5 важнейших факторов, которые надо учитывать при разработке приложений управления движением

При разработке приложения управления движением необходимо учесть множество факторов. Какую использовать технологию двигателей, как организовать в системе обратную связь, как синхронизировать движения и обеспечить обмен данными между различными частями системы – на все эти вопросы вы должны ответить при проектировании высокопроизводительной, надежной системы.

1. Технология двигателя

Важный момент при выборе правильной технологии двигателя – определения правильных требований к крутящему моменту и скорости для конкретного приложения. Производительность системы определяется путем анализа кривой момент-скорость, но кроме производительности необходимо учесть множество других факторов. Например, для некоторых приложений может потребоваться, чтобы двигатель подвергался обработке в автоклаве, или был влагозащищенным, или поддерживал конкретный вид обратной связи, или удовлетворял определенным правовым нормам. После рассмотрения этих требований вы можете выбрать конкретный тип двигателя. Среди самых распространенных типов двигателей – шаговые двигатели, щеточные и бесщеточные серводвигатели и асинхронные двигатели переменного тока. Различные типы двигателей имеют различные характеристики, наилучшим образом подходящие для различных типов приложений.

Для многих приложений все, что вам нужно – это перемещение в заданное положение без каких-либо ограничений, например, без управления коэффициентом демпфирования системы. Для таких приложений сначала рассмотрите вариант с шаговым двигателем, главные преимущества которого – простота использования, низкая цена и высокий крутящий момент при низких скоростях. У шаговых двигателей известно число шагов на оборот и их можно заставить повернуться на заданное число шагов. Поэтому они могут работать без обратной связи, в разомкнутых системах. Шаговые двигатели могут управляться двумя фазами, которые всегда обеспечивают подачу энергии, чтобы двигатель выдавал максимальный вращающий момент. Однако ток в обмотке можно аппроксимировать синусоидальным сигналом, изменяющим поданную на каждую обмотку энергию. Это позволяет двигателю совершать более мелкие шаги и работать более гладко. Такая техника известна как режим дробления шага (микрошаг). Однако размер микрошага имеет границы из-за физических ограничений системы.

Если в вашей системе требуются более высокие вращающие моменты при высоких скоростях, более быстрые движения или меньшее выделение тепла, рассмотрите возможность использования щеточных и бесщеточных серводвигателей. Эти двигатели при правильной настройке обладают великолепной точностью и жестко контролируемыми скоростью и ускорением. Обдумайте их использование для приложений, где требуются короткие, быстрые

перемещения или жестко контролируемые профили движения. Главное отличие между щеточными и бесщеточными двигателями постоянного тока заключается в способе коммутации. В то время, как в щеточных двигателях постоянного тока коммутация происходит с помощью щеток, контактирующих с валом, бесщеточным двигателям необходима электронная коммутация от привода. Это значит, что вы можете управлять щеточным двигателем постоянного тока без внешней обратной связи по скорости, но для управления любым способом бесщеточным двигателем необходима обратная связь или измерение противо-ЭДС.

Существуют и другие технологии двигателей, применяемые в различных приложениях и областях промышленности. Например, асинхронные двигатели переменного тока, как правило, используются в приложениях, где необходимы очень большие двигатели с высокими значениями моментов и энергии. Из-за их конструкции такие двигатели сложно заставить переместиться в заданное положение. Однако они замечательно справляются с управлением по скорости.

