Инженерный вестник Дона, №10 (2025)
ivdon.ru/ru/magazine/archive/n10y2025/10434

Использование нейронных сетей в программируемых логических контроллерах как возможность отказаться от классического программирования

А.Н. Долидзе

Государственный университет аэрокосмического приборостроения, Санкт-Петербург

Аннотация: В статье описывается эксперимент по проектированию нейронной сети для программируемого логического контроллера с целью исключения необходимости привлечения программистов к разработке автоматизированных систем управления. Основная задача программируемых логических контроллеров — это упрощение автоматизации технологических процессов, они практически исключают задачи разработки печатных плат и операции пайки элементов. Очевидно, чем меньше различных задач приходится решать и чем проще эти задачи, тем быстрее пройдёт разработка и запуск новой системы, а себестоимость её будет ниже. С этой же целью, для программирования контроллеров, используются достаточно простые и наглядные языки, это сильно облегчает труд программистов. При современном уровне развития микроэлектроники вычислительные ресурсы контроллеров значительно превышают уровень необходимый для большинства задач автоматизации. Закономерно возникает вопрос, можно ли, используя избыточные вычислительные мощности, однократно разработать некую универсальную программу, способную адаптироваться к любому технологическому процессу. Естественно, такая программа будет работать медленнее и займёт больше памяти, но, в таком случае, задача программирования должна выродиться в задачу настройки готового программного обеспечения. Статья посвящена разработке прототипа такой программы на основе модели однослойного перцептрона. Описаны структура и параметры разрабатываемой нейронной сети с учётом особенностей целевой платформы. Разобран процесс обучения спроектированной нейронной сети. Перечислены и обоснованы ограничения, накладываемые на разработку. Обозначены достоинства и недостатки, а также варианты развития разработки.
Ключевые слова: программируемый логический контроллер, искусственная нейронная сеть, однослойный перцептрон, язык релейной логики, автоматизированная система управления технологическим процессом.

Введение

При создании первых программируемых логических контроллеров (ПЛК) в основу была поставлена идея максимального упрощения разработки систем управления для автоматизации технологических процессов [1]. Так, например, пайку, при монтаже внешних цепей, заменили клеммным соединением. Концепция упрощения также коснулась и языков программирования ПЛК [2], разработчики постарались сделать их максимально простыми и наглядными. Программа ПЛК должна легко

читаться и быть понятной инженерам, работающим на полевом уровне. В результате появился язык релейной логики (язык лестничных диаграмм, LD), дублирующий применяемые до появления ПЛК релейно-контактные схемы. В конечном счёте предполагалось исключить привлечение профессиональных программистов к разработке автоматизированных систем управления технологическими процессами (АСУ ТП).

Технологии непрерывно развиваются, что справедливо и для ПЛК, объёмы доступной памяти увеличиваются, процессоры наращивают быстродействие, так, например, контроллеры ОВЕН серии ПЛК110 [3] получили процессор, работающий на частоте 400 МГЦ, конечно, это достаточно скромно по сравнению с процессором персонального компьютера или смартфона, но более чем достаточно для работы большинства АСУ ТП.

Естественным образом возникает вопрос, а можно ли полностью исключить участие программиста в разработке АСУ ТП? Для решения данной задачи требуется составить некую универсальную программу, которую можно будет обучить работе с любым технологическим процессом без участия программиста. Такую роль может взять на себя искусственная нейронная сеть – вычислительная модель, организованная аналогично биологическим сетям нервных клеток живых организмов [4].

Постановка задачи

С целью установить целесообразность данного предположения проведём эксперимент: реализуем простейшую искусственную нейронную сеть, а именно – однослойный перцептрон [5] с помощью ресурсов ПЛК. Пример однослойного перцептрона представлен на рис.1.

Входные сигналы X поступают на входной слой сети (его единственная функция – распределение входных сигналов по нейронам). Каждый нейрон представляет собой комбинацию сумматора $\sum$ и функции активации φ.

