электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.93'1

Программно-аппаратная реализация нечеткого микроконтроллера

Д.Е. Бекасов, *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: О.В. Рогозин,к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
irudakov@bmstu.ru

Введение

Классическая теория автоматического управления (ТАУ) является научной дисциплиной, изучающей вопросы автоматического управления объектами разной физической природы [1]. В рамках этой дисциплины разработано большое число математических методов анализа и синтеза систем автоматического управления (САУ). Однако, существует целый класс актуальных задач, к которым классические методы ТАУ слабо применимы. Речь идет об управлении слабоструктурированными или плохо определенными объектами, которым присущи такие свойства, как уникальность, отсутствие формализуемой цели существования, отсутствие оптимальности, динамичность, нестационарность структуры и параметров, неполнота или практически полное отсутствие формального описания [2]. Плохо определенные объекты сравнительно мало изучены, что существенно затрудняет их формальное математическое описание и, как следствие, усложняет применение методов ТАУ. Классическая ТАУ детерминированными и стохастическими системами успешно применяется для построения САУ летательными аппаратами, энергетическими установками и т.п., но попытки распространения традиционных методов на такие области, как биосинтез, многофазные химико-технологические процессы, управление в условиях неопределенности и т.п., не дали ощутимых практических результатов, несмотря на все более усложняющиеся математические методы их описания [3].

На практике задачи такого рода успешно решаются человеком-оператором, в силу его способности наблюдать, анализировать, накапливать информацию, делать выводы и принимать решения. Благодаря своему интеллекту, человек может оперировать не только с количественными, но и с качественными неформализованными понятиями, вследствие чего довольно успешно справляется с неопределенностью и сложностью процесса управления [4]. Поэтому, одним из самых важных и перспективных направлений развития теории управления является построение и использование моделей приближенных рассуждений человека, то есть построение интеллектуальных САУ (ИСАУ)[3][5]. Интеллектуальная САУ способна в той или иной степени воспроизводить определенные интеллектуальные действия человека, связанные с приобретением, анализом, классификацией знаний в предметной области управления, а также оперирующих знаниями, накопленными человеком-оператором или самой системой в ходе практической деятельности по управлению объектом. Сфера применения ИСАУ широка – от промышленных роботов до бытовых приборов.

Характерными особенностями ИСАУ является использование информации в символьном виде и наличие возможности выбора (между вариантами в условиях неопределенности). При этом, применение ИСАУ позволяет повысить качество продукции при уменьшении затрат ресурсов и энергии, обеспечивают более высокую устойчивость к воздействию возмущающих факторов по сравнению с традиционными САУ [3].

Развитие теории ИСАУ привело к постановке следующих задач: разработки методов математического описания, анализа устойчивости и оценки качества ИСАУ; создания инструментальных средств проектирования и методов синтеза ИСАУ и т.п. [6].

1. Требования к ИСАУ

В 1989 г. создатель теории интеллектуальных машин Дж. Саридис разработал структурную организацию интеллектуальных систем, в которой ИСАУ представляется тремя обобщенными уровнями в соответствии с принципом IPDI[7]. При этом по мере продвижения к высшим уровням иерархической структуры повышается интеллектуальность системы, но снижается ее точность, и наоборот[3].

Интеллектуальность системы характеризуется способностью системы работать с базой событий с целью выявления знаний, позволяющих уточнить задачу и наметить пути ее решения. ИСАУ, отвечающая этому базовому принципу, архитектурно состоит из организационного, координационного и исполнительного уровней. Каждый из

обозначенных уровней реализуется отдельной подсистемой (также, возможно, иерархической), отвечающей изложенным ниже принципам организации ИСАУ[8].

1 Наличие взаимодействия управляющих систем с реальным внешним миром с использованием информационных каналов связи. Принцип взаимодействия системы с внешним миром позволяет организовать каналы связи для извлечения необходимых знаний с целью организации целесообразного поведения и активного воздействия на среду, если это необходимо.

2 Принципиальная открытость систем с целью повышения интеллектуальности и совершенствования собственного поведения. Открытость систем обеспечивается наличием способностей к самонастройке, самоорганизации и самообучению. Выполнение этого принципа позволяет организовать в интеллектуальной системе процесс приобретения, пополнения и верификации знаний.

3 Наличие механизмов прогноза изменения внешнего мира и собственного поведения системы в динамически меняющемся внешнем мире. Выполнение принципа обеспечивает возможность выхода из критических и непредсказуемых ситуаций.

4 Наличие у ИСАУ многоуровневой иерархической структуры, построенной в соответствии с IPDI. Данный принцип позволяет конструировать ИСАУ в тех случаях, когда неточность знаний о модели объекта управления или его поведении можно скомпенсировать за счет повышения интеллектуальности создаваемых систем или соответствующих алгоритмов управления.

5 Постоянство функционирования (возможно, с определенной степенью деградации) при разрыве связей или потере управляющих воздействий от высших уровней иерархии управляющей структуры. Сохранение автономного функционирования в рамках более простого поведения системы обеспечивает максимальную живучесть ИСАУ.

Степень интеллектуальности системы управления зависит от функциональной насыщенности уровней IPDI. В связи с этим можно уточнить понятийный аппарат. Под интеллектуальной в большем системой будет пониматься система, соответствующая всем перечисленным принципам. Под интеллектуальной в малом системой будет пониматься система, не реализующая указанных принципов, но использующая знания, как средство преодоления неопределенности.

Интеллектуальные в большем и малом системы устанавливают верхнюю и нижнюю границы интеллектуальности управляющих систем. Степень интеллектуальности систем, находящихся внутри этого диапазона, можно определить по наличию или отсутствию тех или иных уровней IPDI [3].

Повышение интеллектуальности системы может быть решением известной проблемы актуальности базы знаний – к моменту начала функционирования системы, база знаний, составленная экспертами, уже является устаревшей. Это происходит, в частности, из-за самой структуры таких правил: «Что было бы, если бы система попала такую-то ситуацию». Единственным способом поддержания базы знаний в актуальном состоянии является непрерывная адаптация системы к изменяющимся условиям окружающей среды.

САУ можно классифицировать так же и по способу взаимодействия с окружающей средой: автономные (нет взаимодействия), формализованные (взаимодействие в рамках идеализированной модели) и информационные (взаимодействие на уровне реальных воздействий). Соответственно, с повышением качества взаимодействия растут так же и требования к интеллектуальности системы.

2. Нечеткая ИСАУ

Любая ИСАУ изначально должна иметь некоторый базовый набор знаний о предметной области функционирования, заданный экспертами при её создании. Учитывая тот факт, что экспертами являются люди, а людям свойственно формулировать свои знания о мире в виде вербальных и нечетких утверждений, одним из самых перспективных подходов для представления знаний является использование лингвистических переменных и аппарата нечетких множеств.[3][9] Такой подход к вычислениям сам автор нечеткой логики Заде относит к классу «мягких вычислений» : «Руководящим принципом мягких вычислений является: терпимость к неточности, неопределенности и частичной истинности для достижения удобства манипулирования, робастности, низкой стоимости решения и лучшего согласия с реальностью»[9]. Аргументами в пользу использования аппарата нечетких множеств так же являются: отсутствие аналитической связи входных и выходных параметров, неполнота знания о правилах поведения системы, неясная связь между параметрами, качественное описание поведения управляемой системы[10][11].

Применение нечеткой логики подразумевает использование нечетких характеристик, описывающих систему, вместо четких, но не всегда однозначных параметров. При этом некоторая характеристика объекта описывается лингвистической переменной, принимающей значение из множества нечетких термов, каждый из которых задает состояние некоторой характеристики объекта. Вывод решения состоит из четырех основных стадий: фазификации(приведения к нечеткости), логического вывода решения из множества нечетких правил вывода, композиции решения и дефазификации(приведения обратно к четкости). При управлении сложными процессами,

не имеющими точного количественного математического описания, нечеткие системы по сравнению с традиционными имеют лучшую помехозащищенность, быстродействие и точность за счет более адекватного описания реальной среды, в которой они функционируют [3].

Данный подход в контексте теории управления называется нечетким управлением. Он применяется при синтезе нечетких регуляторов, гибридных регуляторов, нечетких поисковых систем автоматической оптимизации, нечетких устройств оценивания и фильтрации. Среди причин распространения нечеткого управления, можно выделить[12]:

- особые качества ИСАУ с нечеткой логикой (например, малая чувствительность к изменению параметров объекта управления);
- более простой синтез ИСАУ (в сравнении с традиционными методами)
- «патентная чистота» проектируемых ИСАУ в силу относительной молодости науки.
- популярность нечеткого подхода

3. Типы нечетких ИСАУ.

Основным критерием классификации нечетких САУ является расположение и задачи нечеткого узла в системе управления [12]. Учитывая введенное понятие интеллектуальности и базовую схему САУ, можно определить три типа нечетких ИСАУ[3][12][13].

1 ИСАУ с нечетким контроллером (прямой контроль). Обычная замкнутая система управления с обратной связью, использующая в качестве регулятора нечеткий контроллер. Поскольку нечеткий контроллер осуществляет вывод решения посредством заранее заданной базы знаний и не имеет средств для её модификации, это система с наименьшим уровнем интеллектуальности (интеллектуальная в малом САУ).

2 Гибридные нечеткие ИСАУ. ИСАУ, использующая в качестве регулятора гибридный нечеткий контроллер. Гибридный контроллер представляет из себя двухуровневое устройство, нижний уровень которого вырабатывает управляющее воздействие методами классической ТАУ (например, обычным ПИД-регулятором), а верхний – производит нечеткую адаптацию параметров регулятора нижнего уровня. Таким образом достигается некоторый уровень адаптивности ИСАУ. К этому классу относятся так же системы «с нечеткими комплексными моделями», в которых нижний уровень реализуется ансамблем традиционных регуляторов, а верхний осуществляет их переключение [12]. Еще одним вариантом такой системы является парное использование

нечеткого и традиционного регуляторов, при этом нечеткий регулятор играет роль компенсатора [13].

3 Адаптивные нечеткие ИСАУ. В качестве регуляторов используются адаптивные контроллеры. Это тоже двухуровневые устройства, в которых управляющие воздействия нижнего уровня формируются посредством нечеткого вывода. А на верхнем уровне происходит нечеткая коррекция базы правил нижнего уровня. С помощью корректировки базы правил такая ИСАУ может адаптироваться к изменяющимся условиям окружающей среды и параметрам управляемого объекта.

Вне зависимости от класса нечеткой ИСАУ, основным вопросом при её проектировании является формирование базы знаний.

