НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2018 Том 18 №1 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
January–February 2018 Vol. 18 No 1 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 681.51

ПРИМЕНЕНИЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ ПРИ ПРОЕКТИРОВАНИИ ПРОМЫШЛЕННОГО ОБОРУДОВАНИЯ С ЧИСЛОВЫМ ПРОГРАММНЫМ УПРАВЛЕНИЕМ

М.Я. Афанасьев[a], Ю.В. Федосов[b,a], А.А. Крылова[c,a], С.А. Шорохов[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
[b] ОАО «Российский институт мощного радиостроения», Санкт-Петербург, 199048, Российская Федерация
[c] ООО «ЛАР Технологии», Санкт-Петербург, 197342, Российская Федерация
Адрес для переписки: stratumxspb@gmail.com

Информация о статье
Поступила в редакцию 09.11.17, принята к печати 30.12.17
doi: 10.17586/2226-1494-2018-18-1-87-97
Язык статьи – русский

Ссылка для цитирования: Афанасьев М.Я., Федосов Ю.В., Крылова А.А., Шорохов С.А. Применение микросервисной архитектуры при проектировании промышленного оборудования с числовым программным управлением // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 1. С. 87–97. doi: 10.17586/2226-1494-2018-18-1-87-97

Аннотация

Предмет исследования. Исследован микросервисный подход к разработке системы управления модульного технологического оборудования. Проанализированы его свойства, достоинства и недостатки. В качестве основы для систем числового программного управления предложена гетерогенная компьютерная сеть, где узлы сообщаются между собой посредством очередей сообщений. Представлено описание архитектуры платформы для селективного лазерного отверждения фотополимеров, разработанной в соответствии с микросервисным подходом. **Метод.** Применяемый в работе микросервисный подход является современной интерпретацией сервис-ориентированного подхода. Он позволяет строить систему из небольших слабо связанных модулей, что является одним из основополагающих требований при создании модульного оборудования с числовым программным управлением. Более того, благодаря свойству интероперабельности, подход позволяет без особых затруднений встраивать оборудование в современные производственные киберфизические системы. **Основные результаты.** В качестве основного протокола взаимодействия была выбрана система, обеспечивающая передачу данных за счет использования очереди сообщений. Хранение данных предложено осуществлять за счет использования распределенной нереляционной базы данных. **Практическая значимость.** Предложенный подход может применяться для разработки модульного технологического оборудования для малых предприятий, занимающихся единичным и мелкосерийным производством. Более того, при применении такого подхода оборудование легко совместимо с киберфизическими системами, которые активно развиваются в настоящее время.

Ключевые слова
микросервисная архитектура, модульное технологическое оборудование, очереди сообщений, киберфизическая система, числовое программное управление

MICROSERVICE ARHITECTURE APPLICATION IN THE DESIGN OF INDUSTRIAL EQUIPMENT WITH COMPUTER NUMERICAL CONTROL

M.Ya. Afanasiev[a], Yu.V. Fedosov[b,a], A.A. Krylova[c,a], S.A. Shorokhov[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
[b] JSC “Russian Institute of Power Radiobuilding”, Saint Petersburg, 199048, Russian Federation
[c] Lar Technologies, LLC, Saint Petersburg, 197342, Russian Federation
Corresponding author: stratumxspb@gmail.com

Article info
Received 09.11.17, accepted 30.12.17
doi: 10.17586/2226-1494-2018-18-1-87-97
Article in Russian

For citation: Afanasiev M.Ya., Fedosov Yu.V., Krylova A.A., Shorokhova S.A. Microservice arhitecture application in the design of industrial equipment with computer numerical control. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2018, vol. 18, no. 1, pp. 87–97 (in Russian). doi: 10.17586/2226-1494-2018-18-1-87-97

Abstract
Subject of Research. The paper deals with the microservice approach to development of control system for modular computer numeric control machines. Its properties, advantages, and disadvantages are analyzed. A heterogeneous computer network is proposed as the basis of CNC machine where nodes communicate via message queues. We present a platform architecture description of the selective photopolymer laser curing developed by the microservice approach. **Method.** The microservice approach is a modern interpretation of the service-oriented approach. It allows for system creation from small loosely coupled modules that is essential for modular equipment. Moreover, it makes it possible to embed equipment into modern cyber-physical production systems because of interoperability. **Main Results.** An architecture of the selective photopolymer laser curing device was implemented. Message queue was chosen as the main communication protocol of the system. A distributed non-relational database was used as data storage. **Practical Relevance.** The proposed approach can be applied in the technological equipment development for unit and small-batch production. Moreover, the equipment built according to the approach is easily compatible with cyber-physical systems, which are actively being developed now.

Keywords
microservice architecture, modular technological equipment, message queues, cyber-physical production system, computer numerical control

Введение

Анализируя развитие современной промышленности, можно сделать вывод, что наиболее перспективным направлением совершенствования является создание гибких распределенных автоматизированных производственных линий. В результате четвертой промышленной революции, а также постепенного перехода на киберфизические производственные системы (Cyber-Physical Production Systems, CPPS) сформировалась новая концепция массового производства, а «жесткие» конвейерные линии постепенно уступают мелкосерийной продукции. Более того, все еще продолжается развитие малых инновационных предприятий и так называемых «стартапов».

Первые станки с числовым программным управлением (ЧПУ) появились еще в 50-х годах двадцатого века. Их развитие, продолжающееся и по сей день, в основном было сосредоточено на массовом производстве. Таким образом, устройства ЧПУ (УЧПУ) всегда были сложными, высокопроизводительными и чрезвычайно дорогими. Кроме того, процесс их развертывания является долгосрочным мероприятием, современный срок службы системы составляет десятки лет. Эти обстоятельства усложняют внедрение современных коммуникационных технологий в том смысле, что каждое изменение в производственной линии требует либо реструктуризации всей производственной системы, либо создания дополнительных уровней контроля для связи устаревшего оборудования с современными CPPS. Этот подход наиболее применим в течение переходного периода и, в конечном счете, замещает более старую систему.

