ТЕХНОЛАБ

ОБРАЗОВАТЕЛЬНЫЙ РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(ПРОФЕССИОНАЛЬНЫЙ УРОВЕНЬ)

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

(ПРОФЕССИОНАЛЬНЫЙ УРОВЕНЬ)

К. В. Ермишин
Д. Н. Каргин
А. А. Нагорный
А. О. Панфилов

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

ОБРАЗОВАТЕЛЬНЫЙ
РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(ПРОФЕССИОНАЛЬНЫЙ УРОВЕНЬ)
ОТ 14 ЛЕТ

Учебно-методическое пособие

ЭКЗАМЕН ТЕХНОЛАБ
Издательство ЭКЗАМЕН®

МОСКВА
2014

УДК 372.8:004
ББК 32.816
Е73

Ермишин К. В.

Е73 Методические рекомендации для преподавателя: образовательный робототехнический модуль (профессиональный уровень): от 14 лет / К. В. Ермишин, Д. Н. Каргин, А. А. Нагорный, А. О. Панфилов. — М. : Издательство «Экзамен», 2014. — 256 с.

ISBN 978-5-377-07625-4

Данное пособие предназначено для применения совместно с образовательным робототехническим модулем «Профессиональный уровень». В пособии описываются возможности робототехнического модуля и области его применения. Пособие содержит информацию о назначении робототехнического набора и описание работ по проектированию роботов и робототехнических устройств, которые можно провести совместно с учащимися среднего школьного возраста. Пособие раскрывает базовые теоретические основы функционирования роботов, а также содержит справочную информацию по программированию систем управления роботов и робототехнических устройств, основы обработки информации и показаний датчиков. Особое внимание уделяется описанию основ функционирования основных устройств и узлов робототехнических устройств, а также алгоритмической части систем управления различных роботов. Применение образовательного робототехнического модуля «Профессиональный уровень» позволяет привить учащимся навыки и основы профессионального подхода к решению технически сложных проблем, проведения системного анализа, выработки концепции технического решения и реализации проекта. Рассматриваемые в данном пособии проекты затрагивают решение большинства наиболее часто встречающихся задач в области робототехники, начиная от конструирования роботов и различных механизмов, вплоть до разработки систем управления с использованием различных устройств и программ, реализованных с применением различных сред разработки и программирования. Данное пособие носит рекомендательный характер и тем самым не ограничивает возможности применения робототехнических наборов в образовательной и учебной деятельности, но в свою очередь демонстрирует наиболее яркие примеры проектов и работ, которые возможно реализовать благодаря использованию образовательного робототехнического модуля «Профессиональный уровень».

УДК 372.8:004
ББК 32.816

Подписано в печать с диапозитивов 15.10.2013.
Формат 60x90/8. Гарнитура «Calibri». Бумага офсетная.
Усл. печ. л. 32. Тираж 1000 экз. Заказ №

ISBN 978-5-377-07625-4

Содержание

Введение

Мировые тенденции развития инженерного образования свидетельствуют о глобальном внедрении информационных технологий в образовательный процесс. Робототехника является весьма перспективной областью для применения образовательных методик в процессе обучения за счет объединения в себе различных инженерных и естественнонаучных дисциплин. В результате такого подхода наблюдается рост эффективности восприятия информации учащимися за счет подкрепления изучаемых теоретических материалов экспериментом в междисциплинарной области.

Данное учебное пособие представляет собой методические рекомендации, раскрывающие возможности и особенности применения образовательного робототехнического модуля «Профессиональный уровень». Образовательный робототехнический модуль «Профессиональный уровень» предназначен для углубленного изучения основ проектирования роботов и робототехнических устройств на примере эксперимента, который можно без особого труда выполнить в рамках индивидуальных или групповых занятий.

Образовательный модуль позволяет конструировать модели роботов, робототехнических устройств и производственных механизмов различной сложности. Каждая из моделей предназначена для проведения ряда лабораторных занятий по изучению различных приемов программирования и теоретических аспектов проектирования. Ряд моделей может применяться в различных робототехнических соревнованиях, таких как – соревнования в стиле «сумо», гонки роботов вдоль линии и др.

Предлагаемые лабораторные работы и эксперименты, проводимые в их рамках, основываются на моделях роботов, которые можно сконструировать на базе робототехнического конструктора Bioloid и дополнительных компонентов, производимых корейской компанией ROBOTIS. Наборы данной серии отличает многообразие возможностей, позволяющих реализовать всевозможные задумки начинающих исследователей.

Наличие программируемого блока управления и различных датчиков позволяет сделать полностью автономные модели роботов, которыми также можно управлять вручную с помощью пульта дистанционного управления и модулей беспроводной связи. Широкий спектр доступных компонент и возможностей позволяет пользователю на практике ознакомиться с принципом функционирования различных приводов, контроллеров и сенсорных устройств, а также разработать для них собственную систему управления.

Использование решений из области робототехники в рамках образовательного процесса позволяет формировать технологическую и проектную культуру учащихся, которые также не останутся равнодушными к столь увлекательному образовательному процессу.

Основы изучения среды программирования RoboPlus

Основы изучения среды программирования RoboPlus

ЭКЗАМЕН ТЕХНОЛАБ

Основы изучения среды программирования RoboPlus

Разработка и программирование систем управления – это неотъемлемая часть процесса проектирования робототехнических систем. В настоящее время большинство устройств и механизмов управляются благодаря микропроцессорным устройствам, выполняющим какую-либо программу. Программирование подобных устройств осуществляется в специальных средах для разработки. Как правило, крупные производители микроэлектроники и контроллеров выпускают собственные среды разработки или поддерживают наиболее популярные среди разработчиков программные пакеты.

В нашем случае для программирования робототехнического модуля «Базовый уровень» применяется среда разработки RoboPlus.

RoboPlus CD

Установка RoboPlus

Установка RoboPlus осуществляется с диска, входящего в состав модуля.

Вставьте диск в дисковод и откройте корневую папку диска.

Выберите и запустите файл Setup.exe и запустите процесс установки.

Выберите в процессе установки английский язык (English).

Для русификации RoboPlus скопируйте файлы из папки «Русификатор» в папку установки программы.

В случае удачной установки на рабочем столе вашего ПК появится ярлык RoboPlus.

Запуск RoboPlus

Произведите запуск RoboPlus, использовав ярлык на рабочем столе. На экране появится окно содержащее информацию о программах входящих в состав среды разработки. Обратите внимание, что от версии ПО или года выпуска робототехнического модуля содержание окна может немного отличаться, но это ни каким образом не сказывается на работоспособности набора и возможностях среды разработки.

Вкладки стартового окна содержат информацию о наборах, которые можно программировать в RoboPlus. Среда разработки RoboPlus поддерживает все наборы компании ROBOTIS, на базе которых разработаны образовательные робототехнические модули «Предварительный уровень», «Начальный уровень», «Базовый уровень», «Профессиональный уровень», «Исследовательский уровень».

Состав RoboPlus

В состав среды разработки RoboPlus входят специальные программы, предназначенные для настройки различных устройств, входящих в состав робота; программирования и управления роботами.

RoboPlus Task

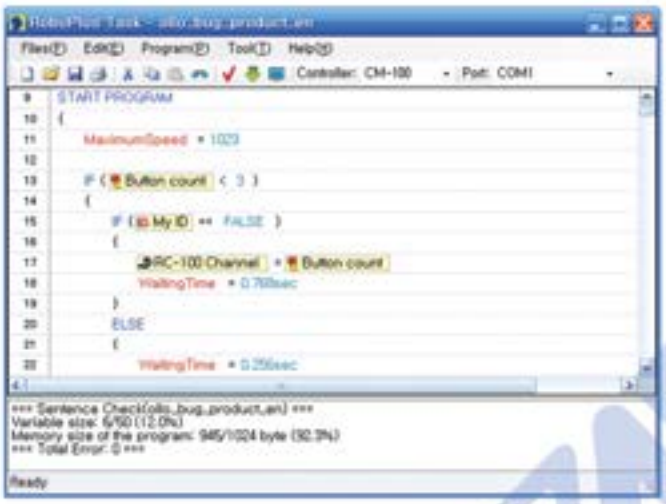
RoboPlus Manager

RoboPlus Motion

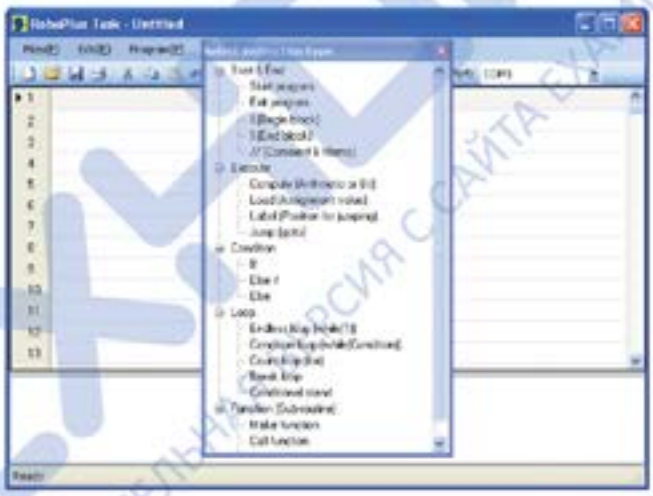
RoboPlus Terminal

Dynamixel Wizard

RoboPlus Task – программная среда для написания и редактирования управляющих программ. Данная программа является основным инструментом для разработки программ для образовательных робототехнических модулей.



Программирование в RoboPlus Task осуществляется с помощью специализированного языка, подобного языку программирования С. Для удобства пользователя в RoboPlus в виде графических блоков реализованы базовые возможности набора, такие так: таймеры, блоки обработки данных с датчиков, блоки передачи данных между устройствами и т.п.



В языке среды RoboPlus все служебные слова, программные блоки, команды и комментарии располагаются в строках. Для того чтобы активировать строку, по ней нужно нажать дважды. После нажатия появится диалоговое окно, позволяющее выбрать различные команды языка.

После выбора определенного набора команд необходимо заполнить пустующие элементы или условия выполнения команд. В пустующие места строк устанавливаются необходимые программируемые блоки. Программируемые блоки выбираются в зависимости от типа программируемого контроллера и числа подключенных внешних устройств. Перечень блоков определяется автоматически и обновляется при подключении новых устройств.

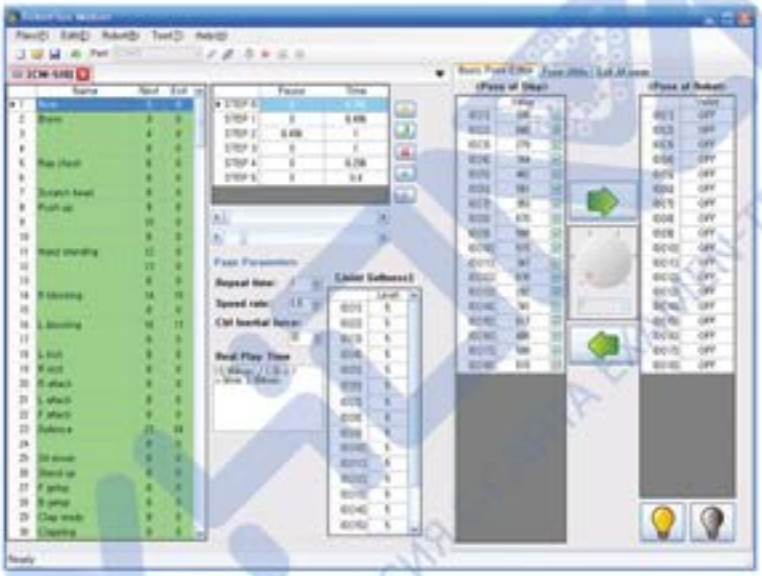
Весь процесс написания программы в RoboPlus Task сводится к дальнейшей загрузке полученных результатов в программируемый контроллер. Для того чтобы компьютер определил тип программируемого контроллера и порт, к которому он подключен, необходимо воспользоваться функцией автоматического поиска.

После того как программа определит тип программируемого контроллера, можно произвести компиляцию программы, тем самым проверить ее на наличие ошибок и подготовить к загрузке в программируемый контроллер.

Полученную программу можно загрузить в контроллер робота, и она запустится сразу же после подачи на него питания.

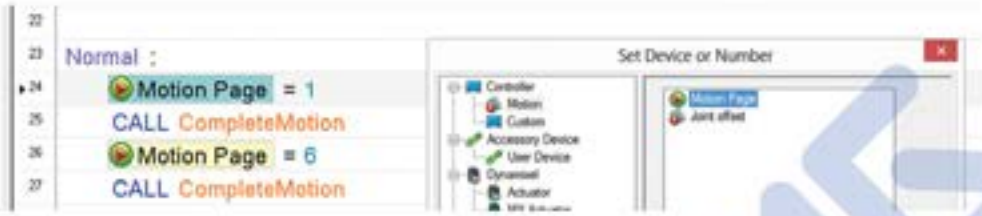
RoboPlus Manager – программа для настройки оборудования, входящего в состав робототехнических конструкторов ROBOTIS. С помощью данной программы RoboPlus обновляет собственные файлы и производит тестирование оборудования, подключенного к компьютеру в данный момент при помощи контроллера или специализированных переходников. Благодаря использованию RoboPlus Manager возможно изменять параметры контроллера, сервоприводов, производить настройку коммуникационных устройств и т.п.

RoboPlus Motion – среда программирования сложных движений робота. Благодаря RoboPlus Motion можно запрограммировать различные действия робота, а после использовать их в основной программе. Зачастую в процессе движения робота участвует множество различных приводов и задать их скорости вращения и углы поворотов вслепую крайне затруднительно.



RoboPlus Motion позволяет промоделировать движение в процессе написания управляющей программы. В специальном окне RoboPlus Motion отображаются все привода, подключенные в данный момент к роботу. Пользователь в режиме реального времени может задать для каждого из приводов скорость и угол поворота, а после запустить программу и увидеть результат ее работы на реальном роботе. Таким образом, можно разработать программу, реализующую сложное движение робота.

Для того чтобы загрузить файл движений робота в память контроллера, воспользуйтесь меню Download Motion вкладки Robot. Очень важно помнить о том, что файл движений (Motion-файл) должен загружаться в память робота до загрузки файла управляющей программы (Task-файл). Нарушение данной последовательности может привести к сбою процесса компиляции основной программы.



Для того чтобы в основной программе использовать заранее разработанное движение, необходимо воспользоваться программируемым блоком Motion Page. В параметрах данного блока следует установить номер строки Motion-файла, отвечающей за необходимое движение.

RoboPlus Terminal – программа, предназначенная для получения и отправки данных посредством терминала операционной системы компьютера. Применяется для отладки управляющих алгоритмов, например для вывода на экран показаний датчиков и т.п., т.е. для отображения той информации, к которой пользователь обычно не имеет доступа в процессе выполнения программы.

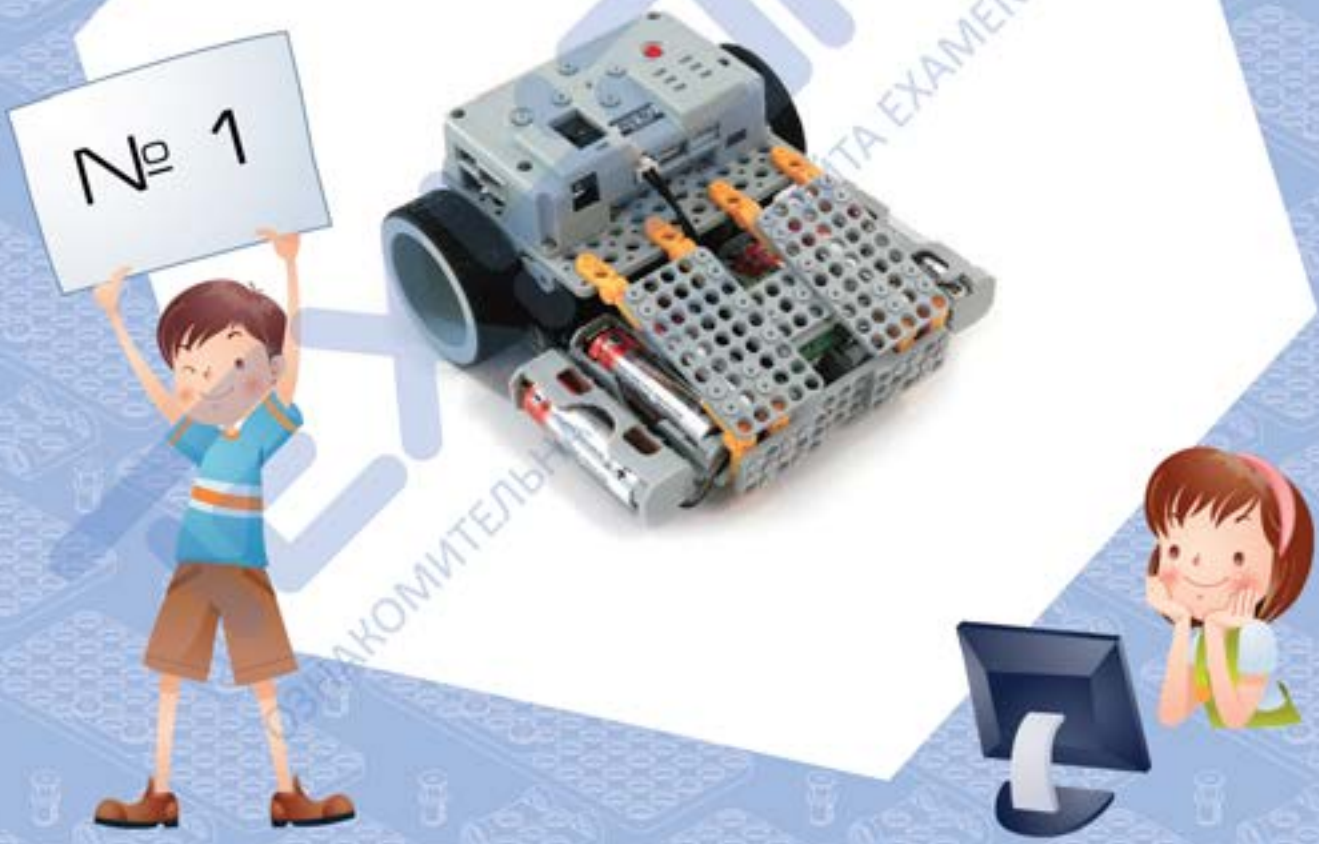
Dynamixel Wizard – программа, предназначенная для настройки и калибровки сервоприводов Dynamixel. С помощью данной программы для каждого из приводов можно задать ограничения скоростей вращения и углов поворота, а также получить код ошибки, препятствующей работе устройства.

Среда разработки RoboPlus содержит в себе все необходимые инструменты для программирования робототехнических наборов на базе конструкторов фирмы ROBOTIS. С ее помощью можно программировать модели на базе наборов серии OLLO, Bioloid, а также отдельные устройства, например сервопривода Dynamixel и модули беспроводной связи.

Лабораторная работа № 1

ЭКЗАМЕН ТЕХНОЛАБ

Управление движением робота с помощью кнопок



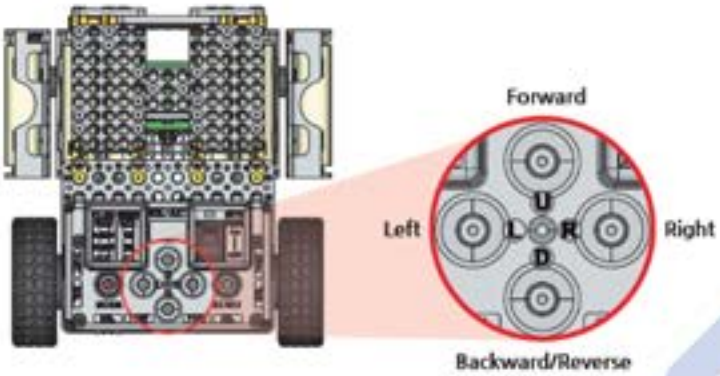
Лабораторная работа № 1
«Управление движением робота с помощью кнопок»

Управление роботами и робототехническими системами достаточно сложный процесс, в зависимости от сложности различают: автоматические системы, полуавтоматические системы и системы ручного управления. Системы ручного управления – системы управления, в которых управление роботом осуществляется человеком, задающим управляющие команды с помощью какого-либо пульта управления.

Существует множество способов задания команд с помощью кнопок управления. Самый простой метод – это выполнение подпрограммы или программы целиком при нажатии на определенную кнопку. Такой метод управления применим при выполнении кратковременных операций, в случае если во время работы нет вероятности возникновения какой-либо ситуации, препятствующей работе робота.

Наиболее часто встречаются ситуации, при которых нажатию кнопки соответствует какое-либо кратковременное действие, например движение робота вперед на небольшое расстояние или включение сигнального светодиода. В этом случае за время выполнения кратковременной задачи мала вероятность того, что может возникнуть ситуация, препятствующая работе робота. К тому же человек-оператор, управляющий движением робота, может скорректировать работу, задав новую команду. Такой метод ручного управления предпочтителен, но требует постоянного внимания со стороны человека-оператора.

Для того чтобы облегчить работу человеку-оператору, очень часто программируются последовательности действий, например: оператор нажимает последовательно на две кнопки и робот выполняет последовательно две команды.

