# Почему проекты IIoT терпят неудачу?

**Если поинтересоваться у людей, используется ли на их предприятии промышленный «Интернет вещей» (Industrial Internet of Things, IIoT), редко можно услышать, что компания с большим успехом внедрила данную технологию. В статье рассматриваются три основные причины, по которым IIoT-проекты заканчиваются неудачами, и предлагается информация, необходимая для успешной реализации проектов, причем она займет считанные дни, а не недели или месяцы.**

## Проблема № 1: так ли хорош IIoT, как о нем говорят?

Проблему № 1 иллюстрирует рис. 1.

### Подключение, сопряжение, интеграция

Когда устройства уже подключены, остается проблема управления ими и преобразования данных с одного «языка» на другой, так как простые инструменты для этого часто отсутствуют. Как вы думаете, легко ли передать данные от программируемой логической интегральной схемы (ПЛИС) в систему планирования ресурсов предприятия (ERP)?

Сделать так, чтобы все устаревшие устройства могли «разговаривать» с современным оборудованием, а затем подключить эту сеть к автоматизированной системе — все это требует написания большого количества заказного кода, поскольку необходимо учитывать все разнообразные протоколы и бренды.

Почему так сложно настроить подключение? Многие фирмы, предлагающие IIoT, обходят этот вопрос молчанием. Они рекламируют все эти «великолепные аналитические методы и сервисы», которые станут вам доступны после «подключения». Как только потребители заглатывают приманку и отваживаются на решительный шаг, они внезапно осознают, что подключить все эти различные устройства — устаревшие, современные, с закрытым и открытым исходным кодом — действительно сложно! Не невозможно, но эта утомительная настройка останавливает многие компании от запуска и работы с IIoT.

Если вам когда-либо доводилось разрабатывать системы на производстве, вы знаете, что там есть такие «вещи», которые станут кошмаром при подключении и интеграции с целым рядом других приложений. Простая задача сбора данных может потребовать недели написания кода.

В условиях растущего спроса на производственные данные в реальном времени в очереди списка потребностей стоит огромное количество задач по сбору данных. Чтобы кто-то действительно мог что-то сделать с этими данными, необходимо быстро передавать их тогда, туда и так, как это вам нужно.

Простая задача сбора данных может занять недели кодирования, чтобы современные и устаревшие устройства и приложения с закрытым и открытым исходным кодом смогли работать совместно.

### Сложность

Вместе с развитием технологий производство эволюционировало. Это привело к значительному совершенствованию автоматизируемых процессов. К лучшему или к худшему, но это развитие также означает большую сложность. И она будет только возрастать.

В сущности, «сложность» — не такая уж плохая вещь. Это понятие мы используем для описания чего-то со многими частями, которые взаимодействуют друг с другом множеством способов,


**Рис. 1.** Подключение устройств: легче сказать, чем сделать

чтобы достигать значительно большего результата по сравнению с тем, когда эти части работают по отдельности. Осознание сложности означает, что мы понимаем, что в IIoT-решениях есть много различных частей, которые мы должны связать вместе, если хотим добиться успеха.

На производстве используется множество устройств различных брендов и с разными протоколами. Комбинация устройств и протоколов уникальна для каждой компании. Многие устройства также используют типы данных с закрытым кодом, что делает очень сложным обмен данными между ПЛК различных брендов. Некоторые машины вообще не были созданы для связи с другими, например станки с ЧПУ.

Не существует стандартного способа, позволяющего соединить все это вместе. В ответ на эту потребность была разработана технология OPC. Но, как мы увидим далее, она тоже несет в себе массу проблем со скоростью и точностью передачи данных.

### Недостатки OPC

Технология OPC была разработана для обеспечения промышленной автоматизации стандартным сетевым протоколом, который требует постоянного опроса для приема данных от устройств. Постоянный опрос — это когда для получения данных система должна опрашивать устройство снова и снова, например ежесекундно или раз в полчаса.

