ИНФОСКАН
системы измерения габаритов и веса

Инфоскан 3D Pro

Руководство пользователя

Инструкция по сборке | Первый запуск | Сетевое подключение | Настройка передачи данных.

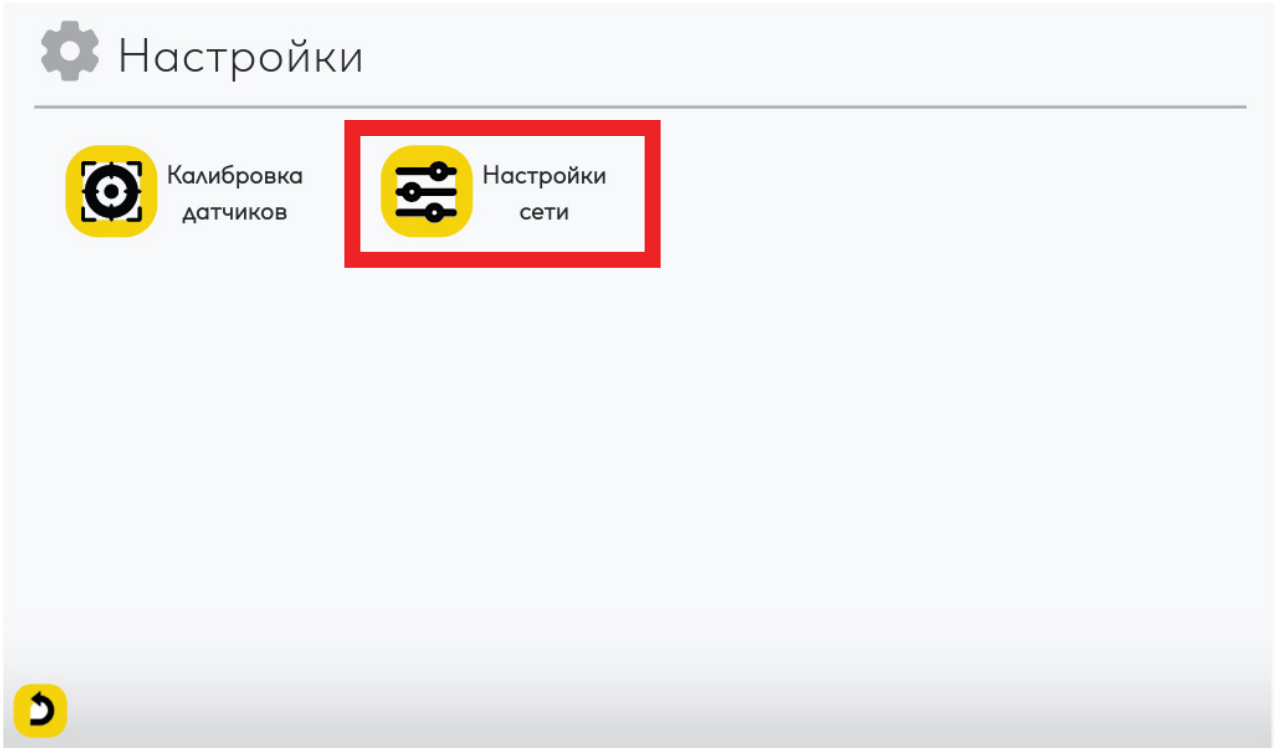
Оглавление

Запуск Инфоскан

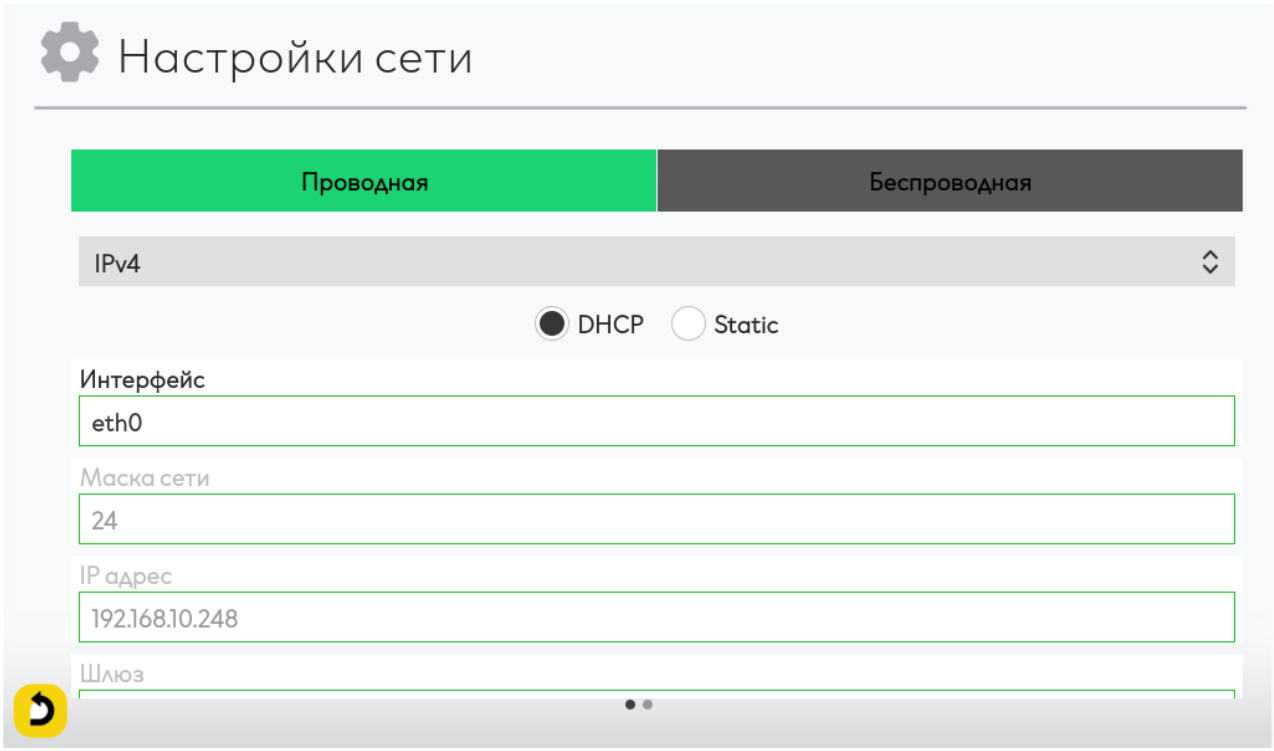
Включить кабель питания в сетевую розетку 220в. Убедиться что на весовой платформе нет посторонних предметов. Нажатием кнопки включения, запустить ПК установленный внизу на кронштейне устройства. Дождаться загрузки системы и на рабочем столе запустить программу **"Infoscan Camera"**. При запуске устройство осуществит автоматическую калибровку.

Подключение к сети

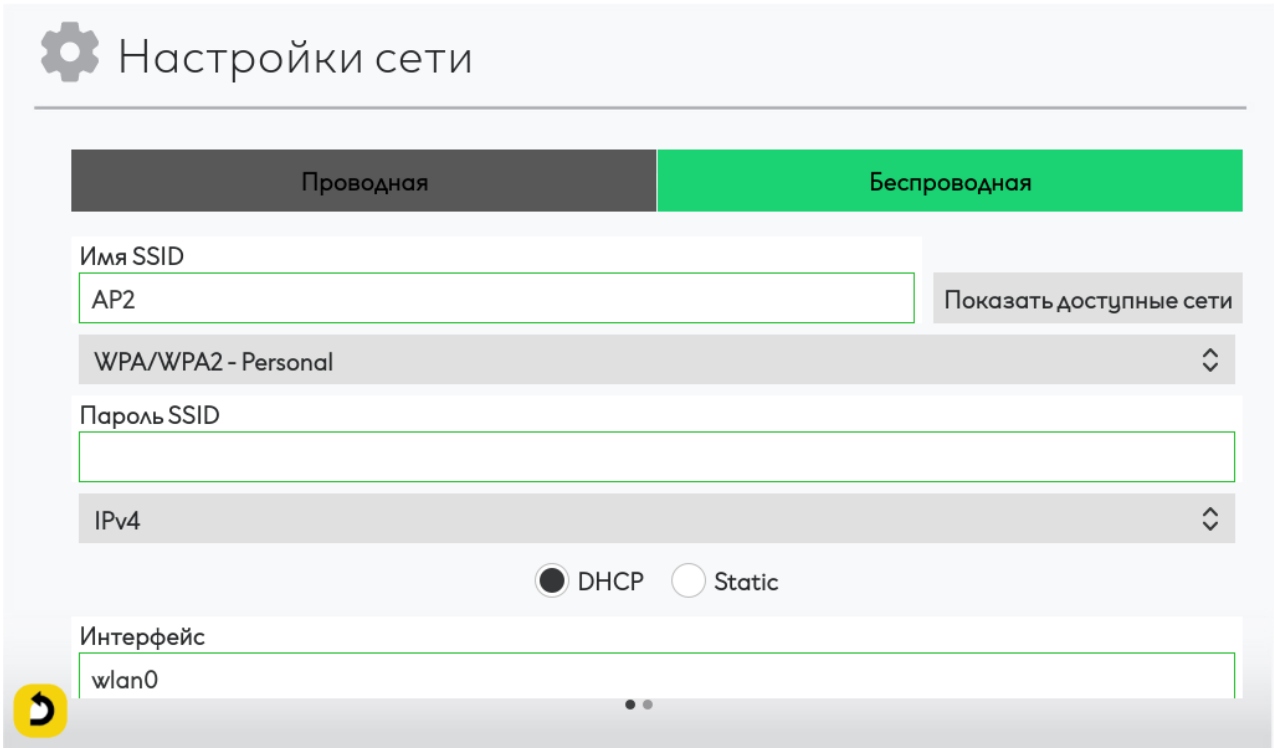
Проводное/Беспроводное сетевое подключение осуществляется через пункт настроек "Настройка сети"



