Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

Санкт-Петербургский горный университет

Кафедра автоматизации технологических процессов и производств

ОСНОВЫ АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ НЕФТЕГАЗОВОГО ПРОИЗВОДСТВА

Методические указания к самостоятельным работам для студентов бакалавриата направления 21.03.01

САНКТ-ПЕТЕРБУРГ
2021

УДК 681.513.2(073)

ОСНОВЫ АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ НЕФТЕГАЗОВОГО ПРОИЗВОДСТВА: Методические указания к выполнению самостоятельной работы / Санкт-Петербургский горный университет. Сост. *Н.И. Котелева.* СПб, 2021. 31 с.

Содержат информацию о применения теории автоматического регулирования, методов и средств автоматического контроля и регулирования при управлении технологическими объектами нефтегазовых производств.

Предназначены для студентов бакалавриата направления 21.03.01 «Нефтегазовое дело» по профилям «Эксплуатация и обслуживание объектов добычи газа, газоконденсата и подземных хранилищ», «Разработка и эксплуатация углеводородных месторождений шельфа», «Эксплуатация и обслуживание объектов добычи нефти».

Научный редактор проф. *В.Ю. Бажин*

Рецензент *В.В. Васильев*

ОСНОВЫ АВТОМАТИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПРОЦЕССОВ НЕФТЕГАЗОВОГО ПРОИЗВОДСТВА

Методические указания к самостоятельным работам для студентов направления подготовки 21.03.01

Сост.: *Н.И. Котелева*

Печатается с оригинал-макета, подготовленного кафедрой автоматизации технологических процессов и производств

Ответственный за выпуск *Н.И. Котелева*

Лицензия ИД № 06517 от 09.01.2002

Подписано к печати 16.03.2021. Формат 60×84/16.
Усл. печ. л. 1,8. Усл.кр.-отт. 1,8. Уч.-изд.л. 1,5. Тираж 75 экз. Заказ 196.

Санкт-Петербургский горный университет
РИЦ Санкт-Петербургского горного университета
Адрес университета и РИЦ: 199106 Санкт-Петербург, 21-я линия, 2

ВВЕДЕНИЕ

Дисциплина «Основы автоматизации технологических процессов нефтегазового производства» призвана сформировать у студентов базовые знания в области автоматизации технологических процессов нефтегазовых производств, подготовить выпускников к решению профессиональных задач, связанных с автоматизированными системами управления и выбором основных средств решения поставленных перед этими системами задач, анализом характеристик и результатов функционирования систем, методами оптимизации, сформировать у студентов современное научное мировоззрение, развить творческий потенциал, естественно-научное мышление, ознакомить студентов с методологией научных исследований.

Представленные методические указания к выполнению самостоятельной работы по дисциплине «Основы автоматизации технологических процессов нефтегазового производства» содержат перечень работ, необходимых к выполнению во время самостоятельной работы студентов, требования к отчетным материалам по выполнению данных работ, основные требования к содержанию разделов, объему, а также процедуре их защиты и критериях оценки.

ОБЩИЕ СВЕДЕНИЯ ОБ ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Самостоятельная работа студентов – обязательная и неотъемлемая часть учебной работы студента по дисциплине «Основы автоматизации технологических процессов нефтегазового производства». Перечень работ, необходимых к выполнению во время самостоятельной работы, а также срок сдачи отчетов по данным видам работ представлен в специальном документе «График самостоятельных работ студента».

По дисциплине «Основы автоматизации технологических процессов нефтегазового производства» программой предусмотрена самостоятельная работа студентов в объеме 40 часов. Перечень работ для организации самостоятельной работы студентов, объем работ и форма отчета представлены в таблице 1.

Таблица 1

Перечень работ для организации самостоятельной работы

Наименование работы	Трудоемкость	Форма отчета/объем, стр.
Расчетно-графическая работа №1	16	Пояснительная записка/ не регламентирован
Расчетно-графическая работа №2	12	Пояснительная записка/ не регламентирован
Расчетно-графическая работа №3	12	Пояснительная записка/ не регламентирован

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №1

Основы программирования программируемых логических контроллеров (ПЛК).

Программируемый логический контроллер (ПЛК) используется в автоматизированных системах управления технологическими процессами (АСУТП) всех отраслей промышленности. В зависимости от функциональных возможностей, технических характеристик и конструктивного исполнения контроллеры можно условно разделить на моноблочные и модульные контроллеры.

Моноблочный контроллер представляет собой микропроцессорное устройство, в едином конструктиве которого располагаются источник питания (не обязательно), центральный процессор (сопроцессоры), память, включающая память программ и память переменных (как правило, энергонезависимая), встроенный порт(-ы) для выхода в сеть, фиксированное число каналов аналогового и (или) дискретного ввода/вывода, встроенный ПИД-регулятор с автонастройкой (необязательно), слот расширения для подключения дополнительных модулей, ЖК-дисплей (необязательно), индикаторы состояния контроллера.

Как правило, контроллеры устанавливаются на DIN-рейку, а соединение с другими модулями, например, с модулем питания, модулем аналогового ввода и др. осуществляется с помощью разъемов или проводников с наконечниками «под винт».

Помимо этих общих характеристик контроллеры отличаются набором встроенных функций, числом базовых команд, способом программирования и т.п. Используя соответствующие модули, легко подобрать конфигурацию ПЛК, максимально удовлетворяющую требованиям АСУ конкретного технологического процесса. Моноблочные контроллеры обладают меньшей степенью гибкости и обычно используются для более простых задач. При необходимости сбора большого количества сигналов на пространственно распределенных объектах, используют удаленные станции ввода-вывода, которые не имеют в своем составе процессора и не выполняют программ, а только обрабатывают входные и выходные сигналы и передают их на главный контроллер по сети. Также к сети могут быть подключены терминалы, установленные непосредственно рядом с объектом управления. Терминал – это микропроцессорное устройство для отображения текущей информации о параметрах технологического процесса и изменения уставок. На терминале расположен дисплей и функциональные клавиши.

Порядок выполнения работы. Режим запуска ПЛК и основные теоретические сведения.

1. Для запуска ПЛК необходимо выполнить команду *Пуск – Proficy Machine Edition*

Программирование контроллеров 90 Micro, как и других контроллеров GE Fanuc производится при помощи программного обеспечения CIMPLICITY Machine Edition. Для того, чтобы начать работу с контроллером, нужно создать проект Machine Edition. Проект содержит конфигурацию контроллера, логику программы и позволяет установить связь между контроллером и ПК. После установления связи можно загружать в память контроллера конфигурацию и логику, считывать их из памяти контроллера, запускать и останавливать контроллер, а также отслеживать статус контроллера и состояние входных и выходных каналов.