Исходя из сказанного выше, необходимо переосмысление парадигмы проектирования оборудования с ЧПУ, и задача внедрения нового оборудования в информационную среду через открытый протокол становится весьма важной. Были предприняты различные попытки реализовать такие подходы. Например, Григорьев и Мартынов в своих исследованиях предлагают подход к разработке гибкого ядра УЧПУ на основе независимой платформы. Открытая архитектура этой системы включает в себя различные абстрактные слои, связанные с различными человеко-машинными интерфейсами, а также возможности описания компонентов системы на разных языках программирования. Подключение системных компонентов устанавливается через «Fieldbus» [1].

Существует описание открытой платформы для создания УЧПУ, состоящее из набора универсальных компонентов, которые могут быть использованы повторно, и набора коммуникационных модулей для их соединения [2]. Аналогичный подход представлен в работе сотрудников Харбинского университета в Китае [3]. А коллектив под руководством Л. Моралес-Веласкеса предложил систему Madcon – платформу с открытой архитектурой на основе многоагентной системы программных и аппаратных компонентов. Аппаратные модули предлагаемой системы объединяют функции управления и контроля, обеспечивающие открытую архитектуру на основе программируемой пользователем вентильной матрицы для реконфигурируемых приложений. Программные компоненты используют язык разметки XML в качестве структуры файлов описания системы, объединяя такие функции, как описательный язык блок-схемы и графический интерфейс пользователя [4].

Работы Н. Вербы [5] и К. Плазереса [6] затрагивают интересное понятие «Fog of Things». Эта концепция – современная интерпретация «Интернета вещей», которая, по сути, является основой многих CPPS. «Fog of Things» позволяет создавать более однородную информационную среду, тем самым улучшая и упрощая протокол связи компонентов.

Описание подхода

Современные системы управления УЧПУ являются сложными и монолитными, каждый элемент в них представляет собой «черный ящик» с жесткой иерархической архитектурой. Они должны обладать высокой надежностью и отказоустойчивостью, а их инертность заставляет разработчиков автоматизированных систем управления предприятием использовать монолитную архитектуру, поскольку встраива-

ние оборудования с распределенным управлением в распределенную производственную среду– непростая задача. Следовательно, акцент делается на концепции интеграции. Другими словами, разработчики сосредоточились на объединении гетерогенных компонентов в одну производственную систему вместо использования принципа интероперабельности, который подразумевает создание открытого интерфейса, позволяющего компонентам сохранять автономное состояние при коммуникационной способности. Соответственно, необходим переход к разработке оборудования с модульной архитектурой.

Модульная архитектура основана на двух основных постулатах: унификации и гибридизации. Унификация предполагает открытую программную и аппаратную структуру, которая позволяет собирать новое оборудование и программное обеспечение, после чего может быть достигнуто разделение одного продукта на несколько взаимозаменяемых строительных блоков с описанием унифицированного ввода и вывода. В итоге любое производственное оборудование содержит следующие основные компоненты: привод управления обрабатывающей головки, координатный стол и блок ЧПУ. Очевидно, что координатный стол является наиболее универсальной единицей, в которой могут быть установлены различные обрабатывающие головки. Таким образом, тип оборудования может быть легко изменен. Также есть возможность создать установку, которая объединяет функции различных типов оборудования, применяя принцип гибридизации.

Однако не следует забывать о том, что замена исполнительного механизма обрабатывающей головки включает в себя изменения в алгоритме управления. Очевидно, что блок ЧПУ должен заранее знать о каждом возможном исполнительном устройстве для обработки, установленном на платформе, и при таком подходе снова будет получена монолитная система с иерархическим управлением. Исходя из этого, необходимо определить основную часть архитектуры и спецификации как для протокола связи, так и для создания новых программных и аппаратных компонентов, которые могут быть динамически подключены к управляющей системе. Основой будет служить алгоритм управления перемещениями координатного стола, а внешние компоненты войдут в оставшуюся часть системы.

Чтобы реализовать подобную архитектуру, необходимо разработать распределенную сеть с программными и аппаратными модулями. Все модули подключены к сети при помощи проводных либо беспроводных соединений. Каждый модуль получает IP-адрес динамически (по протоколу «DHCP[1]»), после чего осуществляет поиск диспетчера через отправку широковещательного пакета. Диспетчер отвечает посредством предопределенного протокола и записывает основные физические параметры модулей в свой реестр.

Описываемый подход дает возможность полностью изменить управление производственной средой. Но, к сожалению, здесь есть свои отрицательные моменты. УЧПУ включает в себя не только алгоритм управления, но и базу данных, пользовательские интерфейсы, а также компоненты связи с физической средой (также известные как встроенные системы и микроконтроллеры). Для реализации такой системы необходим системный подход, и авторы предлагают использовать архитектурный шаблон микросервисов.

Сервис-ориентированная архитектура (Service-Oriented Architecture, SOA) является быстро развивающейся концепцией, все чаще используемой для реализации распределенных программных систем. В основе подобной архитектуры лежит набор слабо связанных заменяемых компонентов, оснащенных унифицированными интерфейсами для взаимодействия по стандартизированным протоколам.

Микросервисная архитектура (Multi-Service Architecture, MSA) является современной интерпретацией SOA. Первые упоминания об MSA относятся к 2012 г., хотя этот термин употреблялся и ранее. Ее используют для создания децентрализованного программного обеспечения. В данной архитектуре под сервисом понимается процесс, выполняемый операционной системой, который взаимодействует с другими процессами по сетевому интерфейсу для достижения общей цели [7]. Микросервисы могут быть построены с применением любого языка или фреймворка, но должны использовать общий протокол, который позволяет скрыть особенности реализации каждого из микросервисов. При этом каждый микросервис работает лишь с небольшой, максимально ограниченной областью задач, выполняя минимум функций.

На сегодняшний день не существует четкого определения MSA. Тем не менее, из всего многообразия встречающихся в литературе формулировок попытаемся выделить те особенности данного подхода, которые особенно важны при проектировании распределенных систем ЧПУ.