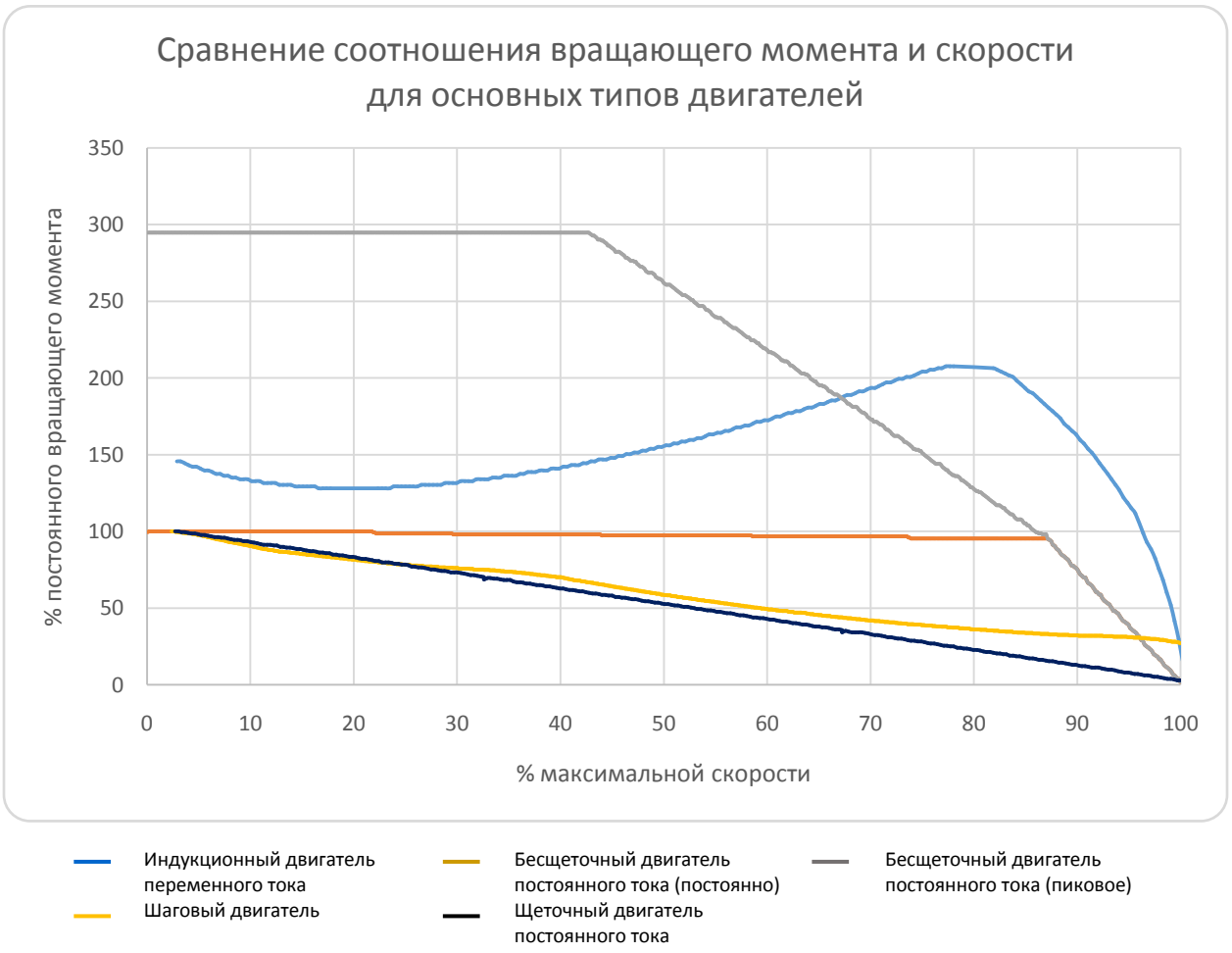

Рисунок 1. Соотношение момента и скорости для наиболее распространенных типов двигателей

После выбора двигателя важно определить, необходимо ли знать точное положение, скорость или ускорение двигателя. Разомкнутые системы с шаговым двигателем могут использоваться для управления положением, но без обратной связи невозможно проверить, находится ли система в заданном положении. Это может быть приемлемо в приложениях, работающих с конвейерной лентой, где допускается проскальзывание. Если важно знать конечное положение, или если этого

требует конструкция двигателя, используйте обратную связь. Для этого существует множество типов устройств обратной связи, например, энкодеры, вращающиеся трансформаторы (резольверы) и другие специальные датчики.

По умолчанию устройством обратной связи считается энкодер. Вы можете использовать оптические энкодеры в широком диапазоне приложений, в зависимости от их требований к температурам, ударам и вибрации. Энкодеры бывают относительными и абсолютными. Абсолютный энкодер сохраняет информацию при выключении и повторном включении питания. Абсолютные энкодеры бывают с несколькими оборотами (multirevolution) и с одним оборотом (single revolution). В системах, где требуется знать положение в начале работы без возможности совершить перемещение в известную исходную точку, лучше использовать абсолютный энкодер, который сохраняет текущее состояние даже при отсутствии питания.

