Контроллер опрашивает раз в 5 секунд есть ли новые заказы, каждая колонка опрашивается по отдельности. Номера команд необходимо предоставить отделано.

struct OrderComReq

{

uint8\_t columnId;

uint8\_t status; // текущий статус

};

status{0 — свободна, 1 — занята}

при изменении статуса колонки, контроллер должен немедленно отправить на сервер статус занята.

Сервис отвечает

struct OrderComRes

{

uint8\_t columnId;

OrderType orderType;

float orderVolume;

float priceFuel;

};

Возможние значения

// Money – заправка на фиксированную сумму, OrderVolume – сумма рублей

// Liters – заправка на литры, OrderVolume – кол-вол литров

// FullTank – полный бак, OrderVolume – максимальная сумма рублей

enum OrderType : uint8\_t

{

NoOrder = 0,

Money = 1,

Liters = 2,

FullTank = 3, (опционально)

};

После получения заказа контроллер проверяет цену.

Если цена отличается выполняет отмену заказа.

struct CanceledComReq

{

uint8\_t columnId;

ReasonType reason;

};

enum ReasonType : uint8\_t

{

other = 0,

ivalidPrice = 1,

};

Если цена корректна отправляет принятие заказа

struct AcceptComReq

{

uint8\_t columnId;

};

на все команды сервис отправляет подтверждение отправки   
struct AcceptComResp

{

uint8\_t columnId;

uint8\_t status;

};

если status не равен 0, то необходимо повторить операцию отправки.

Далее после успешной отправки Accept, контроллер отправляет

struct FuelingComReq

{

uint8\_t columnId;

};

И получает ответ от сервиса:

struct FuelingComResp

{

uint8\_t columnId;

uint8\_t status;

};

При успешном ответе начинается заправка.

По окончании заправки контроллер отправляет

struct CompletedComReq

{

uint32\_t columnId;

float litres;

};

И получает ответ от сервиса

struct CompletedComResp

{

uint8\_t columnId;

uint8\_t status;

};

Если статус не равен 0, то отправку необходимо повторить.

При успешном ответе необходимо снова перейти в режим опроса 1 раз в 5 секунд.