1. Архитектура микросервисов позволяет легко заменять модули, входящие в состав системы. Данная особенность позволяет повысить гибкость оборудования. В качестве примера можно представить человеко-машинный интерфейс для взаимодействия с оборудованием.
2. Все модули организованы вокруг функций. Архитектура микросервисов позволяет разделять функционал каждого блока [8]. В качестве примера рассмотрим фрезерную головку. С точки зрения архитектуры в данном блоке объединены два микросервиса: низкоуровневый (управление приводом

[1] Dynamic Host Configuration Protocol

шпинделя и датчики) и высокоуровневый (интерфейс пользователя с доступом к элементам управления). Это позволяет оградить систему от нежелательных в данный момент действий пользователя.

С другой стороны, каждый из микросервисов отвечает строго за свою функцию системы и ничего не знает о реализации других микросервисов, что, безусловно, приводит к возможному дублированию, поскольку одна и та же функция может быть реализована в разных микросервисах, причем по-разному. Однако это никак не нарушает принципа слабых связей, а относится скорее к дисциплине написания кода. Концепция микросервисов подразумевает возможность бесшовного связывания разных модулей, написанных с применением разных языков программирования, но не запрещает унификацию используемых средств программирования.

Каждый микросервис является эластичным, легко модифицируемым, но при этом законченным программным продуктом. Данное утверждение подразумевает, что при разработке УЧПУ необходимо придерживаться принципа увеличения связности и уменьшения связанности. Это позволяет, с одной стороны, сфокусироваться на разработке и отладке каждого отдельного блока, а с другой – дает возможность упростить методику добавления и модификации функций системы в целом.

В последнее время все чаще появляются публикации, описывающие применение микросервисной архитектуры при разработке реконфигурируемых промышленных систем. В частности, в работе коллектива под руководством Р. да Силвы [9] рассматривается архитектура управления для реконфигурируемых производственных систем на основе MSA и агент-холонического подхода. Предлагаемая система поддерживает обмен знаниями в гетерогенной производственной среде и позволяет управлять оборудованием, находящимся на территориально удаленных производственных площадках. В работе доказано, что используемая архитектура позволяет подобной системе оставаться надежной и отказоустойчивой даже в сложных производственных условиях.

Работа Т. Вреска и И. Чаврака [10] описывает применение MSA для увеличения интероперабельности и масштабируемости Интернета вещей. Известно, что данная концепция в последнее время все чаще ассоциируется с понятием Индустрии 4.0 [11], следовательно, представленный авторами подход может быть применен и при разработке УЧПУ.

Из всего вышесказанного можно сделать вывод о том, что MSA является современным и быстро развивающимся направлением развития CPPS и может быть успешно использована для создания распределенных УЧПУ.

Описание взаимодействия микросервисов

Так как лежащая в основе рассматриваемого подхода децентрализованная сеть является гетерогенной, необходимо использовать такую систему обмена сообщениями между узлами сети, которая не зависит от используемого транспортного протокола. Подобное решение дает возможность проектировать модули системы максимально независимыми и сфокусированными на решении одной конкретной задачи. Также важно помнить о том, что наибольшей производительности сети передачи данных можно достичь за счет применения легковесной шины сообщений и максимального сокращения коммуникаций между компонентами системы [12].

Легко заметить, что подобные сообщения очень похожи на те, которые используются в подобных протоколах, но с одним существенным отличием. Все эти протоколы используются в крупных промышленных приложениях и явно не подходят для встраиваемых систем. Для создания легковесной, открытой и расширяемой системы управления необходимо минимизировать слой абстракции, максимально используя преимущества низкоуровневых сокетов в сочетании с возможностью осуществлять маршрутизацию сообщений. В связи с этим наиболее подходящим способом взаимодействиями модулей проектируемого УЧПУ являются так называемые очереди сообщений.

Очередь сообщений является асинхронным протоколом передачи данных, т.е. отправитель и получатель сообщения взаимодействуют друг с другом не напрямую, а только через очередь. Как правило, очереди сообщений имеют явные или неявные ограничения на размер хранимых сообщений, поэтому данные отправителя передаются получателю в виде одного или более сообщений.

На сегодняшний день существует достаточное количество различных реализаций описанного подхода. Все их можно разделить на коммерческие (проприетарные) и свободные (с открытой лицензией). Среди коммерческих решений следует отметить следующие фреймворки и библиотеки: IBM WebSphere MQ, Oracle Advanced Queuing, Amazon Simple Queue Service, StormMQ, IronMQ. Очереди сообщений реализованы в популярных коммерческих операционных системах реального времени QNX и VxWorks. Наиболее интересными библиотеками с открытым исходным кодом представляются Apache ActiveMQ, Apache Qpid, HTTPSQS, RabbitMQ, Tarantool, NATS Messaging, Nanomsg, RestMQ.

Вследствие того, что разрабатываемая система является открытой, был проведен сравнительный анализ только инструментальных средств с открытым исходным кодом. На основании анализа были сформулированы следующие требования к программной библиотеке.

1. Библиотека должна быть написана на языках программирования C/C++, так как в состав проектируемой системы входят компоненты на основе микроконтроллеров, программирование которых возможно только на этих языках.
2. Библиотека должна обеспечивать доступ к очереди сообщений без участия брокера – специального узла, осуществляющего адресацию, маршрутизацию и массовое обслуживание.
3. Имеется открытый интерфейс прикладного программирования для добавления новых транспортных протоколов.

С учетом вышеизложенных требований, наиболее предпочтительной библиотекой, реализующей очередь сообщений, является Nanomsg. Данная библиотека написана на языке программирования C без дополнительных зависимостей. Nanomsg достаточно компактна, что позволяет использовать ее не только в UNIX-подобных, но и во встроенных системах. Данные в очереди на основе Nanomsg передаются в виде бинарных объектов, поэтому для передачи записей из базы данных предполагается использовать Messagepack – бинарный формат, полностью совместимый со стандартом JSON и позволяющий упаковывать данные на 15–20% эффективнее [13, 14].