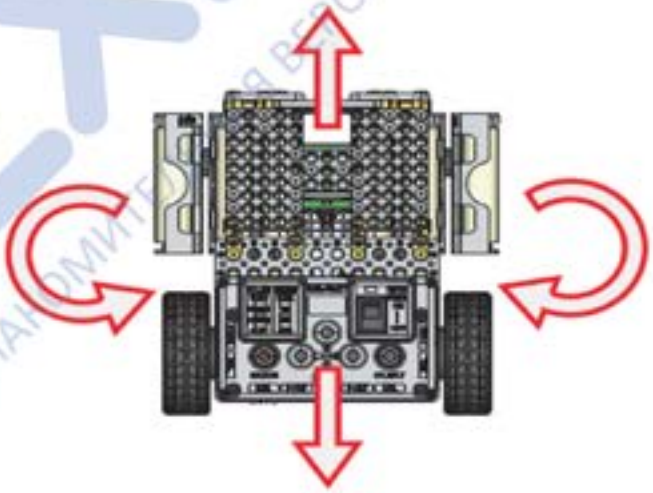


В рамках данной лабораторной работы предлагается сконструировать робота, управляемого с помощью команд, задаваемых кнопками. В качестве устройства, задающего команды, используется панель программируемого контроллера СМ-530. На панели контроллера СМ-530 располагается четыре кнопки управления – U (Up/ Forward), L (Left), R (Right), D (Down/ Backward). Несмотря на то что данные кнопки названы в соответствии с наиболее часто выполняемой функцией, пользователь всегда может присвоить каждой из них определенную задачу. Как это сделать предлагается рассмотреть в данной лабораторной работе.

Часть № 1. Ручное управление мобильным роботом

с помощью кнопок программируемого контроллера СМ-530

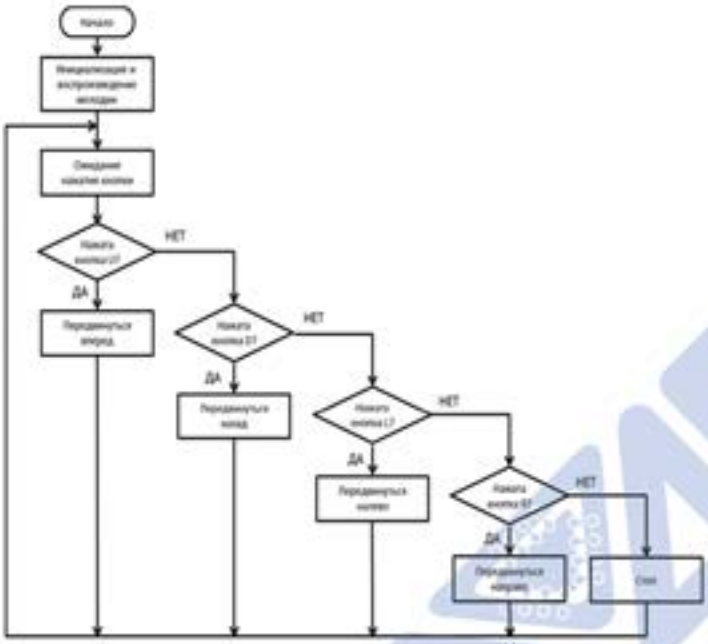
Рассмотрим простейший пример управления мобильным роботом с помощью четырех кнопок на панели контроллера СМ-530. Каждой кнопке контроллера присвоим функцию, выполняющую определенное действие:



Кнопка	Функция	Действие
U	Forward	Воспроизведение мелодии и движение вперед в течение 1 секунды
D	Reverse	Воспроизведение мелодии и движение назад в течение 1 секунды
L	Pivot_left	Воспроизведение мелодии и поворот налево
R	Pivot_right	Воспроизведение мелодии и поворот направо
---	Stop	Остановка робота

С помощью таких простейших движений мобильный робот может выполнить почти произвольный маневр, например объезд препятствия, следование по лабиринту и многое другое. Причем движения робота могут выполняться как поэтапно – после нажатия на соответствующую клавишу, так и непрерывно – по заданной в программе последовательности действий.

Рассмотрим процесс разработки управляющей программы, обрабатывающей нажатия кнопок на панели контроллера СМ-530.



В целом рабочий алгоритм программы можно изобразить в виде блок-схемы, описывающей последовательность выполнения программы.

Сразу же после запуска робота осуществляется инициализация программы управляющего контроллера. В процессе инициализации осуществляется проверка правильности сборки модели робота и присваиваются значения различных таймеров и скоростей движения робота.

После завершения процедуры инициализации программируемый контроллер ожидает нажатия одной из кнопок управления, после чего выполняет запрограммированное действие. Данный процесс повторяется в бесконечном цикле до тех пор, пока программируемый контроллер не будет выключен.

Управляющая программа робота начинается с присвоения значений скоростей движения и временных диапазонов работы таймеров.

```
1:  START PROGRAM
2:  {
3:      // L/R wheel speed (0 ~ 1023)
4:      // Modify speed until robot run straightly in "Straightness_check_mode"
5:      l_wheel_speed  = 400
6:      r_wheel_speed  = 400
7:      forward_time   = 1.000sec
8:      pivot_time     = 0.410sec
9:      standby_time   = 0.600sec
```

Значения скоростей правого и левого колеса задаются для режима прямолинейного движения, который может быть вызван сразу же после запуска программируемого контроллера.

```
11:   // Press START button then press 'D' button until you hear melody to enter "Assembly check mode"
12:   IF ( Button == D )
13:       JUMP Assembly_check_mode
14:
15:   // check Dynamixel's ID and mode
16:   CALL check_ID_mode
17:
18:   // Press START button then press 'U' button until you hear melody to enter "Straightness check mode"
19:   IF ( Button == U )
20:       JUMP Straightness_check_mode
```

В режиме прямолинейного движения мобильный робот едет заданное время со скоростью, определяемой значениями, установленными в начале программы. Данная подпрограмма является тестовой, с помощью которой можно проверить правильность сборки и функционирования робота.

```
73: Straightness_check_mode :
74:     CALL buzzer_straight_mode
75:     Timer = 0.640sec
76:     WAIT WHILE ( Timer > 0.000sec )
77:     ENDLESS LOOP
78:     {
79:         ID[1]: Moving speed = CCW:0 + l_wheel_speed
80:         ID[2]: Moving speed = CW:0 + r_wheel_speed
81:     }
```

Для того чтобы перейти к режиму ручного управления роботом с помощью кнопок на панели контроллера CM-530, необходимо сначала войти в режим проверки правильности сборки робота, нажав на кнопку D.

В этом случае выполнение программы переходит на этап проверки правильности сборки робота и корректности его предварительной настройки.

```
63: Assembly_check_mode :
64:     CALL init_assembly_check
65:
66:     ENDLESS LOOP
67:     {
68:         CALL assembly_check
69:         IF ( assembly_check == TRUE )
70:             JUMP Start_turn
71:     }
```

Проверка качества сборки робота начинается с инициализации максимального и минимального значения ID – номера каждого из приводов в отдельности. Поскольку в модели робота используется всего два привода, максимальное и минимальное значения ID ограничены следующим образом:

```
141:    ID  = 0
142:    ID_MIN  = 1
143:    ID_MAX  = 2
144:    assembly_check  = FALSE
```

Далее последовательно анализируются все возможные ошибки, о которых может сигнализировать сервомодуль Dynamixel, в частности ошибки неправильной инициализации и превышения рабочего момента.

Примечание: для того чтобы проверить алгоритм программы робота или разработать какую-либо управляющую программу не обязательно проводить проверку правильности сборки робота и его тестовые испытания. Данные подпрограммы описываются с целью наглядной демонстрации важности первичной диагностики технически сложного оборудования с целью предотвращения возможных поломок и неисправностей.

После успешного завершения процедуры инициализации программы и проверки правильности сборки робота, выполнение программы переходит к метке Start_turn.

```
22: Start_turn :
23:     Buzzer time  = Play Melody
24:     Buzzer index  = Melody0
25:
26:     ENDLESS LOOP
27:     {
28:         WAIT WHILE ( Button  ==  -- )
```

При старте рабочей программы воспроизводится базовая мелодия, и контроллер переходит в бесконечный цикл ожидания нажатия какой-либо кнопки.

В зависимости от нажатия на одну из кнопок вызывается соответствующая функция, определяющая один из маневров робота, например: при нажатии на кнопку U вызывается функция forward, определяющая прямолинейное движение робота в течение заданного времени.

```
30:     IF ( Button == U )
31:     {
32:         CALL standby
33:         Buzzer time = Play Melody
34:         Buzzer index = Melody0
35:         CALL forward
36:     }
37:
38:     ELSE IF ( Button == D )
39:     {
40:         CALL standby
41:         Buzzer time = Play Melody
42:         Buzzer index = Melody4
43:         CALL reverse
44:     }
```

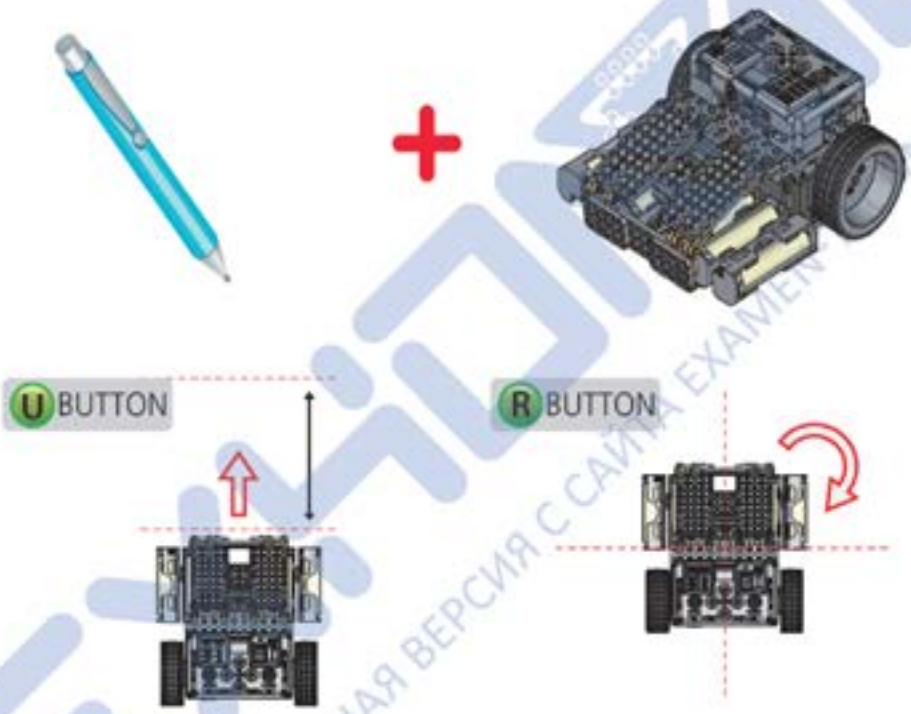
Каждая из функций, задающих движения робота, описывается отдельно. Рассмотрим состав подобной функции на примере функции reverse.

```
93: FUNCTION reverse
94: {
95:     ID[1]: Moving speed = CW:0 + l_wheel_speed
96:     ID[2]: Moving speed = CCW:0 + r_wheel_speed
97:     High-resolution Timer = forward_time
98:     CALL precise_timer_standby
99:     CALL stop
100: }
```

Движение робота задается за счет вращения каждого из приводов с определенной скоростью и в течение заданного времени. С помощью базовых настроек в меню управления задается направление вращения каждого из приводов, скорость движения задается с помощью переменных, указанных в начале программы. Аналогичным образом описываются функции, задающие остальные движения робота.

Воспользовавшись данными рекомендациями, разработайте управляющую программу робота согласно приведенному выше алгоритму. Во время проведения пробных испытаний робота обратите внимание на зависимость движения робота от базовых значений, заданных при инициализации программы:

```
1: START PROGRAM
2: {
3:       // L/R wheel speed (0 ~ 1023)
4:       // Modify speed until robot run straightly in "Straightness_check_mode"
5:       l_wheel_speed  = 400
6:       r_wheel_speed  = 400
7:       forward_time  = 1.000sec
8:       pivot_time  = 0.410sec
9:       standby_time  = 0.600sec
```



Для того чтобы оценка перемещений робота производилась с максимальной точностью, на бампере робота можно закрепить карандаш, который будет очерчивать траекторию движения робота.

Постарайтесь добиться того, чтобы при определенных значениях скоростей вращения колес и значений таймеров, соответствующих движению вперед и поворотам направо и налево, робот перемещался на расстояние кратное собственной длине и поворачивался на угол 90 градусов.

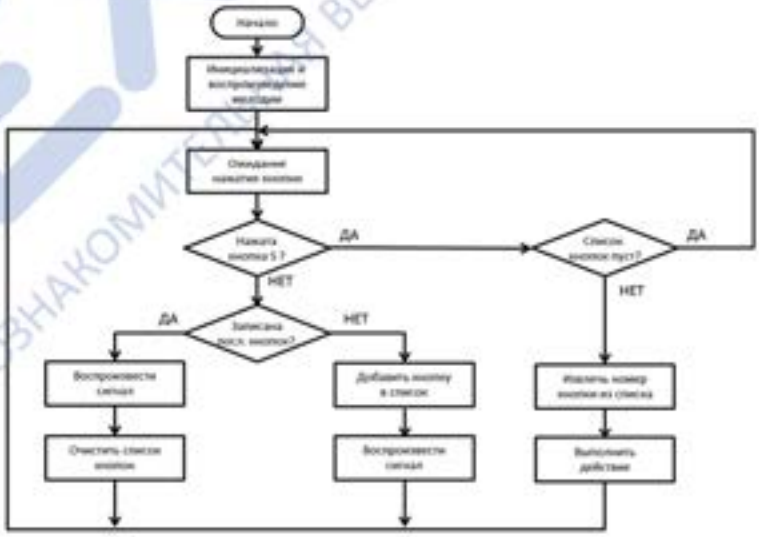
В случае если удастся добиться такой точности движения, мобильный робот будет обладать достаточно высокой маневренностью при собственных габаритах.

Часть № 2. Программирование последовательности действий робота с помощью кнопок контроллера СМ-530

Зачастую алгоритм работы робота сводится к повторению ряда определенных операций. Например, если грузы на конвейере расположены в определенной последовательности, то операция их сортировки сводится к цикличному повторению однотипных операций с каждым из объектов.

В этом случае ручное управление роботом силами человека-оператора слишком утомительный процесс, который следует автоматизировать максимально возможным образом. Если весь цикл сортировочных операций описать в виде последовательности команд, то робот сможет их выполнять автоматически, в этом случае на человека-оператора можно будет возложить обязанности по надзору за процессом выполнения работ.

По аналогии с предыдущей частью лабораторной работы опишем процесс программирования последовательности действий робота с помощью кнопок на панели контроллера СМ-530.



В данном примере предлагается запрограммировать какое-либо движение робота с помощью последовательных нажатий кнопок. Последовательность нажатий заносится в список, содержащий порядковый номер операции и тип нажатой кнопки.

После того как список команд сформирован и нажата кнопка Start, начинается процесс воспроизведения команд из списка до его полного опустошения. Количество команд задается пользователем на стадии написания программы.

Управляющая программа начинается с процесса инициализации начальных значений переменных, процесса проверки правильности сборки робота и тестового режима.

Сразу же после перехода к метке выполнения программы происходит инициализация счетчика команд с помощью функции initialize.

```
23: Start :
24:     CALL initialize
25:
26:     ENDLESS LOOP
27:     {
28:         Buzzer time = Play Melody
```

Функция initialize осуществляет сброс и переход к стартовому значению счетчика команд. Данная функция выделена отдельно, чтобы не допустить случайных ошибок при редактировании программы.

```
141: FUNCTION initialize
142: {
143:     // ********* no need to modify these variables *********//
144:
145:     button_number = 7
146: }
```

После процесса инициализации программа ждет ввода команды, т.е. нажатия одной из кнопок. Поскольку заранее известен перечень всех команд и соответствующих им действий, проводится анализ нажатия определенных кнопок на панели контроллера СМ-530. В нашем случае рассматриваются все кнопки – U, D, L, R, отвечающие за движение и повороты робота.

Какая из кнопок нажата в данный момент, определяется с помощью логической операции «ИЛИ» в теле оператора IF. Если нажата одна из указанных выше кнопок, выражение в теле оператора IF становится равным единице и выполнение программы переходит к следующей строке, где запоминается тип нажатой кнопки.

```
31:     ENDLESS LOOP
32:     {
33:         WAIT WHILE ( Button == ... )
34:
35:         IF ( Button == U || Button == D || Button == L || Button == R )
36:         {
37:             button = Button
38:             CALL button_standby
```

Каждое считанное нажатие кнопки заносится в список команд робота. В данном списке хранится номер текущей операции и ее тип, соответствующий типу нажатой кнопки. Количество программируемых команд определяется длиной списка.

```
40:             IF ( button_number == 1 )
41:             {
42:                 CALL button_init
43:                 button_1 = button
44:                 CALL button_standby
45:                 CALL buzzer_button
46:                 button_number = button_number + 1
47:             }
48:
49:             ELSE IF ( button_number == 2 )
50:             {
51:                 button_2 = button
52:                 CALL button_standby
53:                 CALL buzzer_button
54:                 button_number = button_number + 1
55:             }
```

В рассматриваемом примере список команд содержит пять операций, в случае ввода шестой команды список команд обнуляется и воспроизводится соответствующая мелодия.

```
81:             ELSE IF ( button_number == 6 )
82:             {
83:                 CALL buzzer_failure
84:                 CALL button_init
85:                 button_number = 1
86:             }
```

После того как пользователь задал необходимую последовательность команд, необходимо нажать на кнопку Start, после чего осуществляется выход из цикла и начинается выполнение программы согласно запрограммированной последовательности.

```
89:         ELSE IF ( Button == S )
90:         {
91:             button_number = 1
92:             BREAK LOOP
93:         }
```

Выполнение программы осуществляется последовательно по списку команд. Подобный метод идентичен работе с очередью команд, где выполнение программы производится последовательно с первой заданной и до конца очереди.

```
96:         CALL buzzer_success
97:         move_type = button_1
98:         CALL move
99:
100:        move_type = button_2
101:        CALL move
102:
103:        move_type = button_3
104:        CALL move
105:
106:        move_type = button_4
107:        CALL move
108:
109:        move_type = button_5
110:        CALL move
111:
112:        CALL standby
113:
114:        CALL button_init
```

В процессе выполнения программы осуществляется последовательный анализ очереди команд. На каждом этапе тип выполняемой операции определяется нажатой кнопкой, после чего вызывается функция move, которая определяет, какая из операций должна быть выполнена на данном этапе.

В теле функции move в зависимости от типа нажатой кнопки определяется действие, которое должен выполнить мобильный робот. В зависимости от полученной команды после программируемой задержки вызывается функция, реализующая движение робота.

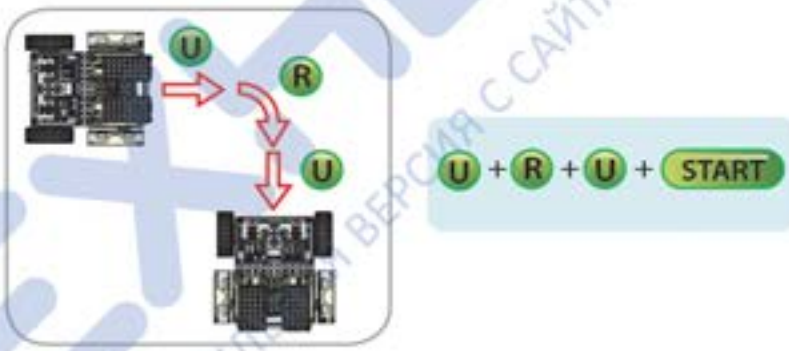
```
153: FUNCTION move
154: {
155:     IF ( move_type == U )
156:     {
157:         CALL standby
158:         CALL forward
159:     }
160:
161:     ELSE IF ( move_type == D )
162:     {
163:         CALL standby
164:         CALL reverse
165:     }
```

Функции, описывающие движения робота, идентичны тем, что использовались в предыдущей части лабораторной работы. С их помощью можно запрограммировать движение робота по произвольной траектории.

В процессе задания траектории движения робота можно использовать любое количество команд, не превышающее величину списка. В случае если величины списка команд недостаточно, следует изменить код программы и расширить величину списка самостоятельно на необходимую величину.

Воспользовавшись данными рекомендациями, разработайте управляющую программу робота согласно приведенному выше алгоритму. Для того чтобы запрограммировать робота с помощью кнопок на панели контроллера CM-530, с помощью кнопки MODE перейдите в режим PLAY и запустите выполнение программы.

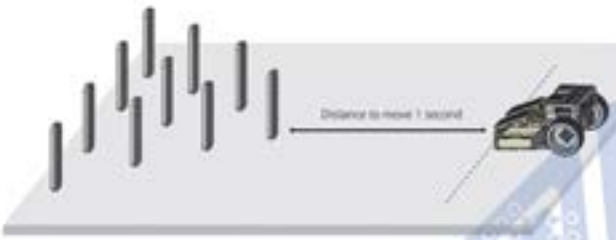
Программирование движений робота осуществляется с помощью кнопок U, D, L, R. В случае если необходимая последовательность команд задана, необходимо нажать на кнопку Start, после чего начнется процесс выполнения программы и робот начнет свое движение.