4. Синтез нечеткой ИСАУ

Основные подходы к формированию базы нечетких правил вывода

Ниже перечислены основные методы формирования базы знаний [12][13].

1 Генерация на основе экспертной оценки [4].

2 Автоматическая генерация на основе наблюдения за действиями оператора [14].

3 Использование зависимостей нелинейных операторов, реализуемых нечеткими системами, от конкретных параметров нечеткого вывода (базы правил, функций принадлежности и проч.) [14].

4 Применение вероятностных методов и методов обратной динамики (построение нечеткой системы в соответствии с обратным оператором объекта)[14].

5 Построение нечеткой модели управления по предварительно полученной нечеткой модели объекта управления [15].

6 Аппроксимация предварительно построенных операторов нелинейного оптимального закона управления нечеткой системой [15].

7 Нечеткие аналоги методов классической ТАУ (передаточной функции, метода Ляпунова и др.)[16].

8 Машинное обучение и экспертные системы [14].

Наиболее часто используемыми являются методы (1-2), как наиболее простые в организации и позволяющие описать систему любой сложности. Основной их недостаток – требование наличия экспертов и субъективность получаемых результатов. Формальные методы, нацеленные на автоматизацию и объективизацию процесса по большей части все еще находятся в стадии разработки и обладают своими недостатками и ограничениями. Например, (3) предполагает наличие нелинейного оператора САУ, (5) – наличие

обратного оператора, а (7) – возможность построения оптимального управления (что возможно далеко не всегда). Метод (6) накладывает ограничение на отсутствие блока дефаззификации – предполагается, что в системе действуют нечеткие сигналы. К тому же указанные методы действуют в рамках ТАУ и дают меньшую гибкость при описании сложно формализуемых алгоритмов поведения. Поэтому на практике, при разработке сложных интеллектуальных систем управления, чаще всего используются (1) и (2) метод в сочетании с (8). Данный подход позволяет сначала сформулировать методом экспертной оценки предполагаемую логику поведения и затем, с помощью методов машинного обучения, выполнить корректировку базы знаний.

Основные подходы к формированию функций принадлежности

Помимо базы правил необходимо сформировать функции принадлежности термов. К этой задаче существует два основных подхода, которые изложены ниже.

1 Прямые методы (например, метод относительных частот, параметрический, интервальный) целесообразно использовать для измеримых свойств и признаков таких как скорость, время, температура, давление и т.п. При использовании прямых методов не требуется точного задания функции. Достаточно зафиксировать вид функции принадлежности и характерные точки, по которым дискретное представление функции принадлежности аппроксимируется непрерывным аналогом – наиболее подходящей типовой функцией принадлежности, используемой в нечетком выводе [4].

2 Косвенные методы (например, метод парных сравнений) используются в тех случаях, когда отсутствуют измеримые свойства объектов в рассматриваемой предметной области [4].

В силу специфики задач при построении нечетких систем автоматического управления, как правило, применяются прямые методы. В свою очередь, в зависимости от числа привлеченных к опросу экспертов как прямые, так и косвенные методы делятся на одиночные и групповые. Наиболее грубую оценку характеристических точек функции принадлежности можно получить путем опроса одного эксперта, который просто задает для каждого значения из области определения соответствующее значение функции принадлежности.

Синтез нечеткого контроллера

В целом к конструированию нечеткого контроллера существует большое число совершенно разных подходов, выбор которых во многом продиктован поставленными целями. Например, в [13] описывается подход, при котором сначала конструируется ПИД-регулятор, затем эквивалентный линейный нечеткий контроллер, который постепенно

приводится к нелинейному. В [17] выполняется построение нечеткого контроллера на основе марковских цепей, а в [18] выполняется построение из блоков, представляющих из себя более простые контроллеры. Интересным и хорошо подходящим для целого класса задач (подразумевающих выполнение последовательности нечетких инструкций), является подход, связанный с использованием понятия нечеткого алгоритма. Нечеткий алгоритм представляет из себя набор нечетких инструкций. Впервые это понятие вводится в работе Заде [19] на примере кулинарного рецепта варки риса. Хорошая формализация нечеткого алгоритма и введение понятия нечеткой грамматики для порождения нечеткого языка представлены в работах [4][20].

Использование понятия нечеткого алгоритма может значительно облегчить задачу разработки нечеткого контроллера, если есть «рецепт» или описание необходимых действий на естественном языке. Помимо самого рецепта необходимо техническое устройство – нечеткая вычислительная машина и нечеткий язык. Основные преимущества такого подхода:

- большая гибкость при описании алгоритма (ограниченная возможностями нечеткого языка);
- естественность задания инструкций управляемому устройству;
- возможность итерационного повышения интеллектуальности САУ без изменения физической архитектуры (путем добавления соответствующих инструкций).

Ниже представлен пример практической реализации указанного подхода к проектированию нечеткой САУ, интеллектуальной в малом и указаны пути повышения её интеллектуальности.

5. Особенности практической реализации нечеткого контроллера

Как указано выше, для задания нечеткого алгоритма работы САУ необходимы три составляющих: описание алгоритма на естественном языке, нечеткая вычислительная машина и нечеткий язык.

Нечеткая вычислительная машина и нечеткий язык

При решении задач управления применяют устройства с жесткой и программируемой логикой. В контексте поставленной задачи необходимо использовать устройства именно с программируемой логикой. Распространенным подходом является использование микроконтроллеров – однокристальных ЭВМ, разработанных для задач управления электронными устройствами.

Для реализации нечеткого алгоритма на микроконтроллерах существует два пути. Первый – аппаратная реализация нечеткого микроконтроллера, второй – программно - аппаратная. Ниже они рассмотрены более детально.

- Аппаратная реализация.

Специально разработанное устройство под конкретные цели.

- Устройство со спроектированной под конкретный алгоритм структурой.
- Промышленный образец
  - Микроконтроллеры специального назначения (например, Fujitsu F2RU-8, VY86C570 (Togai InfraLogic), SAE 81C99 (Siemens))
  - Микроконтроллеры общего назначения с аппаратной поддержкой нечеткой логики (такие как Motorola 68HC12, ST52 Dualogic (STMicroelectronics)).

- Программная реализация.

В силу того, что аппаратная реализация, при наличии таких преимуществ как скорость и удобство работы, не всегда может быть доступна (из-за малой распространенности и высокой цены), часто применятся программная реализация на основе микроконтроллеров общего назначения (AVR, PIC, MSP, ARM).

- Использование специальной «нечеткой» библиотеки, предоставляющей необходимое «нечеткий» API (нечеткий язык)(Free Fuzzy Logic Library, fuzzyCLIPS [5]);
- Использование специальных «конструкторов» нечетких систем и последующая трансляция в код микропрограммы микроконтроллера (Matlab, Matematica, CubiCalc, RuleMaker, FuziCalc, fuzzyTECH [5])
- Использование специализированных программных средств, позволяющих сразу генерировать код с нечеткой системой под любой микроконтроллер.

К общим достоинствам программной реализации можно отнести простоту использования и высокую доступность (из-за высокой распространенности микроконтроллеров общего назначения), гибкость создаваемых решений. Недостатки программной реализации – скорость работы нечетких операций (в отличие от аппаратной реализации) и погрешности в вычислениях (из-за вынужденной дискретизации).

Необходимо отметить, что множество нечетких операций, упоминаемых в перечислении выше, не задаёт нечеткого языка в строгом понимании [4], скорее являясь его некоторой аппроксимацией. Но в контексте данной работы под нечетким языком будет пониматься именно множество нечетких операций микроконтроллера. Можно показать, что нечеткий язык, порождаемый нечеткой грамматикой, строго заданный в [4] и множество нечетких операций вывода позволяют задать эквивалентные последовательности нечетких инструкций, т.е. нечеткие алгоритмы.

В силу практически нулевой доступности аппаратно реализованных нечетких микроконтроллеров был выбран вариант программной реализации. Готовых решений в виде библиотеки нечетких операций, отвечающих заявленным требованиям, на рынке не найдено. Другие варианты представлены широким набором средств в различных ценовых категориях и с различным уровнем функциональности. Однако они обладают рядом серьезных недостатков – высокой ценой пакета, сложностью трансляции нечеткого модуля под конкретную модель микроконтроллера, узостью применения, плохой поддержкой разработчиков.

В силу описанных выше причин был выбран вариант реализации нечеткого микроконтроллера путем разработки специальной нечеткой библиотеки, предоставляющей необходимые для нечеткого аппарата операции.

Был произведен обзор имеющихся на рынке микроконтроллеров общего назначения для выбора целевой микросхемы разрабатываемой «нечеткой» библиотеки. Целевой платформой был выбран 8-битный микроконтроллер AVR. Несмотря на постоянно увеличивающиеся мощности и разрядности современных процессоров, в микроэлектронике все еще очень интенсивно используются 8-битные микроконтроллеры. Это обусловлено их низкой ценой и невысокой вычислительной сложностью возлагаемых на них задач. Решение фирмы Atmel пользуется большой популярностью и является одним из самых распространенных типов микроконтроллеров (наравне с PIC и ARM). Большим преимуществом AVR является широкий набор установленной периферии, единое ядро в рамках всей линейки, что обеспечивает переносимость кода и популярность среди разработчиков, что обеспечивает большое количество разнообразной документации.

Алгоритм на естественном языке

Для практической реализации была выбрана задача поиска пути в неизвестном окружении некоторым интеллектуальным агентом (роботом). Постановка данной задачи, а также описание возможных путей решения произведено в [21]. Ниже приведено краткое описание нечеткого алгоритма нечеткой САУ, интеллектуальной в малом.

- Если Позиция робота такая-то и Позиция Цели такая-то, то направление такое-то.
- Если в окрестности направления есть препятствия, то корректировка такая-то.
- Если скорректированное направление такое-то, то скорости моторов такие-то.

6. Архитектура нечеткого микроконтроллера

Общая структура нечеткой САУ, интеллектуальной в малом, представлена на рисунке 1. Здесь X – входные параметры системы, Y – выходные, R – обратная связь.

Выходные параметры (Y) проходят некоторую обработку перед тем как попасть на вход микроконтроллера.