2. Выбираем нужный шаблон проекта

При создании проекта Machine Edition предлагает выбрать один из шаблонов списка. Шаблон содержит набор компонентов, соответствующих контроллеру, для которого предназначен данный

проект. В данной работе это – Series 90 Micro PLC. Внешний вид диалогового окна при создании проекта представлен на рисунке 1.1.

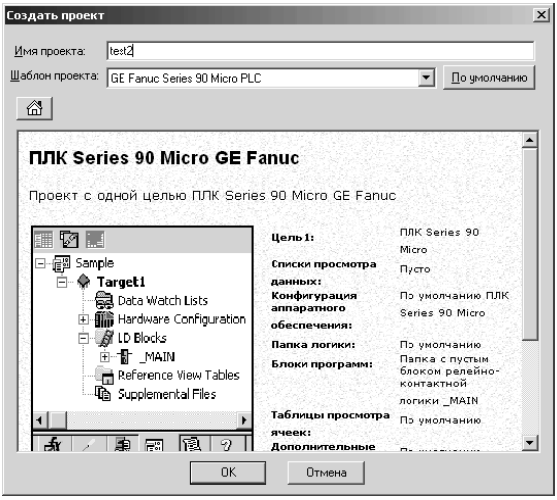


Рис. 1.1. Внешний вид диалогового окна при создании проекта

После нажатия на кнопку ОК создается проект, содержащий цель “Target 1”. Цель – это часть проекта, которая работает с одним контроллером или станцией ввода вывода. Одна из целей проекта является активной. Цель содержит следующие компоненты (рисунок 1.2):

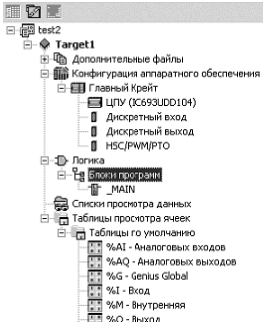


Рис. 1.2. Перечень компонентов программы

3. Проводим конфигурацию аппаратного обеспечения. Сравниваем тип процессорного модуля как указано в методичке. При несоответствии изменяем его.

Конфигурация аппаратного обеспечения по умолчанию для данного контроллера. Поскольку ПЛК 90 Micro является моноблочным, процедура конфигурирования фактически состоит в выборе типа процессора. Если конфигурация контроллера не соответствует конфигурации по умолчанию, следует заменить процессорный модуль. Для этого, щелкнув правой кнопкой мыши на значке ЦПУ, выбрать “заменить модуль”, после чего выбрать соответствующий модуль из каталога.

Тип процессорного модуля указан в документации и на этикетке с левой стороны контроллера.

После замены модуля появляется окно параметров. На соответствующих вкладках можно просматривать и редактировать настройки ЦПУ, режим цикла и конфигурацию памяти контроллера.

4. Открываем блок программ. Пишем программу в специальном поле.

Контроллер 90 Micro не поддерживает использование подпрограмм, поэтому вся логика находится в блоке _MAIN. При двойном щелчке мыши появляется окно написания программы. Программа пишется на языке LD (Ladder Diagram – Лестничных диаграмм), который также называют языком релейно-контактной логики. Основными операторами программы являются контакты и катушки, при помощи которых производится обращение к различным функциональным блокам – таймерам, счетчикам и другим блокам.

Программа на языке LD похожа на электрическую схему. Операторы программы располагаются между двумя шинами питания +24V и 0V.

Создание программ для ПЛК с использованием языка LD

Процесс создания программ для ПЛК состоит из трех этапов:

1. Определение переменных.
2. Написание словесного алгоритма.
3. Перевод словесного алгоритма в программный код.

Рассмотрим каждый из этапов на примере одной задачи по управлению: написать программу управления двумя сигнальными лампами. Управление осуществляется путем нажатия на кнопку «Включить/Выключить сигнальные огни».

Определение переменных.

На данном этапе происходит анализ объекта управления определение входных выходных и других переменных процесса. По окончании данного этапа создается таблица сигналов с подробным описанием всех переменных проекта.

Определяем переменные для поставленной задачи.

Итак, у нас имеется три физических устройства, подключаемых к контроллеру: сигнальная лампа 1, сигнальная лампа 2 и кнопка «Включить/Выключить сигнальные огни». Для начала необходимо определить, Какие устройства подключаются ко входу контроллера, а какие к выходу. К входу контроллера подключаются те устройства, состояние которых изменяется не программно, например человеком по заданию. В нашем примере это кнопка включения/выключения сигнальных огней, а к выходу контроллера подключается те устройства, состояние которых должно изменяться программно по некоторому алгоритму. В нашем примере – это сигнальные лампы. Теперь необходимо определится с типом сигналов. Сигналы бывают дух типов – дискретные и аналоговые. Аналоговые сигналы могут иметь множественные значения в некотором диапазоне, дискретные сигналы имеют два значения (0 или 1, «Истина» или «Ложь», «Да» или «Нет», «Включено» или «Выключено»). Кнопка «Включить/выключить сигнальные огни» имеет два состояния – Включено/Выключено. Сигнальные огни также имеют два состояния – Включены/Выключены.

Таким образом, принимаем следующее решение: пусть первая сигнальная лампа будет подключена к первому дискретному выходу контроллера, вторая сигнальная лампа ко второму дискретном выходу, а кнопка «Включить/Выключить сигнальные огни» к первому дискретному входу контроллера. На рисунке 1.3 показана схема подключений устройств к контроллеру.

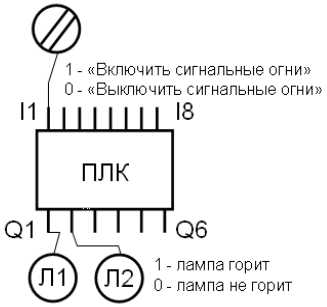


Рис. 1.3 Схема подключений

Для завершения этапа определения переменных и составления таблицы сигналов необходимо знать, каким образом организована память ПЛК, и указать правильный адрес переменной.

Организация памяти ПЛК GE Fanuc 90 Micro.

Основные объекты адресации языков программирования ПЛК представляют собой либо биты, соответствующие дискретным логическим переменным, либо слова (регистры) того или иного формата, соответствующие числовым данным.