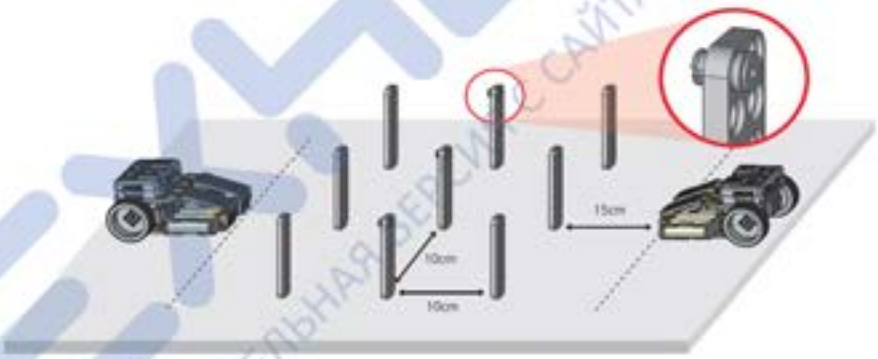
Часть № 3. Подведение итогов

Для того чтобы оценить качество восприятия материала данной лабораторной работы, предлагается выполнить несколько опытных испытаний.

Первое задание заключается в разработке программы робота, которая содержит минимальный список команд для его выполнения. Смысл задания заключается в сбивании флагов, расположенных в рабочей зоне. Победителем является робот, выполнивший задание за минимальное время и минимальное количество команд.



Второе задание заключается в соревновании двух роботов, которые наперегонки должны проехать по рабочему полю и сбить максимальное количество флагов, причем финальной целью является один из отмеченных флагов. Для данного задания могут применять различные правила, оговоренные игроками заранее, например: за сбивание обычных флагов начисляется определенное количество баллов, а за сбивание одного из отмеченных флагов игроку присваивается значительно большее количество баллов.



Программирование робота с помощью последовательности команд, задаваемых с помощью кнопок контроллера СМ-530, простой и быстрый способ корректировки программы робота. В любой момент после запуска робота пользователь может вручную задать алгоритм его работы.

Лабораторная работа № 2

ЭКЗАМЕН ТЕХНОЛАБ

Следование робота вдоль линии

Лабораторная работа № 2
«Следование робота вдоль линии»

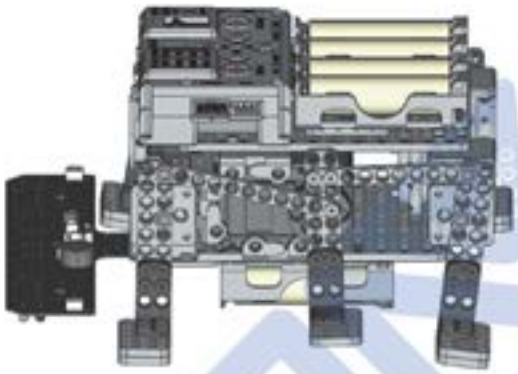
Мобильные роботы применяются для решения различных задач, но основное их предназначение – это перемещение в рабочей зоне. В зависимости от типа рабочей зоны и условий эксплуатации мобильного робота, роботы различаются по типу конструкций шасси. Наиболее часто встречаются колесные роботы, такие роботы обладают наибольшей скоростью и маневренностью. В условиях тяжелой проходимости, пересеченной местности и бездорожья, применяются мобильные роботы с гусеничными шасси. Гусеничные мобильные роботы обладают большей грузоподъемностью и проходимостью, а также способны выполнять развороты на месте, что важно при выполнении маневров в стесненных условиях.

В последнее время в мире стали применяться мобильные роботы с шагающей кинематикой. Чаще всего подобные мобильные роботы представляют собой четырехногое или шестиногое шасси. Интерес к подобным конструкциям обусловлен их повышенной проходимостью. Шагающие мобильные роботы могут передвигаться по пересеченной местности, двигаться среди завалов и подниматься по лестницам. Широкий диапазон применения таких роботов делает их достаточно востребованными в разведывательных и поисково-спасательных операциях, а также при транспортировке грузов в труднодоступные территории.

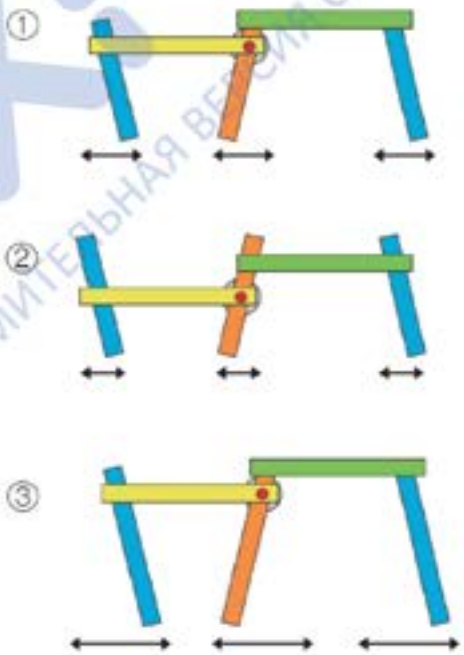
В рамках данной лабораторной работы предлагается рассмотреть конструктивные особенности роботов с шагающей кинематикой и исследовать процесс их движения вдоль линии. Поскольку в состав набора входят два сервомодуля, то существует единственный вариант конструкции шагающего шасси, которое можно разработать в данной работе.

Шасси робота предлагается конструировать по схеме с двумя ведущими приводами, где каждый из приводов будет управлять движением ног робота, расположенных сбоку. Для того чтобы шестиногое шасси робота могло перемещаться, ноги робота должны перемещаться синхронно.

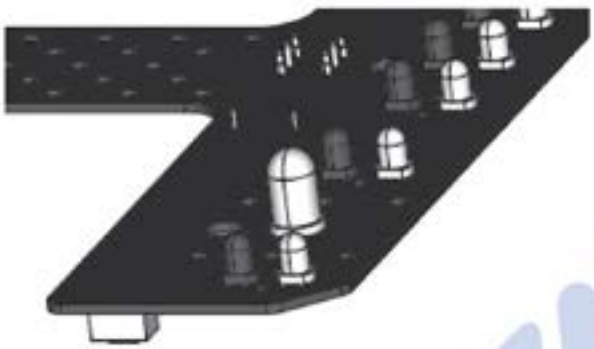
Синхронность движения ног робота-паука достигается за счет механической рычажной передачи. На корпусе робота закреплены ноги робота во вращающихся шарнирах, каждая из них соединена с рычагом, приводящим ее в движение с помощью привода.



Каждая из крайних ног робота-жука образует четырехзвенный механизм с центральной ногой, закрепленной с эксцентриситетом на фланце привода, в результате чего совершаются синхронные движения ног робота при вращении каждого из приводов.



В зависимости от геометрического положения ног робота и их длины зависит скорость передвижения, причем чем больше длина – тем больше шаг и соответственно больше скорость движения робота.

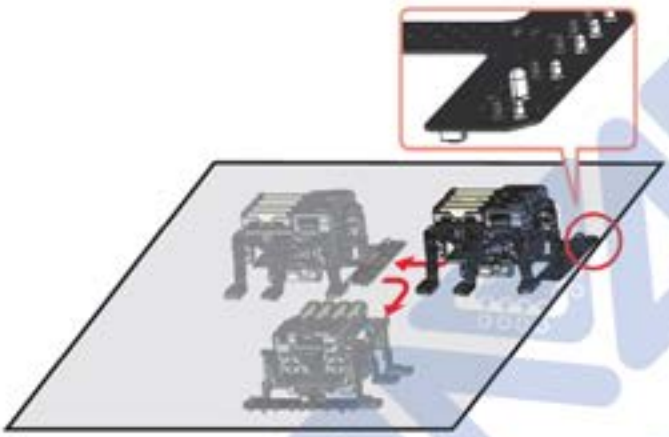


В рамках данной лабораторной работы предлагается разработать модель шестиногого робота-жука, ориентирующегося при движении с помощью ИК-датчиков. В качестве ИК-датчиков используется массив оптронов, состоящий из последовательно установленных светодиодов и фототранзисторов. С помощью подобного устройства робот может обнаруживать объекты, в частности черную линию на своем пути, и передвигаться по очерченным траекториям.

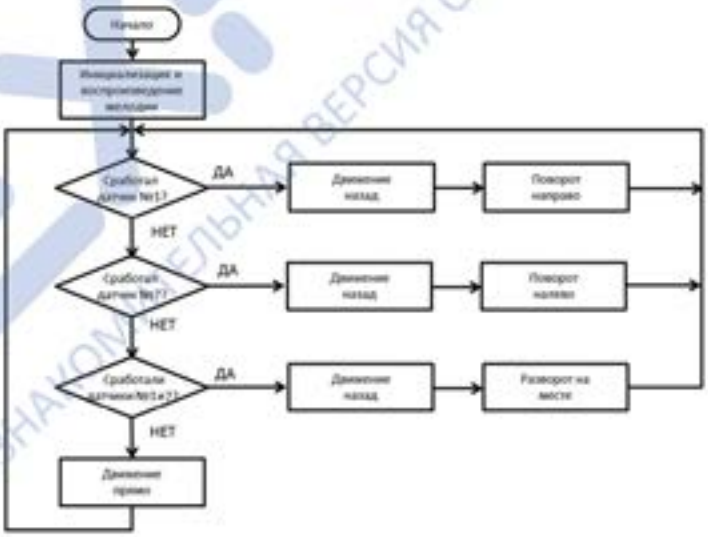
Примечание: для корректной работы ИК-датчиков очень важно осуществлять их калибровку на черный и белый цвет поверхности. Для калибровки массива ИК-датчиков поднимите робота на высоту порядка 5 см от пола и нажмите на кнопку Auto-set. После чего плавно проведите несколько раз над белой и черной поверхностью, добившись срабатывания каждого из датчиков, о чем должно свидетельствовать мигание светодиодов.

Часть № 1. Использование ИК-датчиков для определения черной линии на белом фоне

Рассмотрим простейший алгоритм нахождения черной линии на белом фоне, согласно которому робот должен будет избегать движения по черной линии. При обнаружении черной линии робот должен выполнить любой маневр, изменяющий свое направление на противоположное.



Обнаружение черной линии осуществляется с помощью массива ИК-датчиков, который состоит из 7 пар – светодиода и фототранзистора, работающих совместно. В процессе работы с массивом ИК-датчиков пользователь имеет доступ к показаниям каждого из семи датчиков в отдельности. Воспользуемся этой возможностью в рамках данной работы, для того чтобы определять наличие черной линии вблизи с роботом.



Суть алгоритма сводится к тому, что робот при обнаружении черной линии отходит назад и разворачивается в направлении, противоположном обнаруженной линии.

Для того чтобы обнаружить черную линию и определить ее положение относительно робота, достаточно всего двух датчиков, расположенных по бортам робота. В связи с этим в процессе работы программы будем рассматривать показания 1-го и 7-го датчиков, расположенных по краям массива ИК-датчиков.

Примечание: в приведенном примере выполнение программы начинается с процедуры настройки системы и проверки правильности ее сборки. Для этих целей рекомендуется использовать готовый кусок программы, поскольку он достаточно сложен для разработки в процессе обучения. Также можно разрабатывать управляющую программу без использования функций диагностики и контроля правильности сборки.

Программа управления начинается с инициализации переменных, определяющих скорость движения робота.

```
10: Start :
11:     // L/R wheel speed (0 ~ 1023)
12:     running_speed  = 650
13:     reverse_time   = 1.152sec
14:     pivot_time     = 1.664sec
```

Следом за инициализацией переменных начинается бесконечный цикл, в котором рассматривается, какой из ИК-датчиков массива сработал в текущий момент времени. Тип сработавшего датчика определяется с помощью функции detect_black.

```
16:     ENDLESS LOOP
17:     {
18:         CALL detect_black
```

В данном примере рассматривается случай определения черной линии только 1-м или 7-м датчиком ИК-массива. Работа с массивом ИК-датчиков осуществляется как с устройством с идентификационным номером 100, работающим по общему протоколу шины TTL контроллера CM-530.

```
52: FUNCTION detect_black
53: {
54:     black_result = ID[100]: IR Obstacle Detected
55:     black_1 = black_result & 1
56:     black_7 = black_result & 7
57: }
```

Переменная black_result возвращает результат срабатывания массива ИК-датчиков, т.е. семизначное двоичное число. Для того чтобы определить сработал 1-й или 7-й датчик, производится маскирование (операция «логическое И»), в результате чего переменные black_1 и black_7 становятся равными единице в случае срабатывания соответствующего датчика.

В зависимости от того, какой из датчиков сработал в данный момент, выполняется одно из условий алгоритма. Например, при обнаружении линии слева робот отъезжает назад и поворачивает направо.

```
19:     IF ( black_1 > 0 && black_7 == 0 )
20:     {
21:         CALL reverse
22:         CALL pivot_right
23:     }
```

Движения робота описываются с помощью базовых функций, каждая из которых управляет приводами непосредственно. Скорость движения и временной интервал задаются в начале программы.

```
65: FUNCTION reverse
66: {
67:     ID[1]: Moving speed = CW 0 + running_speed
68:     ID[2]: Moving speed = CCW 0 + running_speed
69:     Timer = reverse_time
70:     CALL timer_standby
71:     CALL stop
72: }
```

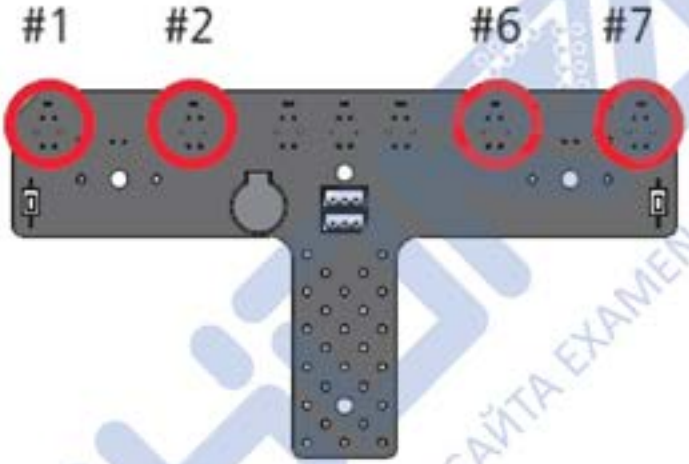
Управление каждым из приводов в отдельности сводится к его вращению с заданной скоростью в течение определенного промежутка времени. В зависимости от скорости вращения и времени работы определяется перемещение робота.

Таким образом, с помощью массива ИК-дальномеров возможно реализовывать функции автономного движения робота вдоль линии по любому произвольному маршруту. Несмотря на кажущуюся простоту это довольно часто встречающаяся в промышленности задача. На большинстве промышленных предприятий применяются робокары – автономные транспортные средства для перевозки различных грузов.

Как правило, подобные роботы перемещаются вдоль специально начерченной линии, причем расположенной так, чтобы минимизировать возможный контакт с людьми. Несмотря на это при проектировании подобных роботов уделяется большое внимание вопросам безопасности. Одна из важнейших задач – обеспечение плавности движения робота вдоль линии и избегание столкновений с препятствиями на пути.

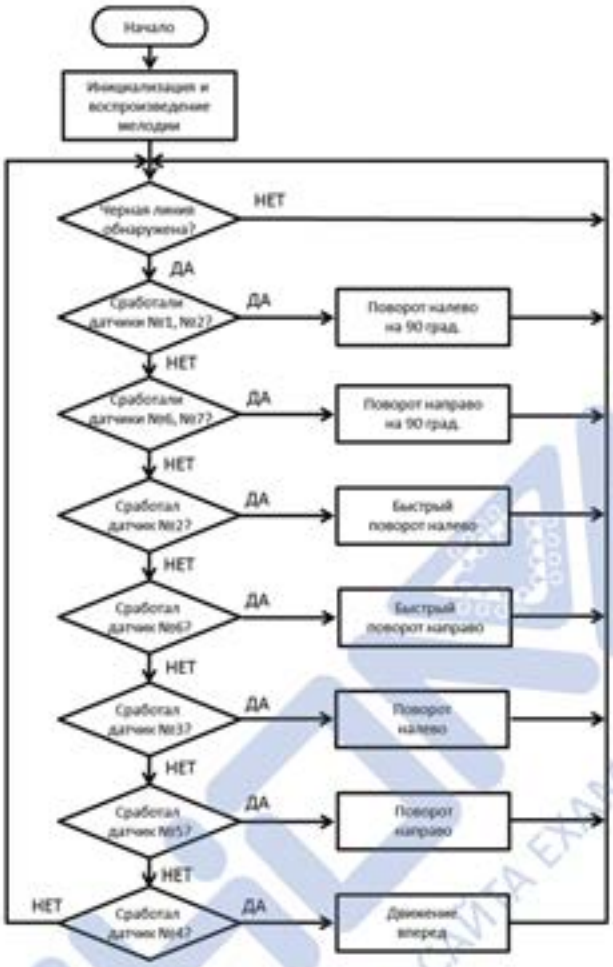
Часть № 2. Использование ИК-датчиков для управления движением робота вдоль линии

Для того чтобы движения робота вдоль линии были плавными, а реакция на изменение кривизны маршрута была достаточно быстрой, необходимо постоянно контролировать как наличие самой линии, так и изменение ее направления. Чаще всего с этой целью используются два ИК-датчика, расположенные справа и слева, тем самым определяя наличие линии по обе стороны от робота.



Для того чтобы положение линии определялось точнее, следует увеличить количество ИК-датчиков. Массив ИК-датчиков содержит семь ИК-датчиков, каждый из которых может определять положение черной линии. В зависимости от того, какой из датчиков сработал, можно определить положение робота относительно черной линии и предпринять какое-либо действие. Например, в зависимости от того, на каком расстоянии от центра робота расположена черная линия, можно регулировать скорость поворота робота.

В общем случае движение робота можно описать с помощью алгоритма, в котором анализируются различные комбинации возможных вариантов срабатывания датчиков. Помимо того, что анализируется положение черной линии, алгоритм учитывает расстояние от нее до центра робота. Принцип работы заключается в условии – чем дальше центр робота от черной линии, тем быстрее должен быть осуществлен возврат к линии, а значит – скорость движения робота и его маневров увеличивается.



Алгоритм сводится к поэтапному рассмотрению всех возможных комбинаций и выполнению заданного действия. Все маневры робота сводятся к движению в противоположную сторону от линии, а скорость движения определяется расстоянием до линии. В случае же, если черная линия находится по центру робота, т.е. обнаружена центральным ИК-датчиком, робот продолжает прямолинейное движение.

Управляющая программа начинается с традиционной для данного модуля процедуры проверки правильности сборки робота. После чего производится процедура инициализации базовых параметров – скорости движения робота и времени работы ИК-датчиков.

```
10: Start :
11:     // L/R wheel speed (0 ~ 1023)
12:     high_speed  = 650
13:     normal_speed  = 200
14:     low_speed  = 0
15:     // Time for judging whether turning is finished
16:     detect_time  = 0.128sec
```

Выполнение программы осуществляется в бесконечном цикле, реализующем приведенный ранее алгоритм.

```
18:     ENDLESS LOOP
19:     {
20:         CALL detect_black
21:
22:         IF ( black_1 > 0 && black_2 > 0 )
23:         {
24:             CALL turn_left_fast
25:             black_detected_number = 3
26:             CALL stop_at_black
27:         }
28:
29:         ELSE IF ( black_7 > 0 && black_6 > 0 )
30:         {
31:             CALL turn_right_fast
32:             black_detected_number = 5
33:             CALL stop_at_black
34:         }
35:
36:         ELSE
37:             CALL follow_line
38:     }
```

Программа состоит из четырех основных итераций. На первой итерации производится вызов функции detect_black. С помощью этой функции определяется, какой из ИК-датчиков массива сработал.

```
88: FUNCTION detect_black
89: {
90:     black_result = ID[100]: IR Obstacle Detected
91:
92:     black_1 = black_result & 1
93:     black_2 = black_result & 2
94:     black_3 = black_result & 3
95:     black_4 = black_result & 4
96:     black_5 = black_result & 5
97:     black_6 = black_result & 6
98:     black_7 = black_result & 7
99: }
```

Каждому из ИК-датчиков присваивается переменная, характеризующая его состояние, например первому датчику соответствует переменная black_1 и т.д. После вызова функции detect_black каждой из переменных black присваивается значение, которое анализируется в дальнейшем.

На второй и на третьей итерации рассматриваются два случая, в котором черную линию обнаруживает одна из пар датчиков – левая пара датчиков № 1 и № 2 или правая пара датчиков № 6 и № 7. Поскольку это самые крайние датчики, их срабатывание означает, что робот съезжает с линии и его движение необходимо скорректировать как можно скорее.

```
22:         IF ( black_1 > 0 && black_2 > 0 )
23:         {
24:             CALL turn_left_fast
25:             black_detected_number = 3
26:             CALL stop_at_black
27:         }
```

В этом случае робот выполняет быстрый поворот до тех пор, пока черная линия не окажется под датчиком № 3 или № 5. Данные датчики расположены в достаточной близости к центру робота, что дает возможность маневрировать вдоль черной линии без опасений насчет того, что робот съедет с нее.

```
51: FUNCTION stop_at_black
52: {
53:     turning_finish = FALSE
54:
55:     LOOP WHILE ( turning_finish == FALSE )
56:     {
57:         WAIT WHILE ( ID[100]: IR Obstacle Detected != black_detected_number )
58:         Timer = detect_time
59:         WAIT WHILE ( Timer > 0.000sec && ID[100]: IR Obstacle Detected == black_detected_number )
60:         IF ( Timer == 0.000sec )
61:             turning_finish = TRUE
62:     }
63: }
```

Остановка робота при совпадении вышеуказанных датчиков с черной линией осуществляется с помощью функции stop_at_black. Данная функция задает время поворота робота до тех пор, пока не сработает ИК-датчик под номером black_detected_number. Факт срабатывания датчика определяется в течение времени detect_time, для того чтобы исключить вероятность ложных срабатываний.

