT {

uint8\_t command;

uint8\_t type;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х7Е, 0х7F }

struct RECVSTRUCT { // при type = 0х01

uint8\_t error;

uint8\_t command;

float variable1\_1;

float variable1\_2;

float variable1\_3;

float variable1\_4;

bool status1;

char constant;

};

struct RECVSTRUCT { // при type = 0х02

uint8\_t error;

uint8\_t command;

int32\_t variable2\_1;

int32\_t variable2\_2;

int32\_t variable2\_3;

int32\_t variable2\_4;

bool status2;

char constant;

};

struct RECVSTRUCT { // при type = 0х03

uint8\_t error;

uint8\_t command;

int32\_t variable3\_1;

int32\_t variable3\_2;

float variable3\_3;

float variable3\_4;

bool status3;

char constant;

};

struct RECVSTRUCT { // при type = 0х7F

uint8\_t error;

uint8\_t command;

char array[random];

char constant;

};

**0х3D** – Обновить параметры (проброс всех доступных настроек из файла)

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х08, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х3Е** – Сохранить настройки

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х08, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**0х3F** – Сброс на заводские настройки оборудования (по умолчанию)

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

};

Возможные варианты ответа error: { 0х00, 0х02, 0х03, 0х04, 0х08, 0х7Е, 0х7F }

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

};

**Подсказки по структурам** (при каких командах используется часто структура)**:**

struct SENDSTRUCT {

uint8\_t command;

char reserv;

char reserv;

char constant;

}; // 0х01, 0х02, 0х1E, 0х1F, 0х22, 0х23, 0х24, 0х25, 0х2F, 0х3D, 0х3E, 0х3F

struct SENDSTRUCT {

uint8\_t command;

uint8\_t tipe || time;

char reserv;

char constant;

}; // 0х03, 0х11, 0х12, 0х13, 0х20, 0х21, 0х27, 0х31

struct RECVSTRUCT {

uint8\_t error;

uint8\_t command;

char reserv;

char constant;

}; // 0x01, 0х02, 0х03, 0х11, 0х12, 0х13, 0х1E, 0х1F, 0х20, 0х21, 0х22, 0х23, 0х24, 0х25, 0х2F, 0х30, 0х3D, 0х3E, 0х3F