К битовым ячейкам относятся:

- %I – дискретный ввод
- %Q – дискретный вывод

Эти биты являются «логическими отображениями» электрического состояния ввода/вывода. Они хранятся в памяти данных и обновляются на каждом сканировании задачи, в которой они сконфигурированы.

- %M – внутренние ячейки памяти
- %T – внутренние ячейки для хранения временных переменных

Внутренние биты используются для хранения промежуточных состояний во время выполнения программы.

- %S – системная память.

Системные биты следят за корректностью операции ПЛК в процессе работы программы приложения.

К регистровым ячейкам относятся:

- %AI – аналоговый ввод
- %AQ – аналоговый вывод
- %R – внутренняя регистровая память

В регистровых ячейках хранятся байтовые переменные, которые позволяют описывать числовые данные.

Адресация переменной величины в стандарте IEC 61131/3 осуществляется следующим образом. Сначала записывается символ стандарта %, затем тип объекта (ввод, вывод или др.), после чего адрес в памяти ПЛК. Составим таблицу сигналов для нашей задачи (таблица 1.1).

Таблица 1.1

Таблица сигналов

Адрес ПЛК	Тип сигнала	Описание сигнала
I0001 (%I1)	Дискретный вход	Состояние кнопки включения/выключения сигнальных огней: 1 - кнопка включена 0 - кнопка выключена
Q0001 (%Q1)	Дискретный выход	Состояние первого сигнального огня: 1 – сигнальный огонь горит 0 - сигнальный огонь не горит
Q0002 (%Q2)	Дискретный выход	Состояние второго сигнального огня: 1 – сигнальный огонь горит 0 - сигнальный огонь не горит

Другие переменные будем определять на каждом промежуточном этапе выполнения задания по мере необходимости.

Составление словесного алгоритма

Словесный алгоритм составляется при помощи конструкции ЕСЛИ…..ТО…. Причем после слова ЕСЛИ перечисляются все условия наступления некоторого действия, а после слова ТО перечисляются сами действия.

Составим словесный алгоритм для трех возможных вариантов управления лампами сигнальных огней:

1 Вариант: при включении кнопки «Включение/Выключение сигнальных огней» должна загореться лампочка первого сигнального огня:

1.) ЕСЛИ кнопка включена (то есть переменная I1=1), то лампочка 1 загорается (Q1=1).

2 Вариант: при включении кнопки «Включение/Выключение сигнальных огней» должна загореться первая лампочка сигнального огня через три секунды, при выключении кнопки «Включение/Выключение сигнальных огней» лампочка первого сигнально огня должна погаснуть через пять секунд

1.) ЕСЛИ кнопка включена (то есть переменная I1=1) и прошло три секунды, то лампочка 1 загорается (Q1=1).

2.) ЕСЛИ лампочка 1 загорелась (Q1=1) и кнопка выключена (то есть переменная I1=0) и прошло пять секунд, то лампочка 1 гаснет (Q1=0).

Система команд языка LD

Язык LD представляет собой графическую интерпретацию процесса разработки релейно-контакторных схем управления. Первоначально на языке LD программировались контроллеры производства компании Allen Bradley. Ввиду его удобства и значительного количества пользователей, обладающих навыками проектирования логических систем на базе реле и контакторов, язык LD был введен в стандарт IEC 61131/3 и в настоящее время является одним из наиболее распространенных языков программирования ПЛК. Этот язык наиболее удобен для программирования небольших задач дискретной логики, поэтому многие контроллеры младших классов имеют язык LD в качестве основного для разработки программ управления. Программы, написанные на языке LD, состоят из последовательности ступеней, которые выполняются ПЛК последовательно, слева направо.

Набор графических элементов языка LD включает:

- Входы/выходы ПЛК (кнопки, датчики, реле, индикаторные лампы и т.д.).
- Стандартные управляющие системные функции (таймеры, счетчики и т.д.).
- Арифметические, логические и специальные операции.
- Внутренние переменные ПЛК.

Контакты. Контакты используются для считывания значений битовых переменных.

Нормально разомкнутый контакт ⊣ ⊢ пропускает поток энергии направо, если связанная с ним переменная находится в состоянии “1”.

Нормально замкнутый контакт ⊣/⊢ (инверсный) пропускает поток энергии направо, если связанная с ним переменная находится в состоянии “0”.

Катушки. Катушки используются для записи значений в битовые переменных.

Катушка —◯— устанавливает связанную с ней переменную в “1”, пока на нее поступает энергия.

Инверсная катушка —⊘— устанавливает связанную с ней переменную в “0”, пока на нее поступает энергия.

Катушка установки —Ⓢ— также устанавливает соответствующую переменную в “1”, но, в отличие от простой катушки, значение сохраняется и при прекращении поступления потока энергии. Для возврата соответствующей переменной в “0” используется катушка сброса —Ⓡ—.

Таймеры. Таймеры обеспечивают задержку по включению или выключению. Таймер начинает счет при наличии разрешающего сигнала на входе и устанавливает выход по прошествии заданного интервала времени.

Рассмотрим работу таймеров на примере одного из блоков – ONDTMR_TENTHS. Реализация работы таймера ONDTMR_TENTHS на языке LD в программной среде CIMPLICITY Machine Edition представлена на рисунках 1.4.

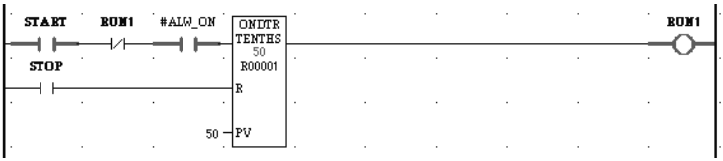


Рис. 1.4. Реализация работы таймера на языке LD
в программной среде CIMPLICITY Machine Edition

Этот таймер накапливает свое значение с шагом 0,1 секунды (TENTHS – десятые).

Вход PV задает время, по достижении которого выход таймера устанавливается в единицу. На этот вход может быть подана переменная или константа. Время задается как количество шагов (приращений) счета. В данном примере заданное время равно 5 секунд (50*0,1сек=5сек)

Вход R является условием сброса. Если этот вход имеет значение = 1, текущее значение и выход счетчика сбрасываются на ноль.

Инверсный контакт RUN1 на входе таймера используется для того, чтобы при достижении заданного значения таймер останавливался.

В контроллерах Series 90 Micro для корректной работы таймеров требуется подключать на вход блока системный контакт #ALW_ON (который всегда активен и при таком подключении не влияет на логику программы). Если используется несколько таймеров, достаточно одного контакта #ALW_ON , подключенного к первому таймеру.