Передача данных с помощью OPC выполняется за несколько шагов. Данные от устройства передаются на ПЛК, затем на OPC-сервер, который, в свою очередь, отправляет их OPC-клиенту. OPC-клиент отправляет данные на локальный сервер или в облачную сеть для дальнейшего использования и обработки. Кроме того, для поддержки серверов, компьютеров и программного обеспечения OPC-серверов и OPC-клиентов потребуются технические специалисты. В дополнение к этому, ПЛК должны иметь отдельные соединения с любым другим устройством или программным приложением. Таким образом, соединения должны быть прописаны от ПЛК 1 к устройству 1, от ПЛК 1 к устройству 2, от ПЛК 1 к устройству 3, а затем повторены для ПЛК 2 и каждого устройства. Если вам нужно передать данные от ПЛК в программу планирования ресурсов предприятия (ERP), то придется прописать соединения ПЛК 1 с ПО ERP 1, ПО ERP 2 и т. д.

Добавьте во все эти слои последовательные опросы, и получите рецепт огромной задержки.

Другая проблема — точность данных. OPC не предлагает никаких функций, гарантирующих точность данных. Одна компания, владеющая фармацевтическим заводом в г. Джэксонвилл, шт. Флорида, столкнулась с этой проблемой. Для опроса устройств они использовали OPC и часто получали 3000 пакетов данных для каждого производственного цикла. У них не было возможности проверить, какой пакет является правильным, соответствующим партии готовой продукции. Чтобы ответить на этот вопрос, техническая служба написала большой объем сложного кода для выполнения двойной проверки источника и получателя данных, чтобы обеспечить соответствие данных. Это потребовало большой работы по созданию и сопровождению.

Если в сети что-то ломается, поиск источника проблемы может выбить из колеи. Это ПЛК? ПЛК какого бренда? Какой OPC-сервер? Клиентское программное обеспечение? Поставщик, написавший драйвер OPC? И нет одного-единственного поставщика, с которым вы могли бы решать эту проблему. Приходится тратить время на созвон с разными поставщиками оборудования и программного обеспечения, а они, как правило, указывают друг на друга. В конечном счете, никто не берет на себя ответственность, и вся тяжесть выявления источника проблемы падает на вас.

### Необходимость модификации устаревших устройств

MQTT быстро становится одним из самых популярных протоколов для IIoT, и многие современные устройства его поддерживают. Однако единственный способ воспользоваться преимуществами MQTT — это купить устройства с его поддержкой. Но никто не хочет отказываться и заменять устаревшие устройства, даже прослужившие 20 или 30 лет, на новые из-за наличия в них MQTT.

Когда в какой-то момент вы захотите добавить новое оборудование, например новейший датчик, то он, скорее всего, будет разработан с учетом MQTT. Недостатком же этого является то, что вам придется провести дополнительную работу, чтобы устройства MQTT могли работать с вашим устаревшим оборудованием. Среди тысяч устройств, имеющихся на вашем производстве, будет, возможно, всего десяток с поддержкой MQTT. Переход всех устройств на MQTT будет медленным и не решит существующую на сегодня проблему соединения устаревших и современных устройств.

В любом случае, понадобится писать все больше кода, чтобы все больше и больше новых устройств могло работать с устройствами, которые с каждым годом все более устаревают.

## Проблема № 2: заказной код

Существование различных поставщиков и протоколов по-прежнему требует использования заказного кода (рис. 2), даже при использовании OPC и MQTT. Это становится вторым фактором, душащим инициативы IIoT.

### Время убивает проекты

Проекты часто откладываются просто из-за того, что они требуют написания слишком большого объема кода. При сопоставлении затрачиваемого времени и важности проекта, время часто может убить проект. Это случается чаще, чем можно было бы подумать.

Кроме того, другие проекты с более высоким приоритетом, уже находящиеся в работе, также откладываются, поскольку программисты не успевают написать очередную порцию кода.

Часто потребность в программисте — это узкое место между узкопрофильным экспертом и необходимым изменением процесса или продукта. Задача эксперта — проанализировать процесс и определить необходимые изменения. Если он знает, что нужно изменить, но не может внедрить это в процесс, это означает упущенную возможность улучшения или увеличения количества выпускаемой продукции.