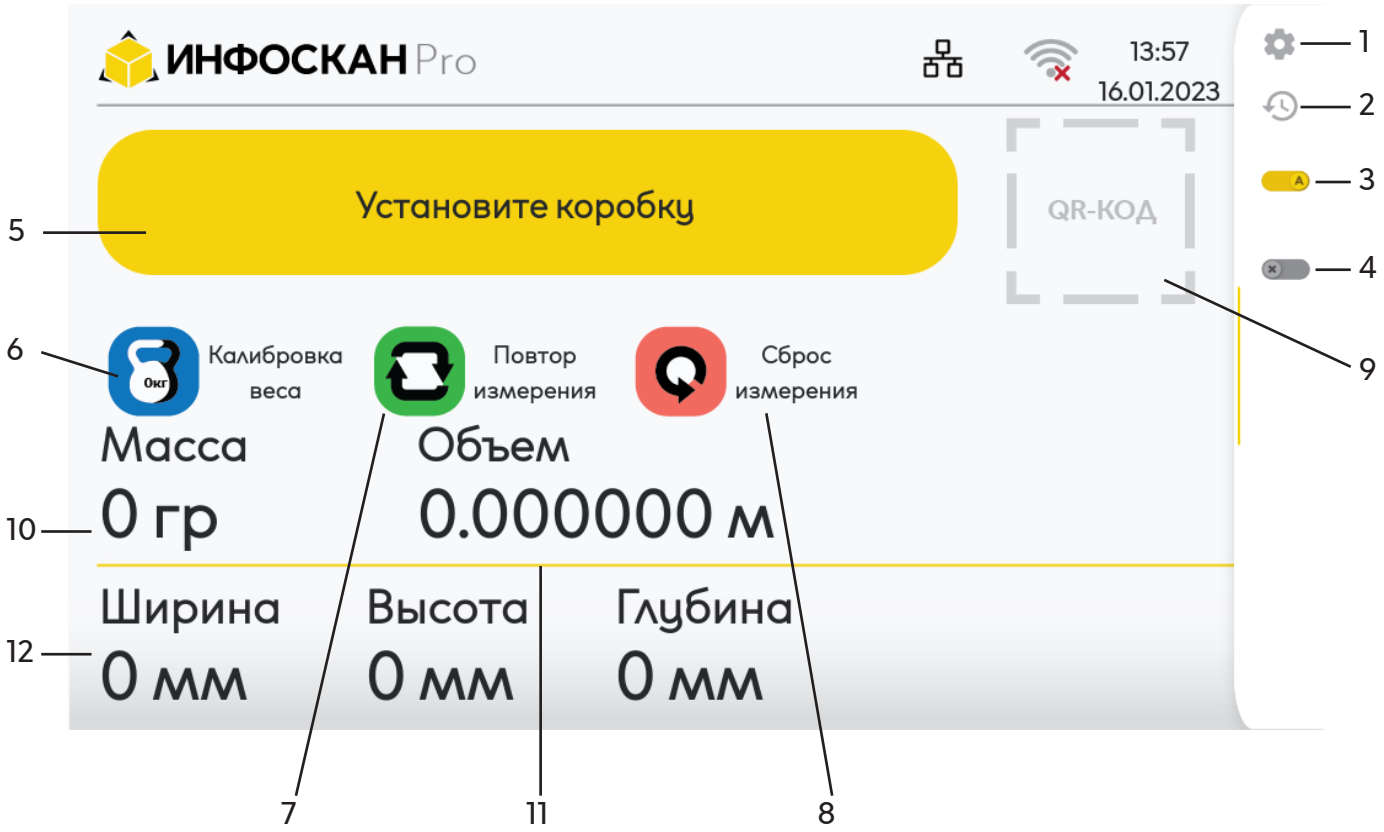
Настройка проводного поденлючения



Настройка Беспроводного подключения



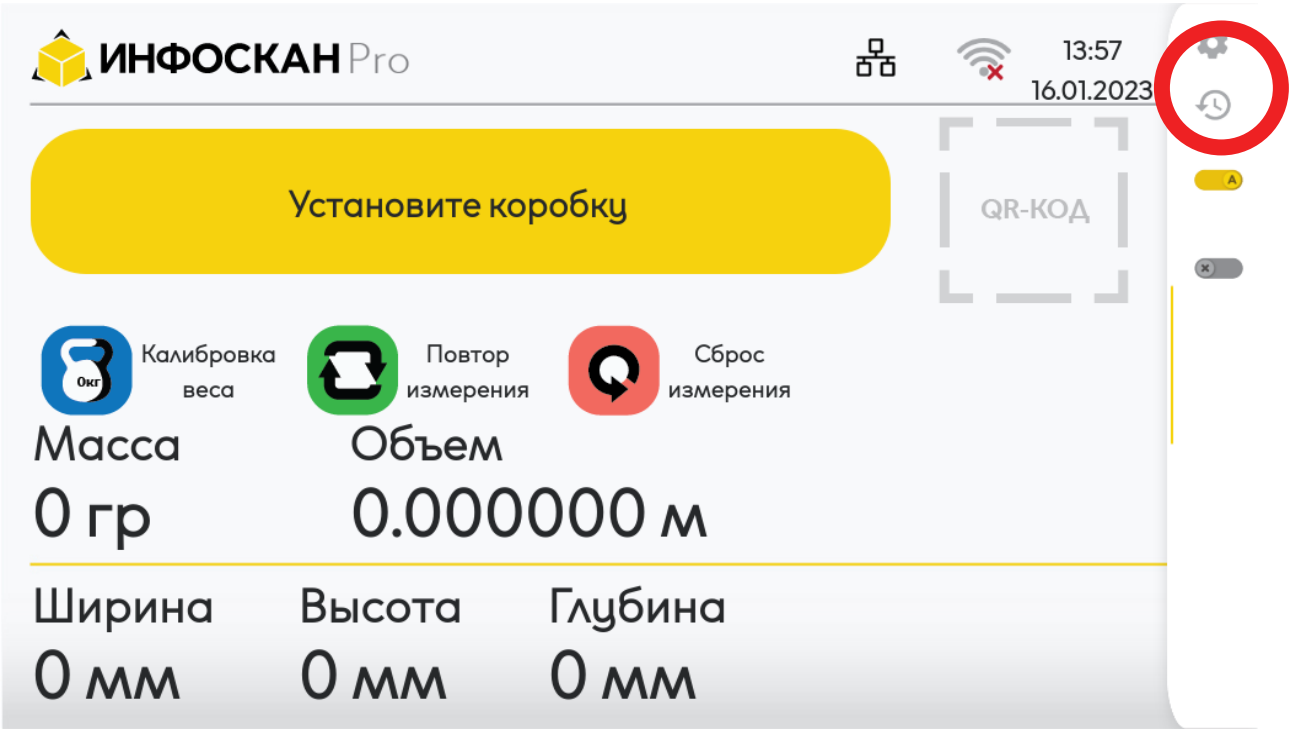
Главный экран Инфоскан



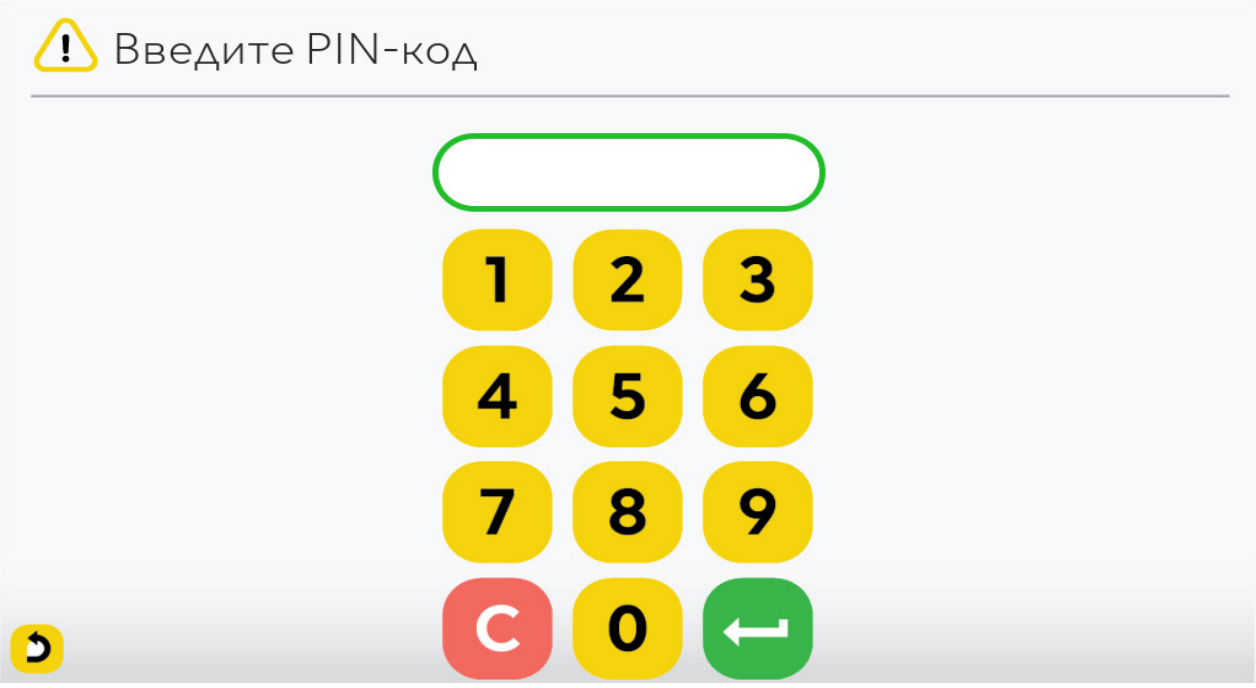
Описание нумерации:

1. Переход в меню настроек устройства;
2. Вкладка просмотра истории последних измерений;
3. Переключатель режимов измерений;
4. Режим работы со сканером ШК;
5. Статус измерения;
6. Кнопка сброса весов на «0»;
7. Кнопка повторения измерений;
8. Кнопка обнуления измерений;
9. Поле генерации QR-кода;
10. Поле вывода массы измеряемого объекта;
11. Поле вывода объема измеряемого объекта;
12. Поля вывода ШВГ измеряемого объекта;

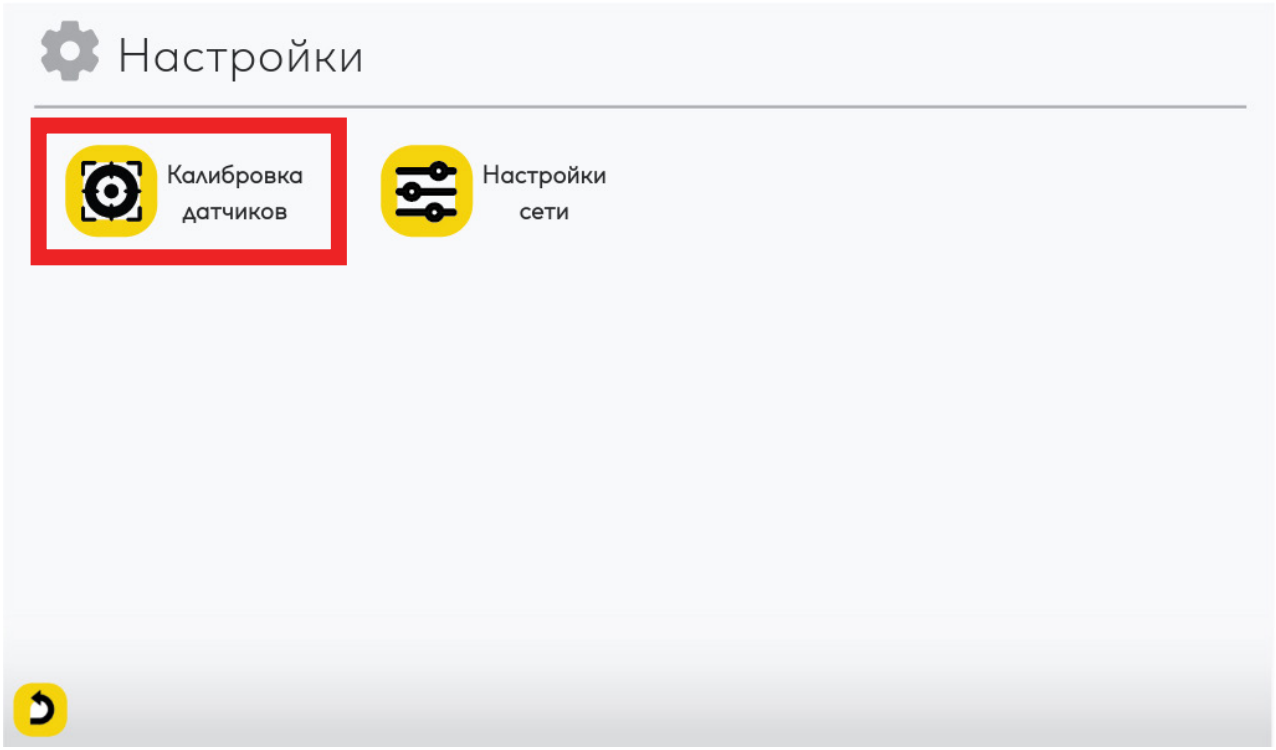
Меню настроек Инфоскан



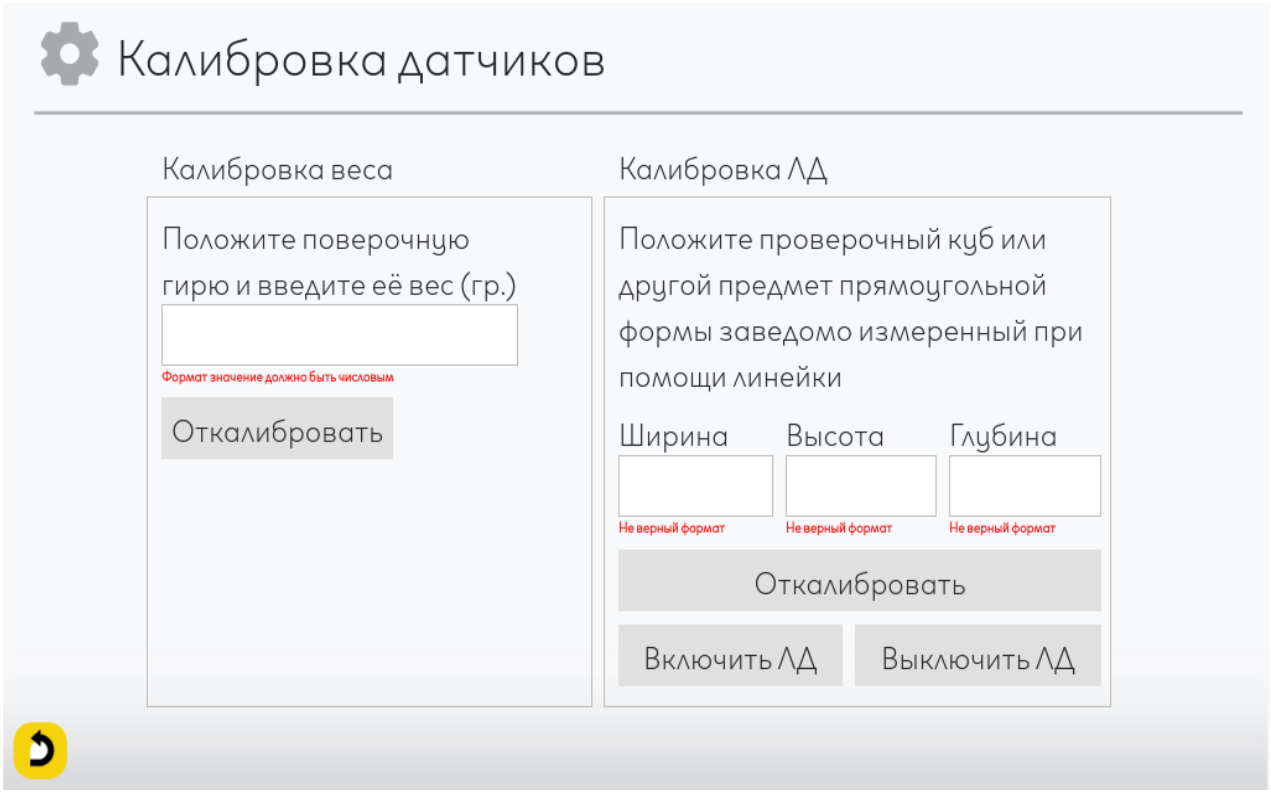
Для перехода в меню настроек необходимо на главном экране нажать на шестерёнку и ввести стандартный пароль «**1234**».



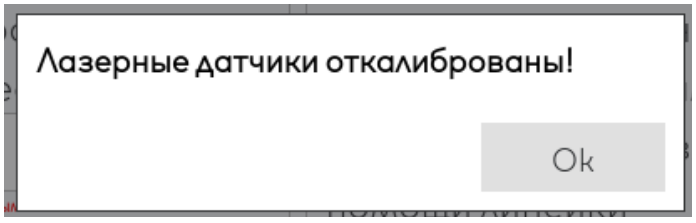
Калибровка датчиков



Выбрать пункт "Калибровка датчиков"
В данном окне осуществляется калибровка датчика веса и лазерных датчиков.



Уведомление об успешной калибровке



Пример отображения параметров габаритов на ***неоткалиброванном*** *устройстве.*

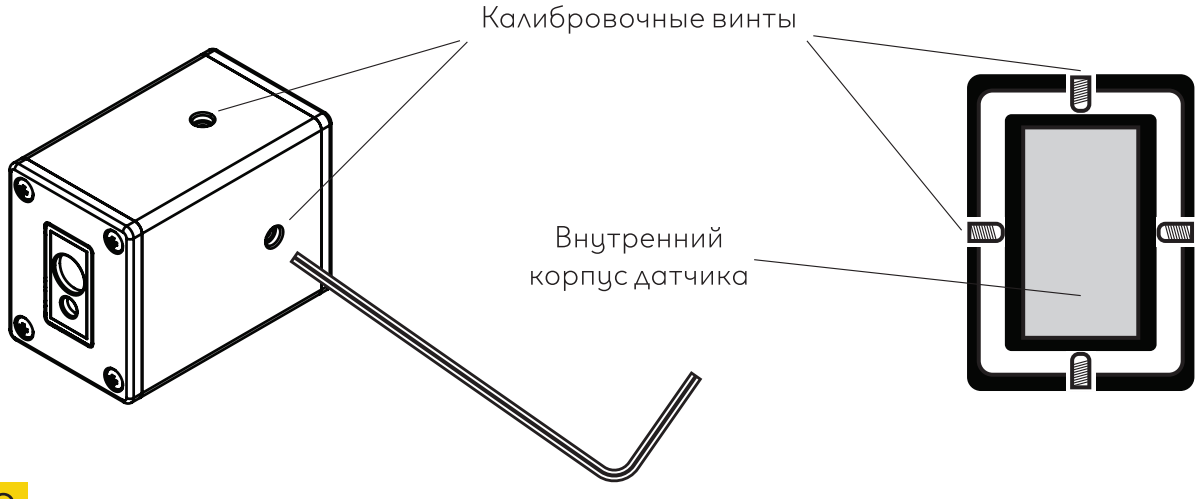
Ширина	Высота	Глубина
20 мм	20 мм	20 мм

Механическая регулировка датчиков.