Таймер занимает три слова регистровой памяти (%R00001). В первом слове хранится текущее значение таймера, во втором – задаваемое значение, третье слово является контрольным. При адресации памяти таймера указывается адрес первого слова, а два последующих адреса должны оставаться свободными.

Для подсчета импульсов используются счетчики.

Счетчики. Счетчик начинает счет при наличии разрешающего сигнала на входе и устанавливает выход при достижении заданного значения. Существует два типа счетчиков: счетчик восходящего счета UPCTR и счетчик нисходящего счета DNCTR.

Так же как и таймер, счетчик занимает три слова регистровой памяти %R и требует подключения контакта #ALW_ON.

Рассмотрим работу счетчика на следующем примере. Блок UPCTR производит подсчет импульсов, поступающих с контакта RUN1, то есть подсчитывает, сколько раз этот контакт перейдет в единицу. Состояние RUN1 задается при помощи таймера TMR_TENTHS. При достижении заданного значения

устанавливается выход счетчика (катушка RUN) и счет прекращается (размыкается контакт RUN). Реализация работы счетчика на языке LD в программной среде CIMPLICITY Machine Edition представлена на рисунках 1.5.

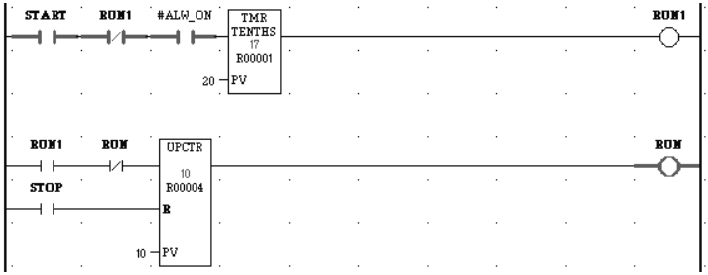


Рис. 1.5. Реализация работы счетчика на языке LDв программной среде CIMPLICITY Machine Edition

5. Смотрим в правом нижнем углу отображается состояние контроллера он находится в режиме Offline

При работе с контроллером мы можем находиться в одном из двух режимов: **online** или **offline**.

В режиме онлайн компьютер, на котором разрабатывается проект, подключен к контроллеру. При этом в проекте отображается статус контроллера, состояние ячеек памяти, обновляется таблица ошибок ПЛК, то есть выполняются те функции, для которых необходим постоянный обмен информации между компьютером и ПЛК.

В режиме offline выполняются команды, не требующие постоянного обмена данных. При этом каждый раз для выполнения команды происходит подключение к контроллеру, а по завершении выполнения – отключение.

В режиме online в строке внизу экрана отображается статус ПЛК:

Логика/Конфигурация EQ – программа/конфигурация в проекте и в контроллере совпадают.

Логика/Конфигурация NE – программы в компьютере и контроллере не совпадают.

Режим работы ПЛК – *Run* или *Stop.*

Команды работы с активной целью вызываются из меню “Цель”. Процедура запуска ПЛК в режим *Run* представлена на рисунке 1.6.

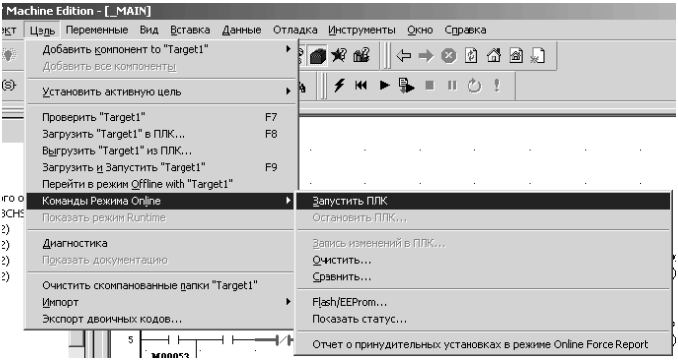


Рис. 1.6. Запуск ПЛК в режим *Run*

Для выполнения программы в ПЛК необходимо загрузить в него логику программы при помощи команды “Загрузить Target 1 в ПЛК”. Перед загрузкой полезно проверить цель на наличие ошибок. Для этого используется пункт меню “Проверить Target 1”. При загрузке программы ПЛК должен находиться в режиме Stop, иначе загрузка прерывается и выдается ошибка. Диалоговое окно пользователя, возникающее при загрузке программы в ПЛК представлено на рисунке 1.7.

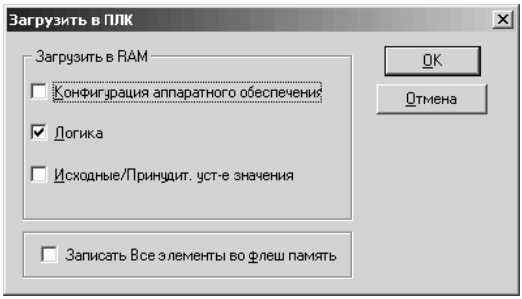


Рис. 1.7.. Диалоговое окно пользователя, возникающее при загрузке программы в ПЛК

При первой загрузке цели в ПЛК необходимо также загрузить конфигурацию.

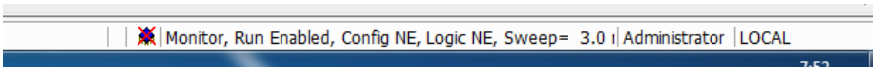
Пуск осуществляется с помощью команды “Запустить ПЛК”.

Останов контроллера производится командой “Остановить ПЛК”. При этом появляется следующее окно, в котором задается вопрос, сохранить ли значения выходов или сбросить их на ноль.

В лабораторных условиях этот выбор не является существенным, однако в случае производственного процесса или действующей установки неправильный выбор может привести к аварии. Например, если дискретный выход связан с положением задвижки и его сброс приведет к ее незапланированному открытию.

6. Вводим контроллер в режим онлайн (соединяемся с контроллером) Нажимаем специальную пиктограмму. В случае отсутствия соединения проверяем включен ли физически контроллер (должна гореть на самом контроллере верхняя лампочка питания pwr).