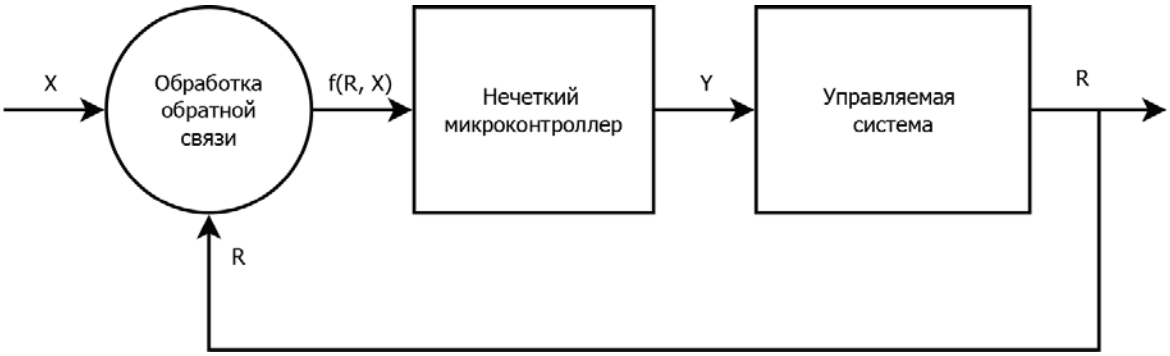


Рис. 1. Общая структура нечеткой САУ, интеллектуальной в малом

При проектировании нечетких микроконтроллеров удобно использовать понятие блока [18]. Тогда можно выделить следующие блоки:

- Блок предварительной обработки
- Блок фазификации
- Блок вывода решения
- Блок дефазификации
- Блок окончательной обработки

Взаимодействие указанных блоков в аппаратно реализованном нечетком микроконтроллере продемонстрировано на рис.2. Такую связь блоков можно назвать последовательной. Здесь X – данные поступающие извне (например, с подключенных датчиков или пульта управления), f(X) – обработанные данные (с учетом обратной связи), f(X)* - нечеткие значения обработанных входных данных, Y* - нечеткие значения выходных данных, Y – четкие значения выходных данных, f(Y) – обработанные выходные данные (управляющие сигналы).

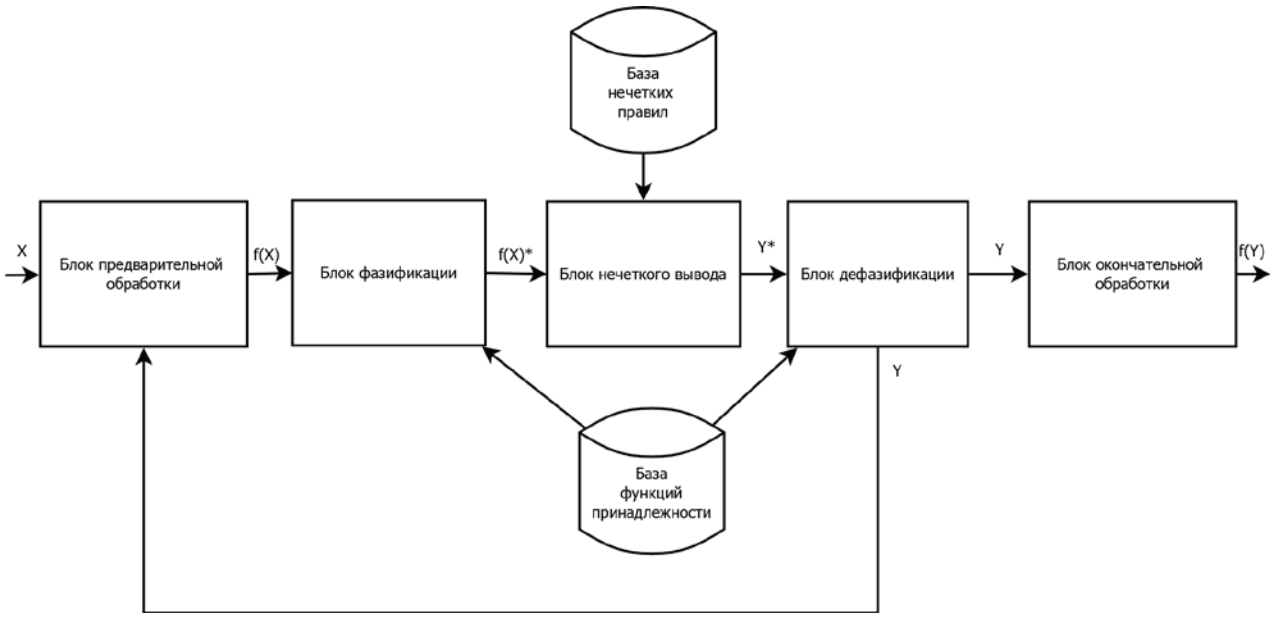


Рис. 2. Последовательная связь блоков в нечетком МК

При аппаратной реализации перечисленные блоки реализуются в виде независимых микроэлектронных устройств, связанных друг с другом. Однако такая схема микроконтроллера не обеспечивает необходимой гибкости и позволяет использовать лишь стандартную последовательность нечеткого вывода. Такие решения могут использоваться в специальных задачах, в которых не планируется последующая модификация.

При программной реализации такие блоки реализуются в виде независимых программных модулей, то есть являются виртуальными. В контексте того, что указанные блоки-модули могут использоваться алгоритмом функционирования микроконтроллера, записанного в память программ в произвольном порядке, схему нечеткого микроконтроллера можно представить по-другому (рис.3). Такую связь блоков можно назвать централизованной.

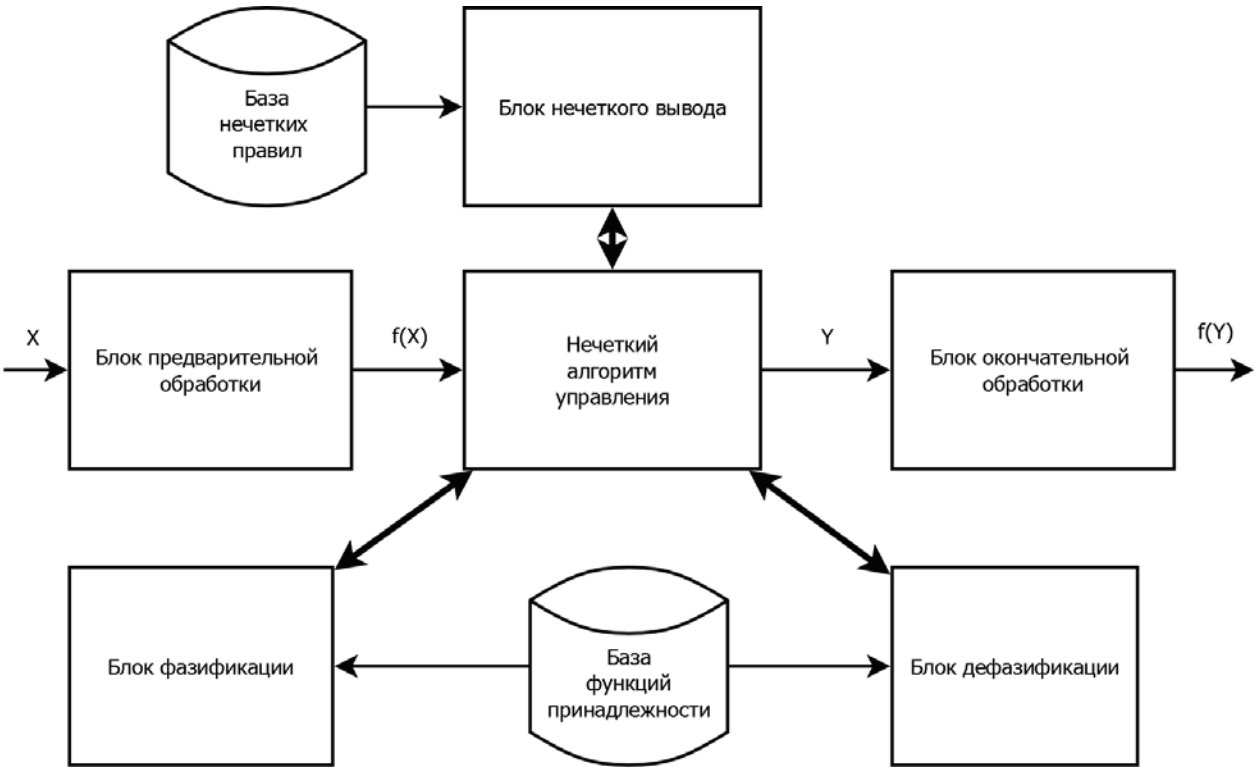


Рис. 3. Централизованная связь блоков нечеткого МК

На схеме, изображенной на рис.3 видно, что алгоритм функционирования управляемой системы связан с блоками, реализующими основные этапы нечеткого вывода, и может с их помощью генерировать необходимое решение. Каждый описанный блок задает некоторое подмножество нечетких операций, описывающих алгоритм.

7. Особенности практической программной реализации нечеткого микроконтроллера

Аппаратные особенности выбранной архитектуры (8-разрядное AVR-ядро) накладывают определенные ограничения на разрабатываемое решение. Гарвардская архитектура микроконтроллера (раздельная память команд (Flash) и данных (RAM и EEPROM)) позволяет выполнять код программы непосредственно с Flash-памяти, не загружая её в ОЗУ. Поэтому весь объем ОЗУ можно использовать для данных. Максимальное число циклов записи Flash-памяти равно 10000, а EEPROM-памяти – 100000. Поэтому такие данные, как база знаний, целесообразно разместить в EEPROM-памяти, а при включении микроконтроллера – загружать её в оперативную память. Такое размещение значительно повышает гибкость решения – данные не зависят от кода. При дальнейшей модификации решения можно реализовать другие способы загрузки базы знаний в ОЗУ.

При выполнении программы крайне желательно, чтобы в оперативной памяти располагалась вся база знаний, т.к. в противном случае её необходимо подгружать с внешнего носителя. Если это EEPROM, то данные операции будут занимать много времени (EEPROM- медленное внешнее устройство с неопределенным временем доступа).

В имеющейся модели МК (Atmega 8515) размеры обоих видов памяти равны (512 байт), что ставит верхнюю границу размера базы знаний в 512 байт. Данное ограничение подразумевает доскональный контроль памяти и использование компактного формата хранения базы знаний. Т.к. вся оперативная память будет занята базой знаний, все промежуточные данные необходимо хранить в регистровой памяти (в ядре AVR – 32 РОН).

Система команд содержит 130 разнообразных RISC – команд: арифметические и логические команды, команды перехода, команды работы с данными и битовые команды. Важно, что в большинстве микроконтроллеров AVR есть аппаратная реализация операции умножения (время выполнения - 2 такта), однако отсутствует аппаратная реализация операции деления. На микроконтроллерах AVR она может быть реализована только программно, поэтому время её выполнения значительно больше 1-2 тактов, что является стандартном для команд AVR-ассемблера. Это означает, что при разработке функций, реализующих этапы нечеткого вывода, необходимо минимизировать количество делений.