Например, на молочном заводе, производящем сыр, можно отслеживать такие данные, как температура сыра, температура в помещении, плотность продукта и т. д. Специалист по продукту просматривает данные и определяет, что процесс необходимо оптимизировать. Возможно, необходимо замедлить или увеличить скорость конвейера. Но он не захочет дополнительной головной боли, потому что придется идти к программисту, который скажет, что он слишком занят, что надо занять место в очереди и прождать несколько недель.

### Ограниченный круг «посвященных»

Когда заказной код является основой управления вашими производственными операциями, возникают определенные риски. Во-первых, трудно вносить изменения в действующую систему, не вызывая при этом простоев. Во-вторых, один или несколько людей написали код, и только они знакомы с тем, как все запрограммировано. Когда они недоступны или уходят из компании, последствия могут принести убытки.

Кроме того, когда некоторые устройства прекрасно функционируют пару лет и дольше, у инженеров нет причин заниматься ими. За такой долгий срок инженеры забывают, как работать с этим конкретным устройством, и им приходится тратить время на повторное изучение всего, что нужно и не нужно, чтобы заставить отказавшее устройство снова работать.

По мере того как с течением времени все больше и больше будет добавляться функциональных возможностей, вы получите приложение-монстр, и только несколько человек будут знать, как


**Рис. 2.** Между устройствами OPC и ERP-приложениями необходим заказной код

## Контрольные вопросы для проверки программной платформы IIoT

**Подключение устройств**

- **Есть ли в платформе IIoT библиотека всех необходимых драйверов (помимо API, OPC или MQTT)?**
- **Есть ли драйверы устройств и приложений?**
- **Может ли платформа обеспечить прямое взаимодействие ПЛК с другим ПЛК?**
- **Могут ли данные передаваться в двух направлениях?**

**Конфигурация системы**

- **Есть ли блок-схема, позволяющая отказаться от написания кода для настройки логики?**
- **Теги автоматически появляются в системе при подключении устройства?**
- **Теги присваиваются в соответствии с универсальным форматом присвоения имен?**
- **Можно ли вносить изменения «на лету», не затрагивая работающую систему в целом?**

**Масштабируемость**

- **Может ли платформа позволить устройствам самостоятельно принимать решения о том, когда и куда отправлять данные?**
- **Сколько данных может обрабатывать программное обеспечение на один экземпляр программного обеспечения?**
- **Можете ли вы сделать все необходимое, используя всего несколько серверов?**
- **Можно ли платформу установить непосредственно на ПЛК, в дополнение к серверу, к серверной стойке или к шлюзу?**

его программно изменить. Таким образом, ваша система никогда не будет быстро переналаживаемой или гибкой.

## Проблема № 3: масштабируемость

Сам по себе IIoT не обеспечивает масштабируемость. Одна из основных идей, лежащих в основе IIoT, заключается в том, что это взаимодополняющая, сложная сеть, которая доходит до самых глубин вашего предприятия: от нескольких датчиков вибрации на конвейерной ленте и до того, что находится за пределами производственного цеха — например, данные о погоде от дронов, летающих над полями вашего поставщика. Теоретически, все может и должно быть связано. Расширение вашей сети — палка о двух концах. Вы хотите сделать ее больше, но она не так легко масштабируется.

### Технические ограничения и затраты на ПО

В большинстве случаев пользователям приходится покупать несколько экземпляров программного обеспечения для промышленной автоматизации (или для IIoT). Это связано с тем, что один узел или сервер может обрабатывать только определенное количество устройств. Для достижения масштабируемости необходимо приобрести несколько узлов или серверов, каждый с экземпляром установленного на нем программного обеспечения. Затраты на каждый сервер и узел, на лицензии для каждого экземпляра ПО, на ИТ для поддержки этих элементов и т. д. в итоге могут составить значительную сумму. По мере развития, когда приходится создавать сложную архитектуру системы, чтобы обрабатывать все больше и больше устройств, масштабируемость становится очень дорогостоящей.