Функция black_detected_number содержит бесконечный цикл, в котором анализируется срабатывание датчика black_detected_number. Фактически функция генерирует временной интервал, во время которого робот осуществляет поворот. Как только указанный датчик срабатывает, изменяется состояние флага turning_finish, который прерывает выполнение бесконечного цикла и приводит к завершению работы функции. После чего робот продолжает анализ показаний других датчиков и движение вдоль линии.

Движение вдоль линии осуществляется с помощью функции follow_line, которая выполняется всегда, за исключением двух вышеописанных критических ситуаций.

```
65: FUNCTION follow_line
66: {
67:     CALL detect_black
68:
69:     IF ( black_2 > 0 )
70:         CALL turn_left_fast
71:
72:     ELSE IF ( black_6 > 0 )
73:         CALL turn_right_fast
74:
75:     ELSE IF ( black_3 > 0 )
76:         CALL turn_left
77:
78:     ELSE IF ( black_5 > 0 )
79:         CALL turn_right
80:
81:     ELSE IF ( black_4 > 0 )
82:         CALL forward
83:
84:     ELSE IF ( black_result == 0 )
85:         CALL forward
```

В зависимости от показаний какого-либо из датчиков выбирается одно из направлений для маневра. Важно обратить внимание на то, что каждая из функций, описывающих движение робота, не содержит таймеров, задающих время ее работы.

```
102: FUNCTION turn_left_fast
103: {
104:     ID[1]: Moving speed = CCW:0 + low_speed
105:     ID[2]: Moving speed = CW:0 + high_speed
106: }
```

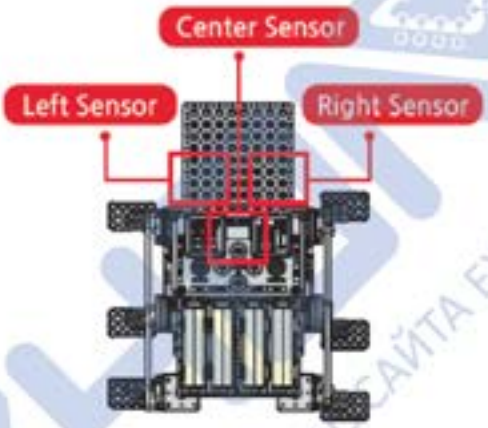
Функции, описывающие движения робота, представляют собой набор инструкций, задающих скорости приводов робота. Это обусловлено тем, что время действия каждой из функций задается алгоритмически за счет перехода от одного условия к другому. Таким образом, робот совершает определенный маневр только во время выполнения соответствующего условия.

Часть № 3. Подведение итогов

Для того чтобы оценить качество восприятия материала данной лабораторной работы, предлагается выполнить несколько опытных испытаний.

В первую очередь необходимо оценить работоспособность предложенных алгоритмов на различных типах поверхности, например: при движении робота вдоль черной линии на белом фоне или на поверхности какого-либо цвета. В рамках таких пробных тестов важно оценить влияние отражающей способности поверхности на работу программы и поведение робота.

На втором важно оценить преимущества применения массива ИК-датчиков на роботе по сравнению с отдельными ИК-датчиками, расположенными по бортам. С это целью предлагается адаптировать конструкцию робота и установить 3 отдельных ИК-датчика : 2 шт. по бортам робота и 1 шт. спереди.



ИК-датчики, расположенные по бортам робота, ориентированы в пол и используются для определения черной линии; датчик, расположенный по центру, ориентирован по направлению движения и предназначен для обнаружения препятствий на пути робота. В связи с этим необходимо скорректировать управляющий алгоритм программы.

Скорректируем программу робота, для того чтобы он стал применим для робототехнических соревнований по правилам «сумо».

1) Если центральный ИК-датчик обнаружил препятствие, а боковые ИК-датчики не обнаружили линию – двигаться вперед.

2) Если правый ИК-датчик обнаружил линию – повернуть налево.

3) Если левый ИК-датчик обнаружил линию – повернуть направо.

4) Если центральный ИК-датчик потерял объект из видимости – повернуть направо и двигаться вперед.

Выше представлен один из простейших вариантов реализации алгоритма управления роботом для соревнований в стиле «сумо». Любой желающий может скорректировать предложенную программу или разработать собственный алгоритм, который приведет его робота к победе.

Данная лабораторная работа была посвящена проектированию роботов, движущихся вдоль линии. Основная цель данной работы заключается в освоении базовых принципов функционирования ИК-датчиков, а также оценки степени влияния на их работу различных посторонних факторов – качество отражающей поверхности, скорость движения робота и т.п.

Лабораторная работа № 3

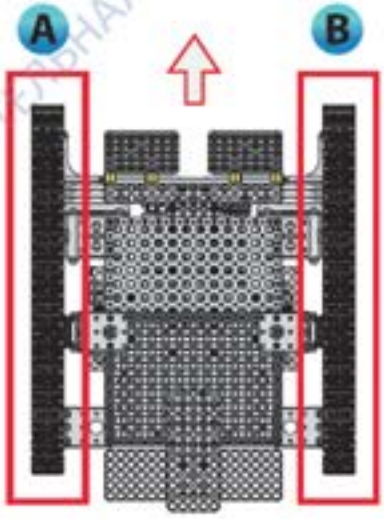
ЭКЗАМЕН ТЕХНОЛАБ

Исследование проходимости роботов

Лабораторная работа № 3
«Исследование проходимости роботов»

Роботы и робототехнические системы предназначены для работы в экстремальных условиях, там, где необходимо облегчить или обезопасить труд человека. Очень часто мобильные роботы применяются в экстремальных ситуациях, например при тушении пожаров, локализации радиоактивных отходов и т.п., и, как правило, работают в труднопроходимой местности.

Решение подобных задач возлагается на мобильных гусеничных роботов, которые обладают высокой проходимостью и грузоподъемностью. Важное отличительное качество гусеничных мобильных роботов заключается в их маневренности. Обладая независимым приводом для каждой из гусениц в отдельности, мобильный робот может легко менять направление собственного движения.



Благодаря тому, что скорость каждой из гусениц регулируется в отдельности, достаточно легко управлять движением мобильного робота. Для задания какого-либо направления движения необходимо изменить относительную скорость приводов.

Тип движения	Правый привод	Левый привод
Движение прямо	(прямо) Vправ= Vлев (прямо)	(прямо) Vлев= Vправ (прямо)
Поворот налево	(прямо) Vправ> Vлев (прямо)	(прямо) Vлев< Vправ (прямо)
Поворот направо	(прямо) Vправ< Vлев (прямо)	(прямо) Vлев> Vправ (прямо)
Поворот на месте налево	(прямо) Vправ = - Vлев (назад)	(назад) Vлев= - Vправ (прямо)
Поворот на месте направо	(назад) Vправ = - Vлев (прямо)	(прямо) Vлев= - Vправ (назад)
Движение назад	(назад) Vправ = Vлев (назад)	(назад) Vлев= Vправ (назад)

Вышеуказанная таблица демонстрирует соотношение скоростей и направлений вращения приводов гусеничного шасси. Важно обращать внимание на положение привода, ведь в зависимости от ориентации в пространстве привода зависит направление вращения его выходного вала, а соответственно и направление движения гусеничных траков. Например, для того чтобы робот двигался вперед, необходимо, чтобы его левый привод вращался «против часовой стрелки», а правый – «по часовой стрелке».

1) Для того чтобы двигаться прямо, необходимо, чтобы правый и левый приводы вращались с одинаковой скоростью в направлении «прямо».

2) Для того чтобы повернуть налево, необходимо, чтобы скорость правого привода была больше, чем скорость левого. Чем больше будет разница скоростей, тем меньше будет радиус разворота при движении.

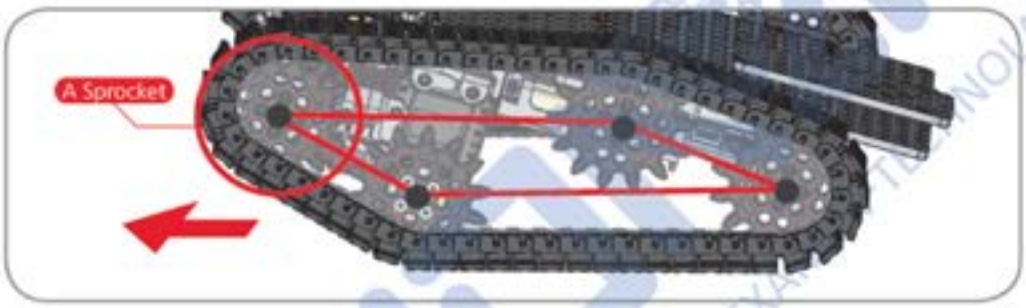
3) Для того чтобы повернуть направо, необходимо, чтобы скорость правого привода была меньше, чем скорость левого. Чем больше будет разница скоростей, тем меньше будет радиус разворота при движении.

4) Для того чтобы повернуть налево на месте, необходимо, чтобы правый привод вращался «прямо», а левый – «назад» с такой же скоростью.

5) Для того чтобы повернуть направо на месте, необходимо чтобы левый привод вращался «прямо», а правый «назад» с такой же скоростью.

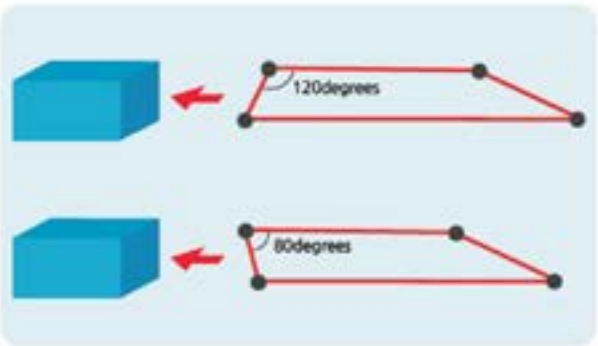
6) Для того чтобы двигаться назад, необходимо, чтобы правый и левый приводы вращались с одинаковой скоростью в направлении «назад».

Помимо высокой маневренности гусеничные шасси обладают повышенной проходимостью. Благодаря хорошему сцеплению гусеничных траков с поверхностью, по которой осуществляется движение, гусеничные роботы могут преодолевать различные неровности поверхности и преграды.



В зависимости от назначения гусеничного робота и степени его проходимости различают различные конструкции гусеничных шасси.

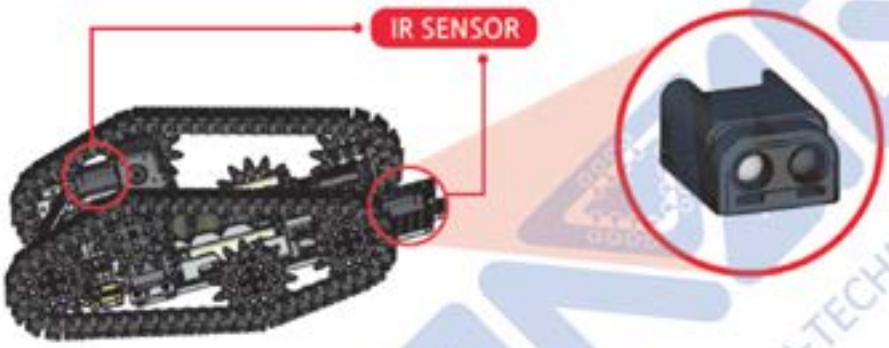
Традиционно гусеничные транспортные средства имеют специальный угол наклона спереди, чтобы въезжать на препятствия по ходу движения. Чем выше проходимость гусеничного робота или транспортного средства, тем, как правило, больше данный уклон.



Иногда для решения специализированных задач применяются гусеничные транспортные средства, состоящие из подвижных относительно друг друга гусеничных траков. С помощью регулировки угла подъема передних гусениц подобные роботы могут преодолевать препятствия различной сложности.

В данной лабораторной работе исследуются способы управления мобильным гусеничным шасси. Разрабатываемая в рамках данной работы модель робота обладает достаточно большой проходимостью для собственных габаритных размеров.

Конструкция шасси робота состоит из гусеничных траков, расположенных под достаточно большим углом к направлению движения, благодаря чему робот может преодолевать препятствия с высотой не менее высоты подъема гусеничных траков.

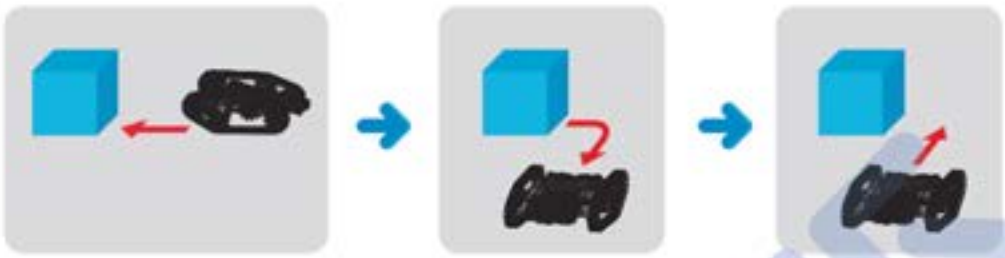


Для того чтобы робот в процессе своего движения не застрял, преодолевая препятствие, его шасси оснащается ИК-датчиком, определяющим наличие объектов на его пути. Если датчик обнаруживает объект, это значит, что высота объекта соизмерима с высотой робота и есть риск того, что робот не сможет преодолеть данный маршрут. В этом случае робот должен предпринять какое-либо другое действие, например объехать препятствие сбоку и т.п.

Данная лабораторная работа посвящена изучению основ движения гусеничных мобильных роботов, исследованию способов их маневрирования и прохождения различных препятствий.

Часть № 1. Выполнение простейших маневров

В данной части лабораторной работы рассматриваются прямолинейное движение гусеничного робота и процесс поиска препятствий на его пути.



Если на пути робота обнаруживается препятствие, это означает, что оно обладает габаритами, которые робот не в состоянии переехать. В этом случае система управления робота должна предпринять какие-либо действия по выполнению маневров с целью избежать столкновения.

Согласно предложенному алгоритму гусеничный робот едет прямо, переезжая все препятствия на своем пути. Если же на пути робота обнаруживается объект, который не пропадает с пути робота в течение 3 секунд, робот останавливается и совершает маневр по объезду препятствия.

```
3:    obstacle_threshold  = 100
4:
5:    // L/R wheel speed (0 ~ 1023)
6:    // Modify speed until robot run straightly in "Straightness_check_mode"
7:    l_wheel_speed  = 600
8:    r_wheel_speed  = 600
9:
10:   // time spent trying to overcome obstacle
11:   obstacle_judging_time  = 3.072sec
```

В начале программы задаются базовые переменные, определяющие пороговое значение до объекта, скорости вращения приводов робота и время ожидания перед обнаруженным объектом (3 сек). С помощью этих значений определяется скорость маневрирования роботов и расстояние, которое робот не доезжает до обнаруженного объекта.

```
29:  ENDLESS LOOP
30:  {
31:      IF ( PORT[3] <= obstacle_threshold )
32:          CALL forward
33:
34:      ELSE
35:      {
36:          CALL forward
37:          Timer = obstacle_judging_time
38:          LOOP WHILE ( Timer > 0.000sec )
39:          {
40:              IF ( PORT[3] <= obstacle_threshold )
41:                  BREAK LOOP
42:          }
43:
44:          IF ( PORT[3] > obstacle_threshold )
45:          {
46:              CALL reverse
47:              Timer = 0.540sec
48:              CALL timer_standby
49:              CALL stop
50:              Timer = 0.128sec
51:              CALL timer_standby
52:              CALL pivot_left
53:              Timer = 0.768sec
54:              CALL timer_standby
55:          }
56:      }
57:  }
```

Программа представляет собой бесконечный цикл, в котором анализируется показания датчика, подключенного к PORT[3]. С помощью переменных obstacle_threshold и obstacle_judging_time задаются максимальные значения расстояния до объекта и время обнаружения объекта. Если объект находится вне зоны видимости, робот продолжает движение под управлением функции forward.

В случае если робот обнаружил объект на своем пути, поочередно вызываются функции reverse, stop, pivot_left с помощью которых робот совершает заданный маневр по объезду препятствия. Функции чередуются с помощью вызова таймера, ограничивающего время работы каждой из них.

```
92: FUNCTION reverse
93: {
94:     ID[1]: Moving speed = CW:0 + l_wheel_speed
95:     ID[2]: Moving speed = CCW:0 + r_wheel_speed
96: }
97:
98: FUNCTION pivot_left
99: {
100:    ID[1]: Moving speed = CW:0 + l_wheel_speed
101:    ID[2]: Moving speed = CW:0 + r_wheel_speed
102: }
103:
104: FUNCTION stop
105: {
106:    ID[1]: Moving speed = CCW:0
107:    ID[2]: Moving speed = CCW:0
108: }
```

Таймеры очень часто применяются для задания времени работы какого-либо устройства или фрагмента управляющей программы. Рассмотрим работу таймера на примере функции инициализации, вызываемой в самом начале программы.

```
78: FUNCTION initialize
79: {
80:     prepare_time = 2.048sec
81:     CALL buzzer
82:     Timer = prepare_time
83:     CALL timer_standby
84: }
```

Данная функция запускает воспроизведение мелодии на время, определяемое переменной prepare_time. Данная переменная инициализирует таймер, который работает в течение заданного времени.

```
119: FUNCTION timer_standby
120: {
121:     WAIT WHILE ( Timer > 0.000sec )
122: }
```

Во время отсчета таймера осуществляется задержка, во время которой выполняется последняя операция, например воспроизведение мелодии. Задержка осуществляется с помощью функции timer_standby, которая с помощью оператора WAIT WHILE ожидает окончания отсчета. Таким образом, можно сгенерировать любую необходимую для работы программы временную задержку.

Важной отличительной особенностью данной программы является отличие функции проверки правильности сборки от функций, рассматриваемых в предыдущих работах. В данной работе перед запуском основной программы определяется, к какому из портов управления подключен ИК-датчик.

Для этого автоматически опрашиваются все порты, и если на одном из них обнаруживаются показания от ИК-датчика, светодиод моргает соответствующее номеру порта количество раз.

```
228: FUNCTION IR_sensor_port_check
229: {
230:     IF ( PORT[1] > 100 )
231:         CALL LED_port_num
```

```
264: FUNCTION LED_port_num
265: {
266:     IF ( Button == ... )
267:     {
268:         Aux LED = TRUE
269:         Timer = 0.256sec
270:         CALL timer_standby
271:     }
272:     IF ( Button == ... )
273:     {
274:         Aux LED = FALSE
275:         Timer = 0.256sec
276:         CALL timer_standby
277:     }
278: }
```

В случае с PORT[1] при обнаружении сигнала с ИК-датчика вызывается функции LED_port_num, которая моргает системным светодиодом.

```
233:     ELSE IF ( PORT[2] > 100 )
234:     {
235:         LOOP FOR ( iterations = 1 ~ 2 )
236:             CALL LED_port_num
237:     }
238:
239:     ELSE IF ( PORT[3] > 100 )
240:     {
241:         LOOP FOR ( iterations = 1 ~ 3 )
242:             CALL LED_port_num
243:     }
```

Если же сигнал с ИК-датчика обнаруживается на каком-то другом порту контроллера CM-530, функция LED_port_num вызывается с помощью оператора LOOP FOR, выполняющегося заданное количество раз.

Часть № 2. Преодоление препятствий на пути

Разрабатываемый в рамках данной работы мобильный гусеничный робот может быть отнесен к сверхлегкому классу подобных роботов. Такие роботы, как правило, применяются для разведки местности и работают на пересеченной местности, среди руин и завалов.

Подобные мобильные роботы перемещаются достаточно быстро среди завалов, преодолевают уклоны и спуски, но из-за малой массы часто опрокидываются и переворачиваются. Несмотря на это роботы должны выполнять поставленную задачу, а значит, как минимум, продолжать движение.

Для того чтобы робот мог продолжать движение после переворачивания, его конструкцию делают абсолютно симметричной.

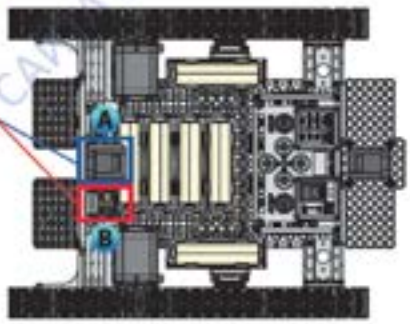
Для того чтобы робот мог двигаться в перевернутом состоянии, необходимо изменить направления вращения его приводов. Поскольку подобные роботы перемещаются автономно или зачастую пользователь не может наблюдать за их перемещением, смена направления вращения должна осуществляться автоматически в зависимости от ориентации в пространстве робота.

С целью определения положения робота устанавливается ИК-датчик, направленный в пол. Соответственно в одном из положений робота он срабатывает, а в перевернутом – нет, и наоборот, в зависимости от способа установки датчика.

Управляющая программа робота идентична той, что рассматривалась в предыдущей части, за исключением части, касающейся определению положения робота.