В разрабатываемой системе используются два вида сообщений: децентрализованный обмен между узлами (или P2P-пакеты) и широковещательные пакеты (рис. 1).

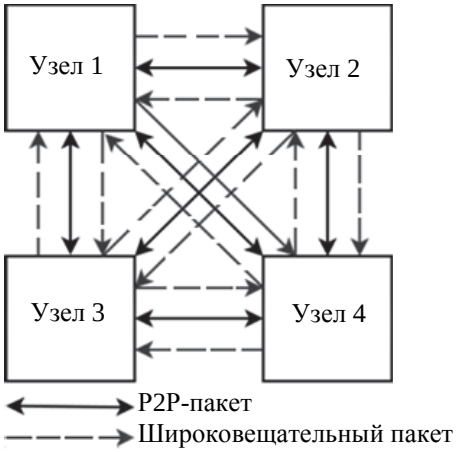


Рис. 1. Взаимодействие узлов через очереди сообщений

Очевидно, что описываемая открытая архитектура не позволяет использовать традиционные реляционные базы данных. В этой связи целесообразно использование так называемых NoSQL баз данных, обладающих такими особенностями, как доступность содержимого и отсутствие схемы описания содержания и структуры данных. На сегодняшний день существуют три базовых реализации парадигмы NoSQL.

1. Хранилища данных вида «ключ-значения», являющиеся обыкновенными хэш-таблицами. Используются в основном как альтернатива распределенным файловым системам и могут быть применены для создания различных файловых кэшей, а также в задачах, использующих «большие данные». К подобным решениям относятся следующие системы: Dynamo, Berkeley DB, FoundationDB, infinityDB, Redis.
2. Документ-ориентированные базы данных, представляющие собой хранилища иерархически структурированных данных. Основные представители данной категории – CouchDB, MongoDB, eXist.
3. Графовые базы данных, используемые для представления данных с большим количеством связей. В качестве примеров подобных баз данных можно привести OrientDB, AllegroGraph, InfiniteGraph.

Что касается применения систем управления базами данных, для реализации микросервисной архитектуры системы управления оборудованием с ЧПУ необходима база данных, совмещающая в себе преимущества документ-ориентированных баз данных и хранилищ вида «ключ–значение». Исходя из этого, были сформулированы следующие требования.

– реализация, не требующая наличия выделенного сервера;
– наличие атомарности при передаче данных и поддержка транзакций;
– возможность хранения всех данных в одном файле без использования временных файлов;
– кроссплатформенность и унификация на уровне интерфейса прикладного программирования;
– свободная лицензия.

По данным критериям был проведен сравнительный анализ. Были рассмотрены такие средства, как MongoDB, Redis и CouchDB, и в результате анализа по совокупности достоинств и недостатков была выбрана база данных UnQLite, основная особенность которой – отсутствие сервера. В отличие от других известных нереляционных движков, UnQLite не требует установки или настройки, а также не запускает отдельного процесса для доступа к данным.

Все данные хранятся в виде одного файла с сериализованными по стандарту JSON данными. В остальном UnQLite является хранилищем пар «ключ–значение» с поддержкой курсоров и возможностью хранить данные как на диске, так и в оперативной памяти. В дополнение к этому UnQLite может работать и как хранилище документов, а также поддерживает транзакции. Движок UnQLite целиком написан на языке C, что позволяет использовать данную систему управления базами данных даже на встроенных системах, используемых в проекте.

В рассматриваемом проекте UnQLite будет применена для хранения структурированных данных (модули системы, связанные с реализацией пользовательского интерфейса контроллера и сложной логики), а также как быстрое хранилище данных, получаемых от датчиков (хранилище вида «ключ-значение»). В последнем режиме UnQLite поддерживает транзакции, а данные, хранимые подобным образом, наиболее подходят для быстрой передачи через очередь сообщений.

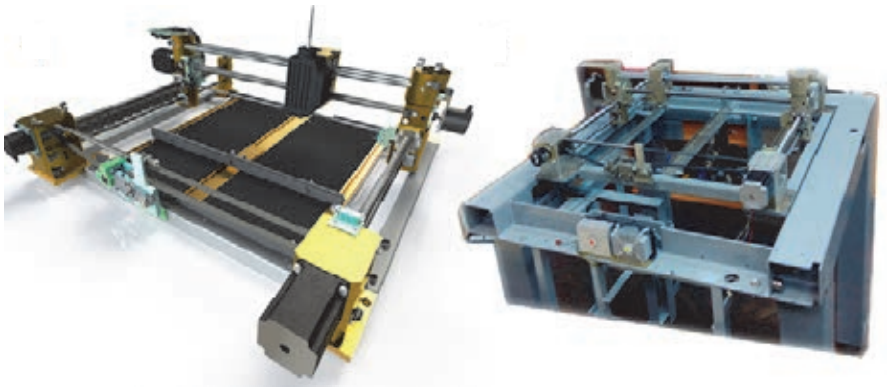
Также необходимо отметить, что при разработке распределенной модульной системы существует ряд задач, решения которых значительно отличаются от тех, которые применяются в монолитных системах. К таким задачам относятся обеспечение согласованности информации, регистрация новых модулей, мониторинг состояния зарегистрированных модулей, а также их поиск в зависимости от выполняемых функций.

Так как разрабатываемая система базируется на микросервисной архитектуре, в первую очередь необходимо обеспечивать согласованность информации о микросервисах. Это означает, что каждый компонент распределенной системы должен обладать постоянным доступом к актуальной информации обо всей системе, а также иметь возможность добавлять данные самостоятельно.

Поскольку модульное технологическое оборудование подразумевает поддержку принципа Plug–and–Produce [15], т.е. возможность подключать и использовать программные и физические модули без предварительной ручной настройки, то требуется обеспечить автоматическую регистрацию микросервисов. Из этого следует, что микросервисы должны самостоятельно сообщать о своем существовании системе управления. Более того, необходима постоянная проверка доступности зарегистрированных сервисов, т.е. мониторинг.