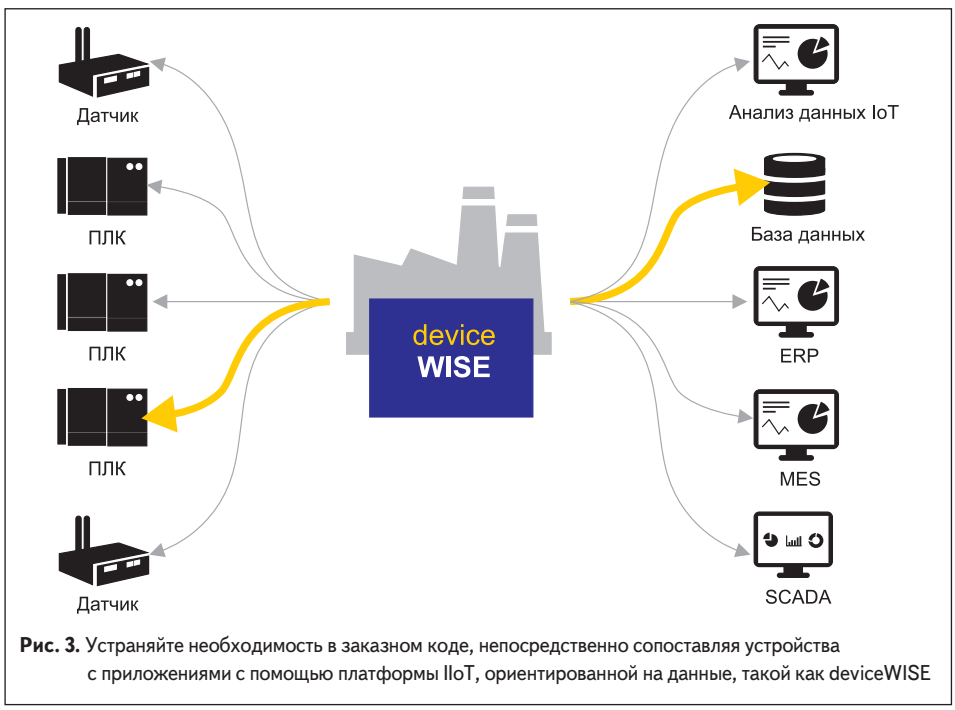


**Рис. 3.** Устраняйте необходимость в заказном коде, непосредственно сопоставляя устройства с приложениями с помощью платформы IIoT, ориентированной на данные, такой как deviceWISE

### Риск модификации развивающейся системы

Большинство изменений и обновлений в автоматизированной системе требуют вмешательства инженера в уже работающую систему. Это означает, что любая модификация должна быть запланирована и выполнена во время планового простоя. При этом нет никакой уверенности, что все заработает без сбоев, когда систему запустят. Это очень деструктивный подход.

Обновления не могут быть выполнены «на лету», в процессе работы. Часто существующая исходная структура нарушается, и инженерам для восстановления работы производственной линии приходится иметь дело с неизвестными переменными в условиях повышенного давления со стороны руководства. Чем больше система, тем труднее ее обновлять и совершенствовать, поскольку слишком многое поставлено на карту.

Теперь, когда мы рассмотрели три самые значительные проблемы IIoT, обсудим, как можно их избежать с помощью программной платформы IIoT, ориентированной на обработку данных.

## Способы обойти проблемы

### Шаг 1. Заставьте IIoT работать в режиме Plug&Play

Как видно из рассмотрения проблем, у вас просто нет возможности непосредственно связывать устройства друг с другом. Это не Plug&Play, на который вы надеялись.

Итак, как же подключить устройства, например ПЛК, непосредственно к приложениям? Для этого потребуется программная платформа IIoT, которая предназначена именно для этого. Платформа, ориентированная на данные, означает, что она была разработана с нуля, чтобы объединить устаревшие и современные устройства независимо от протокола связи и обеспечить основную линию передачи данных для всех устройств и приложений в сети, предоставляя полный контроль над тем, как, где и когда используются ваши данные.