Программа робота состоит из двух симметричных веток, каждая из которых выполняется в зависимости от положения робота. В отличие от программы в предыдущей части, в данной рассматривается единственное дополнительное условие.

```
ELSE IF ( PORT[4] <= obstacle_threshold )
{
    IF ( PORT[3] <= obstacle_threshold )
    {
        CALL reverse
    }

    ELSE
    {
        CALL reverse
        Timer = obstacle_judging_time
        LOOP WHILE ( Timer > 0.000sec )
        {
            IF ( PORT[3] <= obstacle_threshold )
                BREAK LOOP
        }
```



Переход от одной ветки программы к другой осуществляется с помощью ИК-датчика, подключенного к PORT[4], который определяет ориентацию робота в пространстве.

```
ENDLESS LOOP
{
    IF ( PORT[4] > obstacle_threshold )
    {
        IF ( PORT[3] <= obstacle_threshold )
            CALL forward

        ELSE
        {
            CALL forward
            Timer = obstacle_judging_time
            LOOP WHILE ( Timer > 0.000sec )
            {
                IF ( PORT[3] <= obstacle_threshold )
                    BREAK LOOP
            }

            IF ( PORT[3] > obstacle_threshold )
            {
                CALL stop
                CALL slow_reverse
                Timer = 1.004sec
                CALL timer_standby
                CALL stop
                Timer = 0.128sec
                CALL timer_standby
                CALL pivot_left
                Timer = 0.768sec
                CALL timer_standby
```

Также в отличие от программы из части № 1, для функций forward и reverse реализованы аналоги slow_forward и slow_reverse, которые обеспечивают медленное движение робота при маневрах.

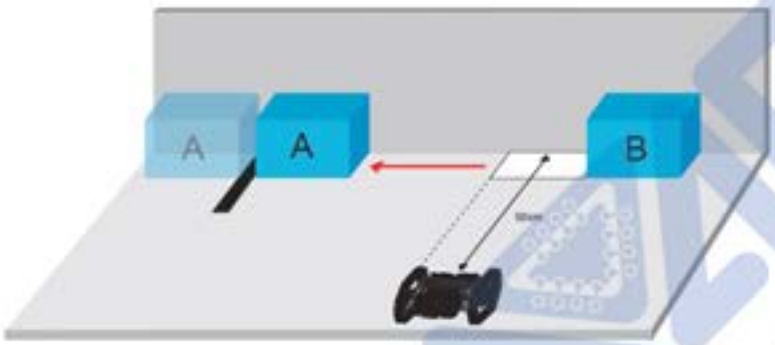
```
FUNCTION reverse
{
    ID[1]: Moving speed = CW:0 + l_wheel_speed
    ID[2]: Moving speed = CCW:0 + r_wheel_speed
}

FUNCTION slow_reverse
{
    ID[1]: Moving speed = CW:0 + l_wheel_low_speed
    ID[2]: Moving speed = CCW:0 + r_wheel_low_speed
}
```

Данные функции оперируют значениями l_wheel_low_speed и r_wheel_low_speed, описываемыми в начале программы.

Часть № 3. Подведение итогов

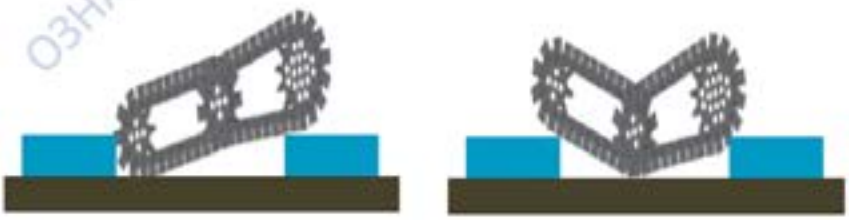
Для того чтобы оценить качество восприятия материала данной лабораторной работы, предлагается выполнить несколько опытных испытаний.

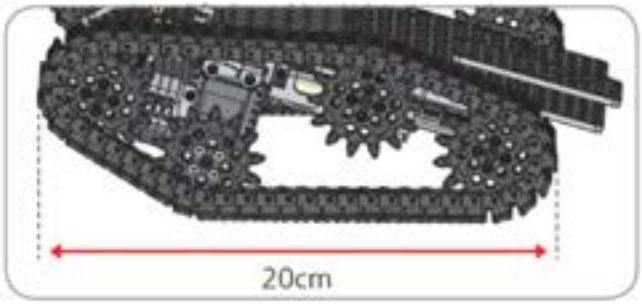
Смоделируйте ситуацию, при которой робот транспортирует груз за пределы черной линии. Для этого оснастите гусеничное шасси двумя ИК-датчиками: спереди – для обнаружения объектов и снизу – для определения черной линии.



По возможности оцените тяговую силу гусеничного шасси по сравнению с колесным и шагающим. Проведите ряд испытаний на поверхностях разного типа и сделайте выводы о том, какой тип шасси обладает большей силой сцепления.

Исследуйте проходимость гусеничных роботов на пересеченной местности, моделирующей ямы или овраги на пути робота. Направьте робота на преодоление прерывистого препятствия, а сами оцените, как зависит ширина преодолеваемого препятствия от габаритов робота.





Помните, что ничего не ограничивает фантазию разработчиков в рамках решения поставленных задач. И если проходимости вашего робота не хватает для преодоления препятствия на его пути, это не повод отчаиваться, а хороший шанс задуматься и усовершенствовать конструкцию.

Лабораторная работа № 4

ЭКЗАМЕН ТЕХНОЛАБ

Поиск маршрута для движения мобильного робота

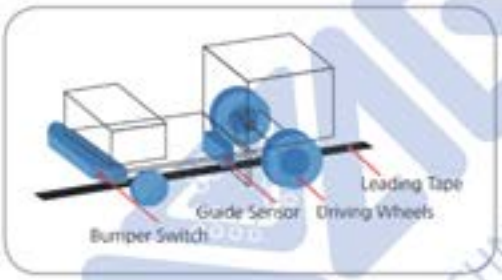
Лабораторная работа № 4
«Поиск маршрута для движения мобильного робота»

Одна из ключевых задач мобильной робототехники – это поиск маршрута для движения и его оптимизация. Перемещаясь в рабочей местности, робот должен постоянно оценивать окружающую обстановку, определять свое положение и положение окружающих его объектов. Существует множество различных способов, с помощью которых робот может определять собственное положение и строить маршрут между точками назначения. При перемещениях на улице применяется технология спутниковой навигации, а окружающие объекты обнаруживаются с помощью камер или дальномеров. В случае перемещения внутри помещений с помощью камер и дальномеров строится виртуальная модель пространства, по которой робот ориентируется в дальнейшем. Вышеописанные методы имеют общий характер и применимы в произвольных ситуациях, но из-за этого они очень сложны в реализации и еще не применяются широко в повседневной жизни.

Как правило, автономные робототехнические системы проектируются под конкретные задачи. Такой подход позволяет формализовать требования к системе и разработать все возможные алгоритмы реакции на изменение состояния окружающей обстановки.

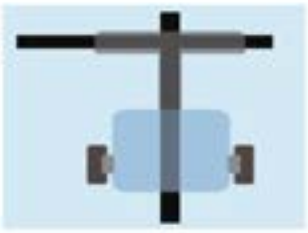
Например, одной из достаточно жестко формализованных задач может быть перемещение объектов внутри производственного помещения. Как правило, при перевозке грузов на складах или в производственных цехах роботы преодолевают один и тот же маршрут постоянно. Соответственно данный маршрут заранее известен и для него можно разработать систему управления движением робота.

Ранее цеховые транспортные средства представляли собой тележки, перемещающиеся по рельсам. С ростом науки и техники на смену им пришли робокары – мобильные роботы разных типов и для различных задач, а рельсы, проложенные вдоль цеха, заменила паутина направляющих линий, начерченных на полу.



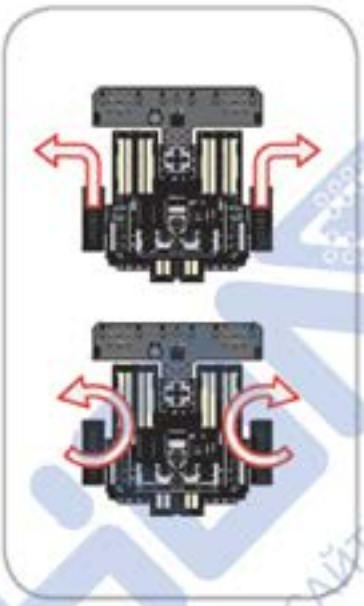
Мобильные роботы, передвигающиеся в цехах вдоль линии, подобно роботам из предыдущих лабораторных работ оснащаются различными сенсорными устройствами для восприятия окружающей обстановки: ИК-датчиками, камерами, датчиками безопасности и т.п. Но в отличие от рассматриваемых ранее роботов, реальные роботы работают отнюдь не в лабораторных условиях – зачастую направляющая линия может быть повреждена или скрыта за каким-либо объектом, некоторые маршруты могут пересекаться или вовсе прерываться частично.

На реальный маршрут могут накладываться различные ограничения, например: некоторые участки маршрута могут быть запрещены для движения, а некоторые могут быть достигнуты только после проезда через другие.

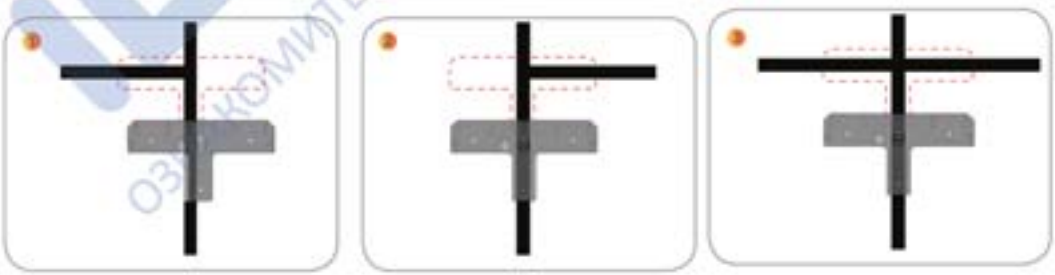


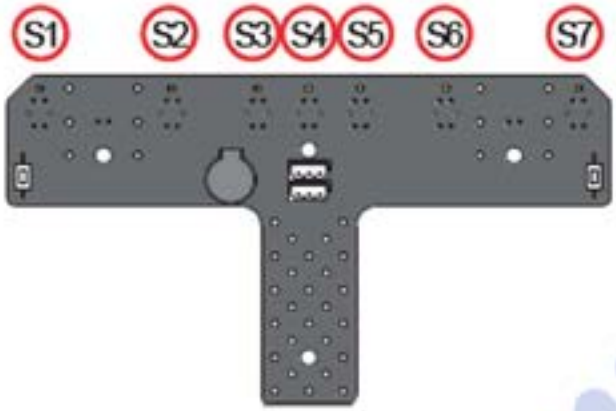
Становится очевидным, что методы движения вдоль линии, представляющей собой замкнутую траекторию, не совсем приемлемы в подобном случае. С примерами различных алгоритмов движения вдоль линии можно ознакомиться в предыдущих работах, но сразу же можно сделать вывод о том, что ни один из них не учитывает прерывистости траектории движения или наличия на ней пересечений.

Если в процессе движения управляющая программа мобильного робота окажется не в состоянии определить наличие пересечения направляющих линий, это может привести к неоднозначности принятия решения.



При переезде пересечения линий система управления мобильного робота получит данные, свидетельствующие о том, что направляющая линия находится одновременно и справа и слева относительно робота. Соответственно процесс принятия решения о последующих маневрах будет нарушен.





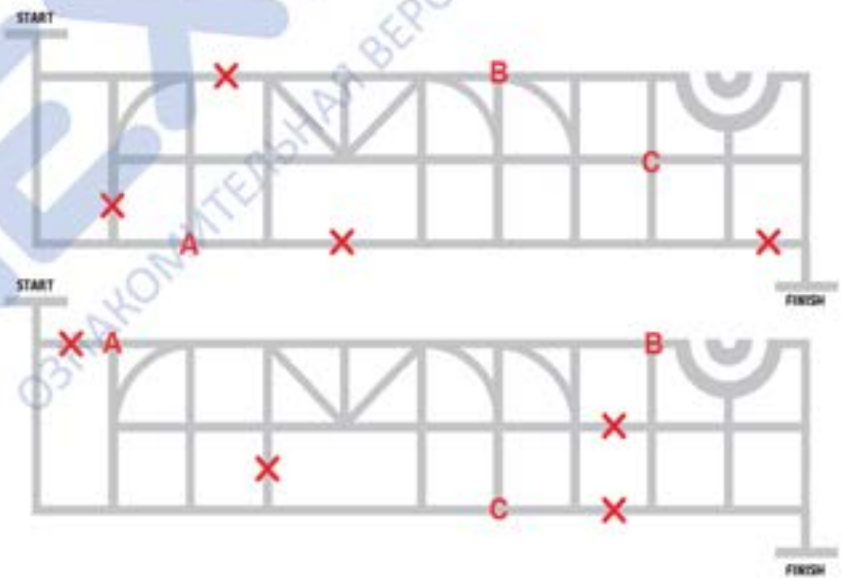
Для того распознавания пересечений направляющих линий может применяться множество различных способов, например в подобных целях достаточно часто применяются камеры. Но обработка изображений требует большой производительности бортового компьютера робота, поэтому такие решения не всегда применимы. В данной работе рассматривается способ управления мобильным роботом с помощью информации, поступающей от массива ИК-датчиков. С помощью массива из семи датчиков S1-S7 становится возможным определить местоположение пересечений линий. Поскольку вариантов наиболее вероятных пересечений достаточно много, следует настраивать управляющую программу робота на максимально возможное количество допустимых вариантов, определенных на основании показаний ИК-датчиков.

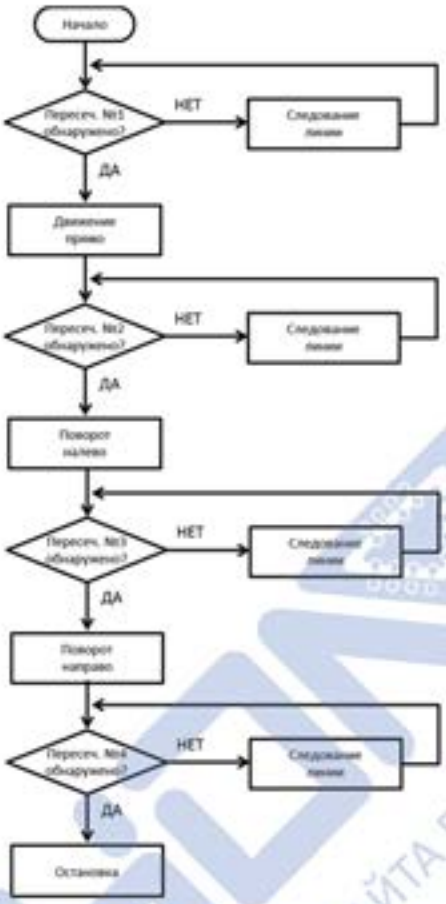
Часть № 1. Выполнение маневров вблизи перекрестков

В процессе движении вдоль маршрута, представляющего собой пересекающуюся линию, помимо отслеживания линии необходимо осуществлять выбор направления движения на каждом из перекрестков. Прохождение правильной последовательности перекрестков дает возможность проехать заданный маршрут верным образом.

В рамках данной работы предлагается разработать программу следования вдоль заданного маршрута. В качестве примера рассмотрим базовый маршрут, приведенный на рисунке. Чтобы разработать программу для движения вдоль данного маршрута на нем необходимо выделить особые точки, являющиеся пересечениями маршрута или его разрывами. Для того чтобы робот доехал до финиша, он должен следовать вдоль линии и пройти четыре указанные точки.

Движение по заданному маршруту определяется алгоритмической последовательностью, которая задает один из маневров в каждой из ключевых точек. В частности, в приведенном примере на первом пересечении маршрута робот продолжает движение в прямом направлении, на втором поворачивает налево, а на третьем – направо и следует до финишной черты.





Аналогичным образом можно разработать программу движения робота вдоль любого из более сложных маршрутов. Для этого всего лишь необходимо задать последовательность движения робота через пересечения маршрута.

Управляющая программа сводится к поочередному поиску каждой из точек заданной последовательности. Каждая из точек описывается собственной функцией, с помощью которой она распознается, а также скоростью прохождения данного участка, которая определяется на стадии инициализации программы.

```
3:  // L/R wheel maximum speed (0 ~ 1023)
4:  // Modify speed until robot run straightly in "Straightness_check_mode"
5:  l_wheel_max_speed  = 420
6:  r_wheel_max_speed  = 420
7:
8:  // time needed to judge if a node is detected
9:  node_judging_time  = 0.025sec
10:
11: // time needed to judge if a T_node is detected
12: T_node_judging_time  = 0.010sec
13:
14: // time for moving forward after detecting a node so that two wheels are on black lines.
15: node_forward_time  = 0.250sec
```

Текст программы представляет собой последовательность поочередных вызовов функций, задающих требуемое движение. Для того чтобы варьировать направление движения робота вдоль линии, необходимо всего лишь поменять очередность вызова функций.

```
37:     CALL l_node_forward
38:     CALL l_node_l_turn
39:     CALL l_node_forward
40:     CALL l_node_l_turn
41:     CALL T_node_r_turn
42:     CALL T_node_r_turn
43:     CALL T_node_l_turn
44:     CALL l_node_l_turn
45:     CALL t_node_r_turn
46:     CALL T_node_forward
47:     CALL T_node_forward
48:     CALL T_node_r_turn
49:     CALL T_node_l_turn
50:     CALL l_node_forward
51:     CALL T_node_r_turn
52:
53:     CALL stop
```

Вышеуказанные функции можно разделить на два основных типа:

1) Функции, осуществляющие поворот робота на Т-образных и Г-образных перекрестках.

2) Функции, с помощью которых робот движется до ближайшего перекрестка без каких-либо действий.

В первом случае функция состоит из двух отдельных операций – вызова функции движения вперед и вызова функции поворота в заданном направлении.

```
155: FUNCTION l_node_l_turn
156: {
157:     CALL l_node_forward
158:     CALL pivot_left
159: }
```

Функция l_node_l_turn предназначена для осуществления поворота налево на ближайшем левом пересечении. Функция состоит из двух других функций: l_node_forward, отвечающей за движение до ближайшего Г-образного перекрестка с поворотом налево, и pivot_left, за сам отвечающей за поворот налево.

```
149: FUNCTION l_node_forward
150: {
151:     LOOP WHILE ( node_detect == FALSE )
152:     {
153:         CALL follow_line
154:         CALL l_node_detect
155:     }
156:     node_detect = FALSE
157:
158:     // fast forward when detect a left node
159:     CALL fast_forward
160:
161:     High-resolution Timer = node_forward_time
162:     CALL precise_timer_standby
163: }
```

Функция l_node_forward в бесконечном цикле ищет точку пересечения траекторий с помощью функции l_node_detect. Во время поиска робот постоянно следует линии с помощью функции follow_line. После обнаружения точки пересечения маршрутов робот совершает кратковременный рывок вперед, ограниченный временем таймера, для того чтобы слегка сместиться для дальнейшего поворота налево. Данное перемещение крайне важно, для того чтобы после маневра робот оказался по центру направляющей линии.

Особое внимание следует уделить процессу распознавания точек пересечения маршрута. Очевидно, что в процессе движения робота по маршруту могут возникнуть различные ситуации, но большинство из них можно описать формальными признаками, например по срабатыванию ИК-датчиков.

Рассмотрим процесс движения робота через участок маршрута с левым поворотом. Очевидно, что в процессе движения робота часть ИК-датчиков попадет на черную линию и возникнет одна из ситуаций, проиллюстрированных ниже.

На рисунке черным цветом отмечены ИК-датчики, расположенные над черной линией, в свою очередь белым цветом – расположенные над белым участком поверхности. В процессе движения робота можно опросить каждый из датчиков и с помощью перебора вариантов определить текущее положение робота.

```
315 FUNCTION l_node_detect
316 {
317     CALL detect_black
318     IF ( black_1 > 0 && black_2 > 0 && black_3 > 0 )
319     {
320         IF ( black_5 == 0 || black_6 == 0 || black_7 == 0 )
321         {
322             // If situation is the same after "node_judging_time", robot considers that it detects a left node
323             High-resolution Timer_ = node_judging_time
324             CALL precise_timer_standby
325
326             CALL detect_black
327             IF ( black_1 > 0 && black_2 > 0 && black_3 > 0 )
328             {
329                 IF ( black_5 == 0 || black_6 == 0 || black_7 == 0 )
330                     node_detect = TRUE
331             }
332         }
333     }
334 }
```

В приведенной выше функции описывается процесс распознавания левого Г-образного поворота. Согласно приведенному алгоритму под подобной точкой маршрута понимается участок траектории, на котором срабатывают ИК-датчики № 1, № 2, № 3.

Подобным образом можно распознать любой участок маршрута. На первый взгляд это может показаться достаточно простой задачей, но стоит уделять повышенное внимание точности распознавания текущего положения. На точность работы программы могут влиять качество рабочей поверхности, скорость движения робота.