Практическое применение подхода

На основе предложенного подхода был разработан прототип системы управления установкой с ЧПУ. Разработанная экспериментальная установка предназначена для селективного отверждения фотополимера при производстве печатных плат. Данная область применения была выбрана по причине того, что в настоящее время не существует устройств, реализующих весь цикл производства плат для единичного и мелкосерийного производства с приемлемыми экономическими и временными затратами.



а б

Рис. 2. Внешний вид установки: в программе Solidworks (а); в собранном виде (б)

Экспериментальная установка представляет собой устройство для обработки лазерным излучением поверхностей произвольной формы (рис. 2). В состав установки входят неподвижный рабочий стол для размещения обрабатываемого объекта, лазерная головка (каретка), размещенная над рабочим столом с возможностью перемещения в горизонтальной плоскости по двум координатам, а также источник лазерного излучения (лазерный модуль), оптически связанный с лазерной головкой [16].

Лазерная головка снабжена гиростабилизированным подвесом, созданным на основе модифицированной платформы Стюарта. Для перемещения платформы используются линейные пьезодвигатели. Подвес позволяет отклонять лазерный луч в двух взаимно перпендикулярных плоскостях, а также изменять фокусное расстояние за счет передвижения объектива головки вдоль оси *Z*. Наличие подвеса позволяет решать сразу несколько задач. Первая из них, как уже было сказано, – это фокусировка лазерного луча в точке обработки. Вторая – возможность обрабатывать поверхности сложной формы за счет наклона оптической оси. Наклон позволяет лазерному лучу всегда оставаться перпендикулярным обрабатываемой

поверхности, соответственно, пятно контакта не искажается. Третья – активная компенсация возникающих вибраций.

Для регистрации вибраций применяются высокоточные одноосевые твердотельные акселерометры, а также камера с телескопическим объективом, которая в реальном времени отслеживает форму пятна. В дополнение к этому лазерная головка оснащена второй камерой, для автоматического выставления нулевой точки установки и наблюдения за зоной обработки.

Система управления установки состоит из следующих модулей (рис. 3).

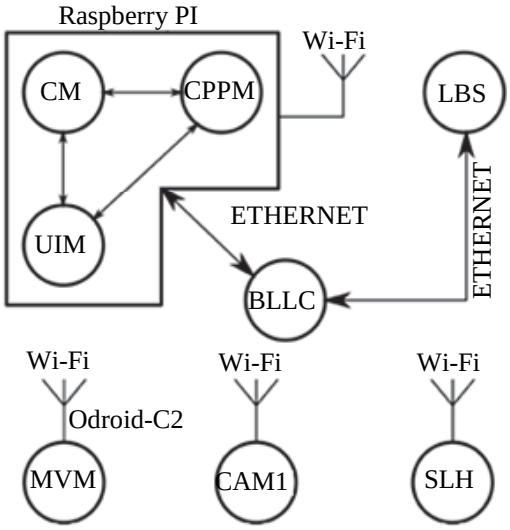


Рис. 3. Модули, входящие в систему управления

Коммуникационный модуль (Communication Module, CM). Физически представляет собой микрокомпьютер Raspberry Pi, с установленной на него операционной системой Debian Linux. Функции данного модуля – регистрация и диспетчирование остальных модулей (обеспечение отказоустойчивости системы в целом).

Основной низкоуровневый контроллер управления (Base Low Level Controller, BLLC). Физически является блоком управления, построенным на базе микроконтроллера архитектуры Cortex-M4 с установленной на него операционной системой реального времени FreeRTOS. Основное назначение BLLC – генерация управляющих сигналов для электрических приводов координатного стола и обработка сигналов от датчиков. Контроллер полностью автономен и может работать даже без подключения к основной управляющей сети.

Входными данными данного модуля являются программы на языке ISO-7bit, иначе называемые G-кодами. G-код загружается по протоколу Nanomsg, в качестве физического протокола используется универсальный асинхронный приемопередатчик. По этому же протоколу BLLC передает в сеть данные от датчиков, которые могут быть получены и интерпретированы любыми узлами управляющей сети. В остальном модуль полностью независим и после загрузки и обработки G-кода работает в автономном режиме. Внутренняя логика BLLC самостоятельно отрабатывает все нештатные ситуации даже при потери связи с управляющей сетью. Также информация обо всех событиях, которые регистрирует BLLC, передается в сеть и записывается в журнал событий.

Помимо управляющих программ BLLC может получать одиночные управляющие сигналы, например: «Переместить каретку в точку с координатами *X* и *Y*», «Двигаться вдоль координаты *X* со скоростью *V*», «Начать процедуру самокалибровки». Подобные команды необходимы для взаимодействия установки с оператором, а также для процедуры автоматического поиска нуля заготовки (реперной точки), инициируемой модулем машинного зрения. В дополнение к этому низкоуровневый контроллер управления имеет отладочный порт, который позволяет оператору взаимодействовать с ним посредством текстовых команд. Данная функция необходима на этапе отладки системы управления и в случае возникновения нештатных ситуаций, связанных с неполадками в сети.

Также BLLC передает по сети управляющий сигнал источнику лазерного излучения. Контроллер может подать команду на включение или выключение лазера или же задать мощность излучения (аналог скорости вращения инструмента). Между BLLC и LBS происходит двусторонний обмен данными. BLLC подает один из управляющих сигналов, LBS получает их и выдает BLLC подтверждение готовности, после чего изменяет состояние лазера. В случае отказа или иной нештатной ситуации BLLC получает соответствующий ответ от LBS. Параллельно этот сигнал получают и другие узлы управляющей сети, а вся система переходит в режим восстановления. То же происходит, если связь с LBS потеряна.

Источник лазерного излучения (Laser Beam Source, LBS). Имеет схожую с BLLC архитектуру, т.е. тоже построен на базе микроконтроллера Cortex-M4 с операционной системой FreeRTOS и также подключен к общей сети. Функции LBS – включать и выключать лазер (осуществляется за счет изменения положения подвижного зеркала), изменять его мощность (за счет широтно-импульсной модуляции), получать и обрабатывать данные от датчиков. LBS имеет внутреннюю логику и способен в автономном режиме отрабатывать нештатные ситуации и передавать данные от датчиков в сеть. В случае сбоев LBS передает сигнал аварийного останова и перехода в режим восстановления.