Устройство оборудованное лазерными датчиками, следует проверить направильностькалибровкилазерныхлучей,длячеговуголустройства устанавливается калибровочный кубик (размер 20х20х20мм), далее в меню Калибровка ЛД выбрать пункт «Включить ЛД». Все 3 лазера должны включиться, лучи каждого лазерного датчика должны быть направлены в центры граней калибровочного кубика.

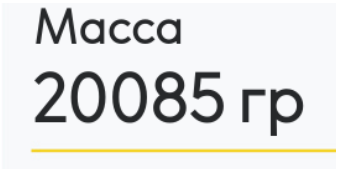
Для регулировки по направлению есть специальные регулировочные винты, находящиеся сверху, снизу и с боков датчика. При регулировке используется шестигранный г-образный ключ H 1.5.
Лазер будет двигаться в сторону того винта, который закручивают по часовой стрелке. Предварительно нужно ослабить противоположный винт, выкрутив его на пару оборотов против часовой стрелки.

Система калибровки направленя изнутри.

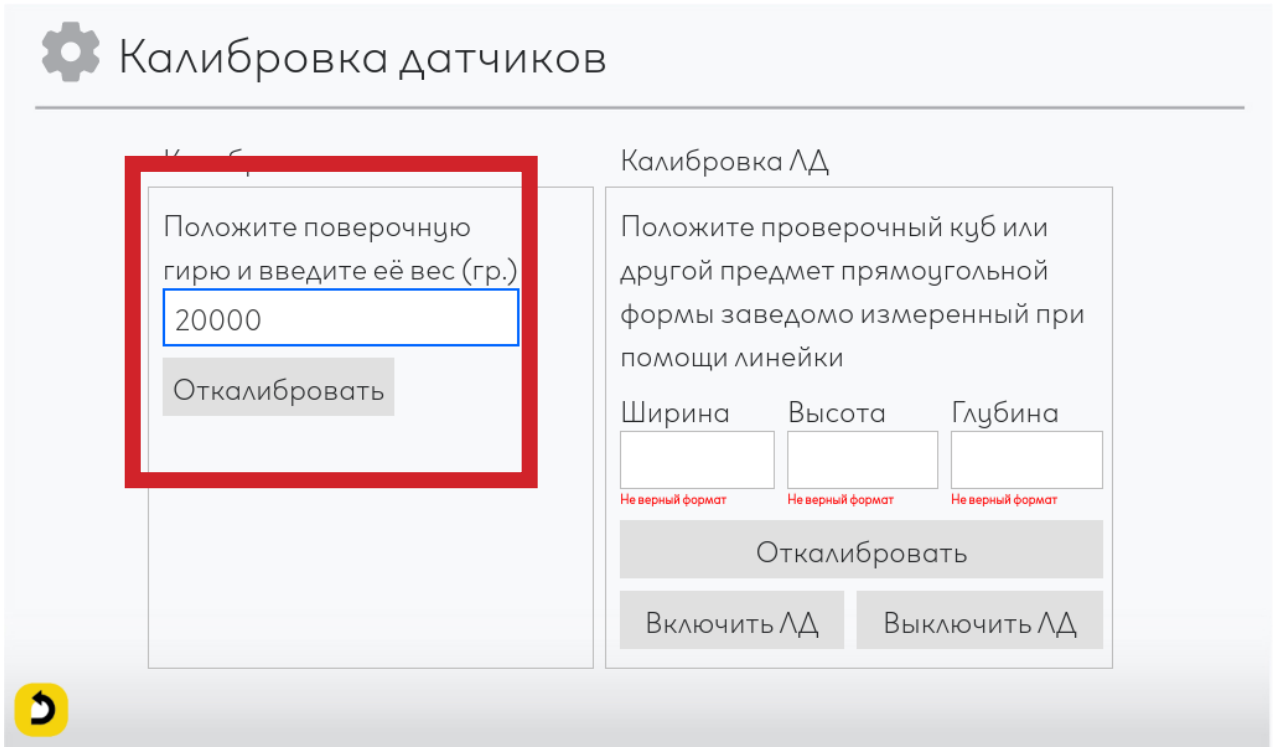


Калибровка весового датчика

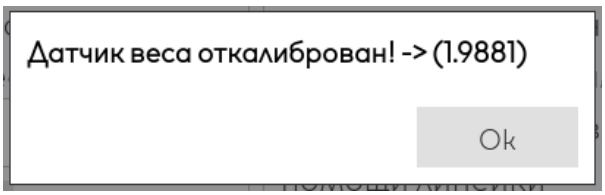
Калибровка веса производится только гирей весом от 20 до 40 кг. После установки объекта на платформу необходимо записать фактическую массу поставленного на платформу объекта (масса объекта должна быть известна заранее) в поле ввода веса для калибровки. После ввода нажать кнопку «**Откалибровать**».

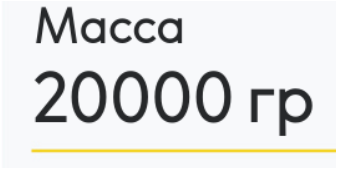


*Пример отображения параметров массы на не **откалиброванном** датчике. При установленной калибровочной гире массой в 20 кг.*



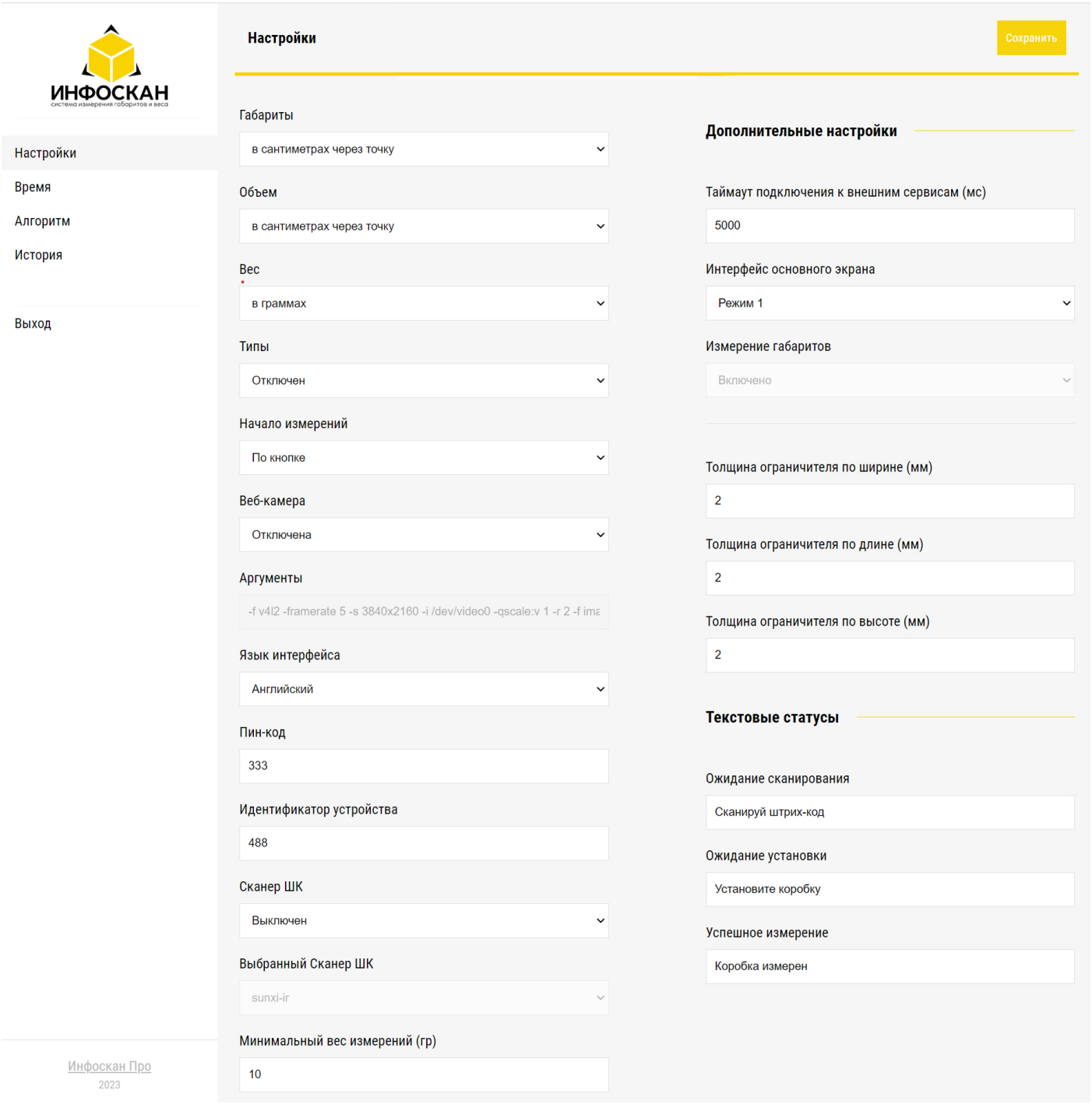
Уведомление об успешной калибровке





*Пример отображения параметров массы на **откалиброванном** датчике. При установленной калибровочной гире массой в 20 кг.*