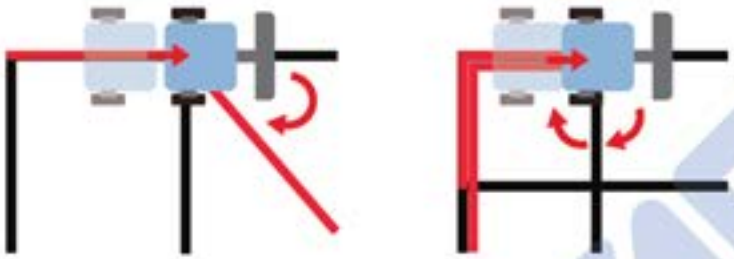
Для повышения точности работы программы в функции l_node_detect реализован механизм защиты от ложных срабатываний. Одно и то же условие проверяется дважды после программируемой задержки.

```
322:        // If situation is the same after "node_judging_time", robot considers that it detects a left node
323:        High-resolution Timer = node_judging_time
324:        CALL precise_timer_standby
325:
326:        CALL detect_black
327:        IF ( black_1 > 0 && black_2 > 0 && black_3 > 0 )
328:        {
329:            IF ( black_5 == 0 || black_6 == 0 || black_7 == 0 )
330:                node_detect = TRUE
331:        }
```

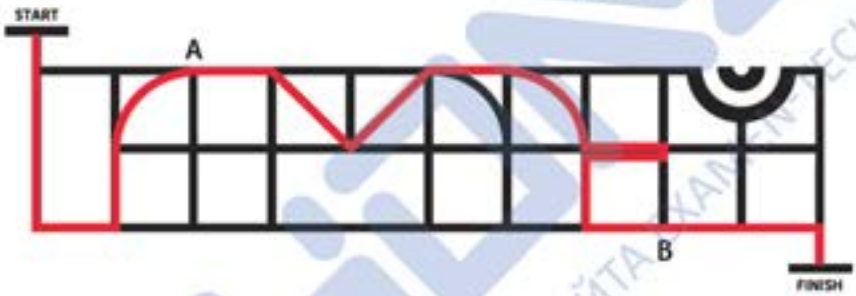
Меры повышения точности работы управляющей программы крайне важны при разработке системы управления. Пренебрежение ими может привести к некорректной работе алгоритма и всей робототехнической системы в целом.

Часть № 2. Выполнение сложных маневров

В реальных ситуациях мобильные роботы перемещаются не только по прямолинейным участкам маршрута, но и по криволинейным траекториям, а также выполняют различные маневры.



Чем больше система управления содержит в себе описаний подобных маневров, тем более сложные маршруты может преодолевать робот. Например, на подобии того маршрута, что приведен ниже на рисунке.



В целом программа управления идентична программе, рассматриваемой в предыдущей части. Также как и для любой другой программы задается последовательность прохождения узлов траектории.

```
49:     CALL r_node_forward
50:     CALL r_node_l_turn
51:     CALL l_node_l_turn
52:     CALL r_node_forward
53:     CALL r_curve_branch_r_turn
54:     CALL r_node_r_turn
55:     CALL diag_corner_l_turn
56:     CALL pivot_left
57:     CALL diag_corner_r_turn
58:     CALL r_node_forward
59:     CALL r_curve_l_turn
60:     CALL T_node_l_turn
61:     CALL pivot_left
62:     CALL T_node_l_turn
63:     CALL T_node_l_turn
64:     CALL l_node_forward
65:     CALL l_node_forward
66:     CALL T_node_r_turn
67:
68:     CALL stop
```

По сравнению с предыдущей частью работы добавились два новых типа движений – движение по дуге окружности и движение по диагонали, причем каждое из этих движений различается по направлениям.

```
493: FUNCTION diag_corner_l_turn
494: {
495:     CALL diag_corner_forward
496:     CALL pivot_left
497: }
498:
499: FUNCTION diag_corner_r_turn
500: {
501:     CALL diag_corner_forward
502:     CALL pivot_right
503: }
```

```
562: FUNCTION l_curve_branch_l_turn
563: {
564:     CALL l_curve_branch_forward
565:     CALL pivot_left
566: }
567:
568: FUNCTION l_curve_branch_r_turn
569: {
570:     CALL l_curve_branch_forward
571:     CALL pivot_right
572: }
```

Каждая из этих функций состоит из функции следования маршруту – diag_corner_forward, l_curve_branch_forward и функции поворота в требуемом направлении – pivot_left, pivot_right.

Контроль за движением робота вдоль линии под углом осуществляется с помощью ИК-датчиков № 1 и № 7, которые задают положение робота над линией. Если же робот оказывается над линией, то запускается функция follow_line, с помощью которой робот отслеживает собственное положение относительно линии и центрируется на ней с помощью ИК-датчика № 4.

```
463: FUNCTION diag_corner_forward
464: {
465:     CALL detect_black
466:
467:     LOOP WHILE ( black_1 > 0 || black_7 > 0 )
468:         CALL follow_line
469:
470:     LOOP WHILE ( black_1 == 0 && black_7 == 0 )
471:         CALL follow_line
472:
```

Следование линии нацелено в первую очередь на движение вдоль нее с ориентацией центра робота над линией. Поскольку центр робота совпадает с ИК-датчиком № 4, функция follow_line стремится минимизировать отклонения ИК-датчиков № 3 и № 5 относительно линии.

```
IF ( black_4 > 0 )
{
    IF ( black_4 > 0 && black_3 > 0 )
    {
        ID[1].Moving speed = CCW:0 + l_wheel_speed_3_4
        ID[2].Moving speed = CW:0 + r_wheel_max_speed
    }
    ELSE IF ( black_4 > 0 && black_5 > 0 )
    {
        ID[1].Moving speed = CCW:0 + l_wheel_max_speed
        ID[2].Moving speed = CW:0 + r_wheel_speed_4_5
    }
    ELSE
    {
        ID[1].Moving speed = CCW:0 + l_wheel_max_speed
        ID[2].Moving speed = CW:0 + r_wheel_max_speed
    }
}
```

Суть данного процесса сводится к выполнению ряда условий:

1) Если ИК-датчик № 4 находится над линией, то робот едет прямолинейно с максимальной скоростью.

2) Если один из ИК-датчиков № 3 или № 5 обнаружил линию, то робот поворачивается в противоположном направлении с минимальной скоростью.

Во время следования линии изменяется, в зависимости от положения робота, скорость его маневрирования. Это сделано из-за того, что в некоторых ситуациях необходимы плавные движения робота, чтобы он не съехал с линии, например при маневрировании между датчиками № 3 и № 5.

В случае же если робот отклонился от линии достаточно сильно, необходимо скорректировать его положение максимально быстро, поэтому скорость его движения увеличивается.

За изменение скорости движения робота отвечает функция change_speed, которая задает скорость вращения приводов в процентном соотношении от значения максимальной скорости.

```
FUNCTION change_speed
{
    // i.e. l_wheel_speed = 400 and speed_percent_1_7 = 50
    // result: left wheel speed = 400*50% = 200
    l_wheel_speed_1 = l_wheel_max_speed * speed_percent_1_7
    l_wheel_speed_1 = l_wheel_speed_1 / 100

    l_wheel_speed_2 = l_wheel_max_speed * speed_percent_2_6
    l_wheel_speed_2 = l_wheel_speed_2 / 100
```

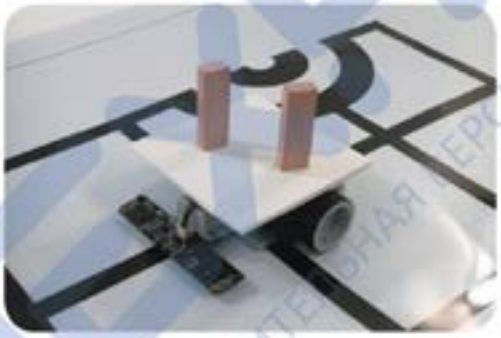
Помните, что соблюдение скоростного режима – одно из важнейших условий, влияющих на движение робота по заданному маршруту. Одно из важнейших требований к алгоритму управления мобильным роботом – соблюдение оптимальной скорости для данного участка маршрута.

Соблюдение скоростного режима – это не просто требование безопасности движения, это в первую очередь требование, позволяющее минимизировать ошибки работы управляющей программы робота. Стоит помнить, что скорость движения робота существенным образом влияет на качество распознавания узловых точек маршрута и ориентацию самого робота относительно направляющей линии.

Основная цель разработчика робототехнических систем – это обеспечение качественной и безотказной работы в процессе функционирования. Ради этого можно пожертвовать многим – производительностью, ресурсоемкостью и т.п., в том числе и быстродействием.

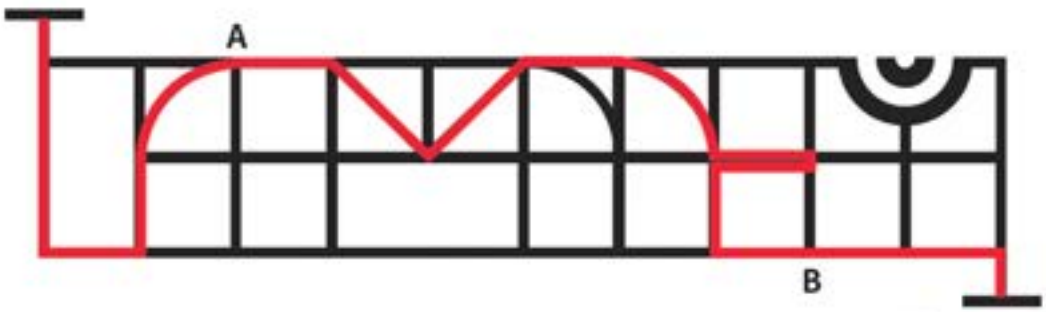
Часть № 3. Подведение итогов

Данная лабораторная работа наиболее важная среди работ, рассмотренных ранее. Это обусловлено не только сложностью преподносимого материала, но и затронутыми важными проблемами, такими как обеспечение точности и качества работы робототехнических систем.

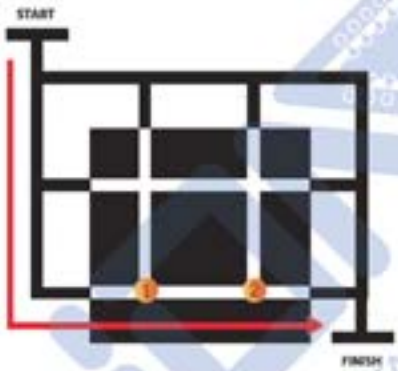


Цель данной работы продемонстрировать, что не только внешние факторы влияют на качество работы того или иного робота. Существенное влияние на процесс выполнения поставленной задачи может оказывать сам робот, функционирующий на основании программы управления.

Рассмотренные в обеих частях программы наглядно демонстрируют влияние скорости движения на качество прохождения заданной траектории. Для закрепления результатов работы можно исследовать движение робота по приведенному ниже маршруту, сочетающему в себе все возможные маневры, рассмотренные ранее.



Помимо сложности маршрута и скорости движения на функционирование робота существенное влияние оказывает качество поверхности, по которой осуществляется движение. Вполне возможна ситуация, что линия, вдоль которой должен передвигаться робот, может оказаться поврежденной или даже закрашенной.



Разработчик робототехнической системы должен предусмотреть все возможные варианты применения своего проектного решения. Чем больше всевозможных влияющих факторов будет учтено на стадии проектирования, тем точнее и качественнее будет функционировать робот.

Лабораторная работа № 5

ЭКЗАМЕН ТЕХНОЛАБ

Управление техническими системами в ручном режиме

Лабораторная работа № 5
«Управление техническими системами в ручном режиме»

Роботы и робототехнические системы бывают разных типов, как правило, они классифицируются по сложности, аппаратной составляющей, конструктивным особенностям, способу управления и многим другим параметрам.

В свою очередь для систем управления роботами существует традиционная классификация. Систему управления по способу управления подразделяются на три типа.

Классификация систем управления		
Биотехнические	**Автоматические**	**Интерактивные**
- Командные	- Программные	- Автоматизированные
- Копирующие	- Адаптивные	- Супервизорные
- Полуавтоматические	- Интеллектуальные	- Диалоговые

Биотехническими называются такие системы управления, в процессе работы которых принимает участие человек.

Командными системами управления называются такие системы, в которых осуществляется управление каждым исполнительным механизмом робота в отдельности, например управление каждым приводом манипулятора в отдельности и т.п.

Копирующие системы управления в качестве задающего движения используют движение либо полностью идентичного макета робота, либо любого другого механизма, задающего нужное движение. Наиболее популярным примером может служить экзоскелет – механизированный костюм, с помощью которого человек может орудовать достаточно тяжелыми грузами.

Полуавтоматические системы основываются на ручном управлении человеком-оператором, но в свою очередь ряд исполнительных механизмов робототехнической системы работает автоматически. Примером такой системы управления может служить захватное устройство манипулятора, приводимое в движение оператором с помощью джойстика. При управлении движением схвата манипулятора пользователь задает вектор и скорость движения, а система управления автоматически рассчитывает закон управления каждым из приводов робота в отдельности.

Автоматическими называются такие системы управления, в процессе работы которых человек не принимает никакого участия.

Программными системами управления называются такие системы, в которых управление исполнительным механизмом робота выполняется по жестко заданной программе. Примерами таких систем управления могут служить системы управления производственных механизмов, например конвейеров или портальных кранов и погрузчиков.

Адаптивные системы управления работают, подстраиваясь под изменение состояния окружающей среды, которое регистрируется с помощью различных сенсорных устройств. Например, система управления робота, движущегося вдоль линии, является адаптивной, поскольку в процессе перемещения робота с помощью датчиков определяется положение линии и относительно нее корректируется положение робота.

Интеллектуальные системы управления – наиболее сложные системы управления, функционирующие на основании информации о состоянии окружающей среды, полученной от различных сенсорных устройств. Отличие интеллектуальных систем управления от адаптивных заключается в том, что в них анализируется состояние системы в данный момент и прогнозируется дальнейший сценарий развития событий. Также к интеллектуальным системам относят системы, работающие с базами данных информации и выдающими собственный закон управления в соответствии с одним из сценариев, заложенным в базу данных и который максимально соответствует текущей ситуации.

Интерактивными системами управления называются такие системы управления, которые работают как автоматически, так и под управлением человека.

Автоматизированные системы управления представляют собой системы управления, которые работают автоматически, но также и частично под управлением человека.

Супервизорные системы управления – системы управления, работающие полностью автономно, в которых человеку отводится роль наблюдателя или контролера качества выполняемых работ.

Диалоговые системы управления работают в прямом контакте с человеком, при этом стратегия работы системы управления выбирается на основании текущего состояния и целей, преследуемых как программой системы управления, так и человеком-оператором. К подобным системам можно отнести системы управления жестами или системы голосового управления.

В рамках данной работы предлагается разработать систему управления электронного пианино. Электронное пианино представляет собой устройство, воспроизводящее звуковые сигналы различной нотной тональности. Управляется данное устройство пользователем в ручном режиме, который нажимает на клавиши – специальные пластинки, расположенные над ИК-датчиками. При нажатии на каждую из таких клавиш происходит срабатывание ИК-датчика, которое регистрирует система управления. В зависимости от номера нажатого датчика воспроизводится одна из мелодий.

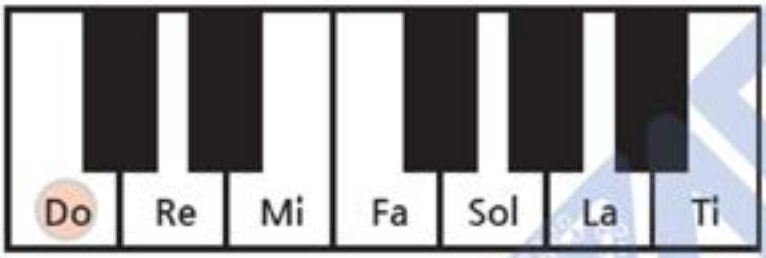
Sensor	S1	S2	S3	S4	S5	S6	S7
Notes	Do	Re	Mi	Fa	Sol	La	Ti

Как обычно данная работа состоит из двух практических частей, ориентированных на детальное изучение материала. В первой части рассматриваются вопросы воспроизведения звуков и задания их тональности. Вторая часть посвящена вопросам изменения октав с помощью кнопок управления, расположенных на контроллере СМ-530.

Быть может, некоторые не смогут отнести разрабатываемое устройство к роботам, но несмотря на это электронное пианино является ярким примером технических систем с ручным управлением, которые встречаются повсеместно в обыденной жизни.

Часть № 1. Ручное управление устройством с помощью массива ИК-датчиков

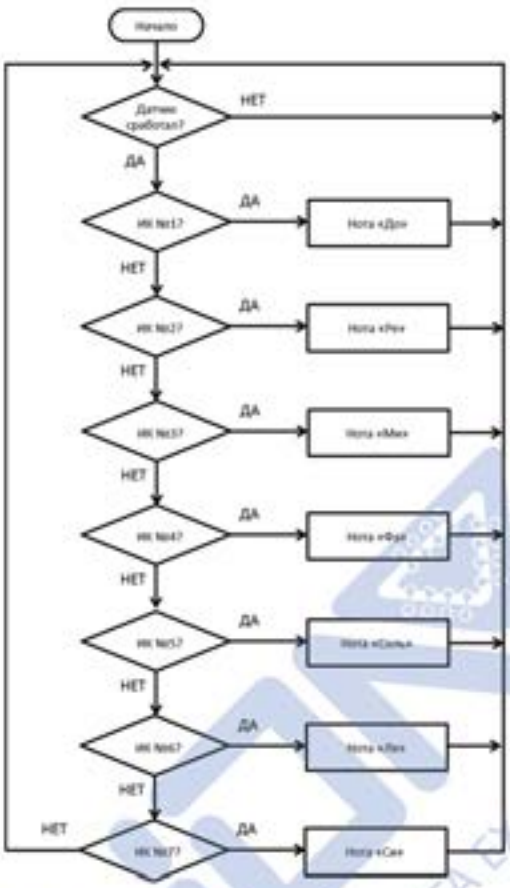
Для ручного управления электронным пианино необходимо спроецировать способ работы с традиционным пианино на технические средства, лежащие в основе устройства. Для моделирования нажатий на клавиши традиционного пианино используется массив из 7 ИК-датчиков, касание которых будет воспроизводить одну из нот.



Воспроизведение звуков и мелодий осуществляется с помощью встроенного в контроллер CM-530 динамика. Каждый звук, воспроизводимый динамиком, получается в результате наложения нот с различными октавами, как обычная музыка.

В среде программирования RoboPlus все возможные звуки реализованы в виде специальных функций, доступных пользователю. Вызов функции осуществляется с помощью окна панели управления во вкладке Musical Scale, где во всплывающем меню можно выбрать одну из них, в том числе и в зависимости от октавы.

В дальнейшем каждая из функций может быть вызвана в соответствующем условии срабатывания одного из ИК-датчиков.



Для воспроизведения каждой из нот в начале программы задается временной интервал. Далее задается порог срабатывания каждого ИК-датчика, причем данная конструкция включает в себя процесс сравнения показаний датчика и заданного значения, т.е. заменяет собой оператор IF.

```
1:  START PROGRAM
2:  {
3:      // ********* the variables below may need its values modified for proper operations*********//
4:      // scale duration timer
5:      scale_play_duration  = 0.5sec
6:
7:      // default threshold values for each IR sensor
8:      // if detected value is higher than default threshold value, the scale will play.
9:      // if problems persist with scale playback,
10:     // press the array's left button to initialize default threshold values
11:     ID[100]: IR Threshold 1  = 150
12:     ID[100]: IR Threshold 2  = 150
13:     ID[100]: IR Threshold 3  = 200
14:     ID[100]: IR Threshold 4  = 250
15:     ID[100]: IR Threshold 5  = 200
16:     ID[100]: IR Threshold 6  = 150
17:     ID[100]: IR Threshold 7  = 150
18:
19:     Buzzer time  = Play Melody
20:     Buzzer index  = Melody3
21:     WAIT WHILE ( Buzzer time  > 0 )
```

Алгоритм программы выполняется в бесконечном цикле, в котором анализируется, какие из ИК-датчиков сработали, в зависимости от нажатой комбинации клавиш.

```
35:  IF ( key == 1 || key == 2 || key == 3 || key == 4 || key == 5 || key == 6 || key == 7 )
36:  {
37:      IF ( key == 1 )
38:          scale = Do(15)
39:      ELSE IF ( key == 2 )
40:          scale = Re(17)
41:      ELSE IF ( key == 3 )
42:          scale = Mi(19)
43:      ELSE IF ( key == 4 )
44:          scale = Fa(20)
45:      ELSE IF ( key == 5 )
46:          scale = Sol(22)
47:      ELSE IF ( key == 6 )
48:          scale = La(24)
49:      ELSE IF ( key == 7 )
50:          scale = Shi(26)
51:
52:      Buzzer time = scale_play_duration
53:      Buzzer index = scale
54:      WAIT WHILE ( Buzzer time > 0 )
55:  }
```

Как было сказано ранее, каждому датчику соответствует определенная нота, которая воспроизводится в случае его срабатывания. Мелодия воспроизводится в течение времени scale_play_duration, а тип воспроизводимой мелодии соответствует значению переменной scale.

Отдельно стоит рассмотреть процесс определения сработавшего датчика. Впервые в цикле лабораторных работ мы сталкиваемся с понятием «прерывание». Прерывание – это сигнал контроллеру, сообщающий о происшествии какого-либо события. Прерывания бывают внешними, срабатывающими на какое-либо воздействие на устройство из вне, например нажатие кнопок или срабатывание датчика. Внутренние прерывания срабатывают на какое-либо изменение в процессе выполнения кода программы, например окончание отсчета таймера, обращение к недопустимым адресам памяти или выполнение запрещенных операций, таких как деление на ноль и т.п. Прерывания бывают также и программными, подобные прерывания представляют собой программы, созданные разработчиками. Как правило, такие прерывания используются в специализированных программах-драйверах для работы со внешними устройствами и сигнализируют о таких событиях, как подключение или отключение устройства, попытка передачи или получения данных и т.п.