Интеллектуальная лазерная головка (Smart Laser Head, SLH). SLH представляет собой программно-аппаратный модуль, построенный на базе микроконтроллера Cortex-M4 с операционной системой «FreeRTOS». Основные функции SLH – компенсация механических вибраций, изменение наклона оптической оси для обработки сложных поверхностей и автофокусировка лазерного луча. Как и все остальные, модуль автономен и самостоятельно обрабатывает все нештатные ситуации с передачей соответствующих сигналов в сеть.

Модуль машинного зрения (Machine Vision Module, MVM). Является программно-аппаратным модулем, построенным на базе микрокомпьютера Odroid-C2 с операционной системой Debian Linux. Функции MVM – поиск реперной точки на заготовке при помощи связанного с ним модуля камеры (обозначена как CAM1 на рис. 3), передача отображаемого в интерфейсе оператора видеопотока. Взаимодействуют BLLC и MVM по следующему алгоритму: перед запуском управляющей программы на исполнения BLLC запрашивает у MVM координаты реперной точки, MVM начинает сканирование рабочей области (за счет передачи управляющих команд BLLC), после нахождения реперной точки возвращает BLLC координаты ее центра и переходит в режим ожидания команд.

Отказ MVM является критичным только в процессе поиска нуля заготовки, в остальных случаях CM просто оповещает оператора, что MVM недоступен, при этом обработка не прекращается. MVM имеет внутреннее хранилище для размещения базы данных, в которой хранятся журнал событий и компоненты пользовательского интерфейса.

Модуль подготовки управляющих программ (Control Programs Preprocessing Module, CPPM). Используется для подготовки управляющих G-кодов непосредственно в браузере. Как и интерфейсы аппаратных модулей, является web-приложением, которое UIM динамически включает в общий интерфейс установки. Физически расположен на том же микрокомпьютере, что и CM с UIM, и может быть вынесен на отдельный сервер.

Модуль интерфейса пользователя (User Interface Module, UIM). Является программным модулем и представляет собой агрегатор элементов управления установкой (экранов и виджетов) для отображения их в браузере. Элементы управления подгружаются в него автоматически в процессе взаимодействия с модулями. Основная идея заключается в том, что вся логика интерфейса написана с использованием стека технологий web-программирования и хранится непосредственно в базе данных каждого из модулей. В процессе инициализации установки все эти статические файлы загружаются в UIM, а затем передаются по сети для отображения в браузере.

С точки зрения реализации UIM – это высокопроизводительный многопоточный веб-сервер, написанный на языке программирования Go, и библиотека отображения элементов управления интерфейса (кнопки, переключатели, ползунки и т.д.), виджетов и экранов. Физически этот сервер расположен на том же микрокомпьютере, что и CM.

Компонент графического интерфейса выступает в роли агрегатора. При выполнении пользователем какой-либо операции компонент должен либо напрямую найти микросервис, отвечающий за ее исполнение, либо оставить задачи поиска отдельному слою абстракции. В разрабатываемом УЧПУ применяется второй способ, поскольку он дает возможность не встраивать механизмы поиска в каждый микросервис и позволяет наглядно реализовать холонический подход[1].

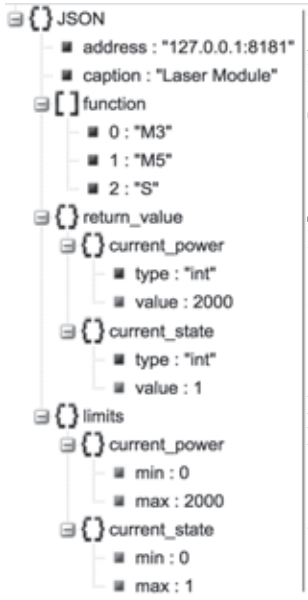
Абстрактный слой работы с микросервисами базируется на распределенном хранилище, в котором каждый микросервис создает набор слотов для работы с ним извне. Слот – это запись типа «ключ–значение», из которой микросервис извлекает необходимые данные. Слот хранится непосредственно в базе данных UnQLite и включает в себя как минимум следующие поля:

- адрес в сети;
- название;
- функции;
- возвращаемое значение;
- пределы возвращаемого значения, свидетельствующие о нормальной работе сервиса.

В качестве адреса выступают IP-адрес и порт, необходимые для взаимодействия с микросервисом. Название должно быть представлено в строковом виде или числовым идентификатором. Доступные для выполнения функции описываются набором G-кодов [17]. Возвращаемое значение необязательно долж-

[1] https://linuxcnc.org/docs/html/gcode.html

но быть одним. Описание каждого из них включает в себя тип (вещественное число или целое число), пределы и значение (рис. 4).



Адрес в сети
Название модуля
Доступные команды в нотации G-кодов
Доступные для чтения данные модуля в сети
Мощность лазера
Тип данных
Значение
Состояние
Пределы возвращаемых значений*
Предел мощности лазера
Значение неаварийных состояний
* применяются для отслеживания ситуаций

Рис. 4. Слот лазерного модуля

Такие записи могут образовывать структуру (холон). В таком случае модули, образующие холон, дополнительно регистрируются под общим именем и объединяют остальные поля. G-коды, посылаемые холону, отправляются на все адреса участников холона, при этом те участники, которые не могут выполнить G-код, игнорируют его.

Заключение

В настоящей работе был рассмотрен новый подход к проектированию архитектуры модульного промышленного оборудования с числовым программным управлением. В качестве основы разрабатываемой системы предлагается использование распределенной коммуникационной сети, каждый узел которой является микросервисом. За счет интероперабельности подобная система может быть легко интегрирована в киберфизическую производственную среду.