Резольверы преобразуют сигнал в аналоговое напряжение путем установки трансформатора на валу двигателя. Они могут быть весьма точны и хороши для использования в неблагоприятных условиях окружающей среды, поскольку менее подвержены физическим повреждениям, чем энкодеры.

Описанные выше устройства, как правило, выдают информацию по цифровому или аналоговому каналу, но существуют и другие механизмы обратной связи, передающие данные по особым шинам. Существует множество отраслевых и патентованных стандартов, например, BiSS и EnDat. Во многих из этих стандартов обмен данными происходит по высокоскоростному последовательному физическому каналу.

2. Шина обмена данными

Учитывая огромное количество возможных топологий систем различных производителей, каждой системе может потребоваться обмен данными через несколько шин. Это особенно важно, если контроллер и привод созданы разными производителями, или если в одном приложении необходимо использовать различные приводы. Выбор шины может быть ограничен приводом, а также требованиями к приложению, например, помехоустойчивостью или детерминизмом.

Долгое время стандартным методом обмена данными между контроллером и приводом было использование аналоговых команд регулирования вращающего момента для серводвигателей и указателей направления для шагового двигателя. В первом случае привод ведет себя аналогично усилителю мощности и источнику ток так, чтобы он соответствовал заданному вращающему моменту, в то время как контуры положения и скорости замыкаются в контроллере. Во втором случае эффект аналогичен: двигатель просто подает питание в соответствующих фазах. Такие способы управления подвержены помехам, но позволяют осуществлять большую часть управления в самом контроллера и поддерживаются большинством приводов.

Последовательная шина также используется для обмена данными с приводами, но чаще всего только как метод конфигурирования. Многие приводы могут подключаться к хост-компьютеру через последовательный порт для настройки различных параметров, например, коэффициентов передачи или режима дробления шага.

Для большинства распределенных систем, требующих большей гибкости, часто используется Ethernet, благодаря его широкой доступности и наличии на большинстве предприятий. Это может быть хорошим решением для систем с большим количеством осей или строгими требованиями к масштабируемости. Одним из недостатков Ethernet является то, что стек Ethernet не является детерминированным, поэтому сложно синхронизировать движение. Однако это может быть неважно для многих приложений, где требуются независимые друг от друга движения, или которые могут передвигаться последовательно.

На основе Ethernet был создан EtherCAT. Он использует такой же физический уровень, но реализует архитектуру главный-подчиненный и известный маршрут передачи пакета. Это обеспечивают детерминированную схему обмена данными, являющуюся главным преимуществом EtherCAT. Также EtherCAT по своей сути более устойчив к электрическим помехам и имеет другие преимущества слоя Ethernet, например, расстояние между источником и приемником данных. EtherCAT требует наличия в сети главного устройства, которое будет исполнять роль контроллера, то есть посылать команды положения, скорости или момента в виде сообщений EtherCAT по цепи. Поэтому приводы должны обладать возможностью реализации контуров внутри и обмениваться данными с контроллером. Многие производители придерживаются стандарта управления движением CiA DS 402, определяющего типовые схемы обмена данными.

3. Координация

В приложениях управления движением часто требуется координация нескольких двигателей для обеспечения нескольких степеней свободы в системе; важно также оценивать профиль движения и требования к приложению, чтобы понять, необходимо ли синхронизировать или координировать несколько осей. Это может побудить использовать специализированные программные архитектуры или детерминированную шину обмена данными. Например, необходимо координировать оси механизмов портального крана или захвата.

Существует несколько способов синхронизировать движения по нескольким осям, но если координация не требуется, цикл генерации траектории может без сопряжения каждой отдельной оси, не заботясь о времени выдачи координат. В самой простой схеме координации траектория вычисляется в N-мерном пространстве, после чего результат раскладывается на соответствующие компоненты по осям. Затем точки этой траектории синхронно отправляются системам управления различных осей. Для многих систем требуются электронные кулачковые или зубчатые передачи, в которых ось будет отмечена как подчиненная другой. В этих случаях траектория подчиненной оси может быть вычислена путем масштабирования уставки главной оси.