7. Фиксируем изменение состояния контроллера. Теперь он находится в режиме Monitor



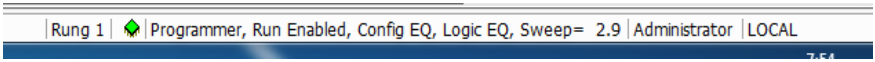
8. Переходим в режим программирования

Нажимаем пиктограмму

9. Загружаем программу в ПЛК. При этом смотрим его состояние (если он запущен (кнопка загрузки и запуска не активна, активна только кнопка стоп – останавливаем его)

Правая пиктограмма обозначает загрузить и запустить контроллер, левая – только загрузить программу

10. В случае удачной загрузки строка состояния должна иметь вид

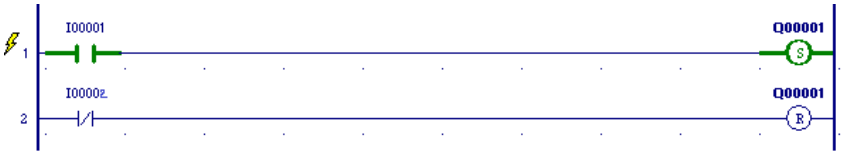


В случае если этого не произошло в отладчики проверьте сообщения об ошибках. Для их детализации необходимо нажать кнопку F4

11. Проверьте исполнения программы. Покрутите соответствующие тумблеры, убедитесь, что программа исполняется в соответствии с заданным алгоритмом. Активные действия подсвечиваются в программе и на панели контроллера соответствующими светодиодами на входах или выходах.

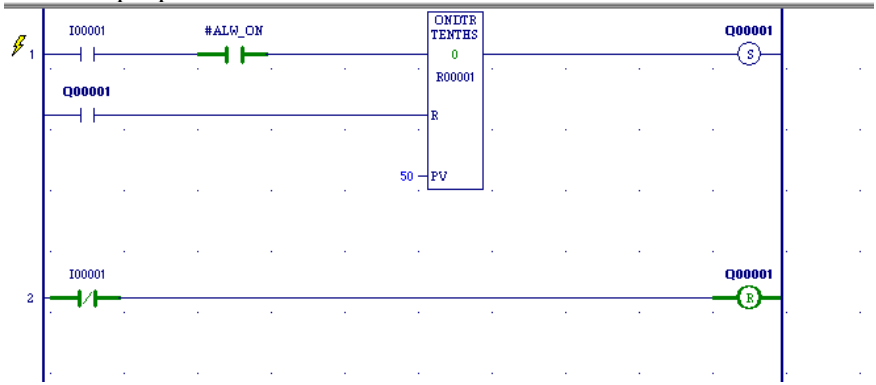
Задание к первой работе – создайте программу по нажатию кнопки включается лампочка. (при этом кнопку определите на 2ой вход ПЛК, а лампочку на 4ый выход)

12. Реализуйте первую строчку программы, указанной выше. Что происходит? Чем она отличается от программы задания 1. Напишите исправленный словесный алгоритм для данной программы. Выполните программу полностью, сделайте выводы.



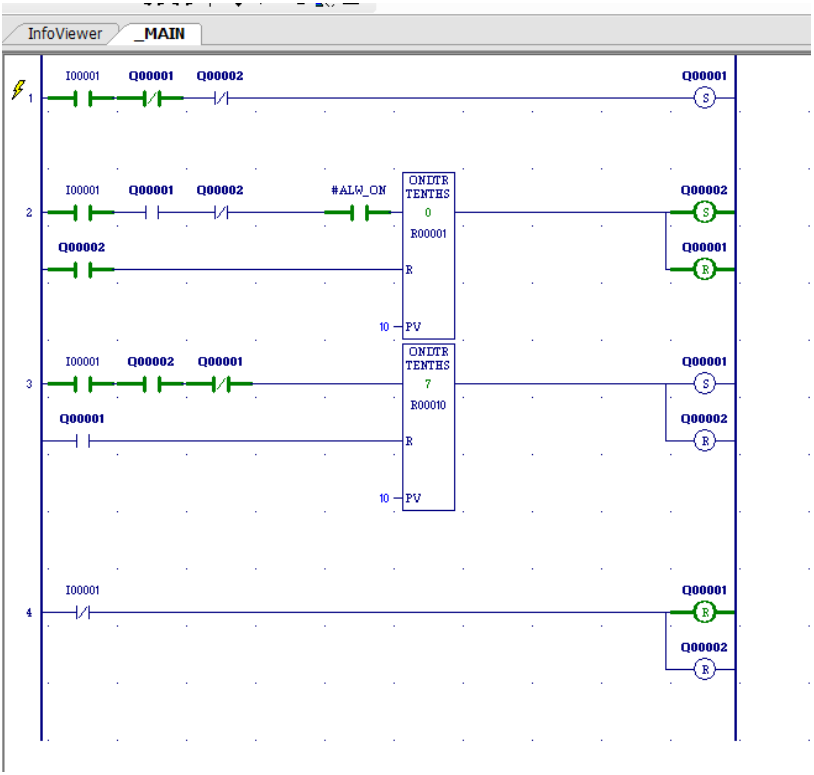
Напишите программу При нажатии кнопки 1 загораются лампочки 1 и 2 одновременно. При нажатии кнопки 2 Л1 и Л2 гаснут одновременно.

13. Выполните программу По нажатию кнопки лампочка загорается через 5 секунд. Найдите объяснение каждому блоку данной программы.



Самостоятельное задание – по нажатию кнопки лампочка 1 загорается через 5 секунд, при выключении кнопки лампочка 2 гаснет через 3 секунды. Используйте те же программные элементы, что и в примере.

14. Напишите программу мигания лампочек, указанную ниже на рисунке.



Варианты заданий:

Вариант 1. Написать программу работы светофора для дневного и ночного режимов. Смена режима производится при помощи ручного переключателя. Дневной режим – обычный цикл светофора, ночной – мигание желтого света.

Вариант 2. Написать программу работы светофора в автоматическом и ручном режиме. Смена режима производится при помощи ручного переключателя. Автоматический режим – обычный цикл светофора. Ручной режим должен обеспечивать задержку после включения красного света, и дальнейший переход к желтому и далее зеленому должен происходить только после подачи импульса со второго ручного переключателя.

Вариант 3. Написать программу работы железнодорожного переезда. При подаче сигнала с первого переключателя на 5 секунд включается двигатель, опускающий шлагбаум. Одновременно включаются сигнальные огни, которые мигают попеременно. При подаче сигнала со второго переключателя на 5 секунд включается двигатель, поднимающий шлагбаум, после чего мигание прекращается. Сигналы подаются в виде импульсов.