Был проведен анализ средств разработки распределенного программного обеспечения с микросервисной архитектурой, протоколов взаимодействия компонентов системы и баз данных. В качестве основного протокола взаимодействия была выбрана система, обеспечивающая передачу данных за счет использования очереди сообщений. Данный подход делает модули системы максимально автономными, упрощает масштабирование, дает возможность сглаживания пиковых нагрузок в сети и обеспечивает повышение отказоустойчивости за счет гарантированной доставки сообщений. Программная реализация протокола основана на библиотеке Nanomsg; используются широковещательные сообщения и пиринговый обмен.

Хранение данных в системе осуществляется за счет использования распределенной нереляционной базы данных UnQLite. Система реализована в виде удобной библиотеки и за счет схожего интерфейса хранения данных легко интегрируется с используемой очередью сообщений Nanomsg.

Представлена реализация предложенного подхода на примере установки для селективного лазерного отверждения фотополимеров на поверхностях произвольной формы. Подробно описаны все модули системы и алгоритм их взаимодействия. Показаны достоинства используемой распределенной микросервисной архитектуры по сравнению с классическими монолитными решениями.

Литература

1. Grigoriev S.N., Martinov G.M. Research and development of a cross-platform CNC kernel for multi-axis machine tool // Procedia CIRP. 2014. V. 14. P. 517–522. doi: 10.1016/j.procir.2014.03.051
2. Li B., Zhou Y., Tang X. A research on open CNC system based on architecture/component software reuse technology // Computers in Industry. 2004. V. 55. N 1. P. 73–85. doi: 10.1016/j.compind.2003.10.011
3. Ma X., Han Z., Wang Y., Fu H. Development of a PC-based

References

1. Grigoriev S.N., Martinov G.M. Research and development of a cross-platform CNC kernel for multi-axis machine tool. *Procedia CIRP*, 2014, vol. 14, pp. 517–522. doi: 10.1016/j.procir.2014.03.051
2. Li B., Zhou Y., Tang X. A research on open CNC system based on architecture/component software reuse technology. *Computers in Industry*, 2004, vol. 55, no. 1, pp. 73–85. doi: 10.1016/j.compind.2003.10.011
3. Ma X., Han Z., Wang Y., Fu H. Development of a PC-based

open architecture software-CNC system // Chinese Journal of Aeronautics. 2007. V. 20. N 3. P. 272–281. doi: 10.1016/S1000-9361(07)60044-2
4. Morales-Velasquez L., Romero-Troncoso R., Osornio-Rios R., Herrero-Ruiz G., Cabal-Yepez E. Open-architecture system based on a reconfigurable hardware/software multi-agent platform for CNC machines // Journal of Systems Architecture. 2010. V. 56. N 9. P. 407–418. doi: 10.1016/j.sysarc.2010.04.009
5. Verba N., Chao K., James A., Goldsmith D., Fei X., Stan S. Platform as a service gateway for the Fog of Things // Advanced Engineering Informatics. 2016. V. 33. P. 243–257. doi: 10.1016/j.aei.2016.11.003
6. Prazeres C., Serrano M. SOFT-IoT: Self-organizing Fog of Things // Proc. 30th Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA). Crans-Montana, Switzerland, 2016. P. 803–808. doi: 10.1109/WAINA.2016.153
7. Rafighi M., Farjami Y., Modiri N. Studying the deficiencies and problems of different architecture in developing distributed systems and analyze the existing solution // Proc. 2nd Int. Conf. on Knowledge-Based Engineering and Innovation (KBEI). Tehran, Iran, 2015. P. 826–834. doi: 10.1109/KBEI.2015.7436151
8. Colombo A.W., Karnouskos S., Mendes J.M., Leitao P. Industrial agents in the era of service-oriented architectures and cloud-based industrial infrastructures / In: Industrial Agents: Emerging Applications of Software Agents in Industry. Boston: Morgan Kaufmann, 2015. P. 67–87. doi: 10.1016/B978-0-12-800341-1.00004-8
9. da Silva R.M., Junqueira F., Filho D.J., Miyagi P. Control architecture and design method of reconfigurable manufacturing systems // Control Engineering Practice. 2016. V. 49. P. 87–100. doi: 10.1016/j.conengprac.2016.01.009
10. Vresk T., Cavrak I. Architecture of an interoperable IoT platform based on microservices // Proc. 39th Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija, Croatia, 2016. P. 1196–1201. doi: 10.1109/MIPRO.2016.7522321
11. Zezulka F., Marcon P., Vesely I., Sajdl O. Industry 4.0 – an introduction in the phenomenon // IFAC-PapersOnLine. 2016. V. 49. N 25. P. 8–12. doi: 10.1016/j.ifacol.2016.12.002
12. Rheddane A.E., Palma N.D., Tchana A., Hagimont D. Elastic message queues // Proc. IEEE 7th Int. Conf. on Cloud Computing. 2014. P. 17–23. doi: 10.1109/CLOUD.2014.13
13. Maeda K. Performance evaluation of object serialization libraries in XML, JSON and binary formats // Proc. 2nd Int. Conf. on Digital Information and Communication Technology and it's Applications (DICTAP). Bangkok, Thailand, 2012. P. 177–182. doi: 10.1109/DICTAP.2012.6215346
14. Popic S., Pezer D., Mrazovac B., Teslic N. Performance evaluation of using protocol buffers in the Internet of Things communication // Proc. 1st Int. Conf. on Smart Systems and Technologies. Osijek, Croatia, 2016. P. 261–265. doi: 10.1109/SST.2016.7765670
15. Danny P., Ferreira P., Lohse N., Guedes M. An AutomationML model for plug-and-produce assembly systems // Proc. IEEE 15th Int. Conf. on Industrial Informatics. Emden, Germany, 2017. P. 849–854. doi: 10.1109/indin.2017.8104883
16. Giret A., Botti V. Engineering holonic manufacturing systems // Computers in Industry. 2009. V. 60. N 6. P. 428–440. doi: 10.1016/j.compind.2009.02.007
17. Федосов Ю.В., Афанасьев М.Я. Устройство для обработки лазерным излучением поверхности объекта произвольной формы // Научно-технический вестник информационных технологий, механики и оптики. 2017. Т. 17. № 1(107). С. 191–195. doi: 10.17586/2226-1494-2017-17-1-191-195

