# Раздел 8 Реальные проекты IIoT на основе SAP HANA

## 8. 2 Система оперативного управления и диспетчеризации ГПС механообработки

В данной работе рассматривается создание производственно-исследовательской системы поддержки ЖЦП сложного изделия, назначение которой – обеспечить доступ максимального количества пользователей к высокотехнологическому оборудованию и мощным компьютерным системам с минимальными затратами.

В отличие от бытового Интернет вещей (IoT -Internet of Things,) проблема интеграции производственных систем (ПС) имеет многолетнюю историю, которая начинается с 70-80-х годов и реализуется различными методами от промышленных шин до методов интеграции процессов в ERP системах.

Существует ряд проблем, которые мешают широкому внедрению IIoT (как у нас в стране, так и за рубежом):

Не разработана теория управления «Производства будущего». Необходимо определить, на каком уровне (исполнительный механизм, единица оборудования, производственный модуль или вся система в целом) функции управления эффективно реализовывать в облачной платформе. Управление ПС должно быть максимально автоматизировано, что предполагает интеллектуальные, мультиагентные системы, которые могут адаптироваться к требуемой конфигурации ПС и ее окружения.

Не разработана теория информационного моделирования производственных задач, методами, позволяющими эффективно обмениваться данными, без использования интерфейсов , разработанных вручную. Концепция объектно-ориентированных CALS-технологий, появившаяся в 90-е годы, в настоящее время уступает место методам онтологического моделирования предметных областей.

Не определена область применения IIoT среди других современных методов интеграции. До сих пор для подключения устройств индустриальной автоматизации к сетям использовались такие специализированные протоколы, как интерфейсы типа RS232, RS485, ControlNet, Profibus и Modbus Plus, CAN –технологии и др. Интеграция систем уровня предприятия и выше реализовывалась на основе COM-технологий, OPC- технологий, SOA и др.

Существует ряд IIoT платформ, которые обеспечивают и работу через интернет и управление облачными вычислениями. Проблема в том, что сейчас не ясно, какие именно IIoT-платформы станут доминирующими.

**Для решения вышеперечисленных проблем предполагается создать в МГТУ им. Н.Э. Баумана Лабораторию промышленного интернета (ЛПИ).**

**Назначение ЛПИ**:

1. Исследование эффективности и выбор наилучших решение по построению промышленных IIoT систем интеграции и управления производством на основе современных методов, технологий и оборудования.
2. Тестирование различных IIoT платформ и методов управления.
3. Создание и эксплуатация учебных курсов промышленных IIoT систем интеграции и управления производством для слушателей в рамках РФ.
4. Разработка конкурентоспособного программного обеспечения и технических средств интеграции промышленных IIoT систем интеграции и управления производством (на уровне патентов РФ и выше)

Можно сказать, что создаваемая система обладает одновременно свойствами и «цифровой» и «умной» систем.

### Состав ГПС

Основой для построения производственно-исследовательской системы послужила существующая на кафедре «Компьютерные системы автоматизации производства» МГТУ им. Н.Э. Баумана система управления ГПС «Denford» и облачная платформа SAP HANA Cloud Platform (SAP HCP) [

ГПС «Denford» состоит из токарного и фрезерного производственных модулей (станок с ЧПУ, робот-манипулятор и накопитель), автоматизированного склада с штабелером, конвейера и системы управления.

Исходя из области применения в АСУ предприятия в ЛПИ должны быть реализованы несколько уровней управления:

нижний уровень — отдельные контроллеры

полевые шины (fieldbus)

средний уровень — цеховые сети;

— уровень работы систем типа SCADA;

уровень АСУП(ERP) — уровень управления ресурсами предприятия;

— уровень инженерной подготовки производства

— уровень взаимодействия между предприятиями.

**Научная новизна**:

1. Построение концепции управления промышленных IIoT систем интеграции и управления производством использованием онтологического моделирования и интеллектуального управления.
2. Разработка и исследование алгоритмов и методов промышленных IIoT систем интеграции и управления производством использованием онтологического моделирования и интеллектуального управления.
3. Разработка учебных пособий, научных монографий по теме промышленных IIoT систем интеграции и управления производством использованием онтологического моделирования и интеллектуального управления.

**Основные функции :**

1. ‣ Дистанционное управление производственным оборудованием, реализация функций SCADA систем в облачной платформе
2. ‣ Гибкое конфигурирование оборудования для решения конкретных производственных задач
3. ‣ Мониторинг выполнения производственных процессов в реальном времени
4. ‣ Контроль состояния оборудования (включая обработку сигналов датчиков и геопозиционирование)
5. ‣ Организация взаимодействия пользователей и программных компонентов ‣ Обработка больших массивов (Big Data) в облачной платформе
6. ‣ Взаимодействие с системами управления предприятием и жизненным циклом продукции (ERP, CAD/CAM/CAE, MES)

ГПС состоит из двух роботов-манипуляторов RV-2AJ компании Mitsubishi Electric токарного и фрезерного станка, конвейера, склада и системы видеонаблюдения (рис. 1).



Рис.1 Состав ГПС

Каждое оборудование управляется из облака с помощью одноплатного компьютера Raspberry Pi. На платах Raspberry Pi находится скрипт на языке программирования Python, который автоматически запускается при их включении в сеть (рис. 2).



### Основные функции ГПС

## Функциональная диаграмма IDEF0 исследуемой ГПС после внедрения промышленного интернета вещей с использованием облачных решений.

Верхний уровень диаграммы представлен на Рис. 3.1.

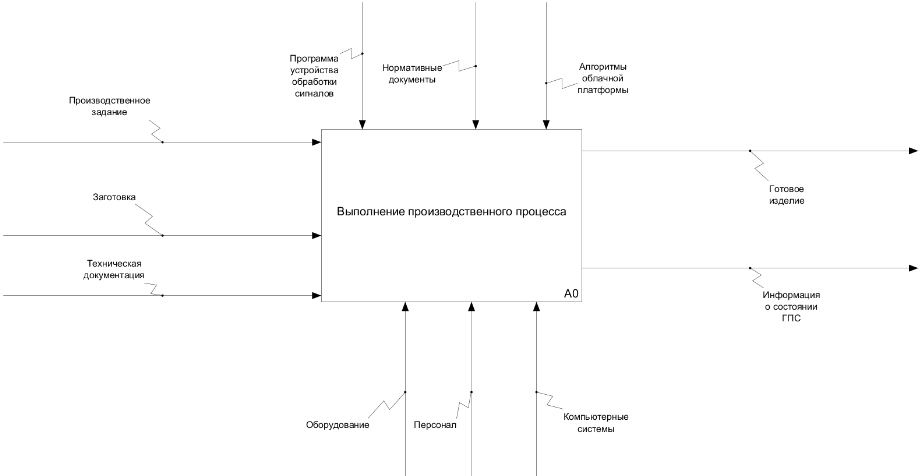


Рис. 3.1 Функциональный блок «Выполнение производственного процесса»

Из рисунка видно, что ПО управляющих компьютеров заменяется на алгоритмы облачной платформы и программу устройства обработки сигналов, а на выходе остаётся только готовое изделие и информация о состоянии ГПС, вместо информации о состоянии станка и общем состоянии системы.

Декомпозиция функционального блока «Выполнение производственного процесса» указана на Рис. 3.2.

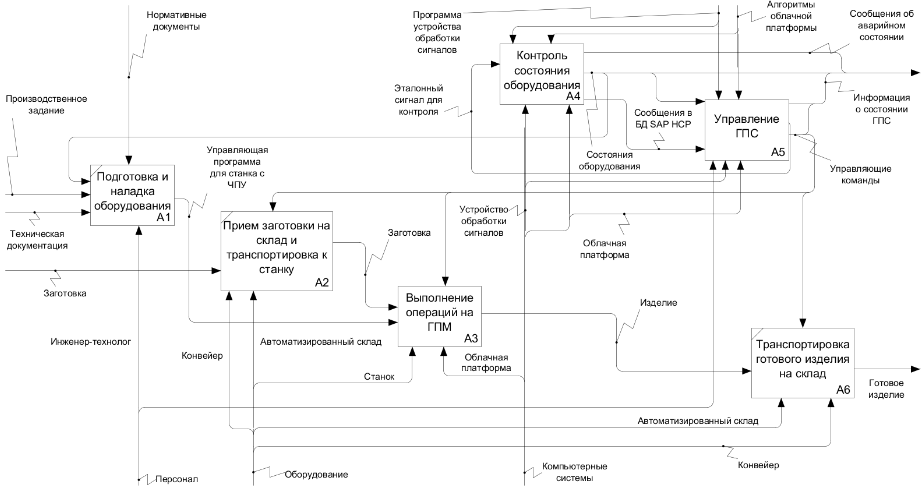


Рис. 3.2. Декомпозиция функционального блока «Выполнение производственного процесса»

Функциональные блоки на декомпозиции блока «Выполнения производственного процесса не изменились».

Декомпозиция функционального блока «Контроль состояния оборудования» указана на Рис. 3.3.

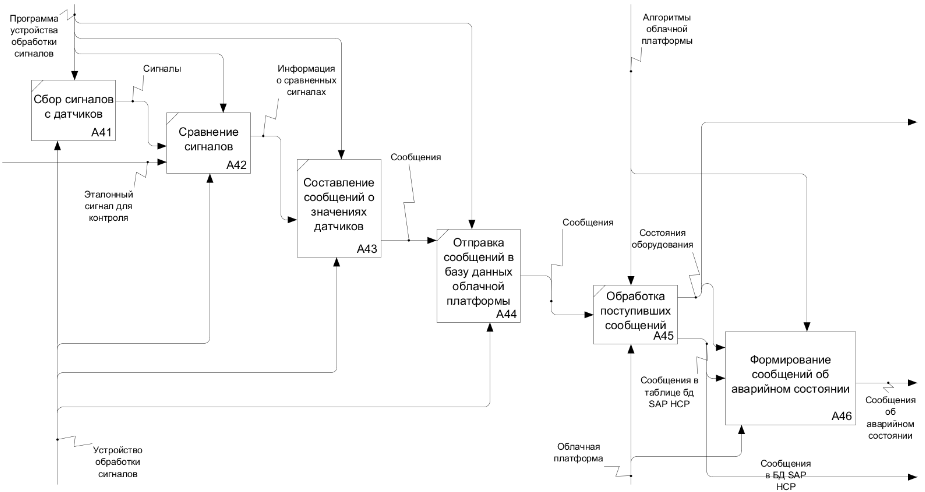


Рис. 3.3. Декомпозиция функционального блока «Контроль состояния оборудования»

По сравнению с контролем состояния до подключения ГПС к облачной платформе, теперь нет необходимости собирать сигнал отдельно со станка и отдельно с конвейера и манипуляторов, теперь данные с датчиков собираются вместе, добавился эталонный сигнал, после сравнения сигналов с датчиков с эталонным сигналом формируются сообщения для отправки в базу данных облачной платформы. В облачной платформе эти сообщения обрабатываются и заносятся в таблицу, также предусмотрен случай с сообщением об аварийном состоянии.

Декомпозиция функционального блока «Управление ГПС» указана на Рис. 3.4.