Многие платформы IIoT ориентированы на анализ данных, но не на сами данные — из-за неспособности быстро подключать устройства. Можно назвать их платформами IIoT, «ориентированными на анализ». Но без наличия больших данных даже самые прекрасные инструменты только разочаровывают пользователей, поскольку из-за неточности или неполноты данных они не могут получить реальные ответы.

Программная платформа, ориентированная на данные, позволит вам действительно получать нужные вам данные и передавать их в любое приложение для анализа данных, уже имеющееся у вас или которое вы бы хотели приобрести в будущем (рис. 3).

Все платформы IIoT будут иметь стандартные инструменты для API, OPC и MQTT, но они часто не содержат встроенных драйверов. Платформа, ориентированная на данные, будет иметь широкий набор встроенных драйверов. Ключевое слово здесь — «встроенных». Не надейтесь, что API и стандартные

протоколы предоставят вам гибкость. Они никогда не позволят избавиться от заказного кода. API, OPC и MQTT — приманка, из-за которой потребуются тысячи и тысячи строк заказного кода. Встроенные драйверы означают, что весь тяжелый процесс написания заказного кода уже сделан за вас.

Если вы выберете пакет программ, в котором уже есть все необходимые драйверы, это позволит реализовать технологию IIoT в считанные дни, а не месяцы (рис. 4). Подключенные к платформе устройства мгновенно смогут взаимодействовать с любым другим устройством или приложением: настоящий Plug&Play.

Одно короткое замечание: при поиске платформы IIoT обратите внимание, включает ли она встроенные драйверы для аналитических программных платформ, таких как SAP, Microsoft Azure и т. д. Это позволит вам не писать заказной код для приложений.

Готовые встроенные драйверы программной платформы IIoT означают, что все ваши устройства и приложения как будто сами подключаются друг к другу. В любое время, в любом месте, так же легко, как добавить новое приложение на свой смартфон и заставить его работать со всем остальным на смартфоне. Разве не таким образом должен работать IIoT?

Без зависимости от API, OPC или MQTT для подключения устройств вам не придется иметь дело с опросами, вызванной ими системной задержкой или заказным кодом. У вас будет более точная и быстрая сеть, в которой обмен данными происходит молниеносно. ПЛК могут общаться с другими ПЛК. ПЛК могут напрямую взаимодействовать с программными приложениями ERP, и наоборот. Вы также сможете мгновенно добавлять новые устройства с MQTT, не беспокоясь о том, как заставить их взаимодействовать с устаревшими устройствами.

Это означает, что вы можете тратить свое время на то, что заставит вашу систему делать что-то значимое, вместо того, чтобы тратить его на то, чтобы просто заставить устройства общаться друг с другом.

### Шаг 2. Конфигурируйте систему с помощью блок-схем

После того как устройство легко подключено, следующим препятствием, которое необходимо преодолеть, становится конфигурирование системы. Для этого может потребоваться написание большого объема заказного кода, даже если для каждого устройства и приложения у вас есть встроенные драйверы.

Программная платформа IIoT, ориентированная на данные, должна не только обеспечивать доступ к данным для каждого устройства и приложения в сети, но и сделать данные доступными пользователям. Не будучи программистом, вы должны иметь возможность контролировать свои данные, а также время, место и способ их использования.

#### *Блок-схема*

Благодаря простой блок-схеме, настраиваемой с помощью мышки, платформа IIoT позволяет упростить создание любой логики без дополнительного программирования. Это означает, что инженеры и даже узкопрофильные специалисты будут иметь возможность полностью настраивать внедрение системы без необходимости внесения изменений программистом (рис. 5).

| API, OPC и MQTT | Встроенные драйверы |
|---|---|
| Требуется заказной код | Программирование не требуется |
| Опрос для устаревших устройств с использованием OPC | Опрос не требуется |
| Задержка из-за опроса | Нет задержки |

**Рис. 4.** Преимущества платформы со встроенными драйверами

Поскольку пользователи получают легкий доступ к управлению системой, понадобятся средства, обеспечивающие внесение изменений только людьми, которые имеют на это право. ПО должно позволять устанавливать разрешения на основе ролей, чтобы только определенные люди могли менять определенные части вашего процесса. Это защищает систему от изменения посторонними.