open architecture software-CNC system. *Chinese Journal of Aeronautics*, 2007, vol. 20, no. 3, pp. 272–281. doi: 10.1016/S1000-9361(07)60044-2
4. Morales-Velasquez L., Romero-Troncoso R., Osornio-Rios R., Herrero-Ruiz G., Cabal-Yepez E. Open-architecture system based on a reconfigurable hardware/software multi-agent platform for CNC machines. *Journal of Systems Architecture*, 2010, vol. 56, no. 9, pp. 407–418. doi: 10.1016/j.sysarc.2010.04.009
5. Verba N., Chao K., James A., Goldsmith D., Fei X., Stan S. Platform as a service gateway for the Fog of Things. *Advanced Engineering Informatics*, 2016, vol. 33, pp. 243–257. doi: 10.1016/j.aei.2016.11.003
6. Prazeres C., Serrano M. SOFT-IoT: Self-organizing Fog of Things. *Proc. 30th Int. Conf. on Advanced Information Networking and Applications Workshops, WAINA*. Crans-Montana, Switzerland, 2016, pp. 803–808. doi: 10.1109/WAINA.2016.153
7. Rafighi M., Farjami Y., Modiri N. Studying the deficiencies and problems of different architecture in developing distributed systems and analyze the existing solution. *Proc. 2nd Int. Conf. on Knowledge-Based Engineering and Innovation, KBEI*. Tehran, Iran, 2015, pp. 826–834. doi: 10.1109/KBEI.2015.7436151
8. Colombo A.W., Karnouskos S., Mendes J.M., Leitao P. Industrial agents in the era of service-oriented architectures and cloud-based industrial infrastructures. In *Industrial Agents: Emerging Applications of Software Agents in Industry*. Boston, Morgan Kaufmann, 2015, pp. 67–87. doi: 10.1016/B978-0-12-800341-1.00004-8
9. da Silva R.M., Junqueira F., Filho D.J., Miyagi P. Control architecture and design method of reconfigurable manufacturing systems. *Control Engineering Practice*, 2016, vol. 49, pp. 87–100. doi: 10.1016/j.conengprac.2016.01.009
10. Vresk T., Cavrak I. Architecture of an interoperable IoT platform based on microservices. *Proc. 39th Int. Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO*. Opatija, Croatia, 2016, pp. 1196–1201. doi: 10.1109/MIPRO.2016.7522321
11. Zezulka F., Marcon P., Vesely I., Sajdl O. Industry 4.0 – an introduction in the phenomenon. *IFAC-PapersOnLine*, 2016, vol. 49, no. 25, pp. 8–12. doi: 10.1016/j.ifacol.2016.12.002
12. Rheddane A.E., Palma N.D., Tchana A., Hagimont D. Elastic message queues. *Proc. IEEE 7th Int. Conf. on Cloud Computing*, 2014, pp. 17–23. doi: 10.1109/CLOUD.2014.13
13. Maeda K. Performance evaluation of object serialization libraries in XML, JSON and binary formats. *Proc. 2nd Int. Conf. on Digital Information and Communication Technology and it's Applications, DICTAP*. Bangkok, Thailand, 2012, pp. 177–182. doi: 10.1109/DICTAP.2012.6215346
14. Popic S., Pezer D., Mrazovac B., Teslic N. Performance evaluation of using protocol buffers in the Internet of Things communication. *Proc. 1st Int. Conf. on Smart Systems and Technologies*. Osijek, Croatia, 2016, pp. 261–265. doi: 10.1109/SST.2016.7765670
15. Danny P., Ferreira P., Lohse N., Guedes M. An AutomationML model for plug-and-produce assembly systems. *Proc. IEEE 15th Int. Conf. on Industrial Informatics*. Emden, Germany, 2017, pp. 849–854. doi: 10.1109/indin.2017.8104883
16. Giret A., Botti V. Engineering holonic manufacturing systems. *Computers in Industry*, 2009, vol. 60, no. 6, pp. 428–440. doi: 10.1016/j.compind.2009.02.007
17. Fedosov Yu.V., Afanasiev M.Ya. Apparatus for surface treatment of free-form object by laser radiation. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2017, vol. 17, no. 1, pp. 191–195. (In Russian) doi: 10.17586/2226-1494-2017-17-1-191-195

Авторы

Афанасьев Максим Яковлевич – кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Scopus ID: 57194081345, ORCID ID: 0000-0003-4061-1407, amax@niuitmo.ru

Authors

Maxim Ya. Afanasiev – PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, Scopus ID: 57194081345, ORCID ID: 0000-0003-4061-1407, amax@niuitmo.ru

Федосов Юрий Валерьевич – кандидат технических наук, инженер, ОАО «Российский институт мощного радиостроения», Санкт-Петербург, 199048, Российская Федерация; доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация; Scopus ID: 57194080548, ORCID ID: 0000-0003-1869-0081, yf01@yandex.ru

Крылова Анастасия Андреевна – аспирант, программист, ООО «ЛАР Технологии», Санкт-Петербург, 197342, Российская Федерация; инженер, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, ORCID ID: 0000-0002-5822-6702, ananasn94@gmail.com

Шорохов Сергей Александрович – аспирант, инженер, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, ORCID ID: 0000-0002-5412-7723, stratumxspb@gmail.com

Yu. V. Fedosov – PhD, engineer, JSC “Russian Institute of Power Radiobuilding”, Saint Petersburg, 199048, Russian Federation; Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation; Scopus ID: 57194080548, ORCID ID: 0000-0003-1869-0081, yf01@yandex.ru

Anastasiya A. Krylova – postgraduate, Software developer, Lar Technologies, LLC, Saint Petersburg, 197342, Russian Federation; engineer, ITMO University, Saint Petersburg, 197101, Russian Federation, ORCID ID: 0000-0002-5822-6702, ananasn94@gmail.com

Sergey A. Shorokhov – postgraduate, engineer, ITMO University, Saint Petersburg, 197101, Russian Federation, ORCID ID: 0000-0002-5412-7723, stratumxspb@gmail.com