Конфигурация устройства

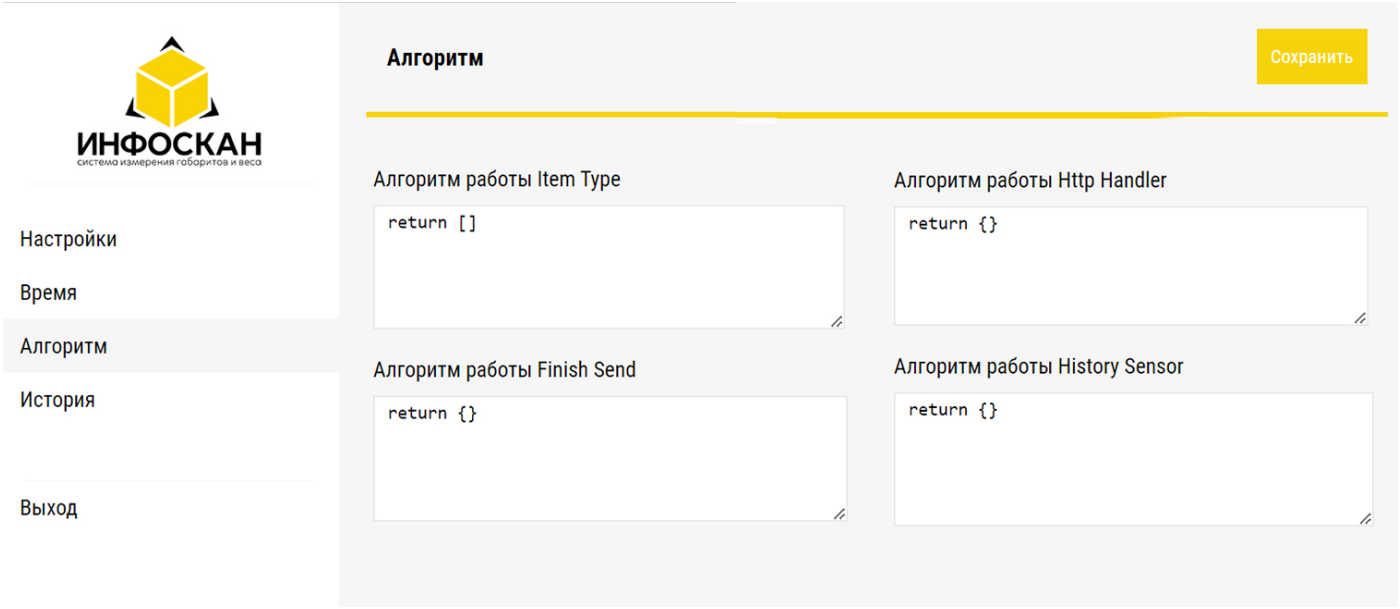


Габариты - единица вывода измерения
Объем - единица вывода объема
Вес - единица вывода массы
Типы - включает обработку расширенного функционала для вывода дополнительных экранов jsScriptItemType
Начало измерений - метод начала измерений
- **Через API команду** - после стабилизации веса и POST запроса на /api/gateway
- **Автоматически** - после стабилизации веса
- **По кнопке** - после стабилизации веса и нажатии на кнопку "Измерить"

Веб-камера - включение/отключение веб-камеры (только для версии оборудования ПРО и ЛАЙТ)
Аргументы - параметры для подключения веб-камеры (только для версии оборудования ПРО и ЛАЙТ)
Язык интерфейса - язык, который используется в пользовательском интерфейсе программного обеспечения или устройства для общения с пользователем
Пин-код - цифровой код для входа в меню настроек через графический интерфейс
Идентификатор устройства - номер устройства, доступный для отображения в шаблоне как переменная DEVCODE
Сканер - метод идентификации измеряемого объекта
• **Выключен** - Сканер штрих-кода не используется
• **Включен** - Сканер штрих-кода включен (на сканере должны быть включены опции: Работать как эмуляция клавиатуры, Отправка символа Enter после считывания штрих-кода)
• **Через API команду** - отправка идентификатора объекта через POST запрос на /api/gateway
Выбранный сканер ШК - список оборудования, работающих в режиме эмуляции клавиатуры
Минимальный вес измерений - минимальная масса измеряемого объекта для начала измерений
Таймаут подключения к внешним сервисам (мс) - время ожидания ответа от внешнего сервиса, настроенного в меню Алгоритм
Интерфейс основного экрана - Интерфейс рабочего экрана
• **Режим 1** - стандартный интерфейс
• **Режим 2** - интерфейс без отображения габаритов
Измерение габаритов - Включение и отключение измерения габаритов при infoscanLitePageMode "Режим 2"
• **Включено** - измерения габаритов включено
• **Выключено** - работа устройства только с массой, без измерения габаритов
Толщина ограничителя по высоте (мм) - установлена стандартная толщина уголка 2 мм (только для версии оборудования ПРО и ЛАЙТ)
Толщина ограничителя по ширине (мм) - установлена стандартная толщина уголка 2 мм (только для версии оборудования ПРО и ЛАЙТ)
Толщина ограничителя по длине (мм) - установлена стандартная толщина уголка 2 мм (только для версии оборудования ПРО и ЛАЙТ)
Ожидание сканирования - текстовый статус баннера при ожидании сканирования штрих-кода
Ожидание установки - текстовый статус баннера при ожидании установки объекта на платформу
Успешное измерение - текстовый статус баннера после успешного измерения

Настройка алгоритма

Алгоритмы отвечают за то, как будет работать устройство и как будут выводиться данные, или же в какой последовательности будут проводиться измерения, также они нужны, чтобы добавить другие функции, такие как генерация QR-кода и тд...



Назначение шаблонов jsScript

jsScriptItemType - шаблон расширенного функционала для вывода дополнительных экранов: Ввод количества упаковок, Выбор типа упаковок(статический или через REST API), Ввод дополнительного штрих-кода, Проверка штрих-кода

jsScriptFinishSend - Шаблон вывода/передачи данных об измерениях по средствам: QR-код, REST API(JSON XML), FTP, FTPS, SFTP, TCP, File

jsScriptHttpHandler - Шаблон настройки отображения данных об измерениях в реальном времени (http://IP/api/data (для Linux) или http://IP:8000/api/data (для Windows))

jsScriptHistorySensor - Шаблон настройки отображения данных об истории измерений (http://IP/api/data (для Linux) или http://IP:8000/api/data (для Windows))

Настройка шаблона jsScriptItemType

Шаблон ввода дополнительного штрих-кода

```
return [
   function(args) {
      return {
         page: 'AdditionalScanPage',
         pageTitle: 'Сканируй штрих-код',
         nameResult: 'resultItemTypeScan'
      }
   }
]
```

Описание шаблона:

- **page: ‘AdditionalScanPage’** – отвечает за вывод экрана с дополнительным сканированием.

- **pageTitle: ‘Сканируйте штрих-код’** – переменная, которая задает название экрана

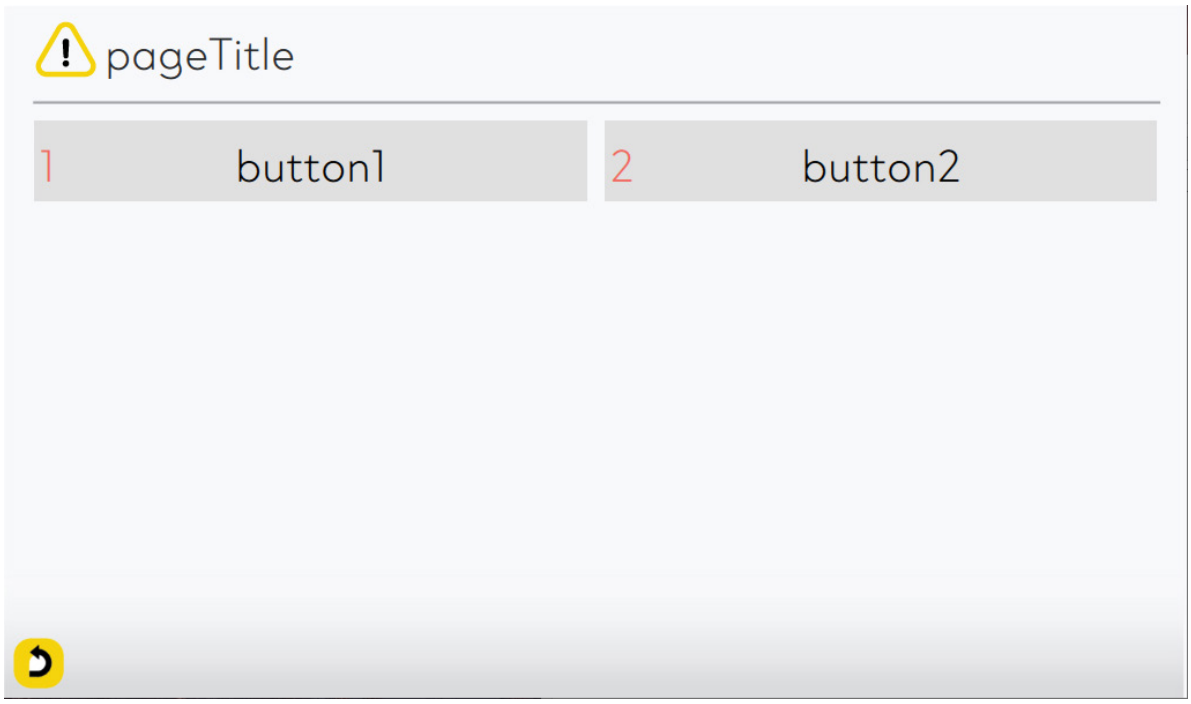
- **nameResult: ‘resultItemTypeScan’** - переменная для получения данных в jsScriptFinishSend, в этой переменной будет содержаться значение, которое считал считыватель штрих-кода

Шаблон выбора типа упаковки (статический)

```
return [
   function(args) {
      return {
         page: 'ListButtonPage',
         pageTitle: 'pageTitle',
         nameResult: 'resultItemTypeList',
         pageButtons: ['button1', 'button2']
      }
   }
]
```

Описание шаблона:

- page: ‘ListButtonPage’ – отвечает за вывод экрана с кнопками
- pageTitle: ‘pageTitle’ – переменная отвечает за название заголовка на экране
- nameResult: ‘resultItemTypeList’ – переменная для получения данных в jsScriptFinishSend, в этой переменной будет содержаться значение, которое содержит наименование нажатой кнопки.
- pageButtons: [‘button1’, ‘button2’] – отвечает за название самих кнопок.