В среде программирования RoboPlus есть специальные средства для реализации программных прерываний.

```
60: CALLBACK
61: {
62:     // when a key is "pressed" it goes from 0 [low] to 1 [high] (high-low logic)
63:     // instead of inverted
64:     key = 1+2+3+4+5+6+7 - ID[100]: IR Obstacle Detected
65: }
```

Конструкция CALLBACK является функцией, реализующей программное прерывание, которое может разработать пользователь. В рассматриваемом примере прерывание работы контроллера вызывается срабатыванием массива ИК-датчиков, т.е. если в какой-либо момент времени сработал ИК-датчик, моментально изменяется значение переменной key, что влечет за собой изменение процесса выполнения основной программы. В данном случае после изменения параметра key происходит воспроизведение соответствующей ему ноты.

К прерываниям предъявляются некоторые требования, обычно требования немного отличаются друг от друга в зависимости от типа программируемого контроллера и среды разработки. В случае использования среды RoboPlus и контроллера CM-530 основные требования следующие:

1) В программе может использоваться только одно прерывание.
2) Функция – обработчик прерывания может быть без имени.
3) Функция – обработчик прерывания должен быть описан в теле главной функции.
4) Объем функции – обработчик прерываний не должен превышать 512 байт.

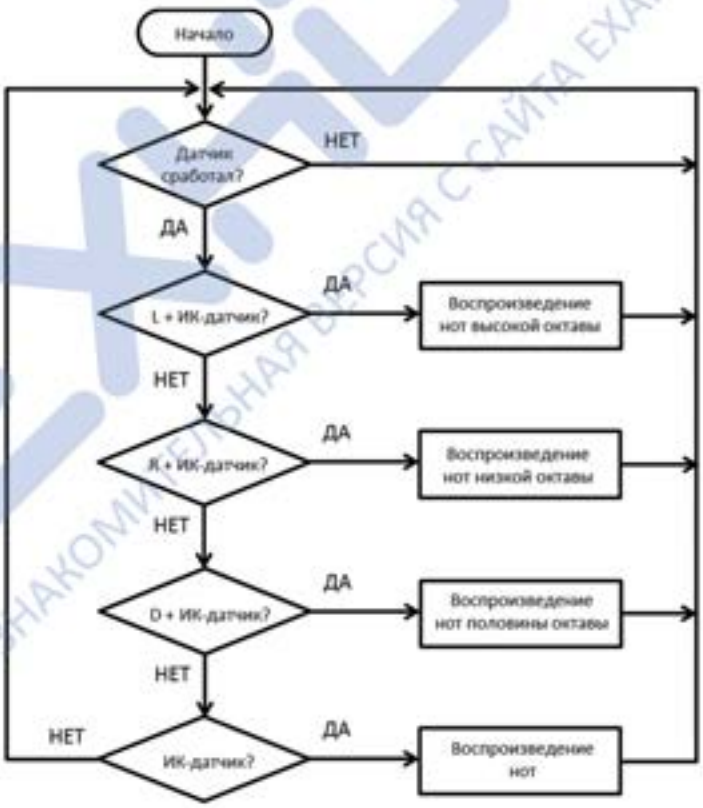
Применение прерываний дает возможность разработать систему управления, оперативно реагирующую на внешние воздействия или команды оператора. Благодаря тому, что переход к выполнению программы обработчика прерываний происходит почти мгновенно, система управления может своевременно отреагировать на внешний сигнал и скорректировать свою работу.

Часть № 2. Ручное управление устройством

с помощью массива ИК-датчиков и кнопок контроллера СМ-530

Рассмотрим более сложный процесс управления электронным пианино, а именно задание в ручном режиме нот и октав одновременно. Ноты задаются таким же образом, как и в предыдущей части работы, а октавы задаются с помощью нажатия кнопок управления контроллера СМ-530.

Запрограммируем кнопки L, R, D контроллера СМ-530 на воспроизведение высокой октавы, низкой октавы и половины октавы соответственно.



Согласно вышеприведенному алгоритму управление электронным пианино осуществляется следующим образом: зажимается одна из кнопок, соответствующих октаве, а потом нажимается «клавиша» пианино, воспроизводящая ноту.

Управляющая программа идентична рассмотренной в первой части. Отличие заключается в добавленном условии, которое рассматривает, какая из кнопок контроллера нажата в данный момент.

```
38:     IF ( Button == L )
39:     {
40:         IF ( key == 1 )
41:             scale = Do(27)
42:         ELSE IF ( key == 2 )
43:             scale = Re(29)
44:         ELSE IF ( key == 3 )
45:             scale = Mi(31)
46:         ELSE IF ( key == 4 )
47:             scale = Fa(32)
48:         ELSE IF ( key == 5 )
49:             scale = Sol(34)
50:         ELSE IF ( key == 6 )
51:             scale = La(36)
52:         ELSE IF ( key == 7 )
53:             scale = Shi(38)
54:     }
```

Мелодии, воспроизводимые в каждом из случаев, выбираются из числа стандартных функций, реализованных в среде RoboPlus.

L +

1	Do	➡ High Do	5	Sol	➡ High Sol
2	Re	➡ High Re	6	La	➡ High La
3	Mi	➡ High Mi	7	Ti	➡ High Ti
4	Fa	➡ High Fa			

R +

1	Do	➡ Low Do	5	Sol	➡ Low Sol
2	Re	➡ Low Re	6	La	➡ Low La
3	Mi	➡ Low Mi	7	Ti	➡ Low Ti
4	Fa	➡ Low Fa			

D +

1	Do	➡ Semi-tone High Do	5	Sol	➡ Semi-tone High Sol
2	Re	➡ Semi-tone High Re	6	La	➡ Semi-tone High La
3	Mi	➡ Semi-tone High Mi	7	Ti	➡ Semi-tone High Ti
4	Fa	➡ Semi-tone High Fa			

Таким образом, комбинируя отдельные ноты и нотные тональности, можно воспроизвести любую мелодию.

Часть № 3. Подведение итогов

В рамках данной работы был рассмотрен процесс разработки системы ручного управления для электронного пианино. Помимо этого были рассмотрены особенности применения прерываний, как одного из важнейших инструментов разработки управляющих программ для программируемых контроллеров.

Для закрепления полученных навыков предлагается рассмотреть программу, с помощью которой можно записать в память контроллера произвольную мелодию. Воспроизводя с помощью «клавиш» электронного пианино звуки, пользователь изменяет значение переменных scale.

```
scale   = 0
scale_1  = scale
scale_2  = scale
scale_3  = scale
scale_4  = scale
scale_5  = scale
scale_6  = scale
scale_7  = scale
scale_8  = scale
scale_9  = scale
scale_10  = scale
```

```
CALL Input
IF ( play_all  ==  TRUE  )
    JUMP Start
scale_1  = scale

CALL Input
IF ( play_all  ==  TRUE  )
    JUMP Start
scale_2  = scale

CALL Input
IF ( play_all  ==  TRUE  )
    JUMP Start
scale_3  = scale
```

Запись воспроизводимых звуков производится до тех пор, пока нажата кнопка S или пока количество воспроизводимых звуков не превысит 30. После этого с помощью повторного нажатия кнопки S можно воспроизвести мелодию.

Рассмотренная в данной работе задача может показаться достаточно простой и не имеющей никакого отношения к робототехнике, но это отнюдь не так. Данная работа иллюстрирует особенности процесса разработки систем управления ручного типа, которые достаточно часто встречаются в повседневной жизни. Причем объектом оправления таких систем управления не обязательно должен быть робот или сложный механизм, в качестве него может быть любое электронное устройство.

Лабораторная работа № 6

ЭКЗАМЕН ТЕХНОЛАБ

Применение сервоприводов для управления движением роботов

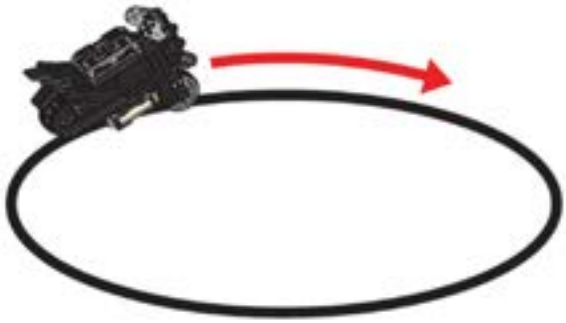
Лабораторная работа № 6
«Применение сервоприводов для управления движением роботов»

Точное управление движением роботов и производственных механизмов – крайне важная задача. От точности позиционирования робота или его исполнительного механизма зависит качество выполняемой работы или производимой продукции.

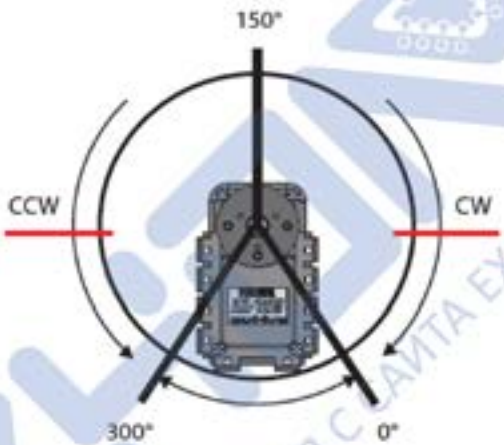
Существует множество методов, благодаря которым обеспечивается точность перемещения исполнительного механизма роботов, но поскольку большинство механизмов приводятся в движение сервоприводами, в первую очередь следует обращать внимание на точность работы привода.

В случае если от привода требуется точное перемещение исполнительного механизма робота на определенный угол, применяются системы управления с обратной связью по положению. Подобные системы отслеживают текущее положение вала привода, благодаря чему привод можно остановить в нужно позиции.

Иногда совокупность привода и системы управления называют сервоприводом. Как правило, данное название относится в первую очередь к модульным устройствам, обычно находящимся в одном корпусе. Также это название прочно закрепилось за модельными приводами, применяемыми в различных радиоуправляемых моделях и роботах.



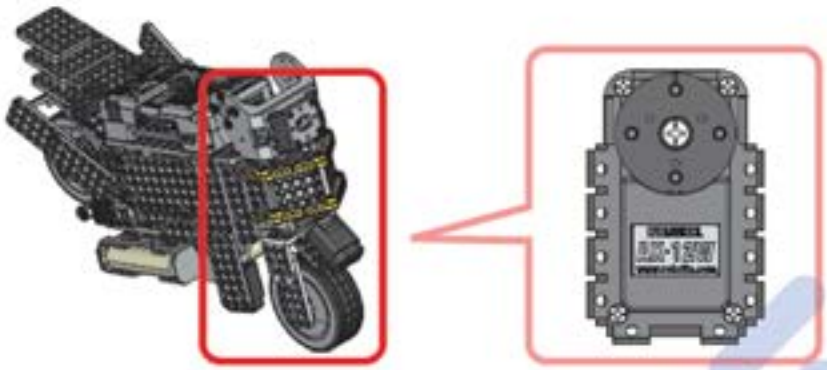
Сервопривода получили широкое применение в моделизме ввиду большой доступности и возможности обеспечивать позиционирование каких-либо составных частей модели точно под определенным углом.



Единственным фактором, ограничивающим применение модельных сервоприводов, является наличие ограничения угла вращения выходного вала. Данное ограничение вызвано тем, что в качестве датчика обратной связи применяется потенциометр, имеющий ограничение по углу вращения. Потенциометр или переменный резистор в зависимости от угла поворота вала изменяет собственное сопротивление, что влечет за собой некоторое падение напряжения на его выводах, которое можно использовать в качестве управляющего сигнала, задающего положение вала однозначно.

Ранее в предыдущих работах нами рассматривались сервопривода Dynamixel в качестве приводов, отвечающих за передвижение робота. В этих работах привода использовались в режиме управления скоростью и могли осуществлять постоянное вращение по и против часовой стрелки.

В рамке данной работы предлагается рассмотреть применение сервопривода в режиме управления положением. В качестве робота для отладки системы управления предлагается использовать модель мотоцикла, у которого рулевой привод должен управляться по положению, чтобы регулировать угол поворота в процессе движения.



При разработке конструкции робота нужно помнить, что при работе в режиме управления положением, сервопривод имеет ограниченный углов вращения, поэтому в конструкции робота нужно учитывать наличие таких положений привода, которые он никогда не сможет достигнуть.

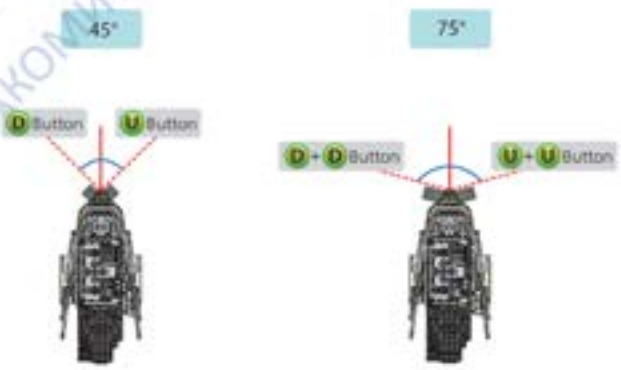
Часть № 1. Ручное управление сервоприводом

с помощью кнопок контроллера СМ-530

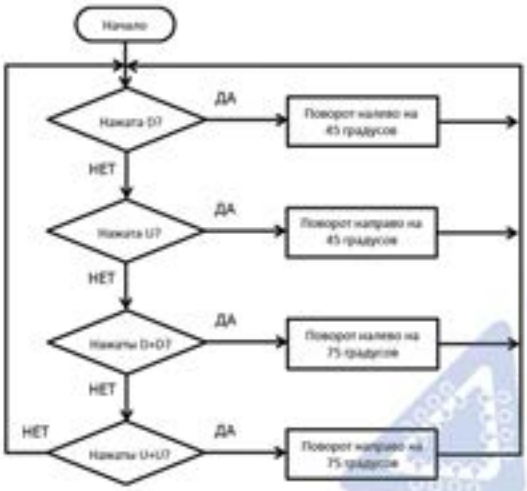
В рамках данной работы предлагается разработать систему управления роботом-мотоциклом, осуществляющую поворот его рулевого привода на один из заданных углов. Управление поворотом робота осуществляется в ручном режиме с помощью кнопок контроллера СМ-530.

Проектируемый робот предназначен для езды вдоль линии. При перемещении вдоль линии чрезвычайно важно своевременно выполнять маневры, чтобы робот не съехал с линии. Поскольку кривизна траектории может быть произвольной, радиус поворота робота оказывается переменным. Поскольку радиус поворота робота определяется поворотом его рулевого колеса, для перемещения по траекториям с малой кривизной зададим минимальное отклонение рулевого механизма, а для перемещения по траекториям с большой кривизной зададим большой угол поворота.

Для удобства управления роботом назначим кнопки D и U в качестве отвечающих за направление поворота, а радиус поворота будет определяться количеством нажатий на них.



В рамках данной работы необходимо написать программу управления роботом, которая в зависимости от команды пользователя осуществляла бы поворот рулевого механизма робота на заданный угол.



Для того чтобы сервопривод осуществил поворот в качестве управляющих параметров, необходимо задать конечное положение в абсолютных координатах от 0 до 1023. Значение переменных, характеризующих положение сервопривода, задается в функции инициализации.

```
75: FUNCTION initialize
76: {
77:     // steer position values to achieve angles
78:     left_45  = 206
79:     left_75  = 103
80:     right_45  = 512
81:     right_75  = 615
82:     straight  = 359
83:
84:     standby_time  = 0.640sec
85:
```

Также на стадии инициализации осуществляется базовая настройка сервоприводов, а именно: задается режим работы сервоприводов, скорость вращения и стартовое положение.

```
86:     ID[1]: Moving speed = CCW:0
87:     ID[2]: Moving speed = forward_speed
88:     ID[2]: Goal position = straight
89: }
```

В разрабатываемом роботе применяется два сервопривода, первый ID[1] применяется в качестве ведущего привода, поэтому он настроен на режим постоянного вращения со скоростью, задаваемой переменной forward_speed; второй сервопривод ID[2] применяется в качестве привода рулевого механизма, и он настроен на режим работы по положению.

Работа программы начинается с вызова функции инициализации и ожидания нажатия одной из управляющих кнопок.

```
18:     CALL initialize
19:
20:     ENDLESS LOOP
21:     {
22:         CALL buzzer_button
23:         CALL standby_D_U
```

Если пользователь нажимает не на ту кнопку, то контроллер воспроизводит предупреждающий звуковой сигнал, вызываемый функцией buzzer_button.

```
91: FUNCTION standby_D_U
92: {
93:     WAIT WHILE ( Button != D && Button != U )
94:     CALL buzzer_button
95: }

189: FUNCTION buzzer_button
190: {
191:     Buzzer time = 0.1sec
192:     Buzzer index = Mi(7)
193:     WAIT WHILE ( Buzzer time > 0.0sec )
194: }
```

Программа управления сводится к бесконечному циклу ожидания нажатия одной из управляющих кнопок.

```
25:         IF ( Button == D )
26:         {
27:             CALL button_standby
28:             CALL standby_S_D
29:
30:             IF ( Button == S )
31:             {
32:                 CALL standby
33:                 CALL left_turn_45
34:             }
35:
36:             ELSE IF ( Button == D )
37:             {
38:                 CALL button_standby
39:                 CALL standby_S
40:                 CALL standby
41:                 CALL left_turn_75
42:             }
```

Как только пользователь нажал на одну из кнопок управления, программа переходит в режим ожидания либо подтверждения текущей команды, либо ожидания следующей команды. Подтверждение текущей команды осуществляется с помощью нажатия на кнопку S, а следующая команда задается повторным нажатием такой же, как и в первый раз, кнопки.

```
97: FUNCTION button_standby
98: {
99:     WAIT WHILE ( Button != -- )
100: }
101:
102: FUNCTION standby_S_D
103: {
104:     WAIT WHILE ( Button != S && Button != D )
105:     CALL buzzer_button
106: }
107:
108: FUNCTION standby_S
109: {
110:     WAIT WHILE ( Button != S )
111:     CALL buzzer_button
112: }
```

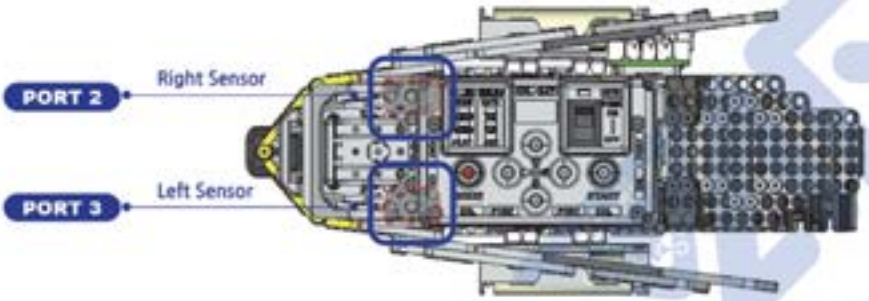
Поворот робота осуществляется в заданном направлении и на заданный угол, что задается функциями, такими как right_turn_45 и им подобными.

```
138: FUNCTION right_turn_45
139: {
140:     ID[2]: Goal position = right_45
141:     WAIT WHILE ( ID[2]: Moving == TRUE )
142:     ID[1]: Moving speed = forward_speed
143:     CALL standby_45
144:     CALL stop
145: }
146:
147: FUNCTION right_turn_75
148: {
149:     ID[2]: Goal position = right_75
150:     WAIT WHILE ( ID[2]: Moving == TRUE )
151:     ID[1]: Moving speed = forward_speed
152:     CALL standby_75
153:     CALL stop
154: }
```

Каждая функция переводит привод ID[2] в требуемое положение, а после запускает привод ID[1] для движения с заданной скоростью. В результате получается, что робот движется по окружности за счет того, что его рулевой привод повернут в сторону, а ведущий привод движется с заданной скоростью.

Часть № 2. Автоматическое движение робота с рулевым управлением вдоль линии

В предыдущей части работы рассматривалась задача ручного управления движением робота. На протяжении всех предыдущих работ нами рассматривались примеры разработки автоматических систем управления и данная работа не исключение. В этой части необходимо разработать модель роботизированного мотоцикла с рулевым управлением и двумя ИК-датчиками для определения черной линии.



Управляющая программа почти полностью идентична ранее рассматриваемым программам, управляющим движением робота вдоль линии с использованием двух ИК-датчиков. Единственным отличием этой программы от ранее используемых программ является использование алгоритма управления рулевым механизмом робота.

```
10: Start :
11:     // sec parameters so the bike can follow the line
12:     return_time  = 0.250sec
13:     steer_speed  = CCW:250
14:     forward_speed  = CCW:500
15:
16:     CALL initialize
17:
18:     ENDLESS LOOP
19:     {
20:         IF ( PORT[3]  <=  left_threshold  &&  PORT[2]  >  right_threshold  )
21:         {
22:             CALL turn_left
23:             CALL turn_standby
24:         }
25:         ELSE IF ( PORT[3]  >  left_threshold  &&  PORT[2]  <=  right_threshold  )
26:         {
27:             CALL turn_right
28:             CALL turn_standby
29:         }
30:         ELSE
31:             CALL forward
32:     }
```

Сразу же после старта программа переходит в бесконечный цикл, анализирующий показания боковых ИК-датчиков, определяющих наличие черной линии. В случае обнаружения линии одним из датчиков, робот выполняет маневр в противоположную сторону.

```
62: FUNCTION forward
63: {
64:     ID[2].Goal position = forward
65:     ID[1].Moving speed = forward_speed
66: }
67:
68: FUNCTION turn_left
69: {
70:     ID[2].Goal position = left
71:     ID[1].Moving speed = forward_speed
72: }
73:
74: FUNCTION turn_right
75: {
76:     ID[2].Goal position = right
77:     ID[1].Moving speed = forward_speed
78: }
```

Маневры робота осуществляются за счет перевода рулевого механизма в одно из положений – прямо, налево или направо, с помощью функций forward, turn_left, turn_right.