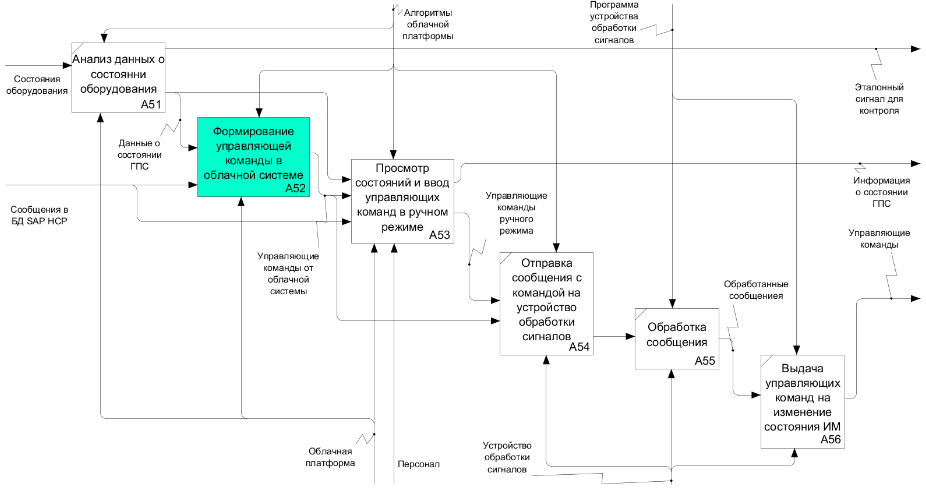


Рис. 3.4. Декомпозиция функционального блока «Управление ГПС»

На этапе управления ГПС с внедрённым облаком, управляющие команды начали формироваться непосредственно в облаке, на основе данных о состоянии ГПС, присылаемых с датчиков. После формирования команд, они отправляются на устройства обработки сигналов, а те в свою очередь, выдают управляющие команды исполнительных механизмам.

Декомпозиция функционального блока «Формирование управляющей команды в облачной системе» указана на Рис. 3.5.

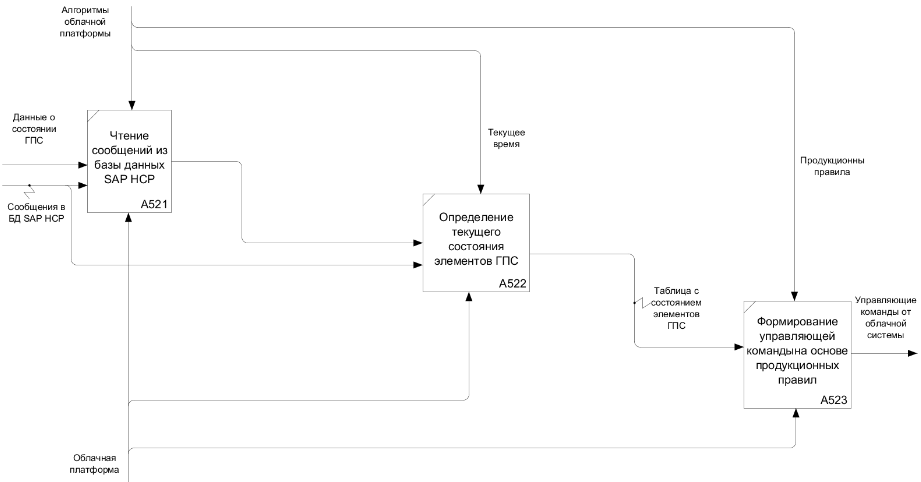


Рис. 3.5. Декомпозиция функционального блока «Формирование управляющей команды в облачной системе»

Схема принятия решения согласно плану

Объекты управления при диспетчеризации, это — ГТМ, склад и конвейер. Взаимодей­ствие между перечисленными организационными частями ГПС реализуется в интервале времени, равном смене. Это связано с тем, что операции комплектации наладок, сборка-разборка спутников и другие требуют в пределах смены значительного вре­мени и поэтому должны выполняться, а значит, и планироваться с опережением по отношению к процессу диспетчеризации, Таким образом, при оперативном управлении ГПС необходимо иметь как минимум два горизонта управления: в реальном масштабе времени и в разрезе смены — сменно-суточное планирование.

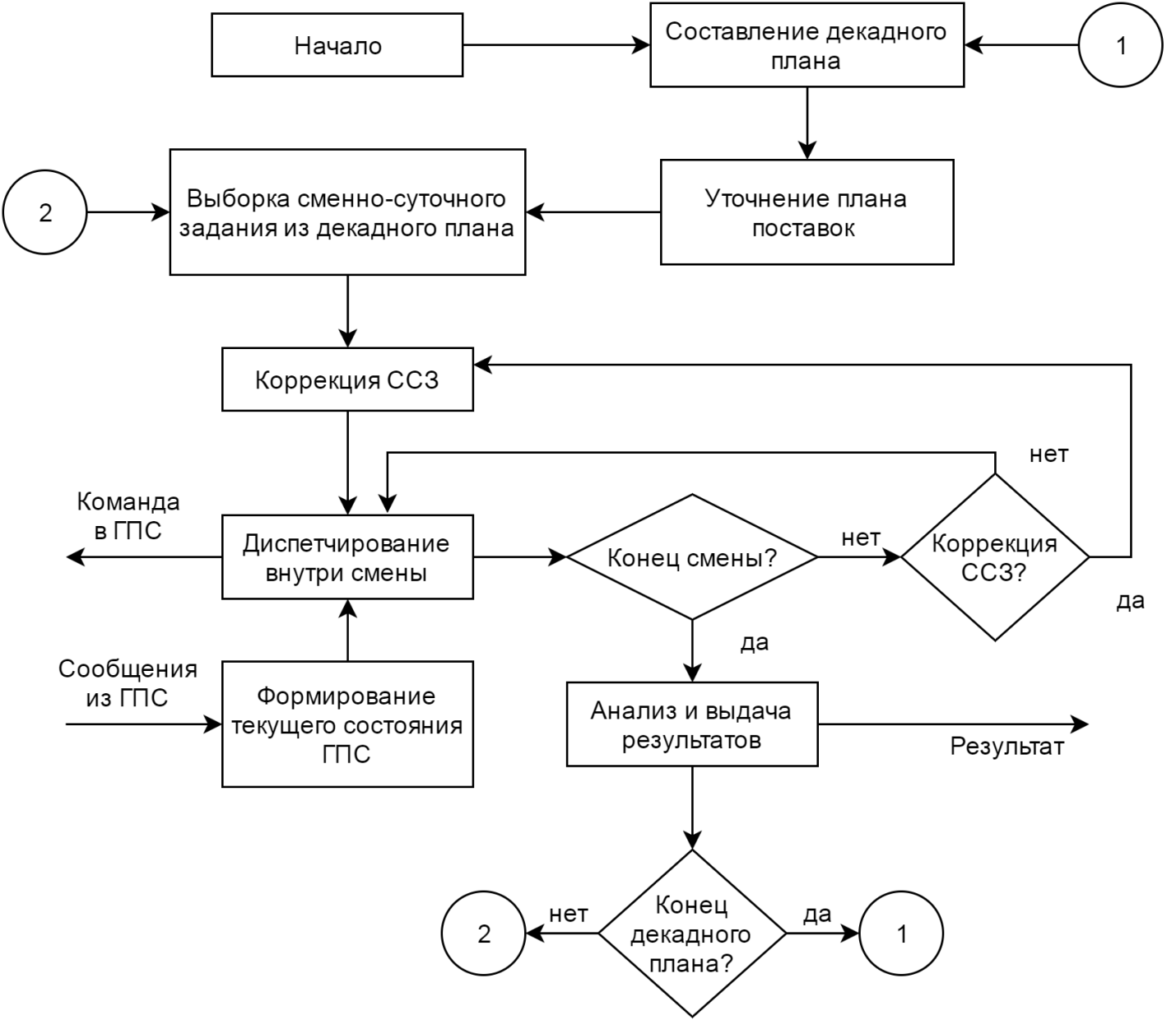


Рис. 3. Алгоритм оперативного управления ГПС

На этапе диспетчеризации производственного процесса его математическая модель детерминированы. Формирование управления для каждой возникшей производственной ситуации должно осуществляться за короткий промежуток времени. В основе методов диспетчеризации лежат правила, использующие только данные о состоянии ПС и плановых заданиях на рассматриваемый момент времени.

При вводе команд для ГТМ проверяется возможность выпол­нения вводимой команды: соответствие состояния ресурсов, ука­занных в команде, их допустимым значениям для данной операции. Например, для операции обработки проверяется наличие передаточной партии заготовок на входном порту ГТМ; работоспособность и наладка станка; стойкость комплекта инструмента.

Если состояние ресурсов соответствует допустимому, вводимая команда обрабатывается и передается на модуль, если какой-либо ресурс отсутствует, выдается ответ оператору.

Под управляющей командой понимается партия информации, передаваемый в XS Server, в котором производится синтаксическая проверка на соответствие типа переменной и формата. При диалоге преду­смотрена возможность повторного ввода при выявлении ошибки. Семантическая проверка осуществляется на нахождение значе­ний параметров команды в области их определения, на верность логических связей между параметрами и на возможность или не­возможность выполнения команды.

Таблица состояний

Таблица состояний элементов ГПС.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | TimeStamp | number | value | crash |  |
| Станок | t | N(1,2,3) | 1 | N(0,1) | начало выполнения программы |
| Станок | t | N(1,2,3) | 0 | N(0,1) | окончание выполнения программы |
| Станок | t | 0 | 0 | N(0,1) | свободен |
| Патрон | t | 0 | 1 | N(0,1) | закрыт |
| Патрон | t | 0 | 0 | N(0,1) | открыт |
| Робот | t | N(1,2,3,4,5) | 1 | N(0,1) | занят |
| Робот | t | 0 | 0 | N(0,1) | свободен |
| Ключ | t | N(паллеты) | 1 | N(0,1) | есть паллета, закрыт |
| Ключ | t | 0 | 1 | N(0,1) | нет паллеты, закрыт |
| Ключ | t | 0 | 0 | N(0,1) | открыт |
| Штабелёр\_1 | t | N(паллеты) | 0 | N(0,1) | нахождение(с паллетой или без) на складе |
| Штабелёр\_1 | t | N(паллеты) | 1 | N(0,1) | нахождение(с паллетой или без) на конвейере |
| Штабелёр\_2 | t | 0 | 1 | N(0,1) | занят |
| Штабелёр\_2 | t | 0 | 0 | N(0,1) | свободен |
| Паллета | t | 0 | 0 | N(0,1) | паллета с заготовкой |
| Паллета | t | 0 | 1 | N(0,1) | паллета с деталью |

Таблица команд для ГПМ

|  |  |
| --- | --- |
| Перечень команд ГПМ | 1. Выполнить программу |
| 2. Прекратить выполнение |
| 0. открыть патрон |
| 1. закрыть патрон |
| 1. Роботу Взять заготовку на паллете |
| 2. Поместить в станок |
| 3. Взять в станке |
| 4. поставить на паллету |
| 5. занять исходное положение |

Таблица команд для конвейера

|  |  |
| --- | --- |
| Перечень команд ключа | 0. открыть ключ |
| 1. закрыть ключ |

Таблица команд для штабелёра

|  |  |
| --- | --- |
| Перечень команд штабелёра | 0. переместиться на склад |
| 1. переместиться на конвейер |