Вариант 4. Написать программу световой рекламы в виде стенки, меняющей свой цвет. На стенке смонтированы светодиоды четырех цветов, расположенные таким образом, что цвет стенки меняется за счет последовательного свечения светодиодов одного цвета. Группа светодиодов одного цвета подключается на один дискретный выход. Частота смены цвета должна меняться в зависимости от положения переключателя.

Вариант 5. Написать программу световой рекламы в виде гирлянды лампочек, по которой справа налево "бегут" огни таким образом, что в каждый момент горит каждая вторая лампочка. Скорость должна меняться в зависимости от положения переключателя.

Вариант 6. Написать программу световой рекламы в виде гирлянды лампочек, по которой справа налево "бегут" огни таким образом, что в каждый момент горит каждая третья лампочка. Скорость должна меняться в зависимости от положения переключателя.

Вариант 7. Написать программу световой рекламы вывески магазина по следующей схеме. Буквы названия должны последовательно загораться и гаснуть с последней до первой, после чего пер-вая остается гореть и начинается пробег с последней до второй и так далее.

Вариант 8. Написать программу световой рекламы вывески магазина по следующей схеме. Буквы названия загораются последовательно с первой до последней с небольшим интервалом, после че-го все буквы несколько раз мигают.

Вариант 9. Написать программу удаленного управления освещением в пространственно протяженном помещении (например, в большом зале или длинном коридоре), имеющем два входа. У

каждого входа находится по импульсной кнопке. Программа должна обеспечивать включение освещения во всем помещении при подаче импульса с любой из кнопок. Выключение освещения производится при следующем нажатии любой из кнопок, независимо от того, какой именно кнопкой оно было включено. При этом для двух управляющих кнопок должен использоваться один дискретный вход контроллера.

Вариант 10. Написать программу для системы управления конвертером. Система может работать в ручном и автоматическом режиме. В автоматическом режиме: поворот конвертера осуществляется по управляющему сигналу в течении заданного времени (во время отладки программы рекомендуется сократить время до 5 секунд). Перед поворотом обеспечивается подача сигнализации в течении 5 секунд. По достижении конечной точки при повороте срабатывает прерывистая сигнализация (3 раза по 1 секунде) и происходит процесс конвертирования в течении некоторого времени (во время отладки программы рекомендуется сократить время до 5 секунд). После завершения процесса конвертирования происходит возвращение конвертера в первоначальное положение для разгрузки. В ручном режиме действия по изменению положения конвертера происходят после подачи импульса с ручного переключателя.

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №2

IoT устройства. Основные принципы программирования и конфигурирования.

Packet Tracer — симулятор сети передачи данных, выпускаемый фирмой Cisco Systems. Позволяет делать работоспособные модели сети, настраивать (командами Cisco IOS) маршрутизаторы и коммутаторы, взаимодействовать между несколькими пользователями (через облако). В симуляторе реализованы серии маршрутизаторов Cisco 800, 1800, 1900, 2600, 2800, 2900 и коммутаторов Cisco Catalyst 2950, 2960, 3560, а также межсетевой экран ASA 5505. Беспроводные устройства представлены маршрутизатором Linksys WRT300N, точками доступа и сотовыми вышками. Кроме того, есть серверы DHCP,

HTTP, TFTP, FTP, DNS, AAA, SYSLOG, NTP и EMAIL, рабочие станции, различные модули к компьютерам и маршрутизаторам, IP-фоны, смартфоны, хабы, а также облако, эмулирующее WAN. Объединять сетевые устройства можно с помощью различных типов кабелей, таких как прямые и обратные патч-корды, оптические и коаксиальные кабели, последовательные кабели и телефонные пары.

Успешно позволяет создавать даже сложные макеты сетей, проверять на работоспособность топологии. Однако, стоит заметить, что реализованная функциональность устройств ограничена и не предоставляет всех возможностей реального оборудования. Cisco Packet Tracer доступен бесплатно для участников Программы Сетевой Академии Cisco.

Микроконтроллер (MCU) - это небольшой компьютер, построенный на системе на микросхеме (SoC). Он содержит ядро процессора, память и программируемые периферийные устройства ввода / вывода. Микроконтроллеры предназначены для встроенных приложений или приложений, которые требуют мало ресурсов компьютера. И наоборот, микропроцессоры, используемые в персональных компьютерах, обычно используются для поддержки других приложений общего назначения, которые требуют больше компьютерных ресурсов.

Примерами приложений, в которых используются микроконтроллеры, являются системы управления автомобильными двигателями, имплантируемые медицинские устройства, пульты дистанционного управления, офисные машины, приборы, электроинструменты, игрушки и другие встроенные системы. Микроконтроллеры со смешанным сигналом также распространены, объединяя аналоговые компоненты, необходимые для управления нецифровыми электронными системами.

Packet Tracer поддерживает эмулятор MCU. Пользователь может запрограммировать PT MCU для выполнения задач, аналогичных MCU реального слова. Чтобы упростить процесс, PT MCU может быть запрограммирован на Java и Python.

Порядок выполнения работы

1. Добавляем и подключаем необходимые устройства.

Для подключения устройств заходим в раздел компоненты (Components) в левом нижнем углу экрана. Перетаскиваем на поле компоненты из разделов Boards, Sensors и Actuators MCU Board, LED и Toggle button соответственно. С помощью провода IOT Custom Cable соединяем устройства с микроконтроллером.

Устройство	Порт MCU
Кнопка (Toggle button)	D0
Светодиод (LED)	D1

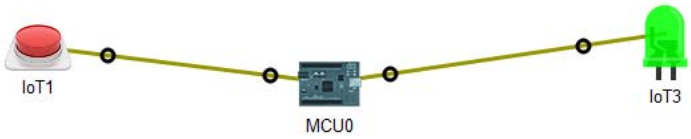


Рис 2.1. Схема работы.

Переходим к программированию микроконтроллера. Для этого двойным кликом мыши щелкаем на контроллер. В появившемся окне свойств переходим не вкладку Программирование (Programming) нажимаем New и загружаем из имеющихся шаблонов программу Blink (Python). Откроется программа main.py (рисунок 2.2)

Рис 2.2. Окно свойств микроконтроллера. Программирование на языке Питон.

Запускаем программу на исполнения кнопкой Run в правом верхнем углу окна свойств микроконтроллера. После запуска программы светодиод должен мигать с периодичностью, заданной в программе. Изменим программный код, который будет активировать мигание светодиода только по нажатию на кнопку.