Микроконтроллер AVR – 8 битный микроконтроллер, а значит все операции в ядре производятся с байтами. Для достижения максимальной производительности необходимо это учитывать: по-возможности, ограничить диапазоны данных размером 1-го байта, а для дробных чисел произвести дискретизацию (аналогично, если это возможно, в 1 байт). Пример – степень принадлежности терму в нечетком выводе задается значением от 0 до 1. В представлении внутри микроконтроллера это будет диапазон 0-255. В данной работе принят именно такой шаг, так как иначе (при работе с числами большего разряда) нельзя использовать машинные команды в чистом виде – должны быть организованы специальные процедуры обработки данных большей разрядности, что значительно снизит скорость выполнения такой программы.

Как упоминалось ранее, разрабатываемое решение должно представлять из себя библиотеку, реализующую функции нечеткого языка под микроконтроллер. Размер памяти программ у рассматриваемой модели – 8 кБайт. Размер 1 команды – 2 байта, значит максимальный размер микропрограммы под данный микроконтроллер – 4 тыс. инструкций. Т.к. разрабатывается библиотека, она не должна занимать всю память программ, её функция – предоставление необходимых сервисов для основного алгоритма,

загружаемого в микроконтроллер. Это так же следует учитывать при разработке данной программы.

Учитывая указанные выше требования по контролю памяти и требования быстродействия, было принято решение вести разработку библиотеки на языке ассемблера.

База знаний нечеткого вывода может занимать практически весь объем оперативной памяти, поэтому использование стека необходимо свести на минимум. Для этой цели из 32 РОН были выделены рабочие регистры (r2-r5,r21,r22), сохранение состояния которых не гарантировалось подпрограммами. Они нужны для промежуточных вычислений между вызовами подпрограмм. Т.к. их состояние не гарантируется, их не нужно сохранять в стеке. Это экономит память и время.

Для передачи параметров в подпрограмму были так же зарезервированы несколько регистров (r19,r23-r25). Были написаны соответствующие макросы, имитирующие вызовы функций на языках высокого уровня, которые копировали значения операндов в специальные регистры операндов и осуществляли вызов подпрограммы. Для возвращения результата использовалась зарезервированная пара регистров r6 и r20. Размещение регистров не по порядку связано с тем, что регистры в AVR не ортогональны, например, r16-r31 более удобны для работы (позволяют загружать числовые значения и данные из ОЗУ напрямую).

Для реализации деления было написано несколько подпрограмм, осуществляющих деление по собственному алгоритму. Для ускорения вычислений было реализовано 3 вариации под конкретные типы задач: деление 0xFF на однобайтовый операнд, деление однобайтового операнда на однобайтовый операнд и деление двухбайтового операнда на двухбайтовый операнд. Деление реализовано в виде циклического определения примерного множителя (в виде степени двойки) путем последовательных сдвигов делимого и делителя, умножения множителя на делитель и вычитания полученного результата из делимого. Данный процесс происходит до тех пор, пока делимое не станет меньше делителя.

8 . Этапы нечеткого вывода

Чтобы задать необходимые нечеткие операции, необходимо реализовать соответствующие блоки. Нечеткий вывод включает в себя 4 основных этапа. Однако второй и третий этап можно объединить в один:

- фаззификация (определение значений лингвистических переменных)
- логический вывод и композиция

- дефаззификация (получение четких значений выходных характеристик)

Итак, программная реализация нечеткого микроконтроллера должна содержать 3 модуля, реализующие данные этапы и задающие требуемые операции. Взаимодействие этих модулей с основной программой продемонстрировано на DFD-диаграмме на рисунке 4. Ниже каждый этап вывод рассмотрен более подробно.

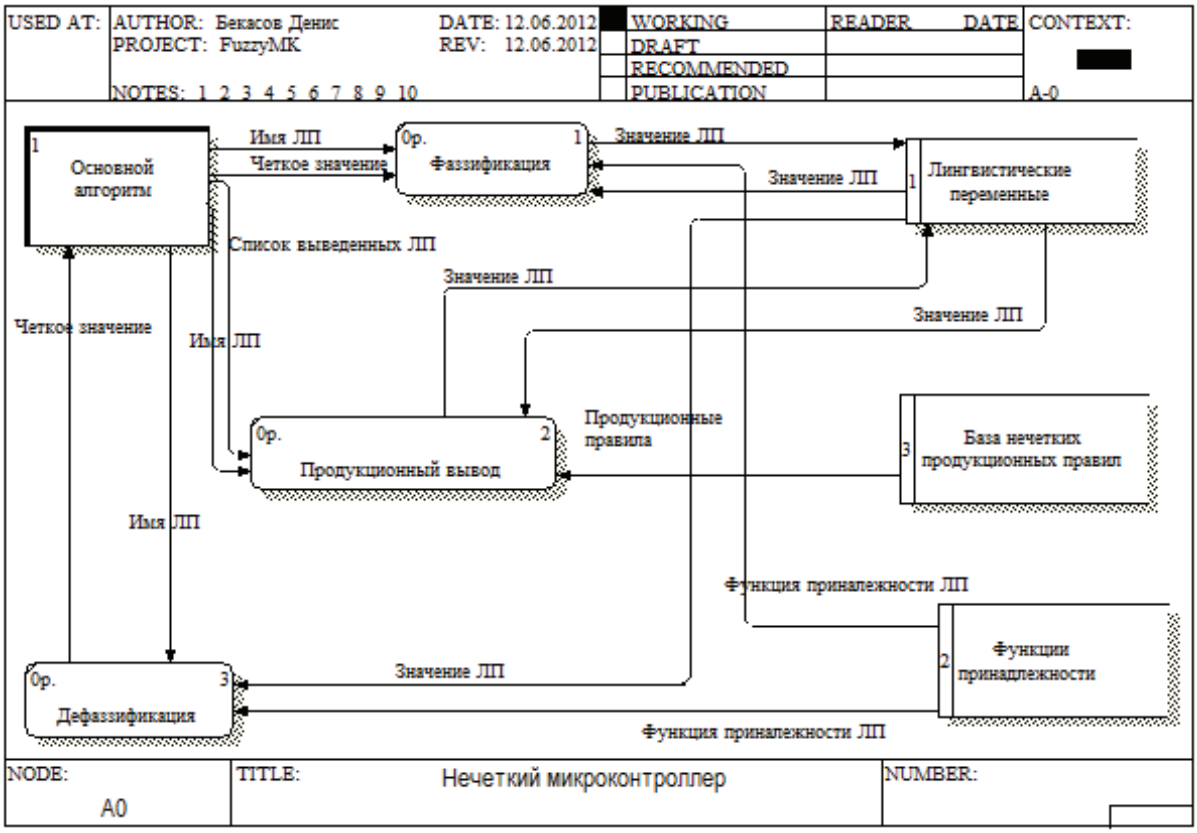


Рис. 4. DFD-диаграмма взаимодействия модулей нечеткого вывода и основного алгоритма

Этап фаззификации

На данном этапе происходит перевод из четкого значения некоторого параметра в нечеткое значение некоторой лингвистической переменной. Для того чтобы осуществить такой переход, необходима функция принадлежности, задающая конкретную лингвистическую переменную. Функции принадлежности могут быть заданы различными способами – монотонными, треугольными, трапецеидальными и сигмоидальными функциями. Учитывая рассмотренные выше ограничения реализации, был сделан выбор в пользу трапецеидальных функций.

Любая лингвистическая переменная определяется на некотором нечетком множестве. Степень принадлежности к каждому терму задаются функцией принадлежности. Таким образом, функция принадлежности – это совокупность трапеций, каждая из которых задает принадлежность конкретному терму. Для экономии памяти,

трапецию целесообразно задавать четырьмя абсциссами, т.к. ординаты у соответствующих точек всегда одни и те же – 0,1,1,0 (или, применительно к принятым правилам дискретизации значений, 0,255,255,0).

Тогда процесс фаззификации заключается в просмотре всех трапеций с целью определения факта вхождения фаззифицируемого значения (x) в соответствующий интервал, а при нахождении такого интервала – определение степени принадлежности (μ). Для любого терма возможны 3 случая:

1 x не попало в область определения трапеции. Тогда μ = 0

2 x попало в средний интервал. Тогда μ = 1

3 x попало в левый или правый интервал. Тогда надо найти ординату пересечения прямой, заданной точками (x_1,0) -(x_2,255) или (x_3,255) -(x_4,0) и прямой x = x. Пусть, для определенности, x попало в левый интервал. Тогда $\mu = \frac{255}{(x_2 - x_1)} * (x - x_1)$.

Таким образом, при фаззификации может возникнуть до N делений, где N – количество термов. Остальные операции простые (занимают 1-2 такта) – последовательный проход по памяти и арифметические сравнения.

Логический вывод.

Цель логического вывода – вывести, с помощью заданной базы правил и известных значений лингвистических переменных, неизвестные значения лингвистических переменных. При описании знаний может быть использована продукционная модель или логика предикатов. В данной работе принята продукционная модель, как наиболее соответствующая естественному языку и идее нечеткого языка. Правила задаются в формате, описанном ниже.

«Если антецедент, то консеквент», где

<Консеквент> = <Выбор терма>

< Антецедент >=<Выбор терма> {“&” <Выбор терма>}

<Выбор терма> = <Лингвистическая переменная> “=” <Терм>

“&” – символ конъюнкции.

В данной работе предполагается, что цель вывода (целевая лингвистическая переменная) в принципе выводима из начальных предпосылок. Это не нарушает общности рассуждений, так как цель логического вывода в случае использования нечеткой логики заключается не в доказательстве или опровержении какого-либо утверждения, а в нахождении значения целевой лингвистической переменной. Это значит, что в базе правил имеется как минимум 1 правило, в состав консеквента которого входит искомая ЛП и все ЛП из его анцедента выводимы из начальных предпосылок.

Продукционный вывод осуществляется обратным методом (поиск в глубину) - на вход процедуре логического вывода подается имя искомой лингвистической переменной и список начальных предпосылок. Процедура просматривает правые части правил в поисках искомого консеквента. Если все ЛП из соответствующего антецендента входят в список начальных предпосылок, то:

- ко всем условиям антецедента применяется Т-норма[22];
- к результату и имеющемуся значению консеквента применяется S-норма[22].

В данной работе за Т-норму принята операция взятия минимума, а за S-норму – операция взятия максимума. Если какая-то ЛП из анцедента не входит в список начальных предпосылок, то она становится текущей целью вывода. Для этого в стеке запоминается текущее состояние вывода (позиция в базе правил, выводимая ЛП) и вывод начинается сначала, но с новой целью. Вывод заканчивается, когда вся база правил просмотрена, а текущая цель вывода равна начальной цели вывода.