Кроме того, надежная программная платформа должна предоставить возможность заглянуть «за кулисы» в те редкие моменты, когда блок-схема не позволяет настроить «вещь» без написания кода. На этом уровне понадобится программист, но не для простых рутинных изменений логической структуры, а для расширения функциональности системы в будущем.

#### *Автоматическое присвоение тегов*

Программная платформа IIoT, ориентированная на данные, может присваивать теги непосредственно в момент подключения устройств и форматировать их в соответствии с универсальной системой присвоения имен с помощью функции ***tag auto-enumeration***.

ПО также делает одинаковыми такие переменные процесса, как, например, температура или давление, независимо от типа ПЛК. В случае если ПЛК устаревший, с линейной адресацией, например Siemens S7-300, DB3[0], функция автоопределения тегов представляет собой простой механизм присвоения имен с использованием более удобной для пользователя структуры именования.

Это позволяет пользователям легко получать температурные теги с любого устройства и сообщать системе, что с этим тегом делать. Это избавляет от проблем, вызванных использованием неправильных имен, что может привести к сбою в логике системы автоматизации.

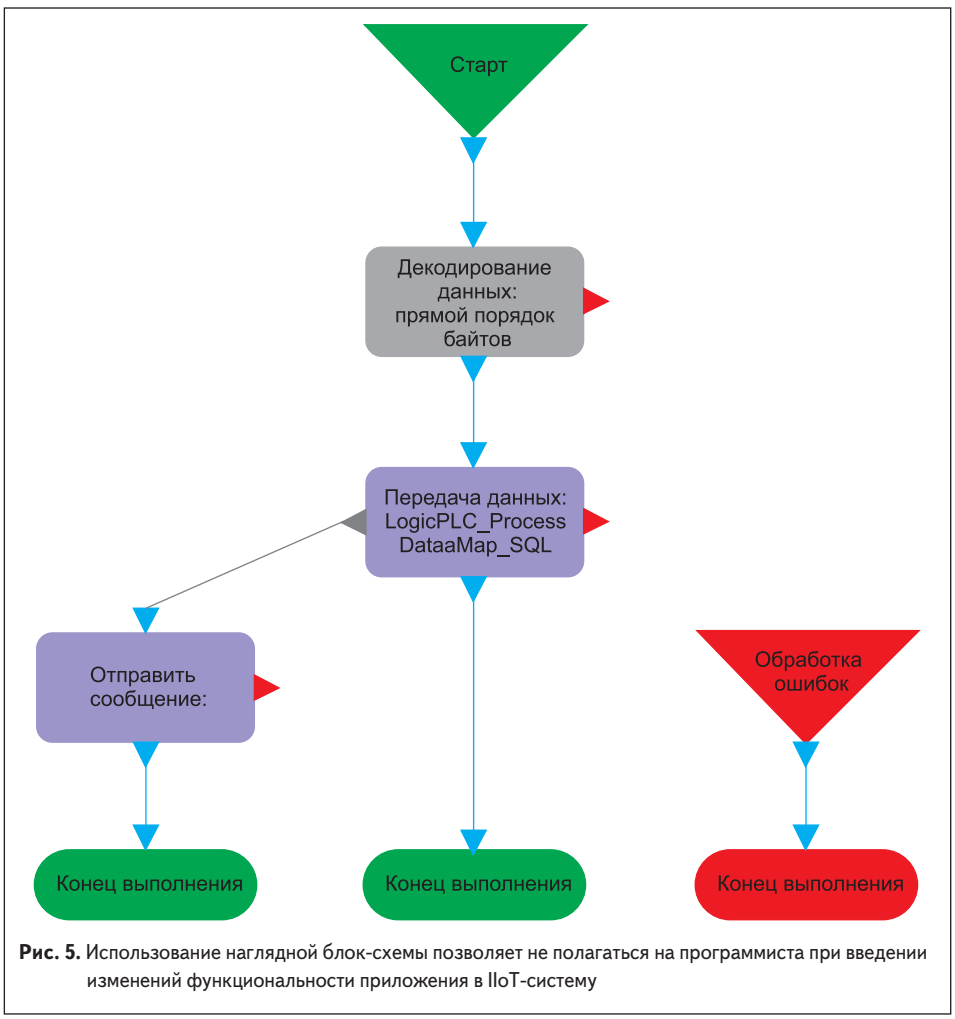


**Рис. 5.** Использование наглядной блок-схемы позволяет не полагаться на программиста при введении изменений функциональности приложения в IIoT-систему