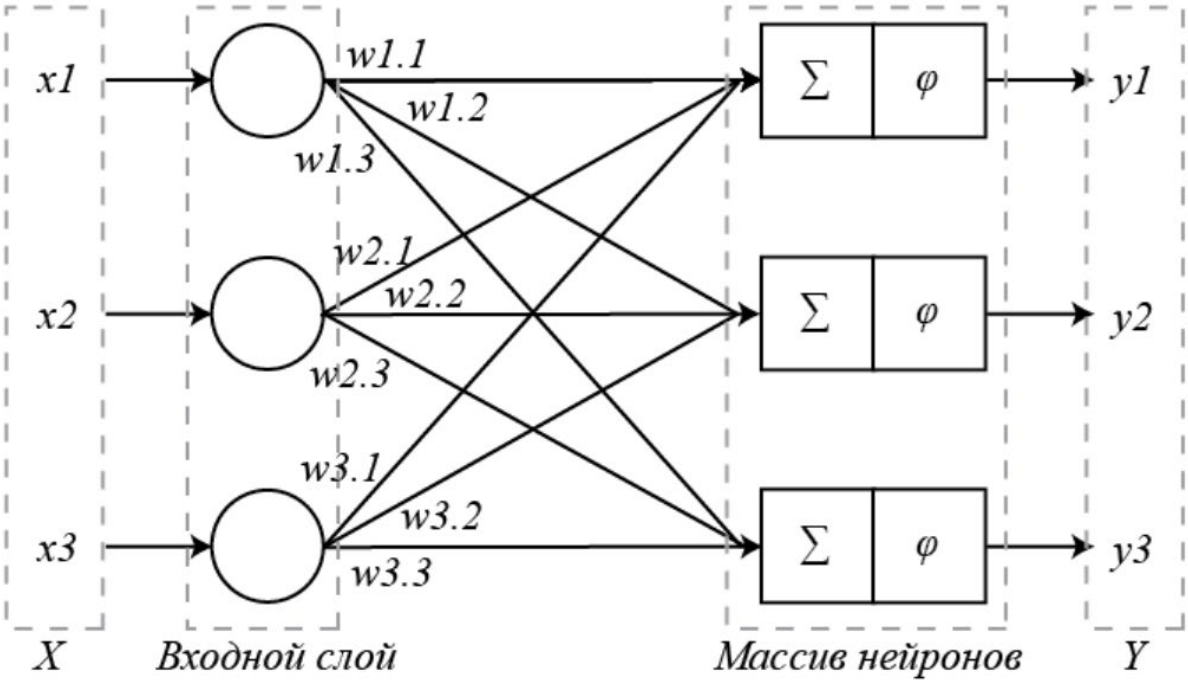

Рис. 1. – Структура однослойного перцептрона из трёх нейронов

Количество входов сумматоров равно количеству входных сигналов, каждому входу соответствует свой весовой коэффициент $w_{i,j}$, на который умножается входной сигнал x_i. Результат работы сумматора обрабатывается функцией активации, в простейшем случае это пороговая функция, то есть, если сумма больше некоторого значения, нейрон считается активированным (возбудившимся) – его выходной сигнал y_i равен единице, в противном случае – нулю. Процесс обучения сети заключается в последовательной корректировке весов таким образом, чтобы на соответствующую входную комбинацию реагировали только нужные нейроны.

Можно легко продемонстрировать, что данная модель подходит для решения задач автоматизации. Представим нейронную сеть в виде чёрного ящика (рис.2). На основе комбинации входных сигналов X формируются выходные сигналы Y – именно это и является основной функцией ПЛК.

К сожалению, на практике всё несколько сложнее: необходимо адаптировать модель к особенностям АСУ ТП.

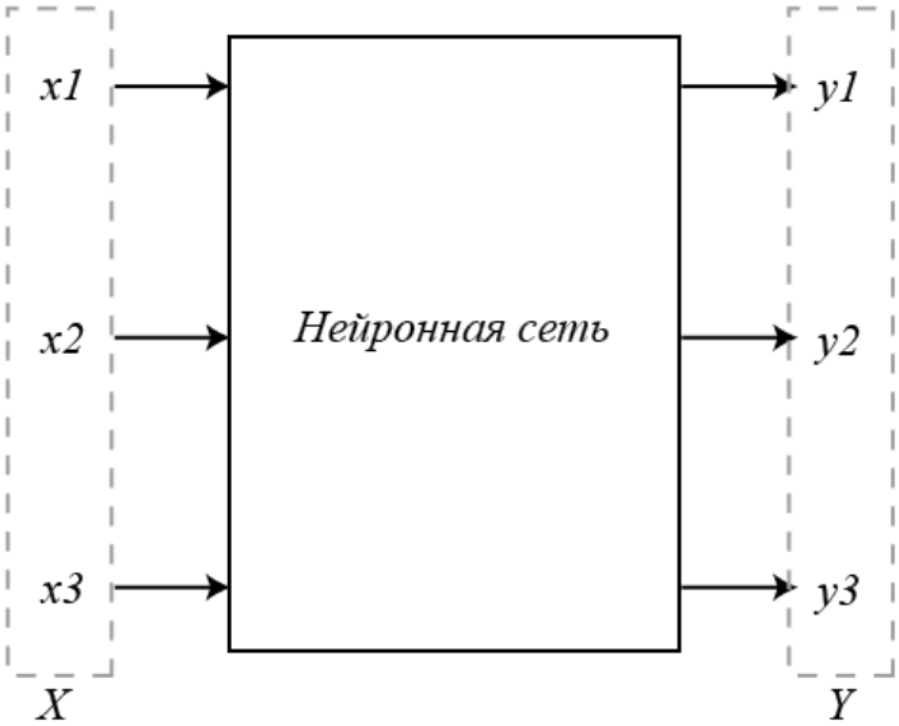

Рис. 2. – Функциональная модель нейронной сети

В терминах технологических процессов Y – выходные сигналы, посылаемые исполнительному оборудованию, допустим, активный нейрон соответствует логической единице на выходе контроллера, иначе на выходе будет ноль. Входные сигналы X, в свою очередь, – это набор датчиков и кнопок, подключённых к контроллеру. Для упрощения введём ограничение и предположим, что разрабатываемая программа будет работать только с дискретными сигналами (в контексте терминов ПЛК, это сигнал, который может принимать лишь два значения: единица или ноль).