Шаблон выбора типа упаковки (через REST API запрос)

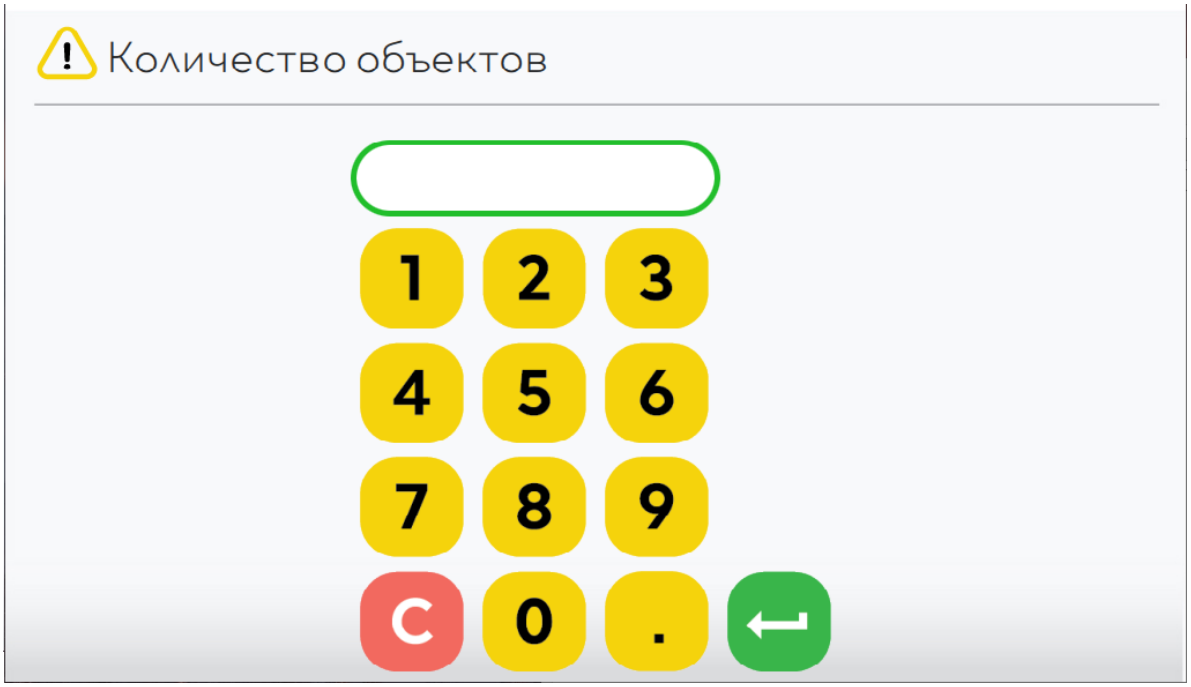
```
return {
    http: function () {
        return {
            url: 'http://IP/endpoint',
            headers: {
                'Authorization': 'Basic ' + HELPER_JS.toBase64('LOGIN' + ':' +
'PASSWORD'),
                'Content-type': 'application/json'
            },
            body: function () { // return string
                return "{\"width\": \"" + SENSOR.widthFormat + "\"," +
                    "\"height\": \"" + SENSOR.heightFormat + "\"," +
                    "\"length\": \"" + SENSOR.lengthFormat + "\"," +
                    "\"weight\": \"" + SENSOR.weightFormat + "\"," +
                    "\"itemType\": \"" + ITEM_TYPE + "\"," +
                    "\"devCode\": \"" + DEV_CODE + "\"," +
                    "\"barcode\": \"" + BARCODE + "\"}"
            },
            resp: function (resp, httpCode) { // return obj
                try {
                    console.log(resp)
                    var json = JSON.parse(resp)
                    if (httpCode == 200) {
                        if (json.status == 'OK') {
                            return {
                                banner: BARCODE + '\n' + json.description,
                                comment: ''
                            }
                        }
                        else if (json.status == 'NOT OK') {
                            throw Error(json.description)
                        }

                        else {
                            throw Error(BARCODE + ' - ошибка отправки данных
на сервер')
                        }
                    }
                    else {
                        throw Error('Ошибка, получен HTTP код: ' + httpCode + ',
вместо ожидаемого HTTP кода 200')
                    }
                }
                catch (e) { // ошибка парсера
                    console.log(e)
                    throw Error(BARCODE + '\n' + json.description)
                }
            }
        }
    }
}
```

Шаблон ввода количества упаковок

```
return [
   function(args) {
      return {
         page: 'CountPage',
         pageTitle: 'pageTitle',
         nameResult: 'resultItemTypeCount'
      }
   }
]
```

Описание шаблона:

- page: ‘CountPage’ – отвечает за вывод экрана ввода количества упаковок.
- pageTitle: ‘pageTitle’ – переменная отвечает за название заголовка на экране
- nameResult: ‘resultItemTypeCount’ – переменная для получения данных в jsScriptFinishSend, в этой переменной будет содержаться значение, которое содержит указанное количество упаковок

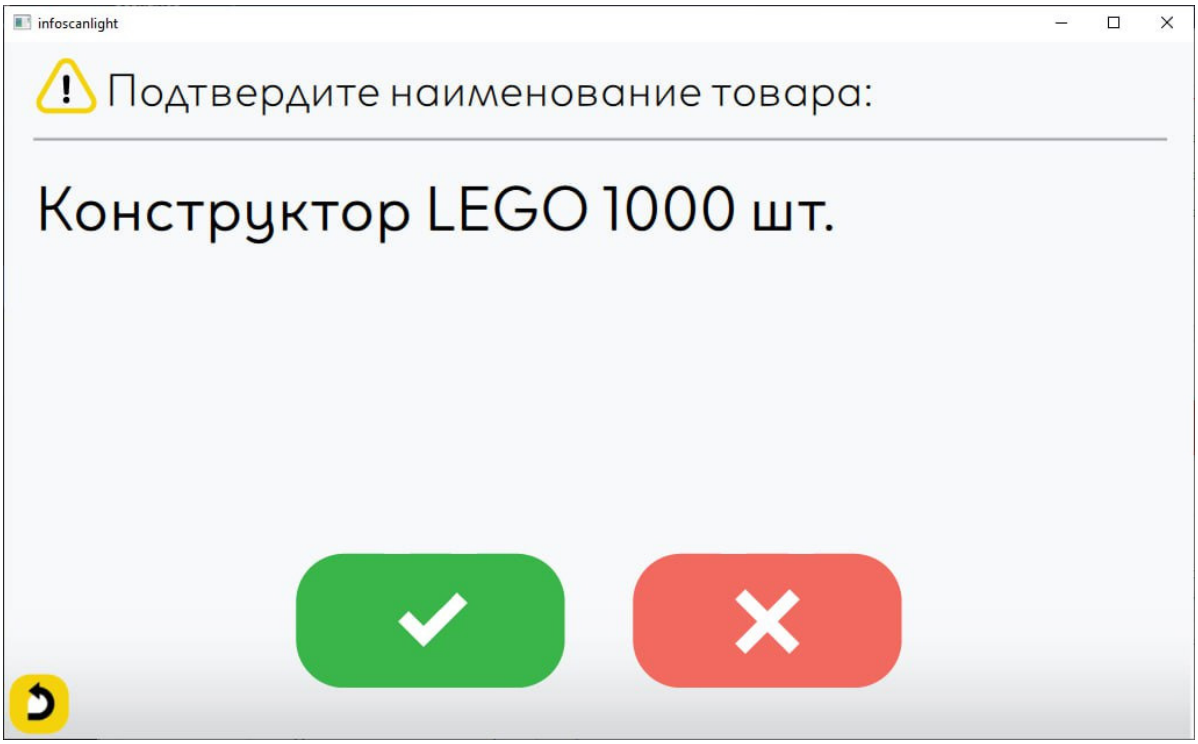


Шаблон проверки штрих-кода, для подтверждения пользователем

```
return [
   function (args) {
      return {
         page: 'ConfirmBarcodePage',
         pageTitle: 'Подтвердите наименование товара:',
         nameResult: 'resultconfirmBarcode',
         http: {
            url: 'http://IP/endpoint',
            headers: {
               'Accept': 'application/xml',
               'Content-type': 'application/xml',
               'Authorization': 'Basic ' + HELPER_JS.toBase64('admin' + ':' + 'admin')
            },
            body: function () { // return string
             return '<CheckBarcode><Barcode>' + BARCODE + '</Barcode></Check-
Barcode>'
            },
            resp: function (resp) { // return object
               var json = HELPER_JS.xmlToJson(resp)
               console.log(resp)
               console.log(JSON.stringify(json, null, 2))
               if (json.CheckBarcodeResponse == null) {
                  throw Error("Ошибка: Нет с данных");
               }
               else if (json.CheckBarcodeResponse.Status == null) {
                  throw Error("Ошибка: Нет с данных");
               }
               else if (json.CheckBarcodeResponse.Status == "OK") {
                  return json.CheckBarcodeResponse.Comment;
               }
               else {
                  throw Error(json.CheckBarcodeResponse.Comment);
               }
            }
         }
      }
   }
]
```

Описание шаблона:

- page: ‘ConfirmBarcodePage’ – отвечает за вывод экрана проверки наименования штрих-кода, для подтверждения пользователем
- pageTitle: ‘pageTitle’ – переменная отвечает за название заголовка на экране
- nameResult: ‘resultconfirmBarcode’ – переменная для получения данных в jsScriptFinishSend, в этой переменной будет содержаться значение, которое содержит подтвержденное наименование товара
- url: ‘http://IP/endpoint’ – адрес веб-сервиса
- headers – заголовки http запроса
- body – тело http запроса
- resp – функция обработки ответа от веб-сервиса, с указанием пути наименования штрих-кода (return только в формате JSON)



Настройка шаблона jsScriptFinishSend

1. Вывод QR-кода на экран.