## Платформа deviceWISE для производства

**deviceWISE для производства компании Telit — это программная платформа, ориентированная на данные. Она гарантирует, что развертывание IIoT начнется с правильных данных, и дает возможность решать, как, когда и где вы хотите эти данные использовать. Ваши устройства станут wise («мудрыми»), поскольку они точно будут знать, как немедленно взаимодействовать с любым другим устройством и приложением через магистраль данных deviceWISE. Причем устройства взаимодействуют напрямую. Любое устройство можно подключить за минуты благодаря огромной библиотеке встроенных драйверов. Особенности платформы:**

- **нет необходимости в замене устройств или отказе от них;**
- **точные данные передаются за один раз;**
- **программирование мышкой;**
- **быстрое создание блок-схем для автоматизации всех аспектов IIoT-решения;**
- **гибкость, достаточная для любого масштабирования.**

### Шаг 3. Упростите и повысьте масштабируемость

***Снижение затрат***

Благодаря предыдущим двум шагам можно легко подключить большее количество устройств и легко управлять большими потоками данных, поэтому последним препятствием становится повышенная нагрузка на ваше оборудование.

Программная платформа IIoT, ориентированная на данные, обрабатывает их значительно эффективнее, снимая тем самым нагрузку с оборудования. В результате вы значительно экономите на количестве компьютеров, которые необходимы для запуска одной и той же системы IIoT. Подумайте о средствах, которые можно сэкономить, сократив количество компьютеров с 1000 до одного (может быть, до двух — для обеспечения резервирования). Стоимость ИТ для администрирования этих компьютеров также сокращается. Ежегодные расходы на электричество для работы этих компьютеров снижаются. И оплата лицензии на ПО для каждого экземпляра также отсутствует.

***Быстрая передача точных данных***

Благодаря программной платформе IIoT, ориентированной на данные, логика узловых устройств становится действительно интересной. Поскольку есть встроенные драйверы для каждого устройства, больше не нужно использовать OPC. Вместо использования OPC для получения данных и отправки устройствам запроса на передачу данных каждую миллисекунду, устройства теперь могут самостоятельно принимать решения о том, когда отправлять данные, в соответствии с предварительно заданными настройками. Происходит только одна транзакция, а не сотни передач туда и обратно. Это гарантирует точность данных и может практически исключить задержки в системе.

Вы даже можете настроить один ПЛК, чтобы он делал что-то на основе данных от другого ПЛК, поскольку они смогут напрямую взаимодействовать друг с другом. Вам больше не нужно перекидывать данные от первого ПЛК на центральный сервер, дожидаться их обработки, а затем ждать, пока центральный сервер даст указание что-то сделать другому ПЛК. Эта сложная передача данных может занять миллисекунды, что кажется не очень значительным временем, но когда эти миллисекунды накладываются на сотни или даже тысячи устройств, происходит перегрузка сети. Итак, подумайте о том, насколько быстрее и точнее ваша система будет работать без центрального сервера, который должен передавать все эти данные туда и обратно: с устройств на сервер, с сервера на устройства.

***Масштабируемость за пределами производства***

Когда вы повышаете масштабируемость за счет более эффективного использования данных от узловых устройств, вы можете расширить систему IIoT до всего, что находится и до, и после вашего производственного процесса. В цепочке поставок ключевым звеном является именно производственный процесс. Подключение автоматизированного процесса к другим пунктам цепочки поставок — вот будущее IIoT. Подумайте о том, как вы могли бы выделить свой бизнес на фоне конкурентов, предлагая новые сервисы в дополнение к товарам. Возможности для инноваций бесконечны.