Итак, на основе набора входных сигналов (условий) контроллер должен запускать или останавливать некоторые процессы. Проблема заключается в том, что изменение входных сигналов не всегда означает изменение выходных. Например, требуется наполнить бак водой, активен датчик нижнего уровня – контроллер подаёт сигнал открыть кран (кран будет открыт, пока активен этот сигнал). Бак начинает наполняться, и датчик нижнего уровня отключается, набор входных сигналов изменился, и

нейронная сеть отключит выходной сигнал – кран закроется (бак наполнился лишь на несколько сантиметров). Правильное условие для завершения наполнения бака – активация датчика верхнего уровня, а не отключение датчика нижнего уровня. В классической программе для ПЛК эта проблема решается с помощью функции RS-триггера (управление устройством разделяется на два независимых сигнала «установить» и «сбросить») [6]. В случае с нейронной сетью можно поступить так же: один нейрон включает выходной сигнал контроллера, второй – выключает. Выше сказанное означает, что нейронов должно быть вдвое больше, чем выходных сигналов ПЛК (даже если все они и не понадобятся в конкретной системе управления, программа должна быть универсальна).

Аналогичная проблема связана с входными сигналами – очень часто необходимо учитывать оба состояния датчика (активен – выполняется одно действие, неактивен – другое). Структура однослойного перцептрона не позволяет учесть неактивное (нулевое) состояние датчика. Выходом из этой ситуации будет ввод дополнительных инвертированных сигналов для каждого входного сигала.

Программная реализация

В качестве среды разработки была выбрана CoDeSys 2.3 [7] – это универсальный инструмент, предназначенный для работы с контроллерами различных производителей, она поддерживает все языки программирования стандарта МЭК-61131-3.

Для обеспечения большей наглядности и упрощения процесса отладки в качестве основного языка программирования будет использован язык релейной логики (LD).

Разрабатываемая система будет обладать следующими характеристиками: четыре входных сигнала и восемь выходных, что, в соответствии со сказанным выше, означает, что в сети должно быть

шестнадцать нейронов и восемь входов. Этого достаточно для проработки концепции, но при необходимости программу можно легко масштабировать.

В первую очередь необходимо разработать структуру данных для хранения весовых коэффициентов. Коэффициенты могут быть как положительными, так и отрицательными, при этом их значение не будет превышать нескольких десятков, поэтому для хранения весов будет выбран самый маленький знаковый тип данных SINT (от -128 до +127). Весовые коэффициенты удобно объединить в двухмерный массив (8 строк, соответствующих входным сигналам, и 16 столбцов, соответствующих нейронам). CoDeSys поддерживает многомерные массивы, но их использование затруднено при работе с языком релейной логики, в связи с чем была разработана структура данных WMatrix, представленная на рис.3, являющаяся набором одномерных массивов.

```
0001 TYPE WMatrix :
0002 STRUCT
0003     Weight1, Weight2, Weight3, Weight4, Weight5, Weight6, Weight7, Weight8: ARRAY [0..15] OF SINT;     (*Массивы весов входного слоя*)
0004 END_STRUCT
0005 END_TYPE
```

Рис. 3. – Пользовательская структура данных для хранения весовых коэффициентов

Весовые коэффициенты должны сохраняться при отключении питания ПЛК, поэтому их необходимо хранить в энергонезависимой памяти. Чтобы создать такую переменную в CoDeSys, необходимо объявить её как глобальную с дополнительными ключевыми словами RETAIN и/или PERSISTENT. Переменные, объявленные как RETAIN, будут записаны при отключении питания или перезагрузке контроллера. Ключевое слово PERSISTENT позволяет переменным сохранять значение при остановке контроллера без отключения питания. На рис.4. представлено объявление энергонезависимой переменной Weight типа WMatrix.

Сигналы, поступившие на входы контроллера, как было ранее сказано, нужно дублировать их обратным значением (рис. 5).