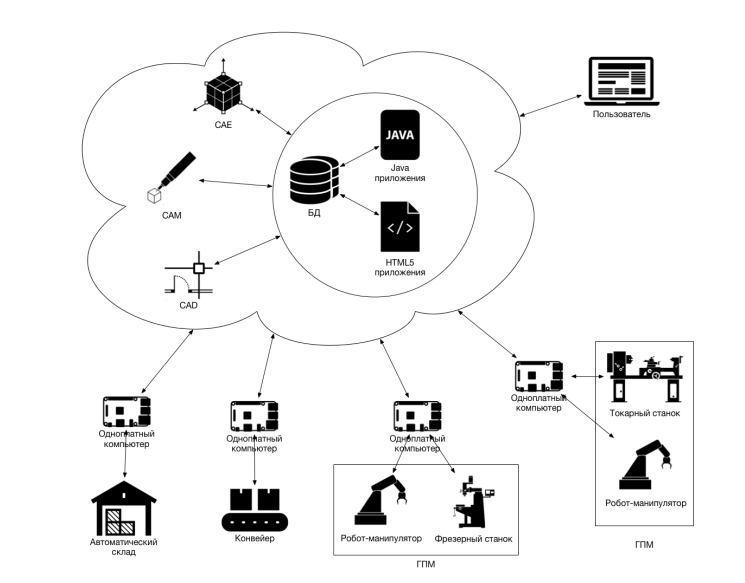
Таблица 5.7. Начальное состояние при запуске

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | TimeStamp | number | value | crash |  |
| Станок | t | 0 | 0 | 0 | свободен |
| Патрон | t | 0 | 0 | 0 | открыт |
| Робот | t | 5 | 0 | 0 | свободен |
| Ключ | t | 0 | 0 | 0 | открыт |
| Штабелёр\_1 | t | 0 | 0 | 0 | нахождение(с паллетой или без) на складе |
| Штабелёр\_2 | t | 0 | 0 | 0 | свободен |
| Паллета | t | 0 | 0 | 0 | паллета с заготовкой |

Таблица 5.8. Управляющие команды

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | TimeStamp | number | value | crash |  |
| Станок | t | 1 | 1 | 0 | Команда 1 (станок выполняет программу) |
| Станок | t | 0 | 0 | 0 | Команда 2 (станок прекращает выполнение) |
| Патрон | t | 0 | 1 | 0 | Команда 3 ( закрыть патрон) |
| Патрон | t | 0 | 0 | 0 | Команда 4 (открыть патрон) |
| Робот | t | 1 | 1 | 0 | Команда 5 (взять заготовку на паллете) |
| Робот | t | 2 | 1 | 0 | Команда 6 (поместить в станок) |
| Робот | t | 3 | 1 | 0 | Команда 7( взять в станке) |
| Робот | t | 4 | 1 | 0 | Команда 8 (поставить на паллету) |
| Робот | t | 5 | 1 | 0 | Команда 9( в исходное состояние) |
| Ключ | t | N(паллеты) | 1 | 0 | Команда 10 (закрыть ключ перед паллетой N) |
| Ключ | t | 0 | 0 | 0 | Команда 11 ( открыть ключ) |
| Штабелёр\_1 | t | N(паллеты) | 0 | 0 | Команда 12( переместиться на склад) |
| Штабелёр\_1 | t | N(паллеты) | 1 | 0 | Команда 13 (переместиться на конвейер) |

### Структура проекта облачного IIoT для ГПС



облачной системы управления производством можно условно разделить на части:

* Приложения в облаке;
* Клиентская часть на физических устройствах.

В облаке размещается HTML-шаблон пользовательского интерфейса, который отформатирования с помощью каскадных таблиц стилей CSS. К HTML-шаблону подключена JavaScript-библиотека JQuery, реализующая функции создания AJAX-запросов.

В облачной части, так же, находятся модуль отображения интерфейса пользователя и модуль работы с данными. Данные модули написаны на языке JavaScript с использование библиотеки JQuery.

В SAP HANA Cloud Platform Cockpit присутствует модуль создания внешних запросов, который служит прослойкой для работы с кросс-доменными запросами с помощью технологии AJAX.

На одноплатных компьютерах Raspberry Pi размещается WebSocket-клиент, который инициирует соединение с облаком SAP HANA Cloud.

Также на одноплатных компьютерах находится модуль обработки команд, который преобразует команды, приходящие из облака в управляющие программы для производственного оборудования.

Завершает цепочку модуль работы с физическими протоколами, который реализует непосредственно отправку управляющих программ на производственное оборудования, считывание их текущего состояния, считывание сигналов датчиков и т.д. Данные режимы работы реализуются посредством стандартного набора протоколов.

**Протоколы взаимодействия**

Соединение устройства с облаком SAP HANA происходит через защищённое соединение по протоколу WebSocket (wss://). Соединение инициируется управляющим компьютером и поддерживается на протяжении всего времени работы устройства. Данная схема подключения в отличие от соединения по протоколу HTTP (с которым также можно работать в сервисе IoT в SAP HANA) позволяет избавиться от необходимости управляющему компьютеру иметь статический IP адрес и значительно ускоряет процесс развертывания системы — достаточно наличия любого подключения к сети интернет.

Программно WebSocket-соединение реализовано с помощью библиотеки WebSocket-client для Python.

Дополнительно была реализована на одноплатных компьютерах функция автоматического переподключения в случае обрыва соединения.

Формат сообщений

Сообщения формируются в формате JSON.

Пример типичного сообщения:

{

"messageType":"m0t0y0p0e3",

"messages":

[

{

"cmd":"switch on",

"operand":"led 1"

}

**Клиентская часть на физических устройствах**

В качестве управляющего компьютера был применен одноплатный компьютер Raspberry Pi 3. Одноплатный компьютер выполняет роль связующего звена между облаком и производственным оборудованием.

На платах Raspberry Pi запускается специальный скрипт, реализованный на языке программирования Python, который решает задачи связанные с управлением оборудованием и получением его состояния.



Рис. 8.11. Одноплатный компьютер Raspberry Pi

Протоколы взаимодействия Raspberri Pi с оборудованием

Одноплатный компьютер управляет производственным оборудованием через стандартные интерфейсы посредством набора стандартных протоколов.

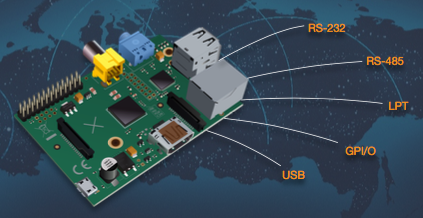


Рис. 8.12. Стандартные интерфейсы управления производственным оборудованием

Такими интерфейсами могут быть:

● RS-232

● RS-485

● LPT

● GPI/O

● USB

Часть из перечисленных интерфейсов может быть реализована через преобразователь, подключаемый к порту USB.

Применяемые в проекте протоколы выбирались исходя из особенностей производственного оборудования. В нашем случае с управлением лабораторной ГПС Denford управление непосредственно оборудованием происходит через COM-порт по протоколу RS-232. Программно управление портом осуществляется с помощью библиотеки Pyserial для Python.

Данные о текущем состоянии оборудования (занятость) считываются через интерфейс ввода/вывода общего назначения (GPIO) на Raspberry Pi. Программно управление интерфейсом осуществляется с помощью библиотеки RPi.GPIO для Python.

Стандарт RS-232 (Recommended Standard 232) - стандарт описывающий интерфейс для последовательной двунаправленной передачи данных между терминальным оборудованием ввода данных ( DTE, Data Terminal Equipment) и конечным устройством (DCE, Data Circuit-Terminating Equipment).

Одноплатный компьютер Raspberry Pi является терминалом, а конечными устройствами являются токарный станок и робот-манипулятор.

В качестве разъема для интерфейса RS-232 на нашем оборудовании располагаются разъемы DE9s (COM-порт).

Протокол имеет ряд переменных параметров, которые должны быть приняты одинаковыми на стороне приемника и на стороне передатчика для успешного обмена данными.

Интерфейс GPIO

Интерфейс GPIO (General Purpose Input-Output) - Это интерфейс ввода-вывода прямого управления. На плате Raspberry Pi находится в углу в виде гребенки из 40 штрихов рядом с видеовыходом. Через интерфейс Raspberry Pi получает и отдает команды внешним устройствам(робот, станок). В рамках проекта данный интерфейс используется для чтения информации о состоянии производственного оборудования.

В этом интерфейсе аппаратно реализованы некоторые базовые промышленные интерфейсы управления оборудованием, такие как: шина I2C, порт RS-232, шина SPI.

На рисунке 5.1.2.1 представлены порты, расположенные на плате

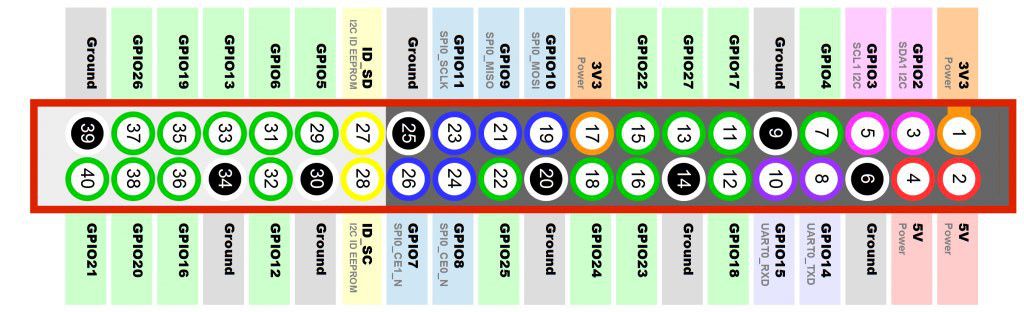
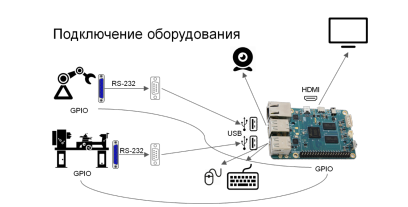


Рис. 8.13 Порты для интерфейса GPIO

### Канал гибкого производственного модуля



### Канал транспортной системы (конвейер)

### Канал автоматизированного складского модуля

### Облачная часть . Диспетчеризация. Интерфейсы и обработка статистики.

Серверная часть. Алгоритм диспетчеризации на основе продукционных правил

Необходимо реализовать обмен информацией(командами и сообщениями), чтобы синхронизировать действие всех элементов ГПС. Система диспетчеризации должна выполнять указанные на Рис. 2.1. функции:

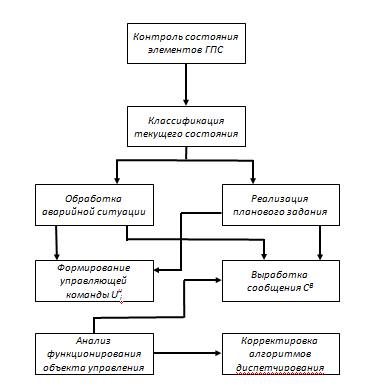


Рис. 2.1. Функции системы диспетчеризации

. Для централизованной системы диспетчеризации ГПС характерно наличие мощной центральной диспетчерской службы (ЦДС) на которую передаётся большинство функций по управлению ГПС в реальном времени. ЦДС производится анализ состояния ГПС, определяет действия для каждого модуля, планирует их начало. Кроме того, выдаётся ряд сообщений о приёме команды, сбоях при выполнении команды. Особенностью решения задачи диспетчеризации ГПС является необходимость определения следующего выполняемого действия сразу после окончания предыдущего. Для реализации централизованной системы диспетчеризации, представляющую собой интеллектуальную систему продукционного типа, нужно установить следующие положения:

Производственный процесс в ГПС представляется как упорядоченное множество основных операций и неупорядоченное множество вспомогательных операций

Операций описываются тремя множествами допустимых значений ресурсов ГПС, в них участвующих: до, во время и после операции

Для элемента ГПС, который выбран в качестве ресурса действия и значения параметров которого не соответствуют множеству, строится ветвь сети, изменяющая состояние ресурса с текущего на требуемое.