Все эти виды схем управления позволяют программисту проделывать вычисления для синхронизации в фоновом режиме. Эти группы называются координатными системами и позволяют разработчику управлять движением в N-мерном пространстве, а не рассчитывать движение по каждой оси, необходимое для выполнения заданного профиля движения. Например, координатная система может получить команду движения по дуге, заданной радиусом и углом, вместо непосредственно расчета кривых скоростей и ускорения, необходимых для построения этой кривой.

Еще один пример синхронизации, которая может потребоваться в системах управления движением – синхронизация с другими компонентами системы, не относящимися к управлению движением. Часто в качестве ведущей в системе управления движением используется система машинного зрения. Алгоритмы машинного зрения находят в пространстве конкретный объект и отправляют его данные системе управления движением, которая осуществляет манипуляции с объектом. Примером такого приложения может послужить извлечение деталей из бункера: камеры обнаруживают детали и их расположение в пространстве, чтобы система управления движением могла их найти.

4. Архитектуры систем

Разобравшись с требованиями к оборудованию и синхронизации, вы можете решить, где будет выполняться управляющий код. Существуют два основных способа установки системы управления движением: или основная часть управляющего кода будет находиться в одном целевом устройстве, или вычисления будут поделены между несколькими узлами.

Примеры первой группы – компьютеры под управление Windows с платами управления движением или системы реального времени, непосредственно реализующие управление. Такие системы несколько проще устанавливать, поскольку последовательность всех движений находится в одном целевом устройстве, которое осуществляет все управление. Однако здесь существуют пределы по времени вычисления и возможностям оборудования. Например, у компьютера может быть ограниченное количество слотов PCI для ввода-вывода или для генерации траектории. Многие лабораторные установки вписываются в эту категорию, поскольку часто находятся в одной комнате и число двигателей в них ограничено.

Но можно разделить генерацию траектории и управление низкого уровня, или и то, и другое. Это можно проделать для приложений, которым требуется масштабируемость и реконфигурируемость, например, для промышленных производственных линий с модульными механизмами и контроллером в роли основного сервера. Иначе объем вычислений может превысить возможности одного контроллера. Распределение только управления низкого уровня будет означать, что один узел осуществляет генерацию траектории, но освобождает ресурсы главной системы; это можно реализовать, если главный контроллер будет посылать команды по вращающему моменту, скорости или положению для комплекта интеллектуальных приводов. Такая схема ограничена только возможностями синхронизации сети. В порядке альтернативы возможно распределить и схему управления высокого уровня, но это становится сложнее, если в системе необходима синхронизация движений, поскольку тогда потребуется и синхронизация команд к узлу и контуров управления. В эту категорию попадают системы с несколькими интеллектуальными приводами, получающими набор команд по положению, скорости или моменту.

5. Разработка программного обеспечения

Когда приняты все решения по оборудованию, остается только решить, каким образом управлять им из программы, а также потребуется ли обмен данными с другими аппаратными средствами. Некоторые системы управления движением работают в основном автономно, в то время как другие необходимо интегрировать в более крупные системы или с другими компонентами,

например, системами машинного зрения. В этих случаях важно обдумать, сможете ли вы использовать одну и ту же среду программирования для всех систем и компонентов.

Стоит подумать о том, возможно ли отделить определение оборудования в программе и программировать собственно движение, чтобы не пришлось переписывать код при замене приводов. Кроме того, стоит предусмотреть возможность тестирования генерируемых программой профилей движения путем запуска фрагментов кода в симуляторе без необходимости запуска двигателя. Обычно для этого точки генерируемой траектории отправляются не в систему управления, а выводятся на график или посылаются на симулируемый привод или модель двигателя и привода. Наконец, использование модели двигателя может помочь протестировать основную часть кода и некоторые настройки прежде, чем перейти физической реализации, однако для этого требуются очень хорошие знания механической системы, что не всегда возможно.

Об авторе

Саймон Перес Санта Мария – инженер-системотехник, сферой интересов автора являются системы управления движением и встроенные системы NI, Остин, Техас.