Теперь входные сигналы нужно умножить на соответствующие весовые коэффициенты.

```
0001 VAR_GLOBAL
0002 END_VAR
0003 VAR_GLOBAL RETAIN
0004     Weight: WMatrix; (*Массив весов входного слоя*)
0005 END_VAR
```

Рис. 4. – Объявление энергонезависимой переменной для хранения массива весовых коэффициентов

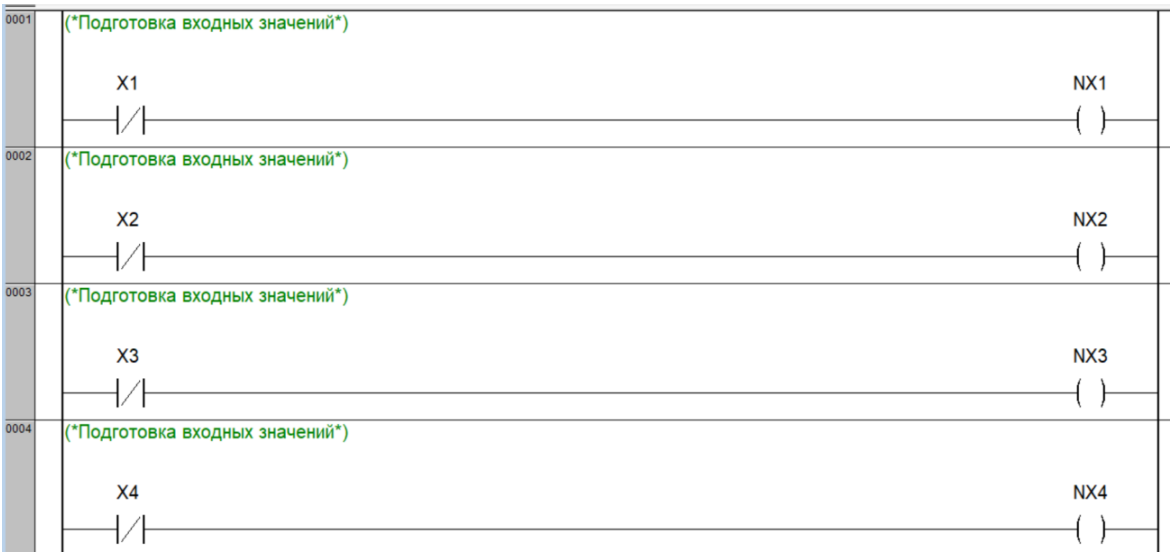

Рис. 5. – Инвертирование входных сигналов

Операция умножения ресурсозатратна (если речь не идёт о умножении на степени двойки, тогда операция вырождается в сдвиг влево), а большинство ПЛК всё же не предназначены для серьёзных математических вычислений. В разрабатываемой модели требуется производить 128 операций умножения на каждом цикле работы программы, это существенно замедлит реакцию ПЛК [8] на внешние события. Ранее было введено ограничение: в данной системе допускаются только дискретные входные и выходные сигналы, из чего следует, что при активном входном сигнале весовой коэффициент умножается на единицу, а при неактивном – на ноль. Очевидно, что при такой постановке задачи умножение вырождается в простую операцию «если». Иными словами, происходит фильтрация массива весовых

коэффициентов, остаются лишь те значения, которые соответствуют активному входному сигналу. Программно данная задача реализуется с помощью набора мультиплексоров (функция SEL). В качестве управляющего сигнала мультиплексора используется входной сигнал *X* или его инверсия *NX*, на первый вход подаётся массив нулей (заглушка, передаваемая на выход мультиплексора, когда входной сигнал неактивен), а на второй вход – массив коэффициентов, соответствующий данному входу. Результат работы мультиплексора сохраняется в массиве Gate, именно эти значения попадут на сумматоры нейронов. На рис.6 представлена программная реализация описанного решения.