Дефаззификация.

Основная цель дефаззификации – сведение значения ЛП обратно к четкости. Для этого применяются разные способы: по среднему центру, по сумме центров, метод центра тяжести, максимума функции принадлежности[6]. До этих пор вывод осуществлялся согласно несколько адаптированному методу вывода Мамдани. В методе Мамдани для дефаззификации традиционно используется метод центра тяжести (центроида):

$$\bar{y} = \frac{\int_Y \ y * max\ \mu_{\overline{B^k}}(y)}{\int_Y \ max\ \mu_{\overline{B^k}}(y)}$$

где B^k-функция принадлежности k –го терма.

Однако в данной работе применение данного метода нецелесообразно в силу его большой вычислительной сложности. Данный метод предполагает отсечение всех функций принадлежности соответствующими μ, объединение получившихся фигур и вычисление центра тяжести объединения. Вместо этого используется метод дефаззификации по среднему центру:

$$\bar{y} = \frac{\sum_{k=1}^{N} \mu_{\overline{B^k}}(\bar{y}^k)\bar{y}^k}{\sum_{k=1}^{N} \mu_{\overline{B^k}}(\bar{y}^k)} \quad (1)$$

При использовании данного метода:

1 Находится отсечение каждого терма ЛП. Для этого находится пересечения прямых (x_1,0) -(x_2,255) и (x_3,255) -(x_4,0) с прямой y = μ. Для левого интервала:

$$x = \frac{\mu}{\frac{255}{(x_2 - x_1)}}$$

Двойное деление в данной формуле используется для того, чтобы не получить значения больше одного байта и, таким образом, не получить переполнение в однобайтовой ячейке памяти.

2 Находится середина полученной верхней площадки трапеции для использования в формуле 2.1.

3 После того, как просуммированы все средние центры и соответствующие μ, применяется формула 1. Здесь происходит деление 2-х байтового числа на 2-х байтовое.

Таким образом, при деффазификации имеется 4N 1-байтовых делений и одно 2-х байтовое деление, где N - количество термов.

6. Реализация нечеткого алгоритма: модель агента и последовательность действий

После того, как была разработана нечеткая платформа (нечеткая вычислительная машина и нечеткий язык), осталось реализовать нечеткий алгоритм на данной платформе. Но для начала необходимо задать модель управляемой системы.

Алгоритм достижения цели оперирует понятием агента. В контексте данной работы агент представляется трехколесным двухприводным мобильным роботом, оснащенным ультразвуковым дальномером, установленном на сервоприводе. Данная структура выбрана по причине своей относительной простоты как в смысле логики связей отдельных блоков, так и в смысле практической реализации. Логическая структура такого робота представлена на рисунке 5.

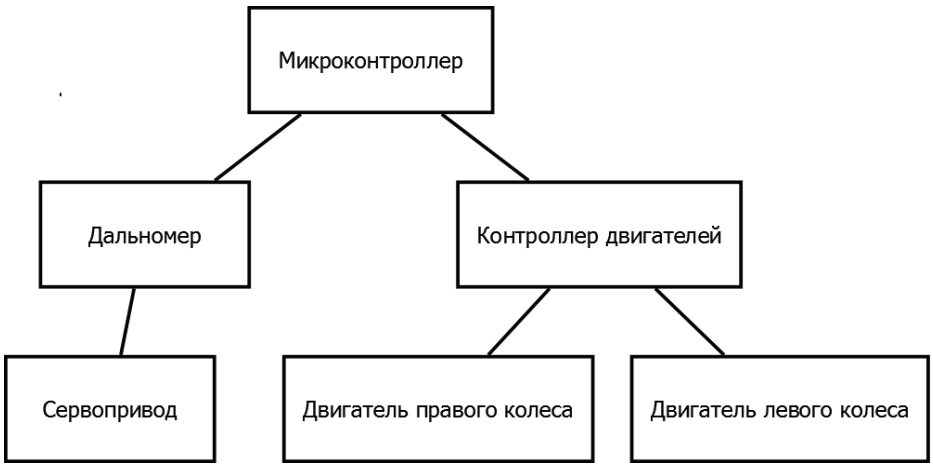


Рис. 5. Логическая структура мобильного робота.

Для получения информации о препятствиях МК посылает дальномеру направление. Дальномер формирует сигнал на сервопривод, посылает ультразвуковой сигнал в пространство и возвращает микроконтроллеру ответ. Для перемещения в пространстве, микроконтроллер посылает контроллеру моторов значения скоростей левого и правого колеса соответственно. Контроллер моторов в свою очередь формирует соответствующие величины напряжений на выходах, к которым подключены двигатели.

Поворот такой системы осуществляется за счет того, что скорости левого и правого двигателей могут быть различными. Это сообщает центробежное ускорение системе и робот осуществляет поворот. Если скорости моторов одинаковы по модулю, но разные по направлению, то робот вращается на месте. В другом случае поворот осуществляется в движении. Схематический вид робота сверху и результирующая траектория поворота изображены на рисунке 6.

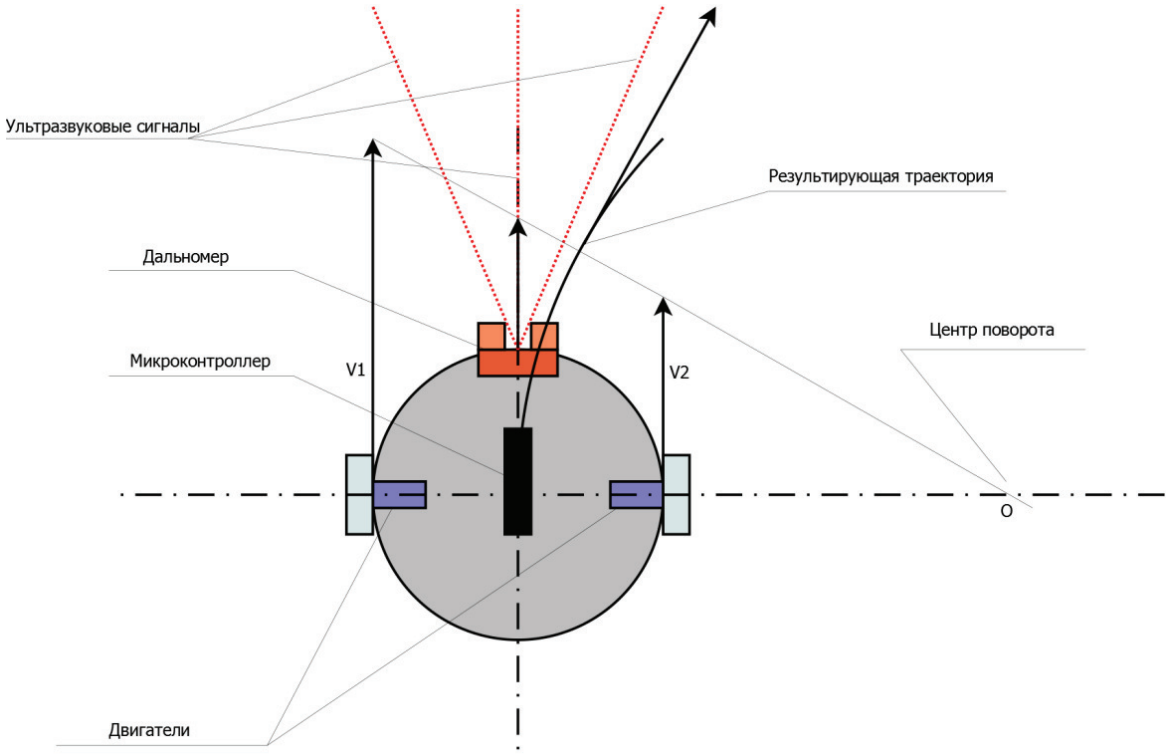


Рис. 6. Схематический вид робота сверху

Исходя из указанных выше рассуждений был предложен нечеткий алгоритм поиска пути в первом приближении. Идея алгоритма состоит в следующем: после начальной инициализации агент запоминает координаты цели и свое начальное состояние. Он переводит их в понятие лингвистических переменных и с помощью базы правил определяет направление на цель. После этого по полученному направлению (+/- 45 градусов) он проверяет наличие препятствий, переводит результат в значения

лингвистических переменных и, опять используя базу знаний, получает значения корректировок. После этого формирует угловую характеристику скорости двигателей и фаззифицирует её. Функция принадлежности данной лингвистической переменной подобрана таким образом, что получившиеся в результате фаззификации степени принадлежности двум термам и равны скоростям двух двигателей. Полученные нечеткие значения посылаются на контроллер двигателей. Данная последовательность действий повторяется до тех пор, пока цель не будет достигнута. На рисунке 7 представлена схема данного алгоритма.