Система диспетчеризации организуется как интеллектуальная система, состоящая из системы принятия решений(системы вывода), базы данных, описывающей цель функционирования ГПС и текущее состояние её элементов, а также базы знаний, содержащей описания операций в форме «правил продукции».

Основной алгоритм программы представлен на Рис. 5.1.

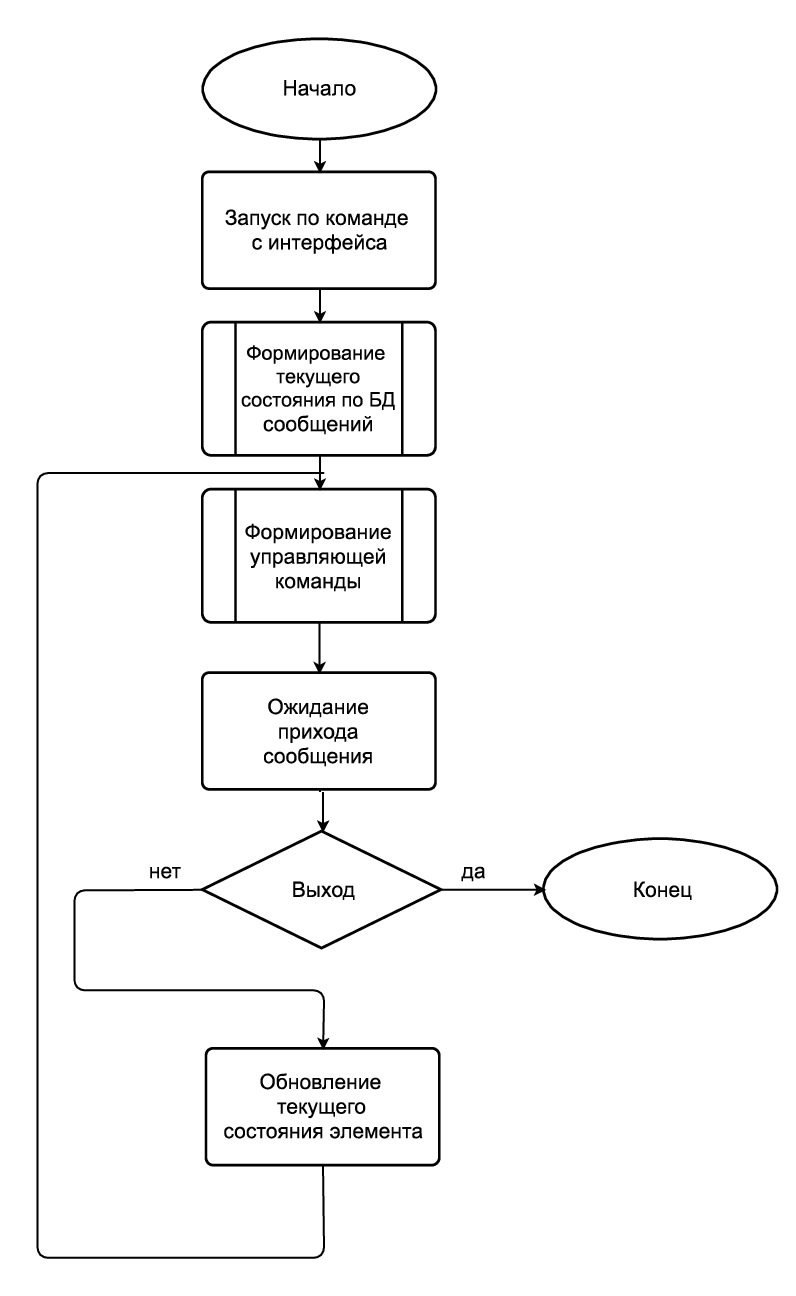


Рис. 5.1. Основной алгоритм работы программы

Алгоритм формирования управляющей программы представлен на Рис. 5.2.

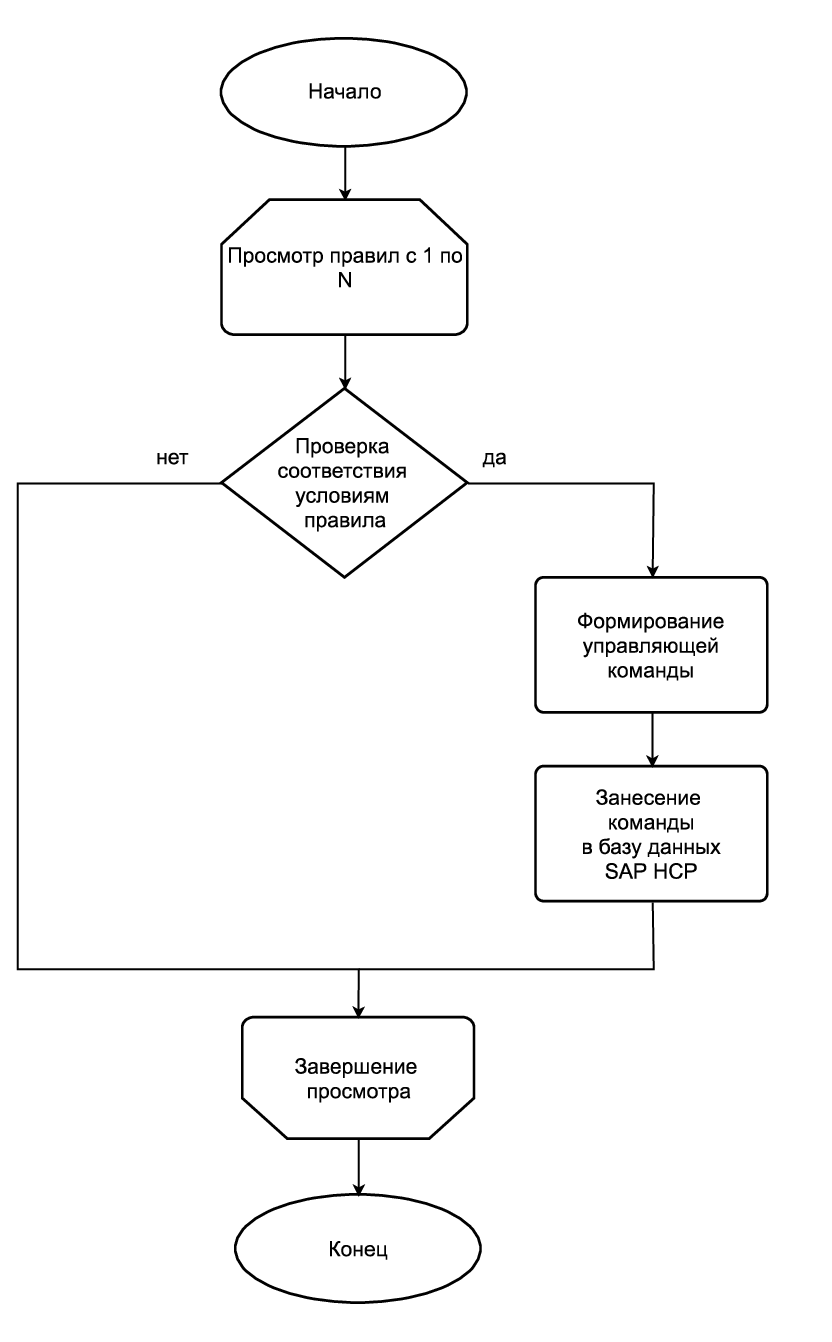


Рис. 5.2. Алгоритм формирования управляющей команды

Алгоритм формирования текущего состояния системы представлен на Рис. 5.3.

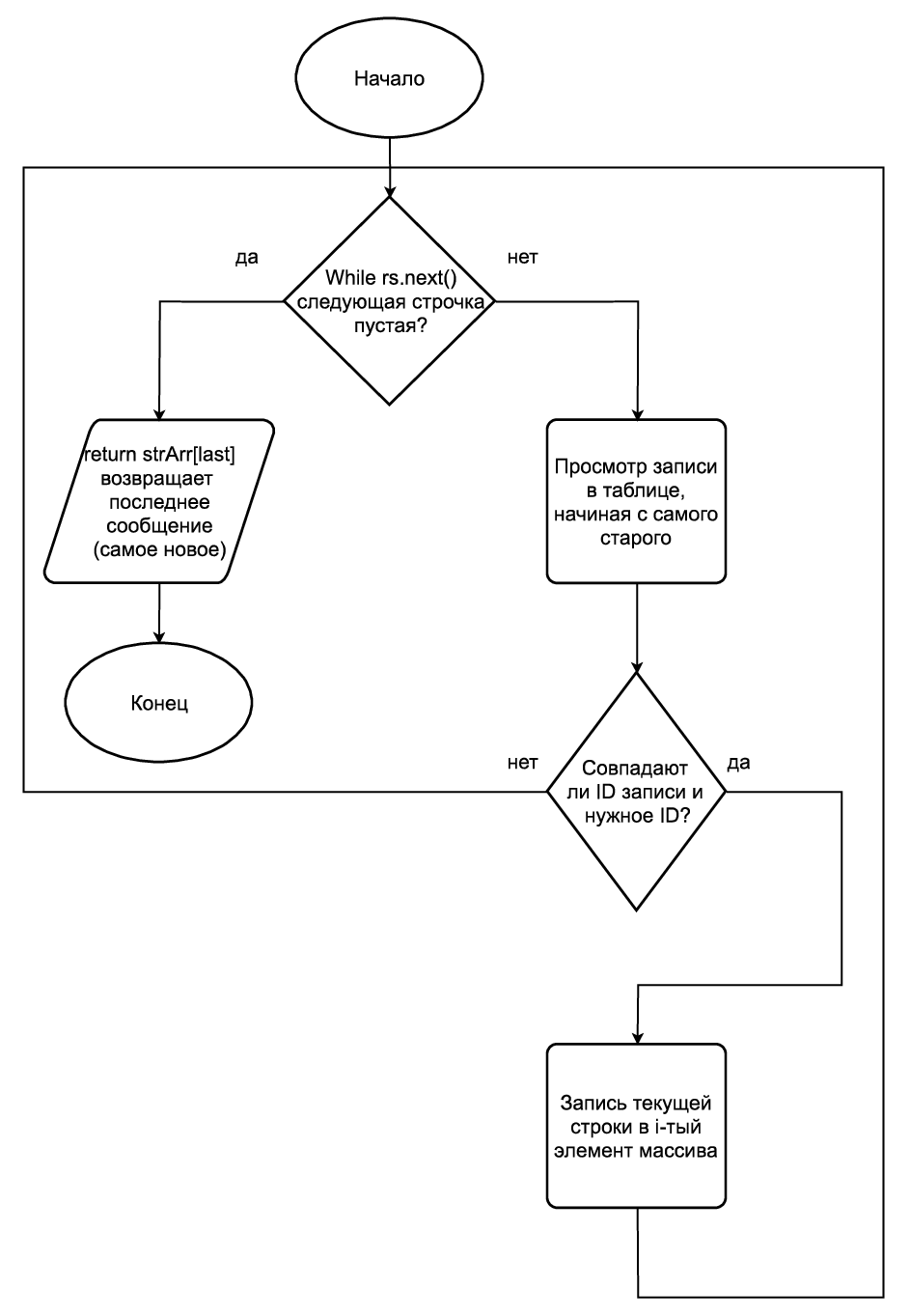


Рис. 5.3. Алгоритм формирования текущего состояния

На основе данных таблиц состояний и команд, были составлены продукционные правила, определяющие управляющую программу в зависимости от состояния системы.