Генератор QR кода используется для вывода данных на графическом экране устройства и находится с права от баннера статусов.
Для генерации формата данных , необходимо отредактировать шаблон согласно вводным данным, описание работы функции qr:

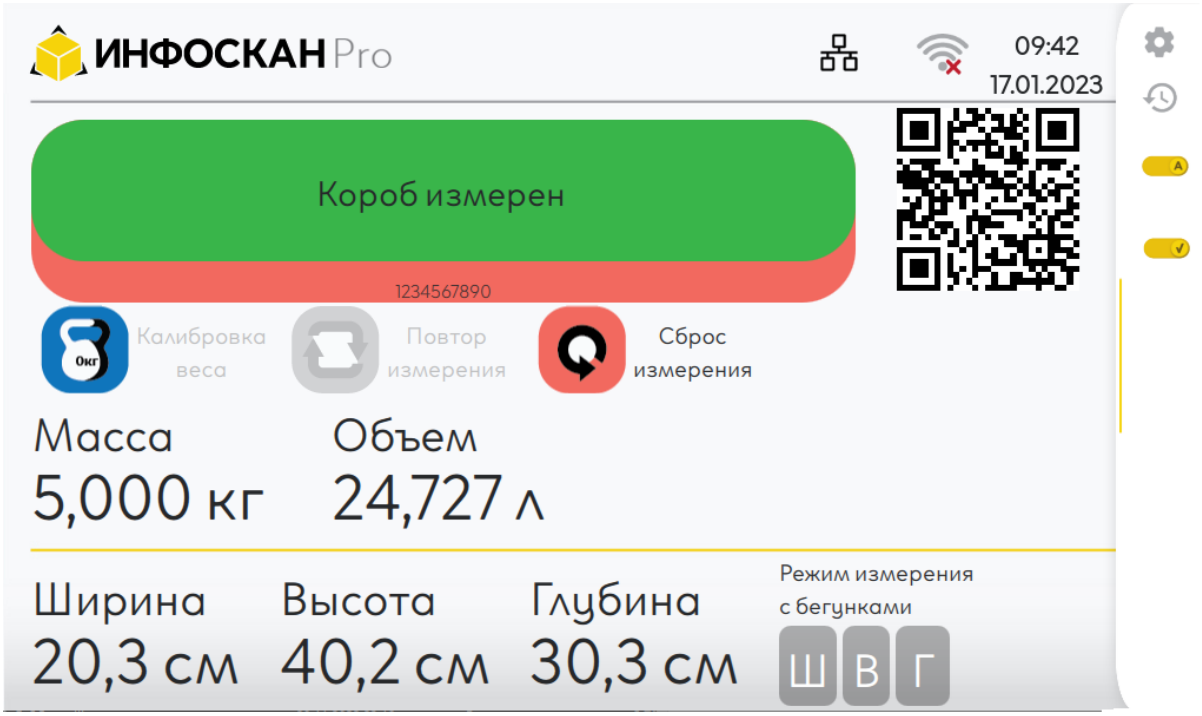
- **body** – тело POST запроса, составляемое в виде XML/JSON
- **resp** – функция ничего не возвращает.

Для вывода успешного результата в баннер статусов используется return

Шаблон для вывода на экран QR кода с результатом измерений

```
_____________________________________________________
return {
   qr: function() {
    return {
      body: function() {
           return BARCODE + ';' + SENSOR.widthFormat + ';' + SENSOR.
heightFormat + ';' + SENSOR.lengthFormat + ';' + SENSOR.weightFor-
mat + ';' + SENSOR.volumeFormat
       },
      resp: function() { // return obj
        return {
            banner: 'Короб измерен: ' + BARCODE,
            comment: ''
          }
       }
    }
  }
}
_____________________________________________________
```

Пример QR кода на экране:
Пример данных в QR коде:
1234567890;20,3;40,2;30,3;5,000;24,727



2. Обмен данными между SOAP/REST сервисами

SOAP/REST обмен работает по принципу Запрос (Request) - Отклик (Response). В отклике – удаленная система должна вернуть в обязательном порядке статус события, который может иметь наименование OK, ER и т.д., а также необязательный комментарий для вывода на экран.
Для передачи данных в форматах SOAP/REST необходимо перед началом отредактировать шаблон согласно вводным данным, описание работы функции http:

url – адрес SOAP/REST сервиса
headers – HTTP заголовки
body – тело POST запроса, составляемое в виде XML/JSON
resp – функция возвращает ответ от сервиса и HTTP код.
Для вывода успешной результата в баннер статусов используется return, а для вывода ошибки используется throw Error

Шаблон обмена в формате XML :

```
_____________________________________________________________
console.log('itemType ->', JSON.stringify(ITEM_TYPE, null, 4));
return {
  http: function () {
    return {
      url: 'http://IP-адрес/путь',
      headers: {
        'Accept': 'application/xml',
        'Content-type': 'application/xml'
      },
      body: function () { // return string
        return '<LoadMeasuremtWithPrinting>' +
               '<Width>' + SENSOR.widthFormat + '</Width>' +
               '<Height>' + SENSOR.heightFormat + '</Height>' +
               '<Length>' + SENSOR.lengthFormat + '</Length>' +
               '<Weight>' + SENSOR.weightFormat + '</Weight>' +
               '<Barcode>' + BARCODE + '</Barcode>' +
               '</LoadMeasuremtWithPrinting>'
      },
      resp: function (resp, httpCode) { // return obj
        if(httpCode == 200) {
          if (HELPER_JS.xmlXPath(resp, '/Detail/code/text()') == '0') {
            return {
              banner: BARCODE + ' - Успешно записан',
              comment: ''
            }
          }

          else {
            throw Error(BARCODE + ' - Ошибка отправки данных на
сервис')
          }
        }
        else {
          throw Error('Ошибка, получен HTTP код: ' + httpCode + ',
вместо ожидаемого HTTP кода 200')
    }
  }
}
```

Шаблон обмена в формате JSON:

```
_____________________________________________________________
console.log('itemType ->', JSON.stringify(ITEM_TYPE, null, 4));
return {
  http: function () {
    return {
      url: 'http://IP-адрес:8000/путь',
      headers: {
        'Accept': 'application/json',
        'Content-type': 'application/json'
      },
      body: function () { // return string
          return "{\"Width\": \"" + SENSOR.widthFormat + "\"," +
                 "\"Height\": \"" + SENSOR.heightFormat+ "\"," +
                 "\"Length\": \"" + SENSOR.lengthFormat+ "\"," +
                 "\"Weight\": \"" + SENSOR.weightFormat+ "\"," +
                 "\"Barcode\": \"" + BARCODE + "\"}"
      },
       resp: function (resp, httpCode) { // return obj
            var json = JSON.parse(resp)
            if(httpCode == 200) {
              if (json.code == '0') {
                  return {
                      banner: BARCODE + ' - Успешно записан',
                      comment: ''
                  }
              }
               else {
                      throw Error(BARCODE + ' - Ошибка отправки данных на
сервис')
               }
            }
            else {
              throw Error('Ошибка, получен HTTP код: ' + httpCode + ', вместо
ожидаемого HTTP кода 200')
       }
     }
   }
```

3. Отправка данных в виде файла на FTP, SFTP, FTPS сервер:

Передача данных в виде файла на FTP сервер при успешном измерении.
Для генерации формата данных , необходимо отредактировать шаблон согласно вводным данным, описание работы функции ftp, sftp, ftps :

Шаблон отправки файла на FTP сервер:

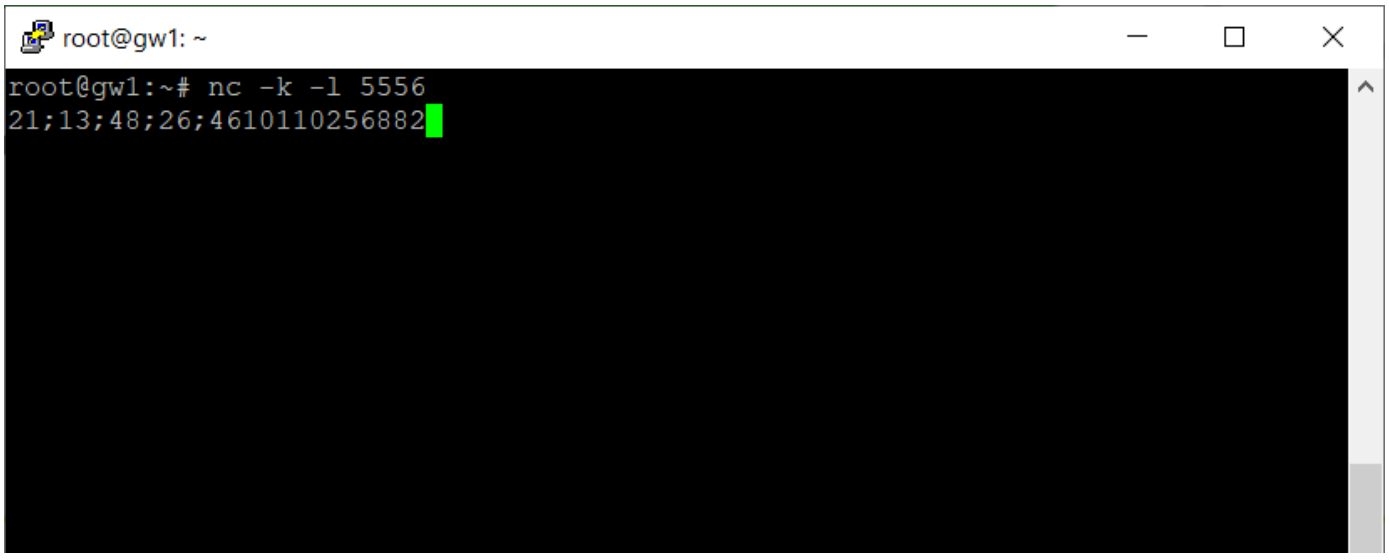
```
console.log('itemType ->', JSON.stringify(ITEM_TYPE, null, 4));

return {
  ftp: function() {
    return {
      address: 'IP-адрес FTP-сервера',
      username: 'Логин FTP',
      password: 'Пароль FTP',
      pathDestination: 'Название файла или переменная' + '.txt ',
      body: function() {
        return BARCODE + ';' + SENSOR.lengthFormat + ';' + SENSOR.height-
Format + ';' + SENSOR.widthFormat + ';' + SENSOR.weightFormat + ';'
      },
      resp: function() { // return obj
        return {
          banner: 'Успешно записан на FTP',
          comment: ''
        }
      }
    }
  }
}
```