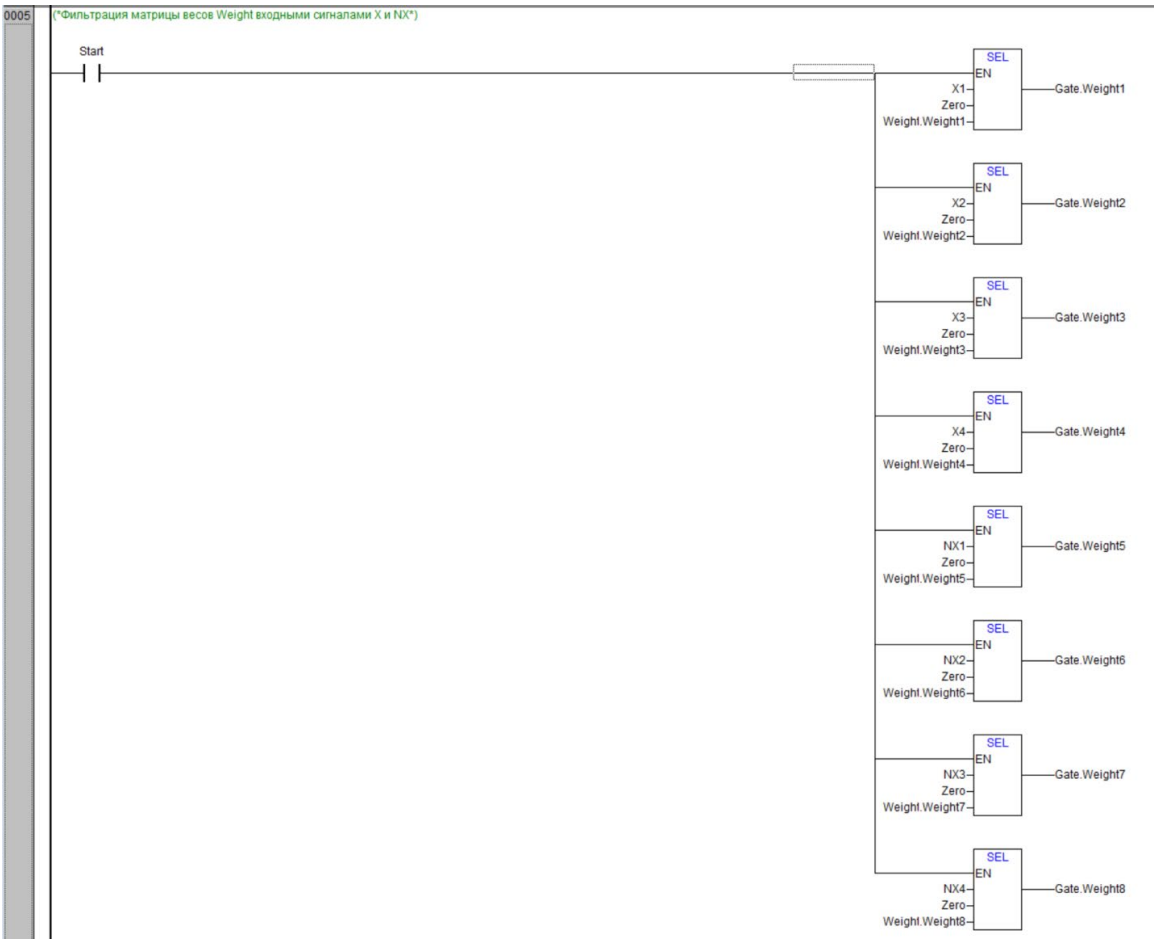

Рис. 6. – Фильтрация массива весовых коэффициентов входными сигналами

Теперь необходимо реализовать сами нейроны. Напомню, что в нашем случае искусственный нейрон состоит из сумматора и функции активации

(рис.7). В качестве функции активации была выбрана пороговая, то есть, если сумма сигналов поступивших на нейрон больше определённого значения, нейрон считается сработавшим. Значение порога для функции выбирается экспериментальным путём, если оно будет слишком маленьким, сеть не сможет обучиться, а увеличение этого числа сказывается на скорости обучения. Учитывая особенности назначения разрабатываемой системы, порог должен быть не меньше числа используемых входов ПЛК (что вдвое меньше числа входов сети). Данное утверждение объясняется тем, что для запуска некоего события условием может быть активация всех датчиков, при этом невозможно, чтобы условием запуска какого-либо события было наличие активного и неактивного состояния одного и того же датчика.

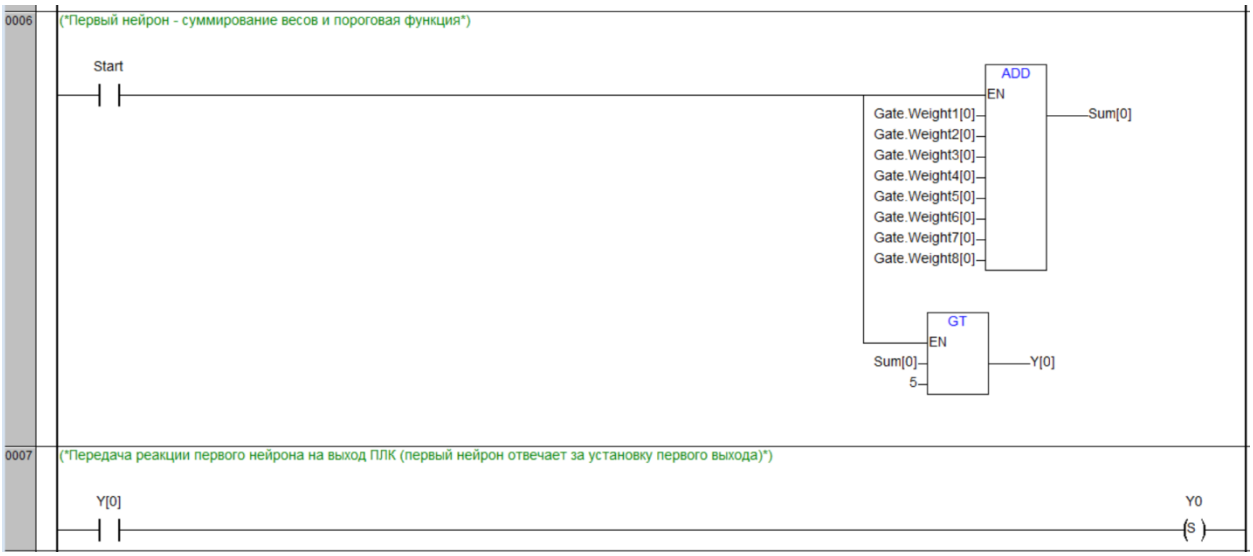

Рис. 7. – Искусственный нейрон (сумматор и пороговая функция) и передача его сигнала на физический выход ПЛК.