1. Если <Ключ.value = 0 & Ключ.number = 0 & Штабелёр\_2.value = 0 & Штабелёр\_1.number = 0 & Робот.value =0 & Робот.number = 5 & Станок.number=0 & Станок.value = 0>, то < выполнить команду 10>
2. Если <команда 10 выполнена>, то < Ключ.value = 1, Ключ.number = N>
3. Если < Ключ.value = 1 & Ключ.number = 0 & Штабелёр\_2.value = 0 & Штабелёр\_1.number = 0 & Робот.value =0 & Робот.number = 5 & Станок.number=0 & Станок.value = 0 >, то <выполнить команду 13>
4. Если <команда 13 выполнена, то <выполнить команду 12>
5. Если < команда 12 выполнена & Ключ.value = 1 & Ключ.number = N & Робот.value =0 & Робот.number = 5 & Станок.number=0 & Станок.value= 0>,то < выполнить команду 5 >
6. Если <Ключ.number != 0 & команда 5 выполнена & Станок.number=0 & Станок.value= 0>, то <выполнить команду 6>
7. Если <Ключ.number != 0 & выполнена команда 6 & Станок.number=0 & Станок.value= 0>, то <Робот.number 9>
8. Если <Ключ.number != 0 & выполнена команда 9 & Станок.number=0 & Станок.value= 0 >, то <Робот.value 0>
9. Если <Ключ.number != 0 & Робот.number = 5 & Станок.number=0 & Станок.value= 0 >, то < выполнить команду 3>
10. Если <Ключ.number != 0 & Робот.number = 5 & команда 3 выполнена>, то < выполнить команду 1>
11. Если <Ключ.number != 0 & Робот.number = 5 & Патрон.value = 1 & команда 1 выполнена >, то < Станок.number=N, Станок.value= 0>
12. Если <Ключ.number != 0 & Робот.number = 5 & Патрон.value = 1& Станок.number=N & Станок.value= 0>, то < выполнить команду 7, Станок.value= 0>
13. Если <Ключ.number != 0 & Патрон.value = 1& Станок.number=0 & Станок.value= 0 & команда 7 выполнена>, то <выполнить команду 4>
14. Если <Ключ.number != 0 & команда 4 выполнена & Станок.number=0 & Станок.value= 0 & Робот.number = 3 & Патрон.value = 0>, то < выполнить команду 8 >
15. Если <Ключ.number != 0 & Патрон.value = 0 & Станок.number=0 & Станок.value= 0 & команда 8 выполнена >, то < выполнить команду 9, Паллета.value =1 >
16. Если <Ключ.number != 0 & Патрон.value = 0& Станок.number=0 & Станок.value= 0 & Робот.number = 5 & Паллета.value =1 >, то <выполнить команду 13, Штабелёр\_2.value= 1 >
17. Если <Ключ.number != 0 & Патрон.value = 0 & Станок.number=0 & Станок.value= 0 & Робот.number = 5 & Паллета.value =1 команда 13 выполнена& Штабелёр\_2.value= 1 >, то < выполнить команду 11 >
18. Если < Патрон.value = 0 & Станок.number=0 & Станок.value= 0 & Робот.number = 5 & Паллета.value =1 & Штабелёр\_1.number = N, Штабелёр\_1.value = 1 & Штабелёр\_2.value= 1 & команда 11 выполнена >, то < выполнить команду 12>
19. Если < Патрон.value = 0& Станок.number=0 & Станок.value= 0 & Робот.number = 5 & Паллета.value =1 & команда 12 выполнена & Штабелёр\_2.value= 1 & Ключ.number = 0, Ключ.value = 0 >, то < Штабелёр\_2.value= 0 >
20. Если <Ключ.number = 0 & Патрон.value = 0 & Станок.number=0 & Станок.value= 0 & Робот.number = 5 & Паллета.value =1 & Штабелёр\_1.number = N, Штабелёр\_1.value = 0, Штабелёр\_2.value= 0 & Ключ.number = 0, Ключ.value = 0 >, то < Штабелёр\_1.number = 0, Паллета.value =0 >

Для реализации кода была выбрана свободная интегрированная среда разработки модульных кроссплатформенных приложений Eclipse Mars с использованием специальных расширений предоставляемых компанией SAP, для взаимодействия с сервисами HANA Cloud Platform.

**Интерфесная часть**

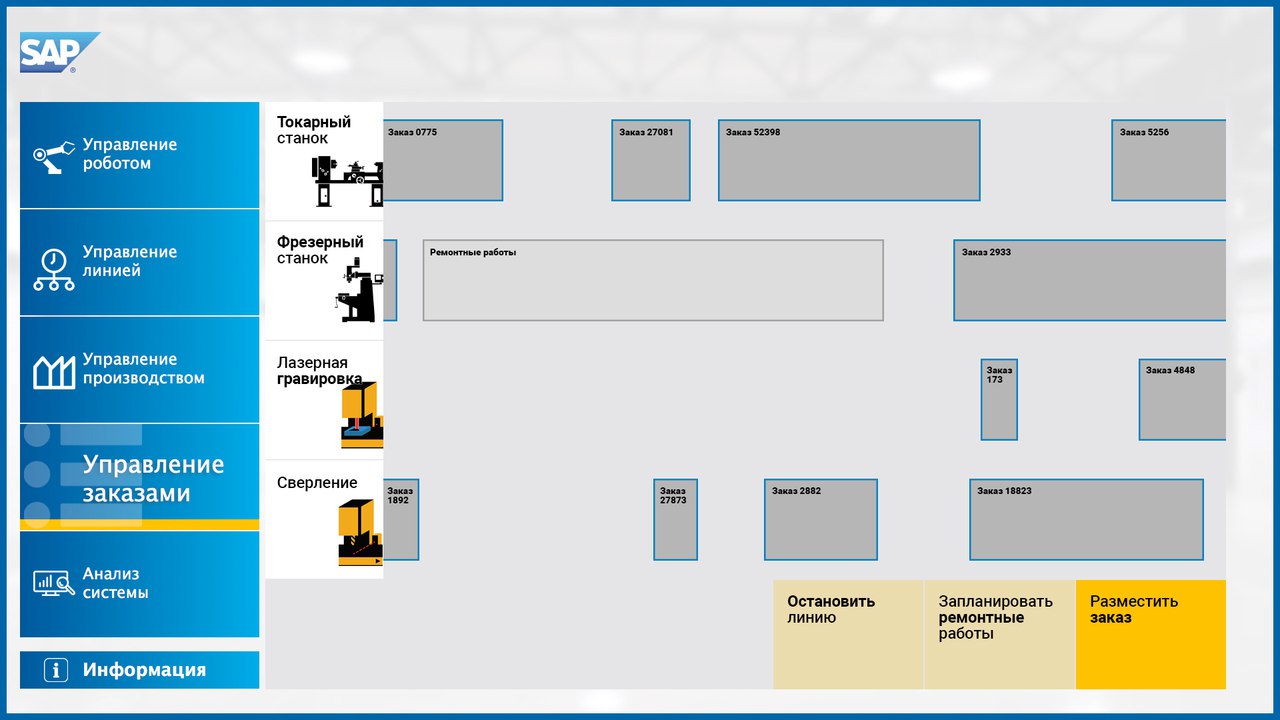


Рис. 13 Общий вид интерфейса диаграммы Ганта

## 8. 2 Система вибромониторинга в HCP

***Надо добавить введение***

Среди всех этих техник диагностики наиболее популярным является вибромониторинг т.к является наиболее эффективным и позволяет обнаружить множество различных дефектов агрегата: дисбаланс, эксцентриситет ротора, износ подшипников и зубчатых передач и многое другое.

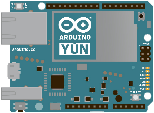
В вибромониторинге как правило измерения производятся с помощью специальных датчиков: акселерометров (датчики измерения ускорения), датчиков виброскорости или же датчиков перемещения (вихретоковые датчики). В отличии от сигнала виброскорости и виброперемещения, измерение виброускорения позволяет отслеживать высокочастотные сигналы малой мощности. Именно поэтому анализ виброускорения позволяет проводить предиктивное ТО.

Существует большое количество методик анализа сигналов вибрации, рассмотрим наиболее популярные из них:

Рисунок 1 Методы анализа вибросигнала

Наиболее информативны с точки зрения полноты и достоверности информации о техническом состоянии агрегата, являются методы вычисления среднего квадратичного значения амплитуды виброускорения и преобразование временного сигнала в частотную зону с помощью быстрого преобразования Фурье.

Архитектура системы выглядит следующим образом:

****

Измерительное устройство

**Акселерометр**

**ADXL345**

**Сервер устройства Arduino Yun**

(Edge Server)

Облачная служба SAP HCP IoT

**Служба MMS**

(служба данных, служба PUSH, служба подтверждения, обработчики входящих сообщений)

**Интерфейс**

I2C

**Сообщения**

(HTTP)

**База данных**

(SAP HANA, SAP Max DB, SAP ASE)

**Пользователь**

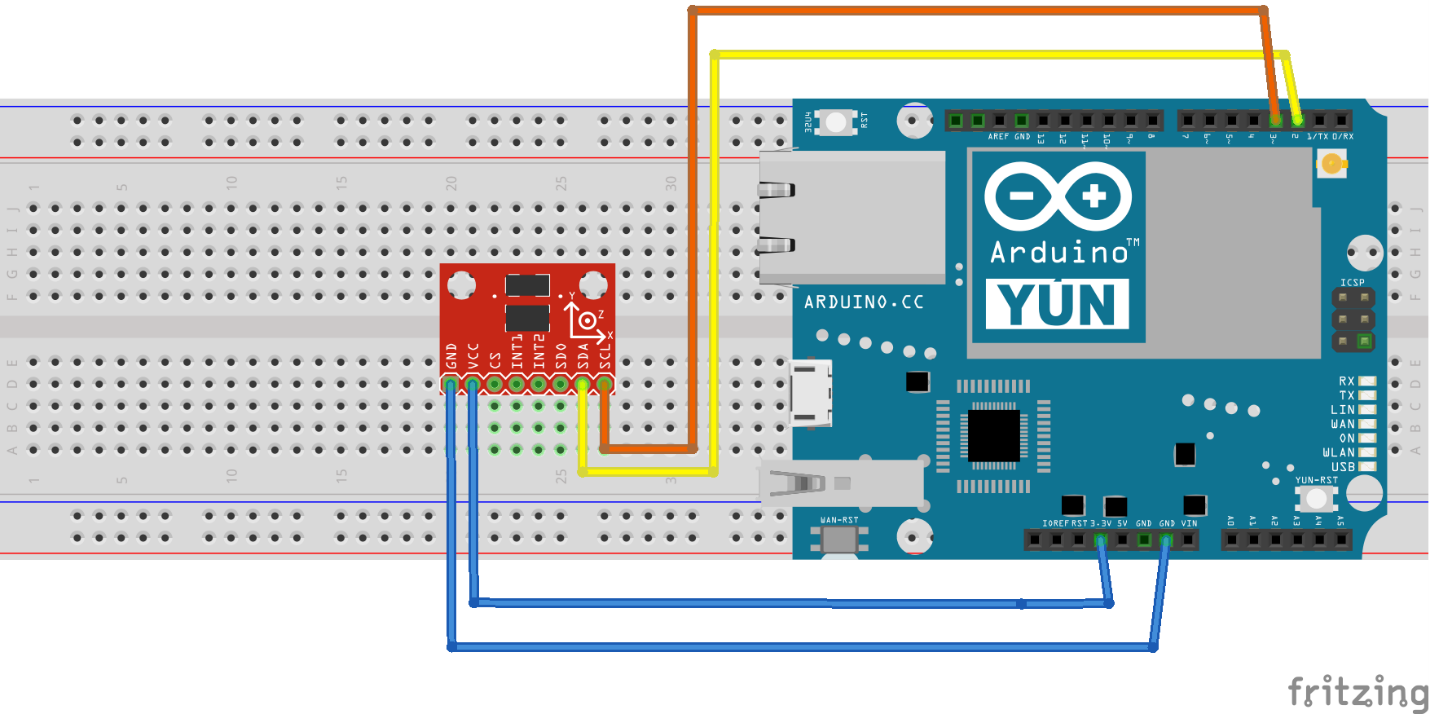
**Приложение IoT** (интерфейс)