Добавим в нужные зоны кода следующие строки:

pinMode(0, IN) (данная строчка определяет нулевую клемму контроллера как вход)

a=digitalRead(0); (данная строчка присваеват переменной ф значение, считанное со входа 0)

```
if a==1:
        digitalWrite(1, HIGH);
        sleep(1)
        digitalWrite(1, LOW);
        sleep(0.5)
else:
        digitalWrite(1, LOW);
```

(Данный циклический алгоритм обеспечивает мигание светодиода, если на вход приходит 1 и выключение светодиода, если на вход приходит 0).

Реализуем данный код. В процессе реализации видим, что код является не рабочим. Запустим отладку кода. Выведем значения а для того, чтобы убедиться, в том, что данные значения состояния кнопки приходят на микроконтроллер. Добавим строчку

print(a)

В строке отладчика видим, что при выключенной кнопке с нее на вход микроконтроллера приходит 0, при включенной кнопке 1023, а не 1 как ожидалось, основываясь на коде. Доказательство выше обозначенного также можно найти на вкладке описание элемента «кнопка» (элемент отправляет LOW (0) или HIGH (1023) на вход устройств).

```
1  from gpio import *
2  from time import *
3
4  def main():
5      pinMode(1, OUT)
6      pinMode(0, IN)
7      print("Blinking")
8      while True:
9          a=digitalRead(0);
10         print(a)
11         if a==1023:
12             digitalWrite(1, HIGH);
13             sleep(1)
14             digitalWrite(1, LOW);
15             sleep(0.5)
16         else:
17             digitalWrite(1, LOW);
18
19
20 if __name__ == "__main__":
21     main()
```

Рис. 2.3. Фрагмент кода

Также можно поменять условия, находящиеся под операторами If и Else, местами. От этого смысл кода и действия и логика, выполняемой программы не изменятся.

Варианты заданий:

Собрать и запрограммировать схему управления IoT устройствами, согласно данным, указанным в задании

Таблица 2.2.

Номер варианта	Количество подключенных устройств, не менее	Обязательное устройство	Количество, функция для каждого устройства, выполняемых программно
1.	3	Кофеварка	2
2.	2	Лампа	3
3.	3	Ветрогенератор	2
4.	2	Солнечная панель	3
5.	3	Вентилятор	2
6.	2	Кофеварка	3
7.	3	Лампа	2
8.	2	Ветрогенератор	3
9.	3	Солнечная панель	2
10.	2	Вентилятор	3

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА №3

IoT устройства. Конфигурирование WEB-интерфейса для мониторинга состояния устройств.

Packet Tracer — симулятор сети передачи данных, выпускаемый фирмой Cisco Systems. Позволяет делать работоспособные модели сети, настраивать (командами Cisco IOS) маршрутизаторы и коммутаторы, взаимодействовать между несколькими пользователями (через облако). В симуляторе реализованы серии маршрутизаторов Cisco 800, 1800, 1900, 2600, 2800, 2900 и коммутаторов Cisco Catalyst 2950, 2960, 3560, а также межсетевой экран ASA 5505. Беспроводные устройства представлены маршрутизатором Linksys WRT300N, точками доступа и сотовыми вышками. Кроме того, есть серверы DHCP, HTTP, TFTP, FTP, DNS, AAA, SYSLOG, NTP и EMAIL, рабочие станции, различные модули к компьютерам и маршрутизаторам, IP-фоны, смартфоны, хабы, а также облако, эмулирующее WAN. Объединять сетевые устройства можно с помощью различных типов кабелей, таких как прямые и обратные патч-корды, оптические и коаксиальные кабели, последовательные кабели и телефонные пары.

Успешно позволяет создавать даже сложные макеты сетей, проверять на работоспособность топологии. Однако, стоит заметить, что реализованная функциональность устройств ограничена и не предоставляет всех возможностей реального оборудования. Cisco Packet Tracer доступен бесплатно для участников Программы Сетевой Академии Cisco.

Микроконтроллер (MCU) - это небольшой компьютер, построенный на системе на микросхеме (SoC). Он содержит ядро процессора, память и программируемые периферийные устройства ввода / вывода. Микроконтроллеры предназначены для встроенных приложений или приложений, которые требуют мало ресурсов компьютера. И наоборот, микропроцессоры, используемые в персональных компьютерах, обычно используются для поддержки других приложений общего назначения, которые требуют больше компьютерных ресурсов.

Примерами приложений, в которых используются микроконтроллеры, являются системы управления автомобильными двигателями, имплантируемые медицинские устройства, пульты дистанционного управления, офисные машины, приборы, электроинструменты, игрушки и другие встроенные системы. Микроконтроллеры со смешанным сигналом также распространены, объединяя аналоговые компоненты, необходимые для управления нецифровыми электронными системами.

Packet Tracer поддерживает эмулятор MCU. Пользователь может запрограммировать PT MCU для выполнения задач, аналогичных MCU реального слова. Чтобы упростить процесс, PT MCU может быть запрограммирован на Java и Python.

Порядок выполнения работы

Для выполнения работы поместите на поле вентилятор (Ceiling Fun) и 2 кнопки (toggle button)

Устройство	Порт MCU
Кнопка 1 (Toggle button)	D0
Кнопка 2 (Toggle button)	D1
Вентилятор (Ceiling Fun)	D2

Основываясь на теоретических сведениях расчетно-графической работы №2 запрограммируйте микроконтроллер таким образом, чтобы одна кнопка управляла включением и выключением вентилятора, а вторая кнопка изменяла скорость ого вращения.

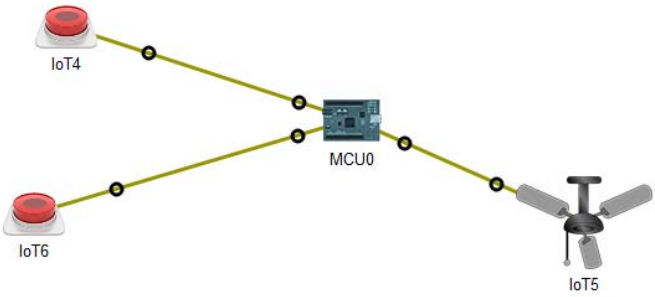


Рис 3.1 Схема соединения устройств.