Ранее уже было сказано, что нейроны в разрабатываемой сети будут работать парами: один для запуска события, другой для остановки. На рис.7 представлен пример нейрона, запускающего событие. Разница между запускающим и останавливающим нейроном заключается в типе выходной катушки при передаче сигнала на физический выход ПЛК. Катушка установки входа обозначается буковой S, а катушка сброса – R.

Аналогичным образом будут выглядеть все шестнадцать нейронов. Если сети будут предоставлены готовые весовые коэффициенты, то она уже будет готова к работе.

Обучение сети

На практике весовые коэффициенты не только заранее не известны, но и вычислить их вручную крайне сложно, поэтому нейронная сеть должна настраивать коэффициенты самостоятельно (обучаться).

В качестве метода обучения был выбран «обучение с учителем» [9]. Так как оператор, который работает с целевой АСУ ТП, знает, какая реакция должна последовать на каждый набор входных данных, он будет сообщать сети, когда она ошиблась, и таким образом обучит её.

Рассмотрим практическую реализацию этого метода. В рамках данного эксперимента предположим, что ожидаемая реакция системы записана оператором в двумерный массив D. Каждая строка этого массива соответствует правильной выходной комбинации на выходе нейронов, а фактическая реакция сети собрана в одномерный массив Y. Необходимо сравнить ожидаемую и фактическую реакцию, то есть поэлементно вычесть из строки массива D массив Y. На практике эта задача является трудоемкой при использовании ресурсов языка LD (массивы D и Y объявлены с разными структурами и типами данных). CoDeSys позволяет использовать разные языки программирования в одном проекте, благодаря чему задачу, трудно решаемую на LD, можно легко решить с помощью функции, написанной на языке ST (рис.8).

Результат сравнения сохраняется в массив (вектор) ошибок E, значение элементов которого определяет корректность реакции каждого нейрона. Значение, отличное от нуля, означает ошибку в работе нейрона. Единица означает, что нейрон не сработал тогда, когда от него этого ожидали,

соответственно, минус единица означает обратную ситуацию (нейрон сработал, когда этого не ожидалось).

```
0001 FUNCTION SUB_ARR : ARRAY [0..15] OF SINT
0002 VAR_INPUT
0003     A: ARRAY [0..15] OF SINT;
0004     B: ARRAY [0..15] OF BOOL;
0005 END_VAR
0006 VAR
0007     i: INT;
0008 END_VAR
```

```
0001 FOR i:=0 TO 15 BY 1 DO
0002         SUB_ARR[i]:=A[i]-BOOL_TO_SINT (B[i]);
0003 END_FOR;
```

Рис. 8. – Функция для сравнения ожидаемой и фактической реакции сети (язык ST)

Сеть нуждается в обучении, если произошла ошибка хотя бы в одном нейроне. Определить наличие ошибки можно сравнив с нулём сумму абсолютных значений элементов массива E (рис.9).

Итак, если ошибка произошла, необходимо скорректировать весовые коэффициенты (рис.10), для этого к строкам структуры Weight (массив весовых коэффициентов в энергонезависимой памяти контроллера), соответствующим активным входам сети, поэлементно прибавляется массив ошибок E. Сложение элементов массивов, подобно процедуре сравнения реакций, производится с помощью пользовательской функции, написанной на языке ST. После нескольких итераций описанной процедуры сеть начинает правильно реагировать на предложенную комбинацию входных сигналов (обучается), после чего устанавливается новая комбинация и процесс повторяется заново.

Стоит отметить, что обучаясь на новой комбинации, сеть может «забыть» ранее изученные, изменив задействованные в них весовые коэффициенты, поэтому процесс обучения повторяется циклично, пока сеть не перестанет ошибаться на всех входных комбинациях.