Сохранение сообщений

Отправка PUSH-сообщений устройству и получение сообщений

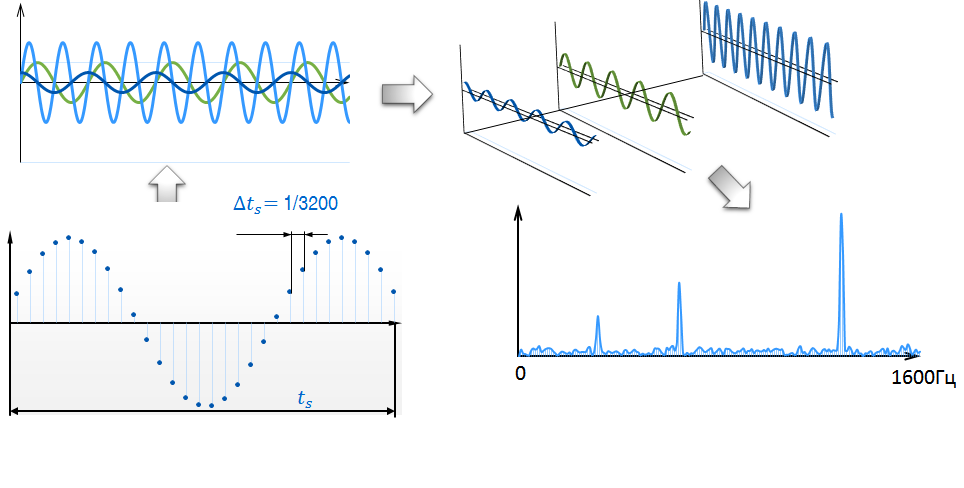
Получение сообщений

Схема подключения датчика к плате Arduino:



Полоса частот акселерометра ADXL345 ограничена 1600 Гц, что может оказаться недостаточным для некоторых машин и дефектов. В этом случае можно использовать ADXL001 с более большой полосой частот до 22 кГц и аналоговым выходом.

Для того чтобы построить спектр интересующего диапазона частот (1600 Г ц), необходимо организовать запись данных амплитуд во временном сигнале вибрации с датчика частотой 3200Гц (т.е в два раза больше). Данное условие регламентировано теорией цифровой обработки сигнала, а именно теоремой Котельникова [8.2].



Данный диапазон опроса настраивается функцией (более подробно в разделе написание программной части на Edge Server)

void setDataRate(ADXL345\_DATARATE\_3200\_HZ);

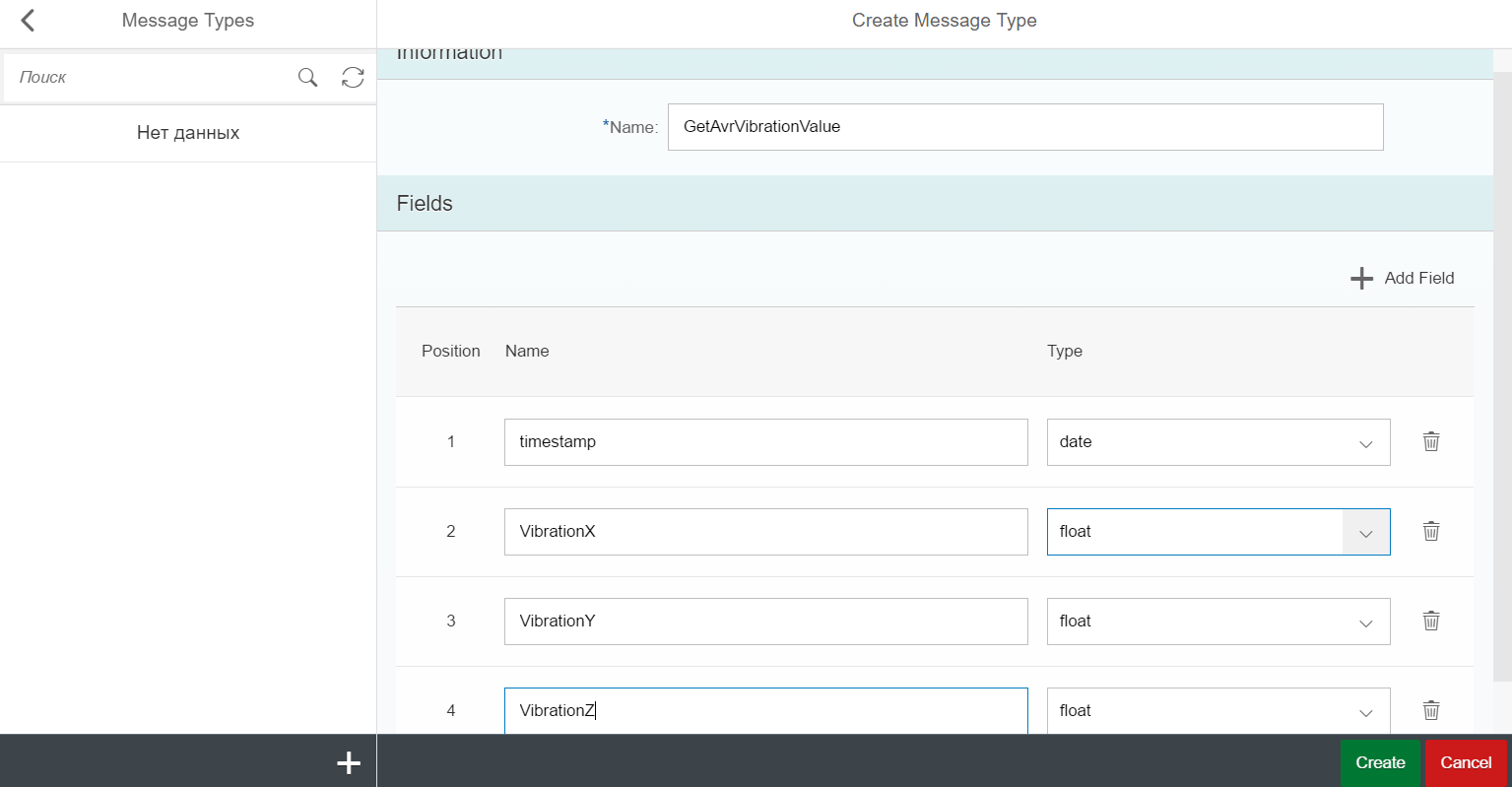
следующий обмен сообщениями для построения облачной системы вибромониторинга:

1. **Передача средних значений амплитуд вибраций по осям X, Y, Z (*GetAvrVibrationValue*).**

*Тип сообщения*

|  |  |  |  |
| --- | --- | --- | --- |
| Позиция | Имя поля | Тип информации | Пояснение |
| 1 | Timestamp | date | Временная метка сообщения |
| 2 | VibrationX | Float | Ср.значение амплитуды по оси Х |
| 3 | VibrationY | Float | Ср.знаечние амплитуды по оси Y |
| 4 | VibrationZ | Float | Ср.значение амплитуды по оси Z |

Определяем данный тип сообщения в службе IoT Service:

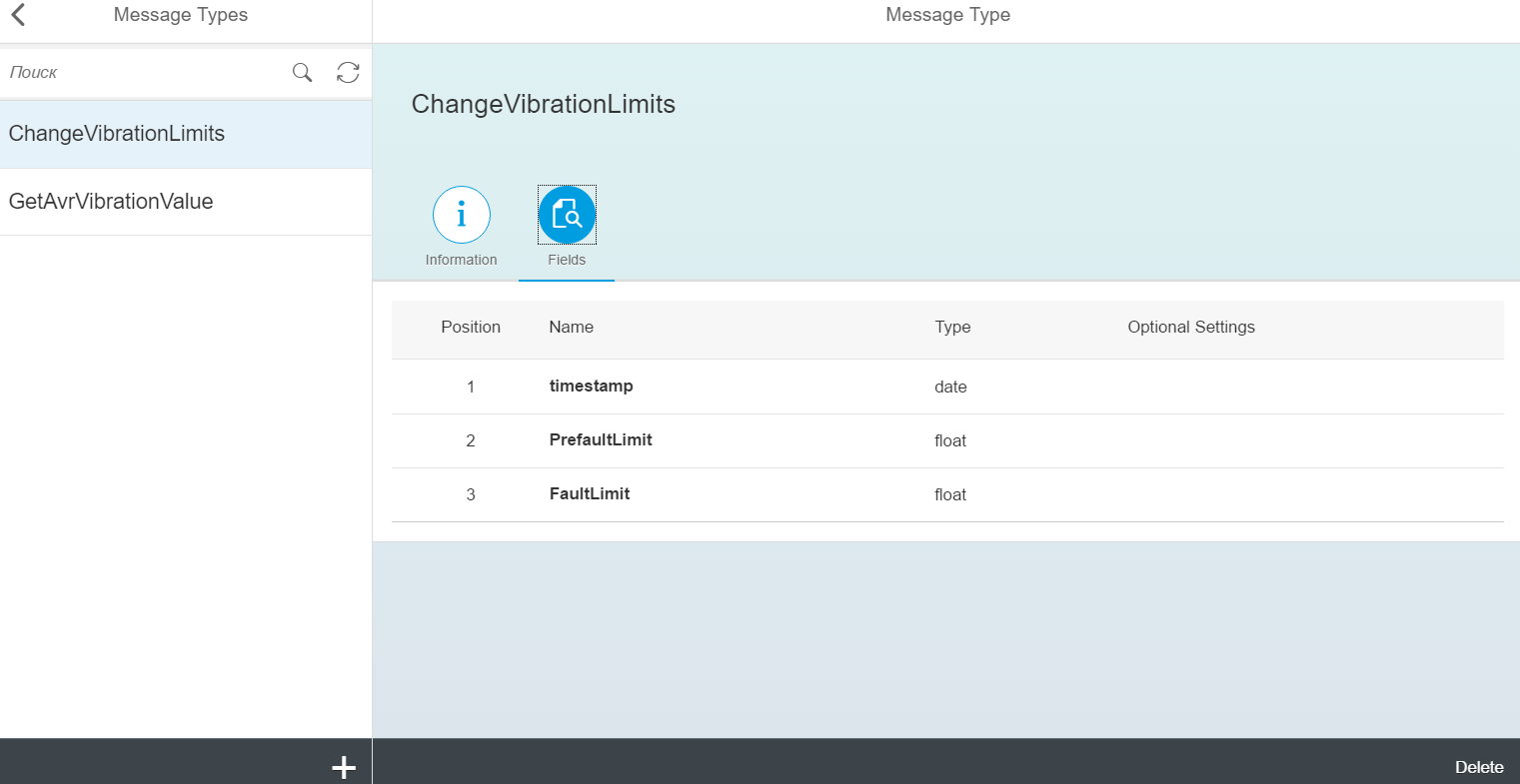


ID сообщения: a371657a832e78082613

1. **Изменение предельных значений амплитуд**

*Тип сообщения*

|  |  |  |  |
| --- | --- | --- | --- |
| Позиция | Имя поля | Тип информации | Пояснения |
| 1 | Timestamp | date | Временная метка сообщения |
| 2 | PrefaultLimit | Float | Новый предаварийный уровень |
| 3 | FaultLimit | Float | Новый аварийный уровень |