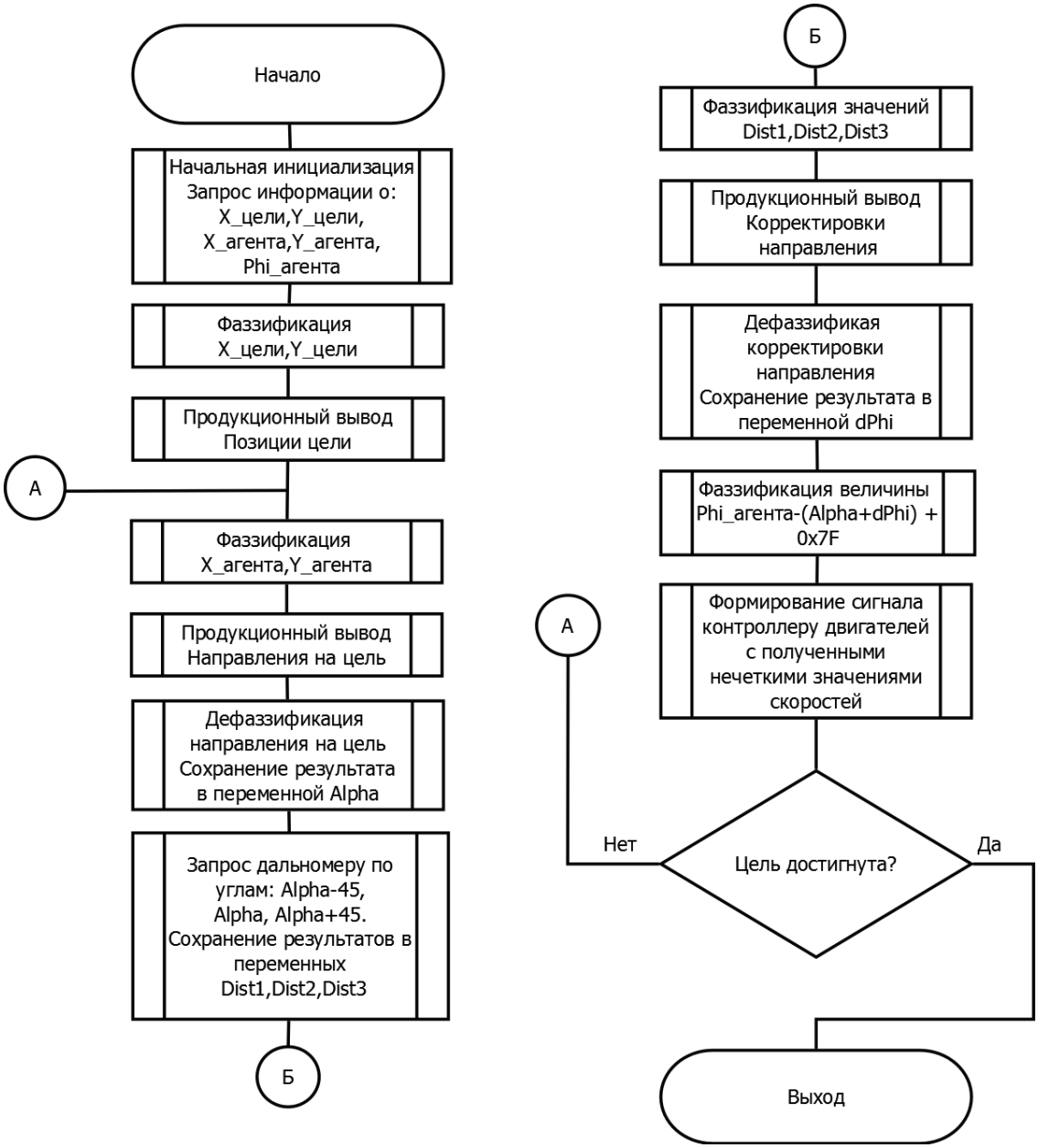


Рис. 7. Схема нечеткого алгоритма поиска пути

9. Реализация нечеткого алгоритма: база знаний

В теории нечетких множеств любая величина задается некоторым множеством её возможных значений, характеризующихся той или иной степенью принадлежности (с помощью так называемой функции принадлежности) объекту, описываемому этим нечетким множеством [21]. В описанном выше алгоритме введены различные лингвистические переменные. Ниже следует их описание с функциями принадлежности.

1 X и Y - задают координаты в пространстве задачи.

Пространство задачи ограничено по X и по Y интервалом значений [0,255]. Нечеткое множество задается на 3 термах – «лево», «прямо», «право». Функция принадлежности представлена на рисунке 8.

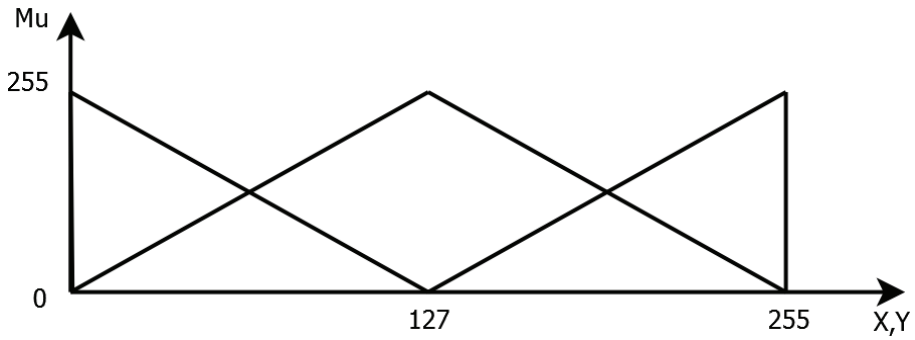


Рис. 8. Функция принадлежности лингвистических переменных X и Y

2 $P_{цели}$ и $P_{агента}$ – задают положения агента и цели в пространстве задачи.

Для экономии памяти в базе знаний, данные ЛП не имеют функций принадлежности, т.к. это промежуточные переменные, используемые в логическом выводе. Введены для более компактного задания правил.

3 Alpha – направление до цели.

Угол задаётся в интервале от 0 до 2π. Однако, учитывая требования дискретизации, в МК данный интервал отображается на интервал [0,255]. Данная лингвистическая переменная задается на 16-ти термах, каждый из которых представляет одно из возможных направлений движений робота. Функция принадлежности представлена на рисунке 9.

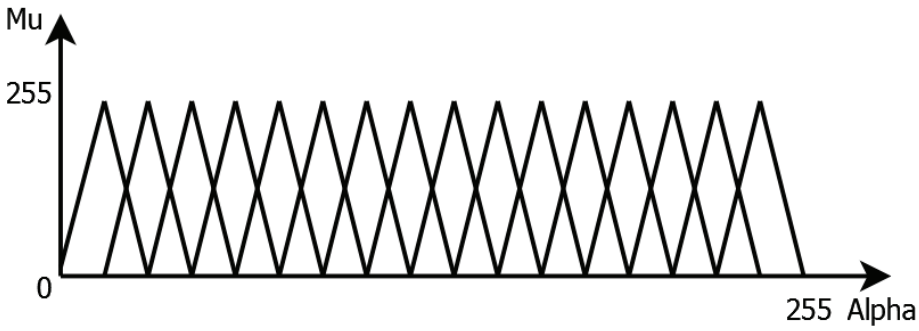


Рис. 9. Функция принадлежности лингвистической переменной Alpha

4 Dist1,Dist2,Dist3 – расстояния до препятствий по трем направлениям.

Задаются на трех термах: «близко», «средне», «далеко». От соотношения расстояний по трем направлениям зависит корректировка направления. Функция принадлежности представлена на рисунке 10.

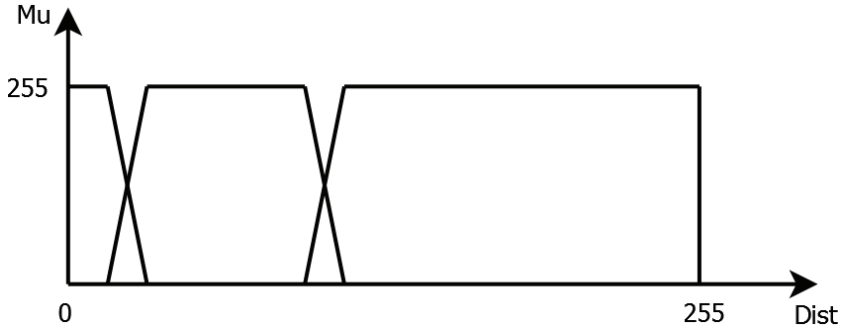


Рис. 10. Функция принадлежности лингвистических перемнных Dist1, Dist2, Dist3

5 dAlpha – корректировка направления.

Корректировка направления в зависимости от ЛП Dist1, Dist2, Dist3. Корректировка заключается в измени угла на +/- 90 градусов. Следовательно, область определения ФП для dAlpha: [-64,64]. Т.к. в данном решении рассматриваются только положительные значения, то с учетом сдвига область определения: [0, 128]. ЛП задается на 5 термах: резко влево, чуть влево, прямо, чуть вправо, резко вправо. Функция принадлежности представлена на рисунке 7.4.

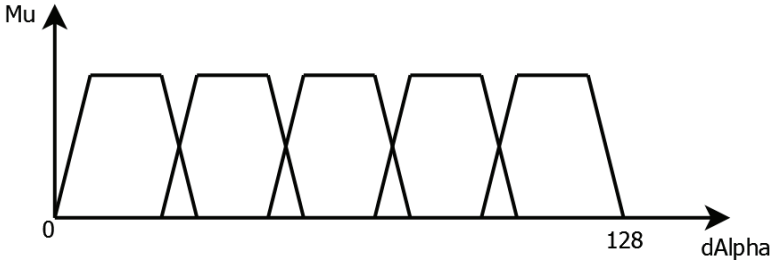


Рис. 11. Функция принадлежности лингвичтической переменной dAlpha

6 Moto – скорости для обоих моторов.

Эта переменная представляет из себя угловую характеристику Phi_агента – (Alpha+dPhi) +127, при фазификации которой получаются такие значения степеней принадлежности термам, которые могут напрямую использоваться как управляющие воздействия для скоростей 2-х двигателей. Переменная задается на двух термах: «левый двигатель» и «правый двигатель». Функция принадлежности представлена на рисунке 12.

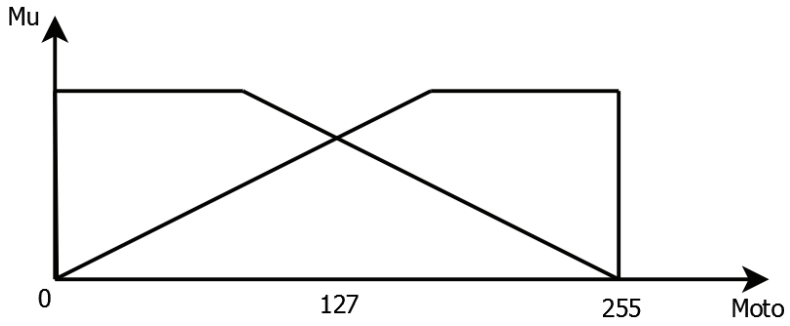


Рис. 12. Функция принадлежности лингвистической переменной Moto

База нечетких правил, используемая при выводе. Состоит из следующих смысловых блоков:

- Блок вывода позиции из значений координат.
  Содержит правила вида: «Если X = «Лево», Y = «Лево», то P = «Левый нижний квадрат»»
- Блок вывода направления из значений позиций цели и агента.
  Содержит правила вида: «Если P_агента = «Левый нижний квадрат», P_цели = «Правый верхний квадрат», то Alpha = «направление на 45 градусов»»
- Блок вывода корректировки направления из значений препятствий.