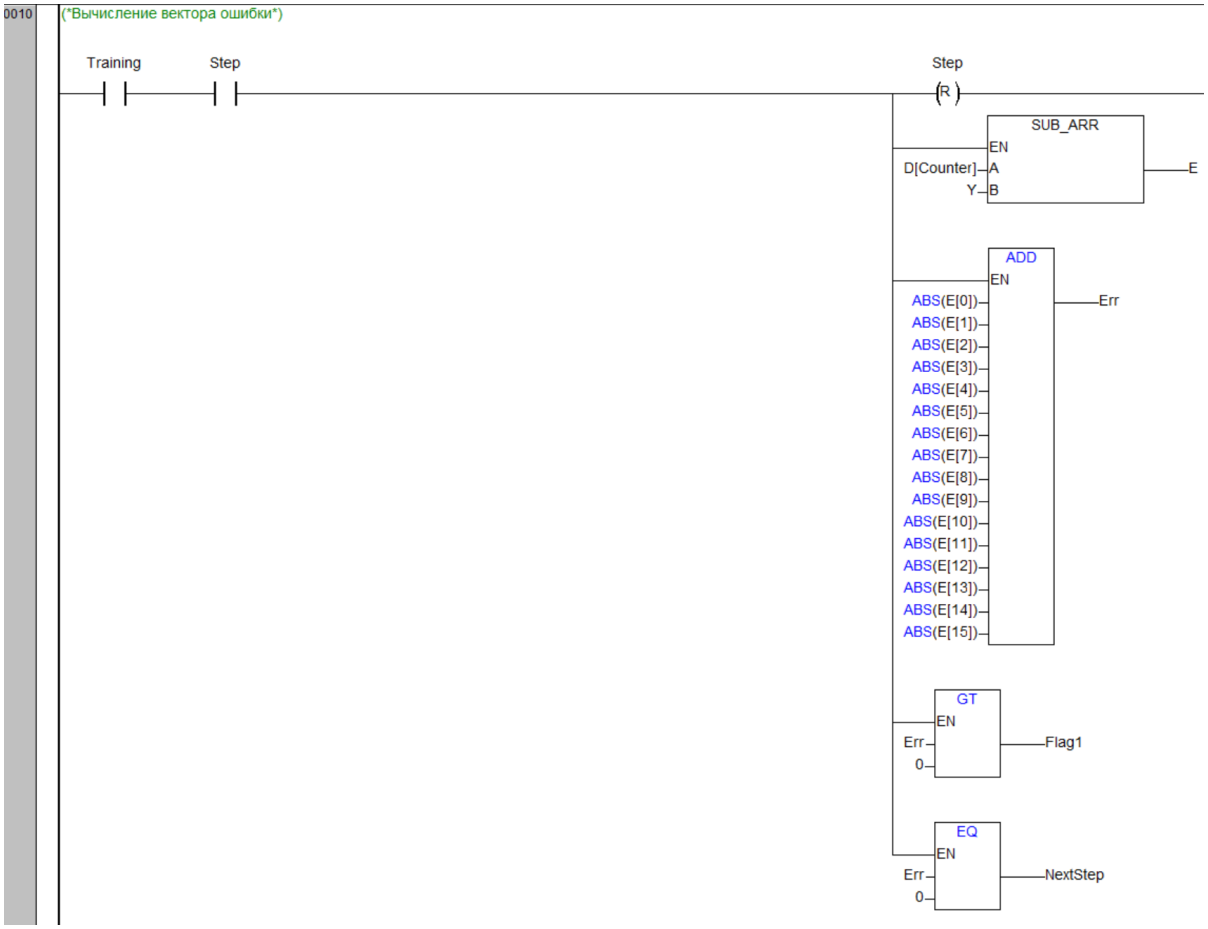

Рис.9. – Сравнение ожидаемой и фактической реакции с последующим формированием вектора ошибок и фиксацией факта наличия ошибки

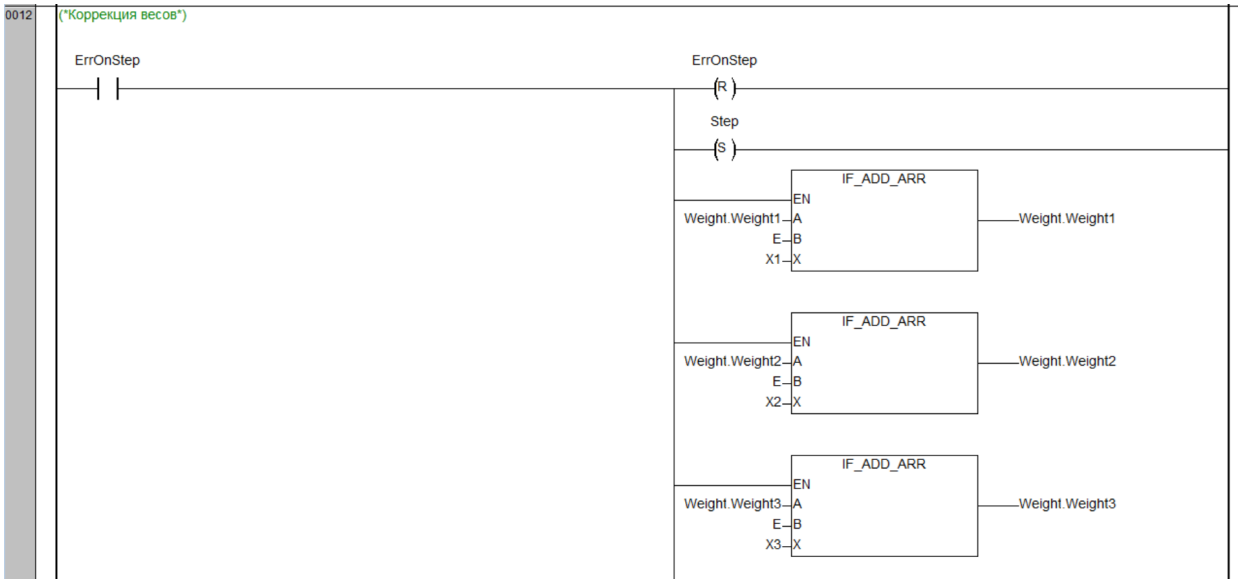

Рис.10. – Коррекция весовых коэффициентов (фрагмент)