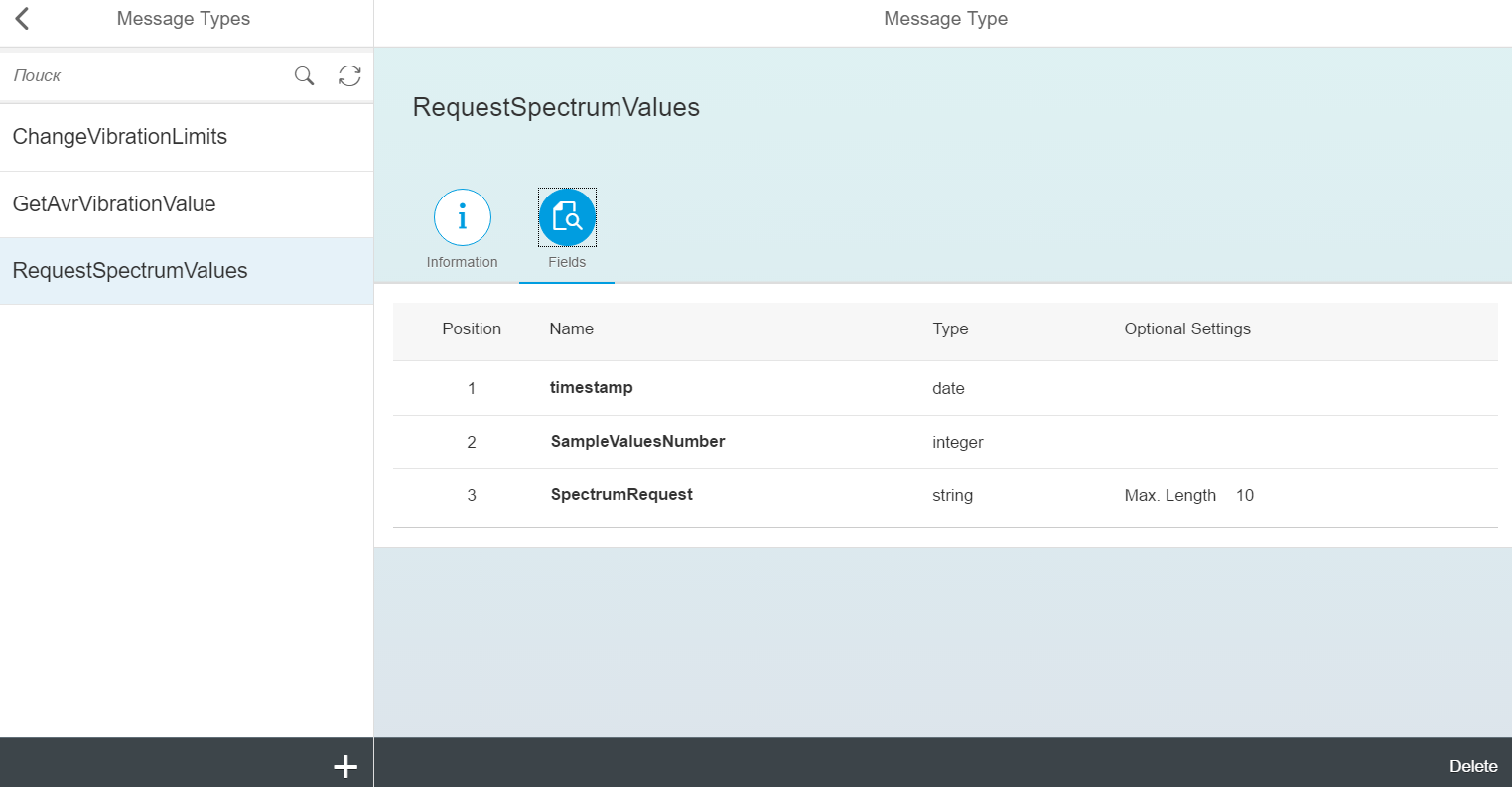
Определяем данный тип сообщения в службе IoT Service:  ID сообщения: df9288ebd989c5b30154

1. **Запрос на выдачу амплитуд спектра вибрации**

*Тип сообщения*

|  |  |  |  |
| --- | --- | --- | --- |
| Позиция | Имя поля | Тип информации | Пояснения |
| 1 | Timestamp | date | Временная метка сообщения |
| 2 | SampleValuesNumber | Integer | Новый предаварийный уровень |
| 3 | SpectrumRequest | String | Новый аварийный уровень |

Определяем данный тип сообщения в IoT Service:



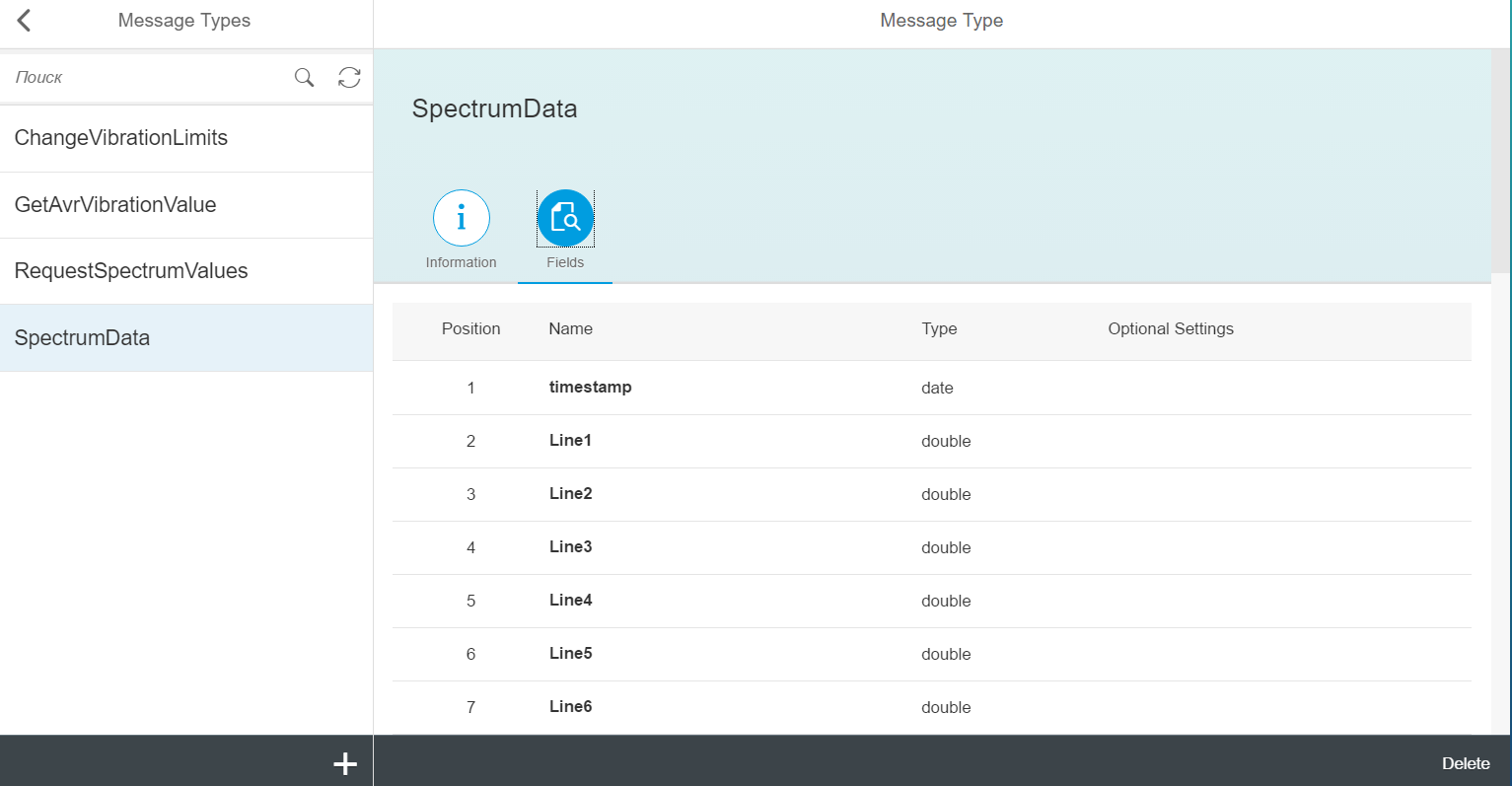
ID сообщения: 63a97febad59c6143507

1. **Сообщение передачи амплитуд спектра вибрации**

*Тип сообщения*

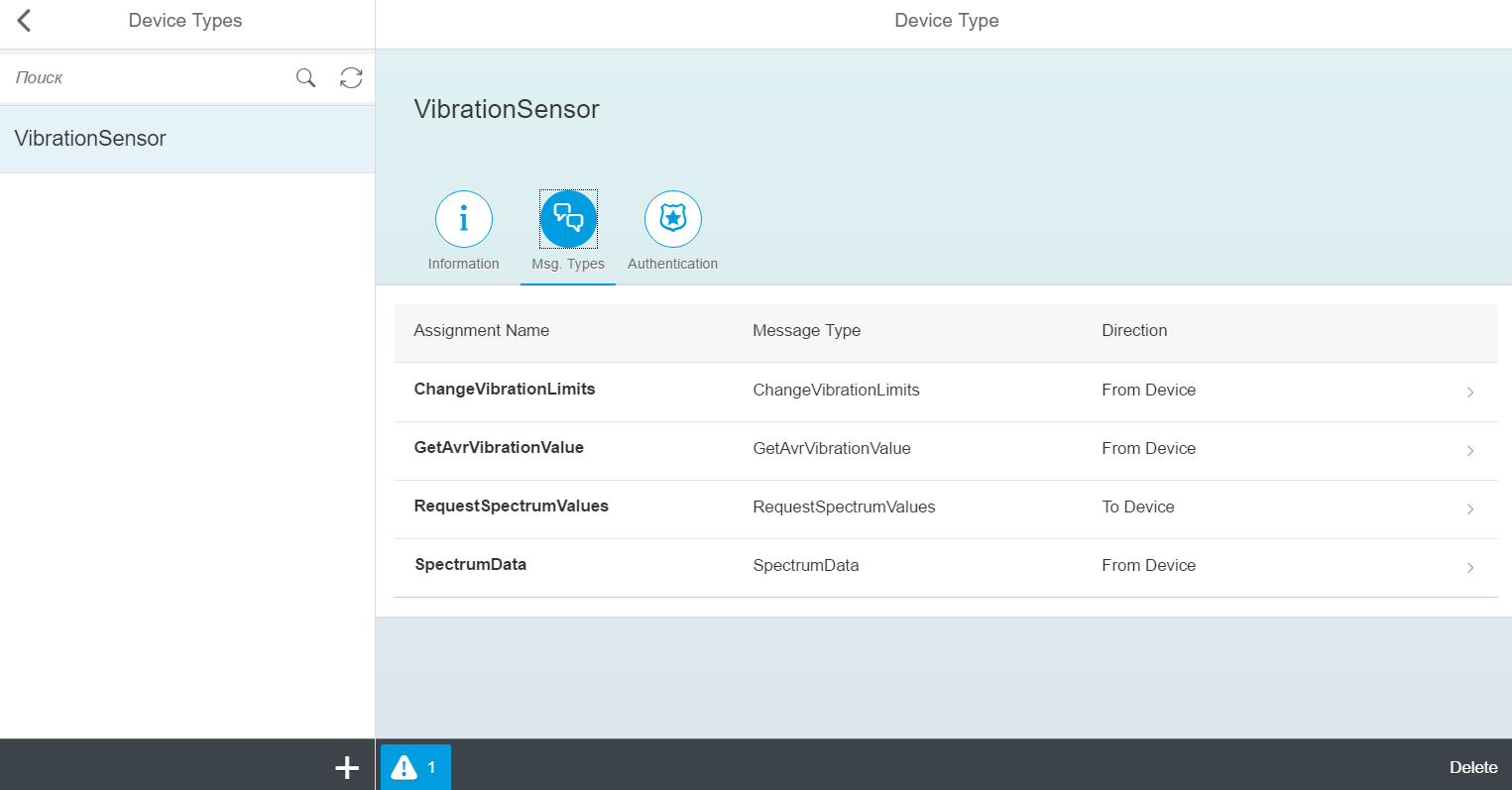
|  |  |  |  |
| --- | --- | --- | --- |
| Позиция | Имя поля | Тип информации | Пояснения |
| 1 | Timestamp | date | Временная метка сообщения |
| 2 | Line1 | double | Амплитуда  1 линии спектра  0-50 Гц |
| … | … | … | … |
| 33 | Line32 | double | Амплитуда 32 линии спектра 1550-1600 Гц |