Для управления вентилятором через Web интерфейс необходимо подключить устройства (используя кабель «витая пара») и назначить им IP адреса, согласно данным таблицы

Устройство	IP адрес
Коммутатор (PT-Switch)	10.0.0.5
ПК (PC)	10.0.0.4/24
Сервер (Server)	10.0.0.2/24
Вентилятор (Ceiling Fun)	10.0.0.3/24

Для задания IP адреса в каждом устройстве открываем таблицу свойств, переходим на вкладку «Конфигурация» >> FastEthernet0 в разделе Gateway/DNS IPv4 выбираем Static и в поля вбиваем IP адрес и маску. В случае правильно выполненных условий пи наведении мыши на устройство в специальном поле можно увидеть IP адрес.

Далее на уровне сервера в окне свойств необходимо зайти на вкладку сервисы (Services) и включить IoT сервис.

На уровне вентилятора необходимо зайти на вкладку конфигурации (Config) и прописать настройки сервера для подключения IoT устройств.

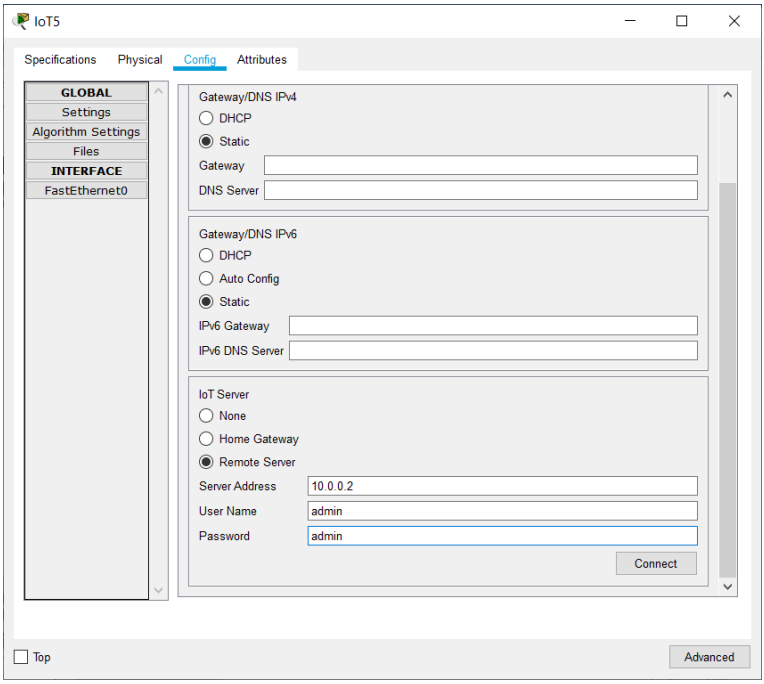


Рис. 3.2. Окно с настройками IoT сервера

После установки необходимых настроек пробуем подсоединиться к серверу. Нажимаем Connect (Подключение). Видим, что подключение не проходит. Это связано с тем, что на IoT сервере не зарегистрирован аккаунт с пользователем admin. Зарегистрируем его. Для этого зайдем через ПК в браузер. В строке браузера наберем адрес сервера. На Web странице увидим надпись «Don't have an IoE account? Sign up now» Нажмем Зарегистрировать аккаунт сейчас (Sign up now) и введем настройки аккаунта. Теперь снова попробуем подключиться. Подключение должно проходить в этом случае.

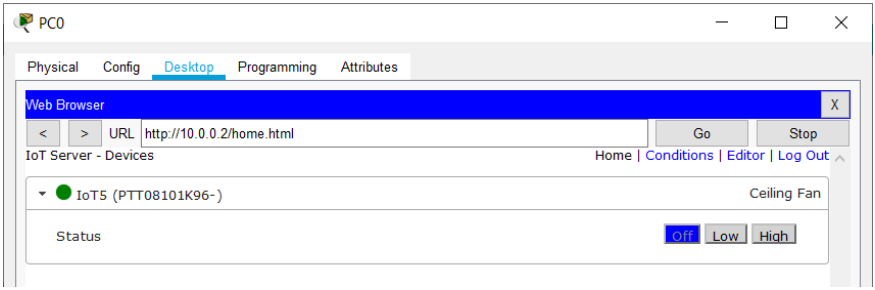


Рис. 3.3. Web страница для управления вентилятором.

Теперь попробуем поменять разметку страницы. Изменим надпись на кнопках управления. Для этого заходим в свойства вентилятора. Переходим на вкладку дополнительно. Далее программирование. Запускаем код на Питоне main.py. Находим то место в коде, которое отвечает за наименование кнопок. Изменяем его. Проверяем результат. Вводим еще насколько элементов для закрепления полученных навыков.

Задание: для схемы, собранной в ГРЗ №2 на базе имеющейся WEB- страницы для каждого устройства создать WEB-страницу с добавлением не менее чем 10 элементов.

РЕКОМЕНДУЕМЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Основная литература

1. *Храменков, В.Г.* Автоматизация управления технологическими процессами бурения нефтегазовых скважин : учебное пособие [Электронный ресурс] : учеб. пособие — Электрон. дан. — Томск : ТПУ, 2012. — 416 с.

https://e.lanbook.com/book/10326

2. *Храменков, В.Г.* Автоматизация производственных процессов: учебник [Электронный ресурс] : учеб. — Электрон. дан. — Томск : ТПУ, 2011. — 343 с.

https://e.lanbook.com/book/10325

3. *Задорожная, Н.М.* Характеристики типовых звеньев систем автоматического регулирования [Электронный ресурс] : учеб. пособие / Н.М. Задорожная, В.А. Дудоладов. — Электрон. дан. — Москва : МГТУ им. Н.Э. Баумана, 2014. — 37 с.

https://e.lanbook.com/book/62016

4. *Медведев, А.Е.* Автоматизация производственных процессов : учеб. Пособие [Электронный ресурс]: учеб. пособие / А.Е. Медведев, А.В. Чупин. — Электрон. дан. — Кемерово : КузГТУ имени Т.Ф. Горбачева, 2009. — 325 с.

https://e.lanbook.com/book/6606

Дополнительная литература

1. *Смирнов, Ю.А.* Технические средства автоматизации и управления [Электронный ресурс] : учеб. пособие — Электрон. дан. — Санкт-Петербург: Лань, 2017. — 456 с.

https://e.lanbook.com/book/91063

2. *Гайдук, А.Р.* Теория автоматического управления в примерах и задачах с решениями в MATLAB [Электронный ресурс] : учеб. пособие / А.Р. Гайдук, В.Е. Беляев, Т.А. Пьявченко. — Электрон. дан. — Санкт-Петербург: Лань, 2017. — 464 с.

https://e.lanbook.com/book/90161

СОДЕРЖАНИЕ