4. TCP

Передача данных на TCP порт при успешном измерении с последующим закрытием подключения после передачи данных.

```
return {
  tcp: function() {
    return {
      address: 'IP:порт',
      body: function() {
        return SENSOR.widthFormat + ';' + SENSOR.heightFormat + ';' + SEN-
SOR.lengthFormat + ';' + SENSOR.weightFormat + ';' + BARCODE
      },
      resp: function(resp) {
        return {
          banner: 'Успешно отправлен на TCP',
          bannerFontColor: '#FF00FF',
          comment: ''
        }
      }
    }
  }
}
```

address - IP адрес и порт сервиса
body - данные
banner - текст баннера при успешной отправки данных
bannerFontColor - цвет баннера (не обязательное)
comment - поле вывода комментария (не обязательное)

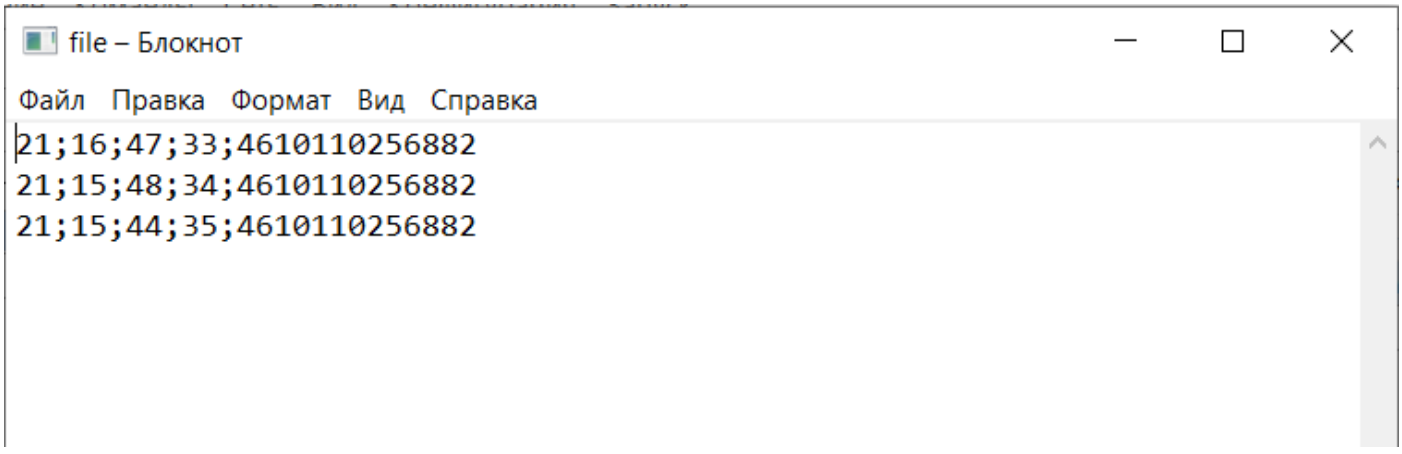


5. File

Сохранение данных в файл (только для версий на ОС Windows)
Запись данных в файл при успешном измерении с перезаписью или дозаписью.

```
return {
  file: function() {
    return {
      pathDestination: 'C:/file.log',
      append: true,
      body: function() {
        return SENSOR.widthFormat + ';' + SENSOR.heightFormat + ';' + SEN-
SOR.lengthFormat + ';' + SENSOR.weightFormat + ';' + BARCODE + '\n'
      },
      resp: function() { // return obj
        return {
          banner: 'Успешно отправлен в файл',
          bannerFontColor: '#FF00FF',
          comment: ''
        }
      }
    }
  }
}
```

pathDestination - путь и имя файла
append - флаг для дозаписи занных в файл
body - данные
banner - текст баннера при успешной записи данных в файл
bannerFontColor - цвет баннера (не обязательное)
comment - поле вывода комментария (не обязательное)



Настройка шаблона jsScriptHttpHandler

Запрос данных с устройства методом GET на URL адрес устройства http://IP/api/data (для Linux) или http://IP:8000/api/data (для Windows) в режиме реального времени согласно шаблону

Шаблон вывода данных

```
return {
  handlerDataResp: function () {
    try {
      if(SENSOR == null) {
        return {
          width: '0',
          height: '0',
          length: '0',
          weight: '0',
          barcode: '',
          devCode: DEV_CODE
        }
      }
      return {
        width: SENSOR.widthFormat,
        height: SENSOR.heightFormat,
        length: SENSOR.lengthFormat,
        weight: SENSOR.weightFormat,
        barcode: BARCODE,
        devCode: DEV_CODE
      }
    }
    catch (e) {
      console.error(e);
    }
  }
}
```

Пример GET запроса с ответом:

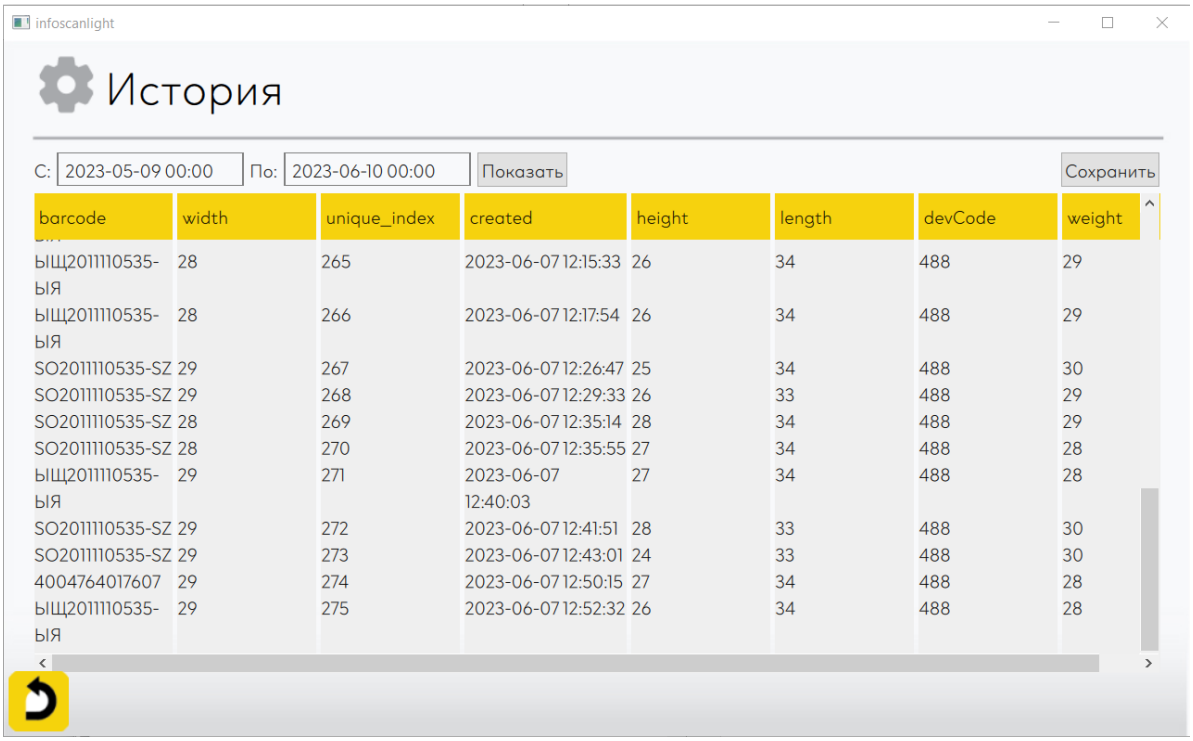
```
GET /api/data HTTP/1.1
Host: 192.168.10.41

HTTP/1.1 200 OK
```

Настройка шаблона jsScriptHistorySensor

Запрос данных с устройства методом GET на URL адрес устройства http://IP/api/data (для Linux) или http://IP:8000/api/history (для Windows) в режиме реального времени согласно шаблону

Шаблон вывода данных

```
return {
  historySensor: function() {
    var obj = {
      width: SENSOR.widthFormat,
      height: SENSOR.heightFormat,
      length: SENSOR.lengthFormat,
      weight: SENSOR.weightFormat,
      barcode: BARCODE,
      devCode: DEV_CODE,
      _order: ['barcode', 'width', 'unique_index', 'created', 'height', 'length', 'devCode',
'weight']
    };
    return obj
  }
}
```



barcode	width	unique_index	created	height	length	devCode	weight
ЫЩ2011110535-ЫЯ	28	265	2023-06-07 12:15:33	26	34	488	29
ЫЩ2011110535-ЫЯ	28	266	2023-06-07 12:17:54	26	34	488	29
SO2011110535-SZ	29	267	2023-06-07 12:26:47	25	34	488	30
SO2011110535-SZ	29	268	2023-06-07 12:29:33	26	33	488	29
SO2011110535-SZ	28	269	2023-06-07 12:35:14	28	34	488	29
SO2011110535-SZ	28	270	2023-06-07 12:35:55	27	34	488	28
ЫЩ2011110535-ЫЯ	29	271	2023-06-07 12:40:03	27	34	488	28
SO2011110535-SZ	29	272	2023-06-07 12:41:51	28	33	488	30
SO2011110535-SZ	29	273	2023-06-07 12:43:01	24	33	488	30
4004764017607	29	274	2023-06-07 12:50:15	27	34	488	28
ЫЩ2011110535-ЫЯ	29	275	2023-06-07 12:52:32	26	34	488	28