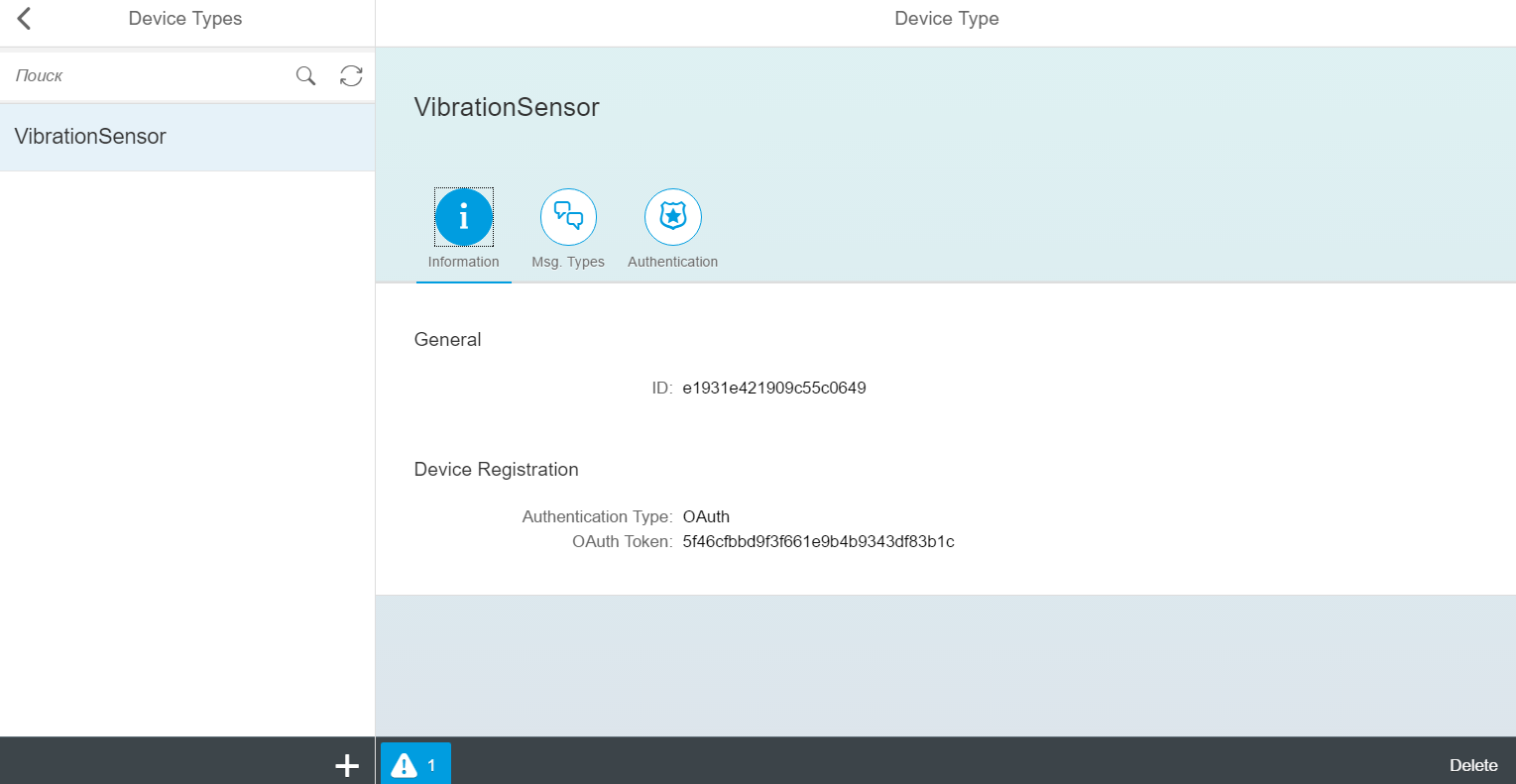
Определяем данный тип сообщения в IoT Service:

****

ID сообщения: f5de3efcbc5fab4df687

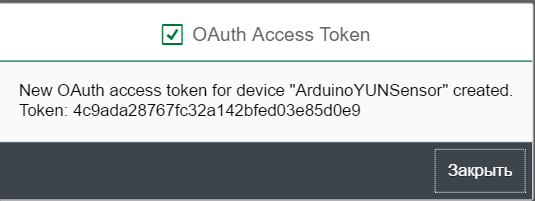
Регистрируем тип устройства.

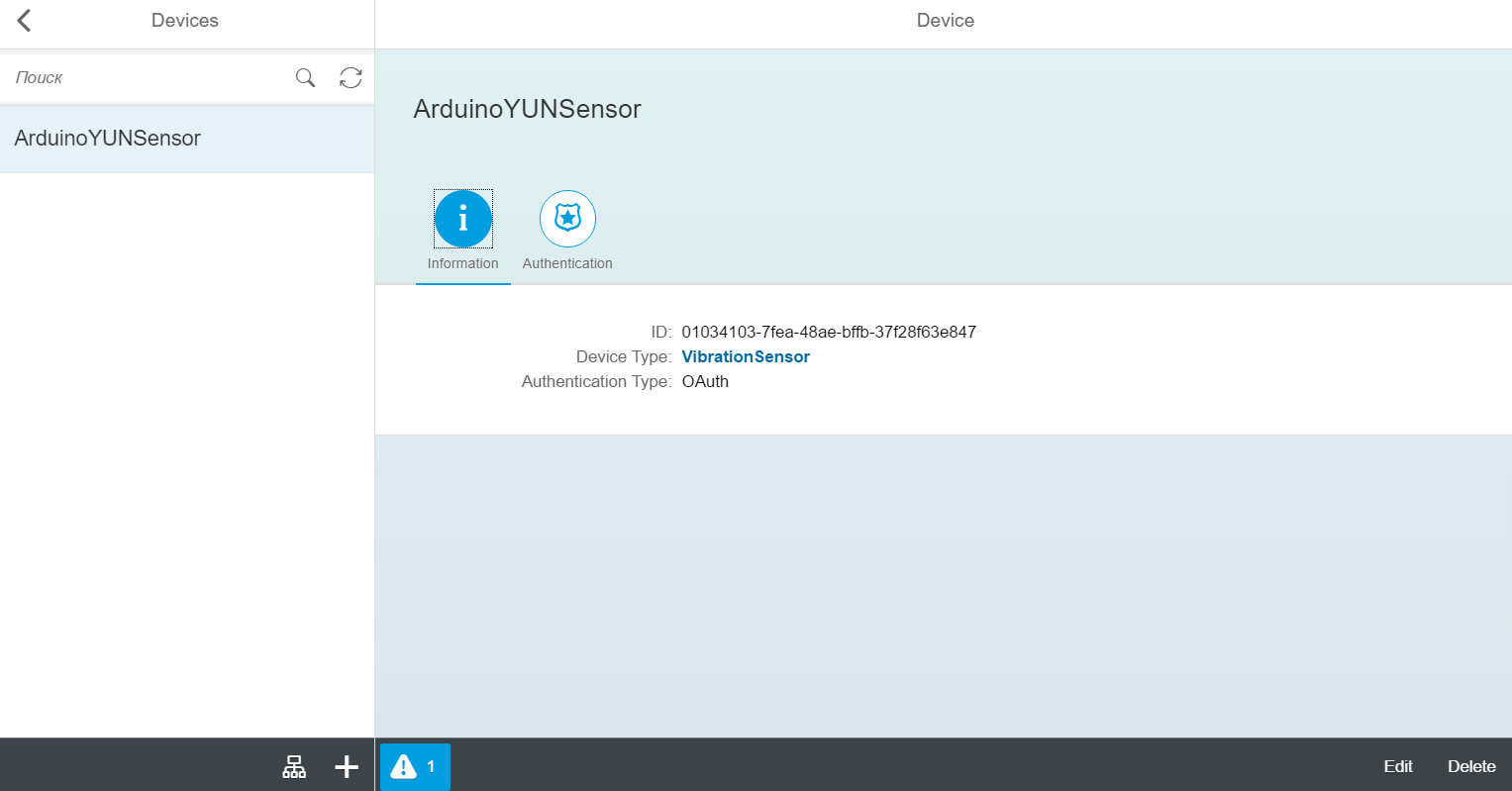




И назначаем ему только что созданные типы сообщений **GetAvrVibrationValue, ChangeVibrationLimits, RequestSpectrumValues, SpectrumData**. Направление сообщений: От устройства, т.к передавать значения будет наш Edge Server (Arduino YUN) в IoT Service во всех типах кроме типа **RequestSpectrumValues,** т.к это запрос от облачной части к Edge Server для подготовки и выдачи сообщением **SpectrumData** значений амплитуд вибрации в линиях спектра.

Далее создаем устройство с именем **ArduinoYUNSensor** и назначаем ему только что созданный тип устройства **VibrationSensor**.





В результате в процессе настройки получили необходимую информацию для дальнейшей настройки сообщений:

* Токен доступа oAuth –**4c9ada28767fc32a142bfed03e85d0e9**
* ID устройства – **01034103 – 7fea- 48ae –bffb-37f28f63e847**

**Дополнительая литература**

1. Горнев В.Ф., Емельянов В.В., Овсянников М.В. Оперативное управление ГПС. – М.: Машиностроение, 1990. – 256 с.
2. Басараб М. А., Зелкин Е. Г., Кравченко В. Ф., Яковлев В. П. Цифровая обработка сигналов на основе теоремы Уиттекера-Котельникова-Шеннона. —М.: Радиотехника, 2004. - 72с.