Выводы

В результате экспериментальной разработки удалось получить функционирующий однослойный перцептрон, работоспособность которого проверена на контроллере ОВЕН ПЛК110. Система выполняет поставленные функции и может легко масштабироваться, но, по сравнению с классическими подходами к программированию ПЛК, является значительно более ресурсозатратной и, соответственно, менее быстродействующей [10], но всё ещё достаточно эффективной. Разработанная система обладает рядом значительных ограничений, не позволяющих ей считаться полностью универсальной программой, а именно: невозможность использования аналоговых сигналов, таймеров и счётчиков. Также в реальных АСУ ТП, в качестве условий срабатывания событий, используются не только входные сигналы системы, но и выходные. Описанные ограничения будут сняты в последующих разработках. В рамках данного эксперимента предполагалось, что ожидаемая реакция на входные комбинации уже записана в контроллер, а обучение системы происходит практически вручную (оператор должен устанавливать входные комбинации и переключать итерации циклов самостоятельно), в ходе последующих экспериментов планируется разработать методы автоматизации обучения.

Литература

1. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. Москва: Солон-Пресс, 2016. 254 с.

2. Programmable controllers – Part 3: Programming languages. International Electrotechnical Commission, International Electrotechnical Commission, 2013. 435 с.

3. Программируемый логический контроллер ПЛК110(М02): Руководство по эксплуатации // Owen.ru 2024. URL: owen.ru/downloads/re_plk110_m02.pdf (дата обращения: 26.08.2025).

4. Нейронные сети // bigenc.ru: научно-образовательный портал «Большая российская энциклопедия» URL: bigenc.ru/c/neironnye-seti-e734b3 (дата обращения: 26.08.2025).

5. Freund, Y., Schapire, R. E Large Margin Classification. Using the Perceptron Algorithm // Machine Learning. 1999. №37(3). С. 277-296.

6. Hugh Jack Automating Manufacturing Systems with PLCs. Lulu.com, 2010. 644 с.

7. ПЛК1xx [M02] Программирование в CODESYS: Руководство пользователя // Owen.ru 2024. URL: owen.ru/uploads/467/rp_plk1hh_m02__1-ru-75044-1.34.pdf (дата обращения: 26.08.2025).

8. Долидзе А.Н. Методы решения проблемы замедления рабочего цикла контроллера при увеличении объёма программы // Инженерный вестник Дона, 2025, №2. URL: ivdon.ru/ru/magazine/archive/n2y2025/9841.

9. Романов Д.Е. Нейронные сети обратного распространения ошибки // Инженерный вестник Дона, 2009, №3. URL: ivdon.ru/ru/magazine/archive/n3y2009/143.

10. Сафаров И.М., Богданова Н.В., Латыпов Т.И. Комплексный критерий оценки эффективности программируемых логических контроллеров // Инженерный вестник Дона, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8706

References

1. Petrov I.V. Programmiruemye kontrollery. Standartnye yazyki i priyomy prikladnogo programmirovaniya [Programmable controllers. Standard languages and techniques of application programming]. Moskva: SOLON-Press, 2016. 254 p.

2. Programmable controllers – Part 3: Programming languages. International Electrotechnical Commission, International Electrotechnical Commission, 2013. 435 p.

3. Programmiruemyy logicheskiy kontroller PLK110 (M02): Rukovodstvo po ekspluatatsii [PLC110 (M02) Programmable Logic Controller: User Manual]. 2024. URL: owen.ru/downloads/re_plk110_m02.pdf (accessed 26/08/25).

4. Neyronnye seti [Neural networks]. URL: bigenc.ru/c/neironnye-seti-e734b3 (accessed 26/08/25).

5. Freund, Y., Schapire, R. E Machine Learning. 1999. №37(3). pp. 277-296.

6. Hugh Jack Automating Manufacturing Systems with PLCs. Lulu.com, 2010. 644 p.

7. PLK1xx [M02] Programmirovanie v CODESYS: Rukovodstvo pol'zovatelya [PLC1xx [M02] Programming in CODESYS: User Manual]. 2024. URL: owen.ru/uploads/467/rp_plk1hh_m02__1-ru-75044-1.34.pdf (accessed 26/08/25).

8. Dolidze A.N. Inzhenernyj vestnik Dona, 2025, №2. URL: ivdon.ru/ru/magazine/archive/n2y2025/9841.

9. Romanov D.E. Inzhenernyj vestnik Dona, 2009, №3. URL: ivdon.ru/ru/magazine/archive/n3y2009/143.

10. Safarov I.M., Bogdanova N.V., Laty`pov T.I. Inzhenernyj vestnik Dona, 2023, №9. URL: ivdon.ru/ru/magazine/archive/n9y2023/8706.

Дата поступления: 17.08.2025

Дата публикации: 25.09.2025