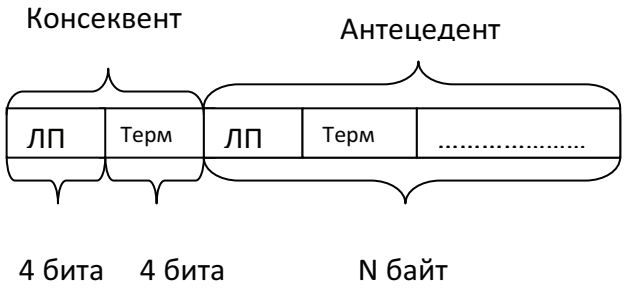
Содержит правила вида: «Если Dist1 = «Средне», Dist2 = «Близко», Dist3 = «Близко», то dAlpha = «Чуть влево»»

В микроконтроллерах AVR объем EEPROM колеблется от 512 байт до 4 кБайт. Т.к. EEPROM является основным носителем базы знаний, то необходим очень компактный способ представления. Кроме того, БЗ во время работы программы планируется хранить в ОЗУ (т.к. циклы обращения к EEPROM очень медленные), поэтому необходимо предусмотреть место для хранения значений ЛП. Формат базы знаний представлен в таблице 1.

Таблица 1

Формат базы знаний

Адрес	Название блока	Содержание
1	2	3
0x0 : 0x4	Заголовок	Указатель на блок ЛП (LV_ptr) Указатель на блок БЗ (KB_ptr)
0x5 : LV_ptr – 0x1	Блок ФП (функции принадлежности)	Функции принадлежности в формате последовательностей из 4 байт, каждый из которых – соответствующая абсцисса трапеции.
LV_ptr : KB_ptr – 0x1	Блок ЛП (лингвистические переменные)	Лингвистические переменные в формате:  Где N – кол-во термов данной ЛП (максимум – 16). Данный формат предоставляет возможность хранить 15 различных ЛП.

KB_ptr : 0xXXXX	Блок БЗ (база знаний)	Содержит терминальные символы и продукционные правила. Формат терминального символа: 0x0N, где N – число условий в данном блоке БЗ. 0x00 указывает на конец БЗ. Формат продукционного правила:  Антецедент задается конъюнкцией указанных условий. Такое правило читается как «Если ЛП1 = Терм1, то ЛП2 = Терм2»

База знаний, закодированная в данном формате, занимает 495 байт. Это значит, что у программ есть только 17 байт свободного места для использования стека. Программа написана таким образом, что максимальное погружение в стек осуществляется на 12 байт (в функции дефаззификации). Таким образом, остается еще 5 буферных байт защиты от повреждения данных.

10 Результаты

Для проверки работоспособности нечеткого микроконтроллера была разработана специальная моделирующая программа, моделирующая описанную выше структуру агента и неизвестное окружение. Связь с микроконтроллером устанавливалась по специально разработанному протоколу через интерфейс RS-232. Архитектура этого комплекса представлена на рисунке 13.

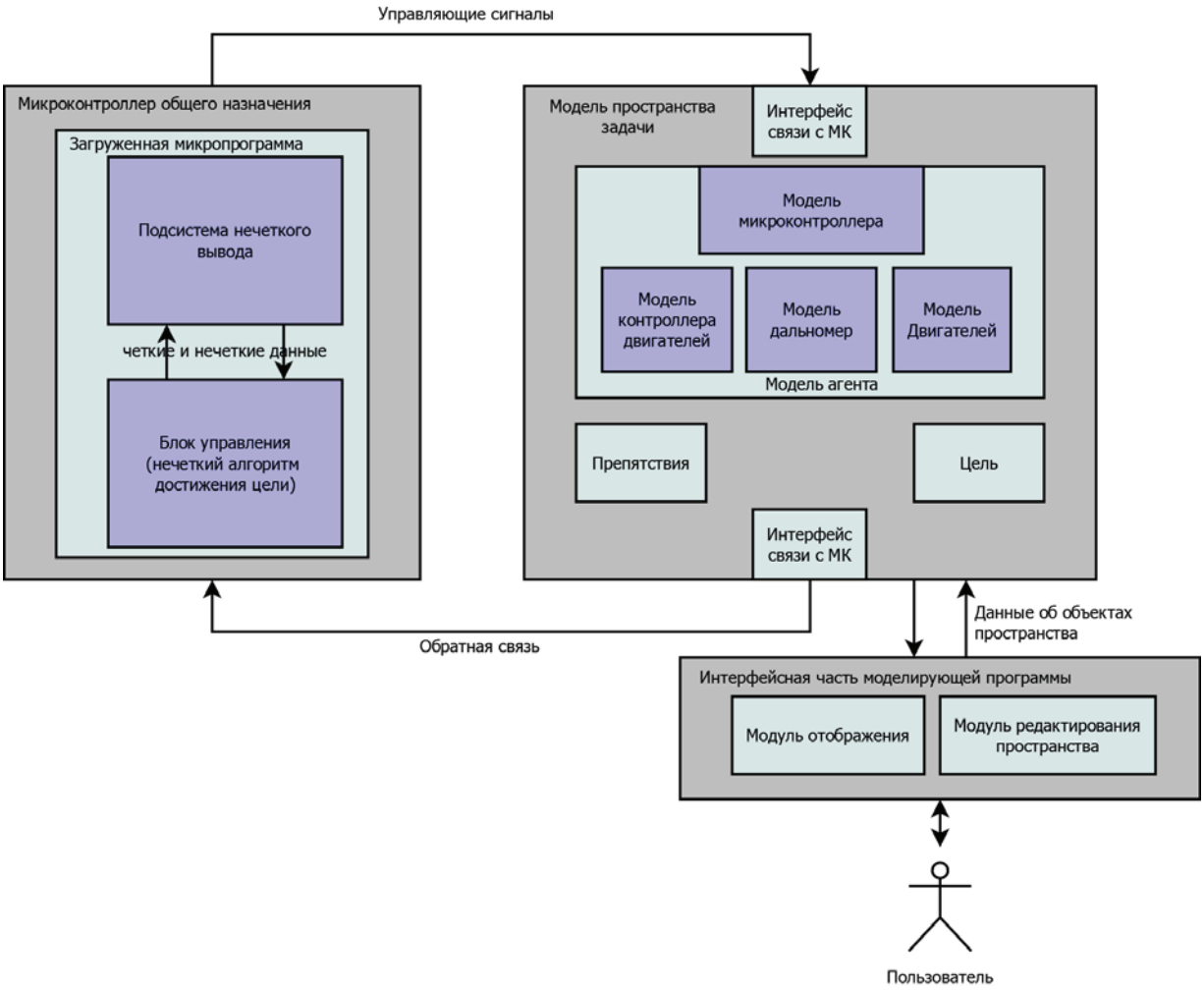


Рис. 13. Архитектура разработанного комплекса

На рисунке 14 представлены результаты работы предложенного алгоритма. Производилось тестирование системы на последовательно усложняющейся карте препятствий. Таким образом, можно отследить изменение траектории движения агента. На этом рисунке видно, что траектория достаточно плавно изменяется с почти прямой (1-я иллюстрация) до волнистой линии (6 – я иллюстрация). Как и ожидалось, траектория имеет характер плавной линии (гладкой функции).

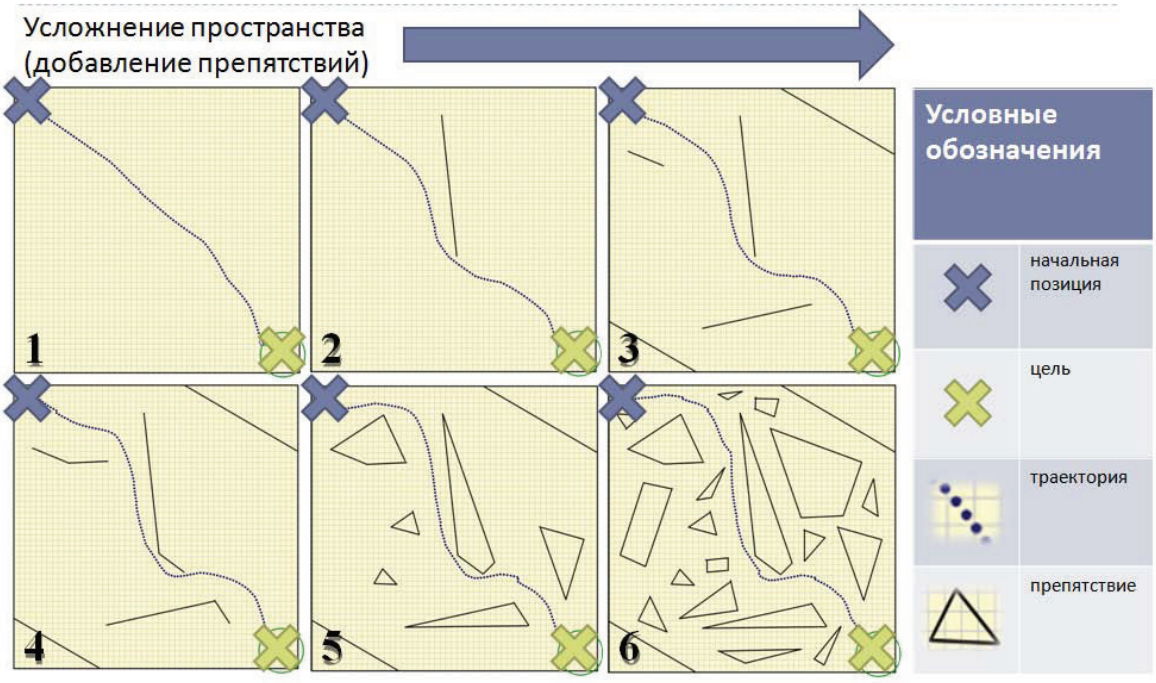


Рис. 14. Пример работы алгоритма на последовательно усложняемом пространстве

На рисунке 15 продемонстрирована работа алгоритма при различных начальных условиях, а на рисунке 16 – приведены функции ошибок (расстояния до цели) для различных случаев. Например, 16-б – случай объезда большого препятствия, перекрывшего цель, а 16-г – случай движения вдоль препятствия.