***Гибкость благодаря IIoT***

Программная платформа IIoT, ориентированная на данные, фокусируется не на встраивании в заказной код, а на обеспечении бесперебойной передачи данных параллельно с процессами, протекающими в системе, и управляющими процессами вашего предприятия. Это фундаментальное разделение делает вашу команду более гибкой и эффективной.

Когда данные находятся в безопасной «магистрали», которая отделена от работающей системы, изменения могут быть вынесены непосредственно в процессе работы. Не нужно «тревожить» работающую систему, не нужно планировать время простоя, чтобы перейти на новый код, и не придется сталкиваться с вынужденным простоем из-за того, что что-то в коде повредило систему. Появляется возможность просто и без труда вносить изменения, не нарушая работу какой-либо части системы. ■

*Оригинал статьи опубликован на сайте www.telit.com/no-more-coding*

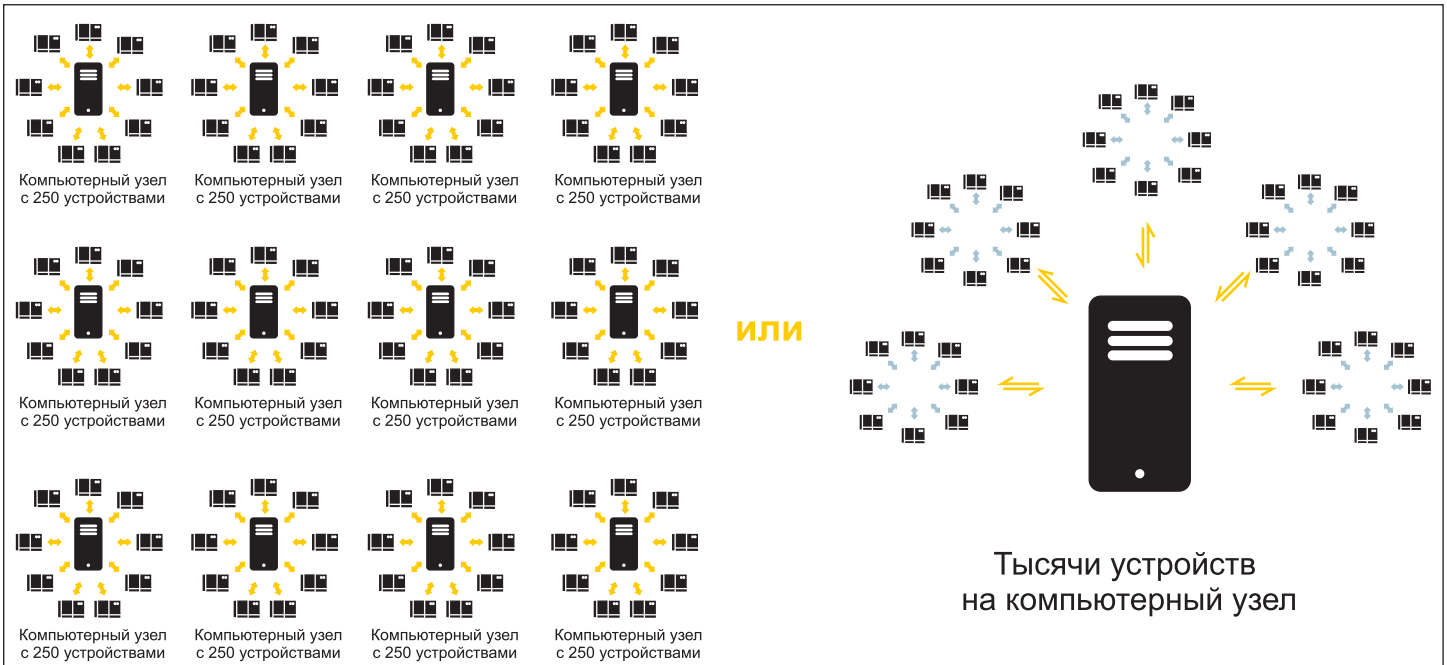


**Рис. 6.** Настройка логики устройств значительно снижает нагрузку на оборудование