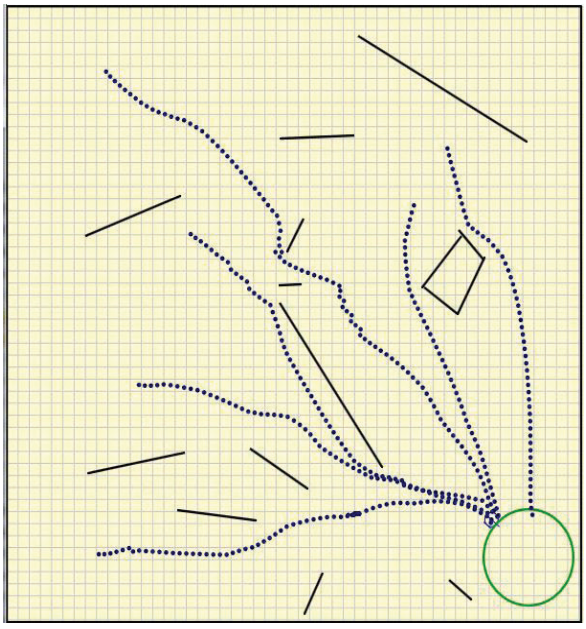


Рис. 15. Пример работы алгоритма при разных начальных условиях

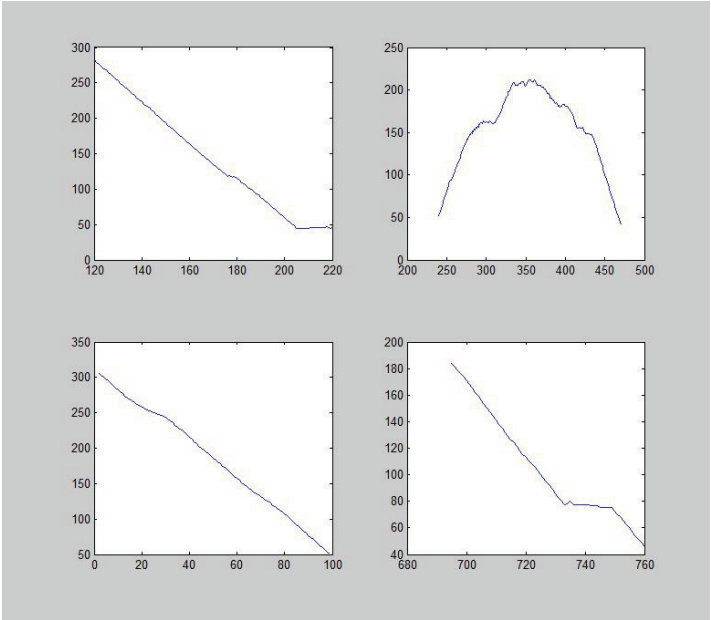


Рис. 16. Функции ошибок (удаления от цели) для разных случаев движения агента

На рисунке 17 приведены эмпирически построенные функции зависимости скоростей левого и правого моторов от близости препятствий. На графиках хорошо видно, что чем ближе препятствие, тем больше скорость мотора для осуществления соответствующего поворота.

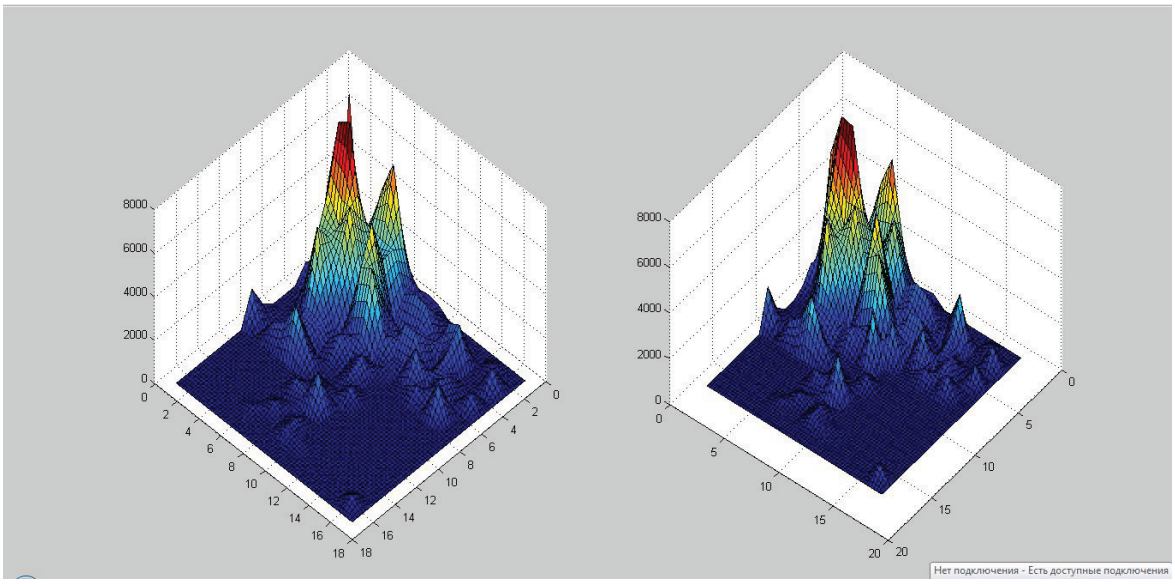


Рис. 17. Эмпирически построенные функции зависимости скоростей левого и правого моторов от близости препятствий

С помощью эмулирующей среды было произведено небольшое тестирование производительности полученной САУ. После проведения нескольких тестов были

получены следующие усредненные временные характеристики для этапов нечеткого вывода при решении данной задачи:

- Фаззификация: 200 - 400 тактов (зависит от количества операций деления);
- Логический вывод: 5000 – 20000 тактов (зависит от глубины доказательства);
- Дефаззификация: 500-1000 тактов (зависит от количества операций деления).

Показателем производительности нечетких вычислительных машин является скорость обработки правил. В базе правил конкретного нечеткого алгоритма 92 правила, следовательно, при погружении в стек не более чем на одно правило, примерное количество обрабатываемых правил – 184. Пусть скорость выполнения этапа логического вывода 10000 тактов. Тогда за 1 секунду при тактовой частоте процессора, установленной в 8 МГц (максимум для энергосберегающего режима МК) операция дефаззификации может быть выполнена 800 раз. Следовательно, скорость обработки правил: 140 тыс. правил в секунду.

Заключение

В результате работы была разработана нечеткая САУ, интеллектуальная в малом, в виде программно-аппаратной реализации нечеткого микроконтроллера, решающего задачу поиска пути в неизвестном окружении. Несмотря на достаточно грубую дискретизацию (8 бит), многочисленные округления и малый размер базы знаний, удалось получить неплохие результаты, демонстрирующие возможность использования аппарата нечеткой логики даже на микроконтроллерах общего назначения и решения с помощью такой реализации практических задач.

Разработанная программная реализация нечеткого микроконтроллера обеспечивает приемлемую скорость обработки правил (140 тыс. в секунду) и операций фаззификации /дефаззификации. Несмотря на дискретизацию и округления (неизбежные, при использовании операций целочисленного деления), получаемые результаты совпадают с ожидаемыми.

Синтезированный нечеткий контроллер для решения задачи поиска пути в неизвестном окружении является системой, интеллектуальной в малом, т.к. в нем отсутствует возможность к адаптации к изменяющимся условиям функционирования. Однако, благодаря тому, что нечеткий контроль осуществлен посредством нечеткого алгоритма на псевдонечетком языке нечетких операций, данный недостаток является исправимым. Повышение интеллектуальности может производится путем усложнения алгоритма. В этом случае необходимо расширить базу правил и включить в неё блок правил корректирующего или организационного уровня, производящих адаптацию существующих правил исполнительного уровня. Помимо правил, корректировке могут

подвергаться функции принадлежности термов с помощью специальных правил в зависимости от нечеткой функции ошибки системы.

Список литературы

1. Теория автоматического управления //Wikipedia.URL. http://en.wikipedia.org/wiki/ Теория_автоматического_управления (дата обращения: 03.03.2013)
2. 2 Поспелов Д.А. Ситуационное управление: теория и практика.–М.:Наука, 1986.–288 с.
3. Рубано в В. Г. Интеллектуальные системы автоматического управления. Нечеткое управление в технических системах: учебное пособие. — 2-е изд., стер. — Белгород: Изд-во БГТУ им. В. Г. Шухова, 2010. — 170 с.
4. Нечеткие множества в моделях управления и искусственного интеллекта / под. ред. Д.А.Поспелова.–М.:Наука, 1986.–312с
5. Кондратенко Г. В., Кондратенко Ю. П., Мухортова К. В.Синтез нечетких регуляторов на основе объектно-ориентированных технологий// ААЭКС. Оптимальное управление объектами и системами.2005. №1(15).
6. Пивкин В.Я., Бакулин В.П., Кореньков Д.И. Нечеткие множества в системах управления. – Новосибирск: изд-во НГУ, 1998.–75 с.
7. Saridis G.N. Analytical formulation of the principle of increasing precision with decreasing intelligence for intelligent machines // Automatics. 1989. V.25. №3.
8. Захаров В.Н., Ульянов С.В. Нечеткие модели интеллектуальных промышленных регуляторов и систем управления: Эволюция и принципы построения // Известия РАН: Техническая кибернетика. 1993. №4. С.189-205
9. Заде Л.А. Роль мягких вычислений и нечеткой логики в понимании, конструировании и развитии информационных/интеллектуальных систем. – Новости Искусственного Интеллекта. №2-3. 2001. С. 7 – 11.
10. Гриняев С. Нечеткая логика в системах управления // Компьютерра. 2001. №38.
11. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976.
12. Усков А.А. Принципы построения систем управления с нечеткой логикой // Приборы и системы. Управление, Контроль, Диагностика. 2004. № 6. С. 7-13.
13. Jantzen J. Design Of Fuzzy Controllers//Technical University of Denmark, department of automation, lyngby Denmark, technical report.1998.

14. Ульянов С.В., Тятюшкина О. Ю., Колбенко Е. В. Нечеткие модели интеллектуальных промышленных регуляторов и систем управления. Методология проектирования // Электронный журнал «Системный анализ в науке и образовании».2011.№2.
15. Алиев Р.А., Церковный А.З., Мамедова Г.А. Управление производством при нечеткой исходной информации. М.: Энергоатомиздат, 1991
16. Кудинов Ю.И. Нечеткие системы управления // Изв. АН СССР. Техническая кибернетика. 1990. № 5. С. 196-206.
17. Dimirovski G.M. Fuzzy-petri-net reasoning supervisory controller and estimating states of markov chain models//Computing and Informatics, Vol. 24, 2005, 563–576.
18. Mehmet Karakose, Erhan Akin. Block based fuzzy controllers//Department of Computer Engineering, University of Firat , Elazig, Turkey, 2010.
19. 19Zadeh L.A. Communication Fuzzy Algorithms// Information and control.1968. 12.C.94-102.
20. Castro J .L., Delgado M., Mantas C.J. Fuzzy grammar for handling fuzzy algorithms //International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.1999.Vol.7.C.277-286.
21. Бекасов Д.Е. Поиск пути в неизвестном окружении // Студенческий научный вестник. Сборник докладов участников общенаучной научно-технической конференции «Студенческая весна - 2012», посвященной 165-летию Н.Е. Жуковского. МГТУ им. Н.Э.Баумана /М.: НТА «АПФН»,2012. – С.412-417.
22. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры: Учеб. пособие для вузов . – 2-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2004. – 400 с.: ил. – (Информатика в техническом университете).