Проведите испытания разработанного робота на различных трассах при езде вдоль линии разной кривизны. Очевидно, что для разных случаев необходим поворот робота на определенный угол – в первом случае угол поворота рулевого механизма наибольший, а в третьем – наименьший.

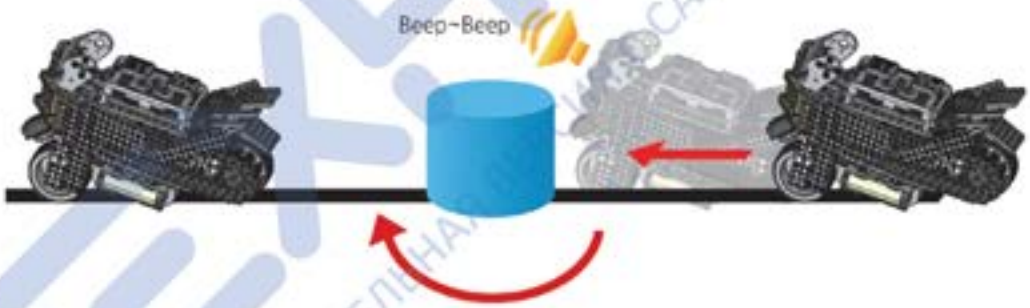
В ходе испытаний установите минимальный радиус поворота робота, оптимальный для данной трассы, и настройте угол поворота рулевого механизма таким образом, чтобы робот отрабатывал маневры наиболее точно.

Часть № 3. Выполнение сложных маневров

В предыдущих работах мы неоднократно рассматривали ситуации, в которых робот объезжал различные препятствия на его пути. Данная работа также не исключение, в данной части предлагается рассмотреть процесс выполнения маневров по объезду препятствий роботом с рулевым управлением.

Для того чтобы робот мог обнаружить объект на своем пути, он оснащается ИК-датчиком, расположенным по ходу движения.

Объекты, препятствующие движению робота, обнаруживаются с помощью ИК-датчика, расположенного на арке переднего колеса.



При движении вдоль направляющей линии робот с помощью двух ИК-датчиков определяет собственную ориентацию относительно маршрута. В случае обнаружения объекта на своем пути робот останавливается напротив него, воспроизводит звуковой сигнал и выжидает 3 секунды. После чего, если объект по прежнему препятствует движению, робот осуществляет маневр по его объезду.

```
18:  ENDLESS LOOP
19:  {
20:      IF ( PORT[1] > front_threshold )
21:      {
22:          CALL stop
23:          CALL buzzer
24:
25:          Timer = obstacle_judging_time
26:          WAIT WHILE ( PORT[1] > front_threshold && Timer > 0 )
27:
28:          IF ( PORT[1] > front_threshold )
29:          {
30:              CALL turn_left_obstacle
31:              CALL turn_standby
32:              CALL turn_right_obstacle
33:              WAIT WHILE ( PORT[3] > left_threshold )
34:          }
35:      }
36:
37:      ELSE IF ( PORT[3] <= left_threshold && PORT[2] > right_threshold )
38:      {
39:          CALL turn_left
40:          WAIT WHILE ( PORT[2] > right_threshold )
41:      }
42:
```

Объезд препятствия осуществляется с помощью функций turn_left_obstacle и turn_right_obstacle, задающих направление движения. Время движения задается с помощью функции turn_standby, с ее помощью можно задать продолжительность времени, необходимого для объезда определенного препятствия.

После того как робот выполнил маневр, ему необходимо вернуться на линию и продолжить движение вдоль нее.

```
37:      ELSE IF ( PORT[3] <= left_threshold && PORT[2] > right_threshold )
38:      {
39:          CALL turn_left
40:          WAIT WHILE ( PORT[2] > right_threshold )
41:      }
```

С помощью ИК-датчиков PORT[3] и PORT[2] робот определяет свое положение относительно линии и выполняет поворот в одну из сторон, чтобы объехать центрироваться относительно нее.

Часть № 4. Подведение итогов

В рамках данной работы были изучены основы управления сервоприводами и рассмотрены особенности их применения при проектировании роботов. В рамках практической части был разработан робот с рулевым управлением и разработана его система управления.

Основная задача данной работы – иллюстрация границ применимости сервоприводов. Важно понимать, что сервопривод по сравнению с обычным приводом – это не только технически сложное устройство, но и крайне дорогое. Поэтому применение сервоприводов при проектировании роботов должно быть технически и экономически обосновано.

Применение сервоприводов обосновано в системах, где необходимо обеспечить точность перемещения исполнительных механизмов или осуществлять перемещение исполнительных механизмов робота в ограниченном диапазоне рабочей зоны.

Лабораторная работа № 7

ЭКЗАМЕН ТЕХНОЛАБ

Основы локальной навигации мобильных роботов

Лабораторная работа № 7
«Основы локальной навигации мобильных роботов»

Работа мобильных роботов сопряжена с постоянным перемещением в пределах рабочей зоны. В настоящее время подобные роботы встречаются все чаще и с каждым днем они становятся все более функциональными и сложными. По мере роста функциональности роботов растет и сложность решаемых ими задач. В настоящее время уже повсеместно встречаются автономные мобильные роботы – робокары, перемещающиеся по производственным помещениям, также стали довольно востребованы сервисные роботы – роботы для проведения экскурсий в музеях и выставочных центрах, роботы – официанты в кафе и ресторанах. Такие роботы работают в тесном контакте с окружающими объектами и людьми, поэтому к точности и безопасности их перемещений предъявляются слишком высокие требования.

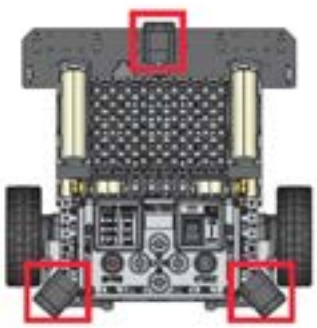
В предыдущих работах уже рассматривались примеры решения задач перемещения мобильного робота в рабочей зоне. Также уделялось внимание процессу выполнения простейших маневров по объезду препятствий, встречающихся на пути.

Зачастую в реальных ситуациях мобильный робот сталкивается с препятствиями, расположенными произвольным образом в рабочей зоне, а также и с перемещающимися объектами. Каждый из таких объектов может быть потенциальным препятствием, поэтому в процессе своего движения мобильный робот должен иметь информацию обо всех объектах вблизи, чтобы в случае необходимости иметь возможность отреагировать на них.

Сбор, обработка и систематизация информации об объектах вблизи робота называется составлением локальной карты окружающего пространства. Использование локальной карты в задачах планирования маршрута робота или объезда препятствий называется локальной навигацией.

При разработке мобильных роботов нужно учитывать, какие задачи поставлены перед мобильным роботом, чтобы подобрать необходимые сенсорные устройства для их решения. Так, например, для решения задачи перемещения внутри рабочей зоны робот может быть оснащен дорогими лазерными сканирующими дальномерами и GPS-устройствами для определения собственного положения, тогда как для решения задачи локальной навигации мобильный робот может быть оснащен простыми ультразвуковыми или инфракрасными датчиками по периметру.

Рассмотрим типичную для роботов из данного образовательного модуля задачу. Нам уже не раз встречались мобильные роботы для перемещения вдоль линии, каждый из них был оснащен набором ИК-датчиков или ИК-массивом для обнаружения линии. Поскольку задача перемещения вдоль линии является главной, то в качестве основной сенсорной системы, на основании показаний которой перемещается робот, выбираем систему ИК-датчиков.



В случае если передвижению мобильного робота могут препятствовать различные объекты, по периметру робота можно расположить ИК-датчики, с помощью которых робот может обнаруживать препятствия с каждой из сторон. Количество подобных датчиков определяется возможностями программируемого контроллера, габаритами робота и объектов в зоне его функционирования.

В рамках данной работы предлагается сконструировать робота, оснащенного тремя ИК-датчиками, расположенными спереди и по бокам робота. С помощью этих датчиков робот должен обнаруживать препятствия, возникающие по ходу его движения.

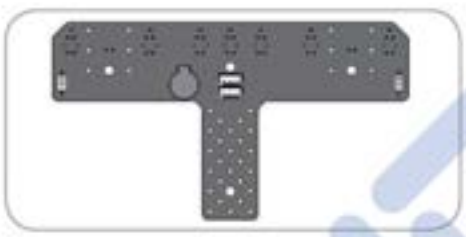
Предлагается рассмотреть ряд приемов, которые могут быть использованы разработчиками как метод усовершенствования системы управления мобильного робота, благодаря чему она сможет выполнять более широкий спектр задач.

Часть № 1. Обнаружение объектов или неровностей поверхности в процессе движения

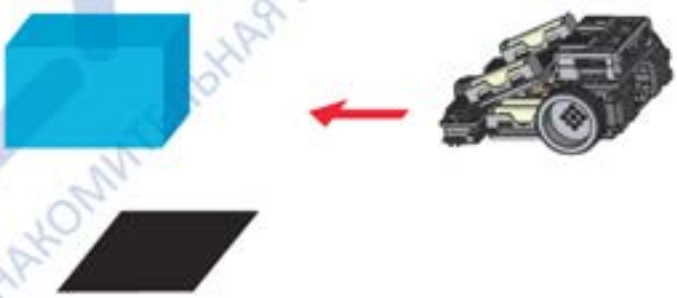
Достаточно часто при движении мобильных роботов по пересеченной местности возникают ситуации, в которых робот не может преодолеть то или иное препятствие на своем пути.

Очень часто разработчики мобильных роботов акцентируют свое внимание на проблемах взаимодействия робота и окружающей среды в процессе движения – это могут быть сенсорные системы для обнаружения препятствий, системы определения уровня вибраций и системы стабилизации и др.

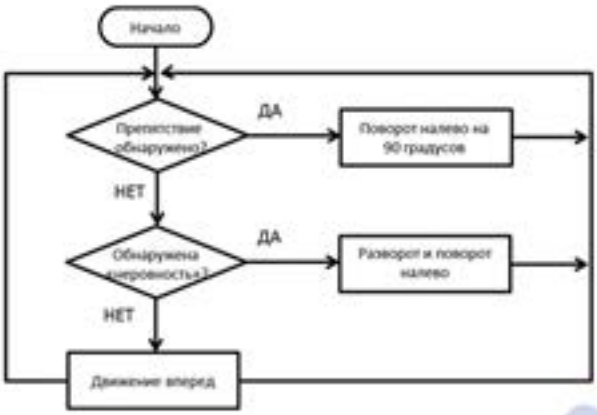
В рамках данной части предлагается рассмотреть модель робота, анализирующего наличие препятствий на собственном пути, а также оценку возможности дальнейшего перемещения. Под этим понимается анализ рабочей поверхности, например поиск обвалов и ям на пути и т.п.



Разрабатываемый нами робот оснащается массивом ИК-датчиков для исследования поверхности по которой он перемещается и тремя ИК- датчиками, расположенными по периметру робота, для обнаружения препятствий во время движения.



Алгоритм движения робота достаточно прост – робот движется прямолинейно и если он обнаруживает препятствие у себя на пути, он совершает поворот налево, если же на пути робота встречается обрыв или область черного цвета («неровность» поверхности), робот разворачивается и едет в противоположную сторону и налево.



По большому счету программа сводится к единственному бесконечному циклу, анализирующему показания ИК-датчика, подключенного к PORT[6], а также срабатывание массива ИК-датчиков в режиме поиска препятствий.

```
23: Start :
24:     ENDLESS LOOP
25:     {
26:         IF ( PORT[6] > front_threshold )
27:             CALL pivot_left
28:         ELSE IF ( ID[100] IR Obstacle Detected > 0 )
29:         {
30:             CALL reverse
31:             CALL pivot_left
32:         }
33:         ELSE
34:             CALL forward
35:     }
```

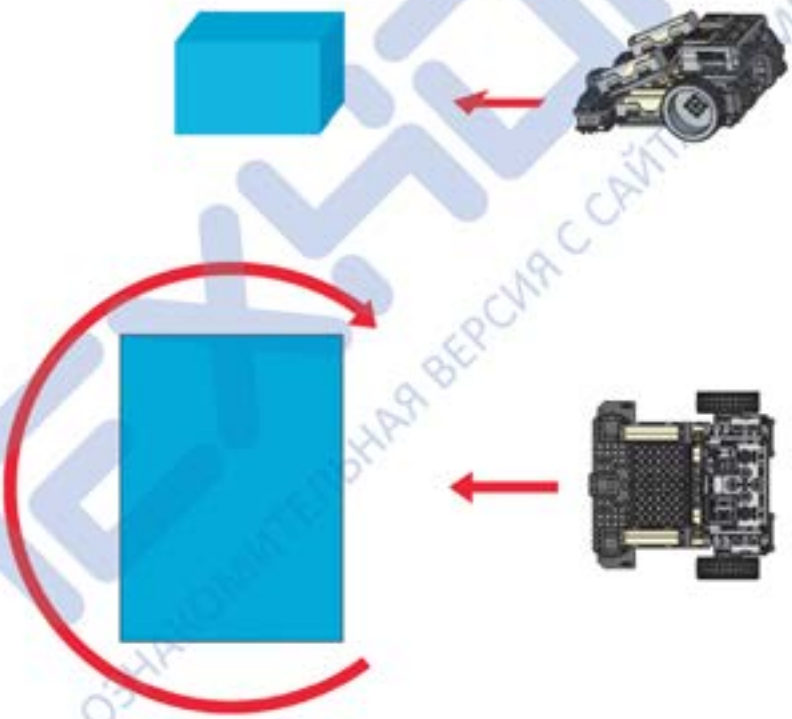
Режим IR Obstacle Detected – это один из базовых режимов работы массива ИК-датчиков, в котором автоматически определяется факт срабатывания одного из 7 датчиков. Данная функция выбирается в меню панели управления на ряду с другими, такими как: возврат текущего значения или срабатывание по пороговому значению.

Таким образом, используя даже такие простые средства, как в данном наборе можно смоделировать и исследовать процесс применения мобильного робота в произвольной рабочей зоне. В качестве задания для закрепления результатов можно рассмотреть процесс объединения двух задач воедино – задачи следования по линии как основной рабочей и задачи, рассмотренной в данном разделе.

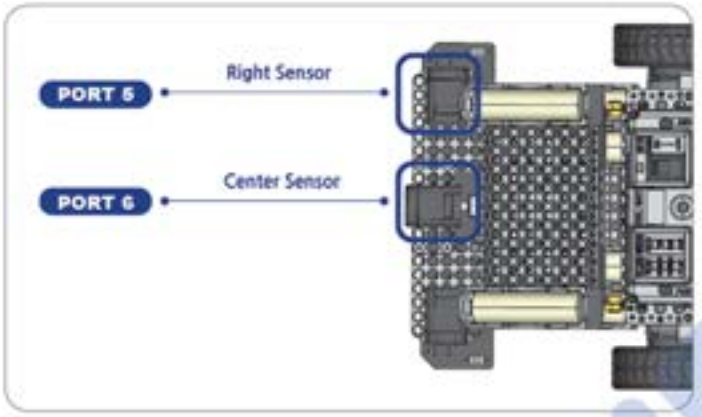
Часть № 2. Объезд препятствия по периметру

Ранее нами рассматривались задачи объезда препятствий в процессе движения робота по маршруту. В предыдущих работах акцентировалось внимание на алгоритмическую составляющую – поиск препятствия и принятие решения о маневре, а под препятствием понимался объект, который объезжался роботом за один маневр.

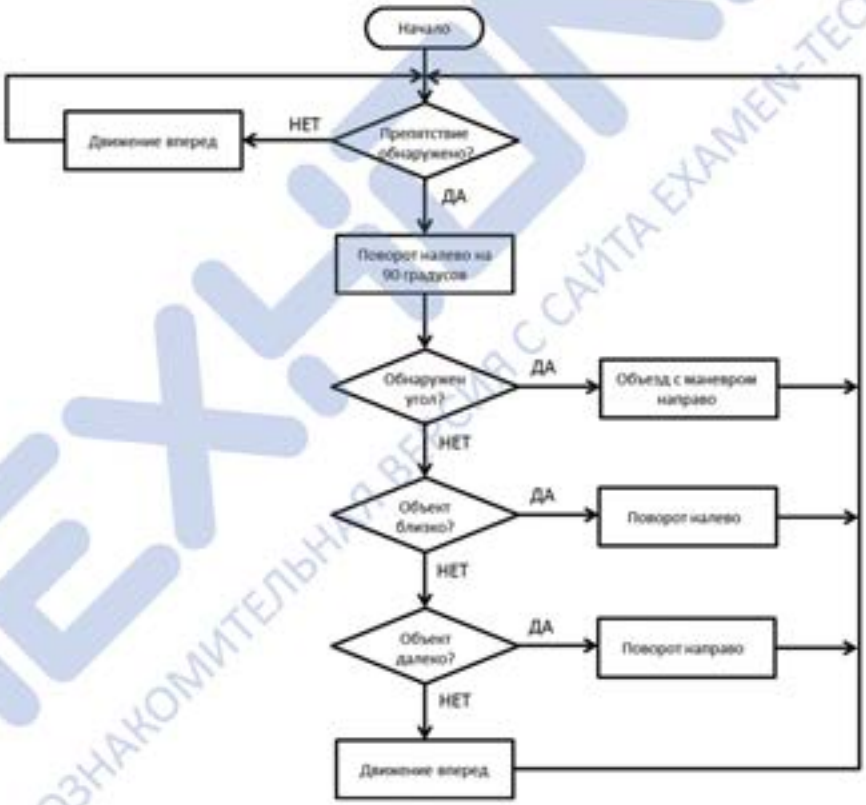
В реальной ситуации объекты, встречающиеся на пути робота, могут обладать большими габаритами и объезд их может быть затруднен. В связи с этим, необходимо предусматривать ситуацию в которой робот будет перемещаться вокруг объекта с целью вернуться на заданную траекторию и продолжить движение дальше.



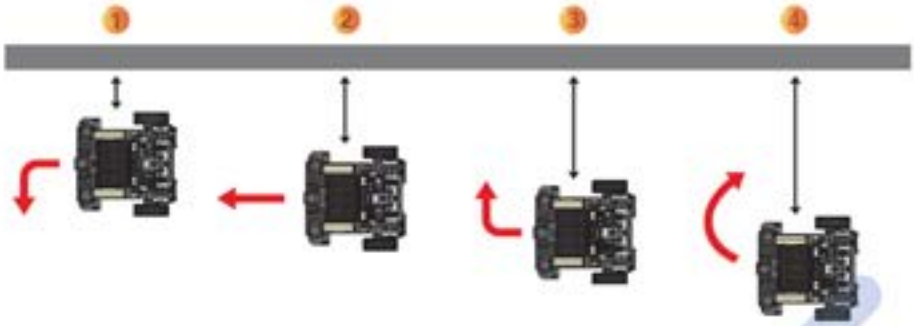
Для того чтобы мобильный робот мог обнаруживать препятствие в процессе его объезда, необходимо расположить один из ИК-датчиков сбоку. В этом случае, подъехав к объекту и начав маневр по его объезду, система управления робота будет постоянно контролировать расстояние до объекта.



Расположим три ИК-датчика на переднем бампере робота, два из них направим по обе стороны от робота, а центральный – в направлении движения. С помощью этих датчиков робот может обнаруживать объект и контролировать расстояние до него при маневрировании вокруг.



При движении робот постоянно анализирует расстояние до объекта, и в случае если расстояние меньше заданного – он отъезжает от него левее, а если больше – приближается, поворачивая направо. Если в процессе движения вдоль объекта он исчезает из виду, робот поворачивает направо, чтобы приблизиться к объекту либо объехать его с другой стороны.



Вышеуказанная процедура выполняется в цикле, описываемом программой, состоящей из четырех последовательных условий. Каждое из условий соответствует одному из рисунков: в первом случае происходит вызов функции l_slight_turn для поворота налево, во втором случае вызывается функция forward для прямолинейного движения, в третьем случае вызывается функция r_slight_turn для поворота направо и в последнем случае осуществляется поворот направо с помощью функции r_corner_turn.

```
31: Start :
32:     CALL forward
33:     WAIT WHILE ( PORT[6] <= threshold_2 )
34:     CALL pivot_left
35:
36:     ENDLESS LOOP
37:     {
38:         IF ( PORT[5] <= threshold_3 && PORT[5] >= threshold_2 )
39:             CALL forward
40:         ELSE IF ( PORT[5] > threshold_3 )
41:             CALL l_slight_turn
42:         ELSE IF ( PORT[5] < threshold_2 && PORT[5] >= threshold_1 )
43:             CALL r_slight_turn
44:         ELSE IF ( PORT[5] < threshold_1 )
45:             CALL r_corner_turn
46:     }
```

Движения робота задаются традиционным образом с помощью функций, изменяющих направление и скорость вращения колес.

```
91: FUNCTION l_slight_turn
92: {
93:     ID[1]: Moving speed = CCW:0 + l_wheel_normal_speed
94:     ID[2]: Moving speed = CW:0 + r_wheel_high_speed
95: }
96:
97: FUNCTION r_slight_turn
98: {
99:     ID[1]: Moving speed = CCW:0 + l_wheel_high_speed
100:    ID[2]: Moving speed = CW:0 + r_wheel_normal_speed
101: }
```

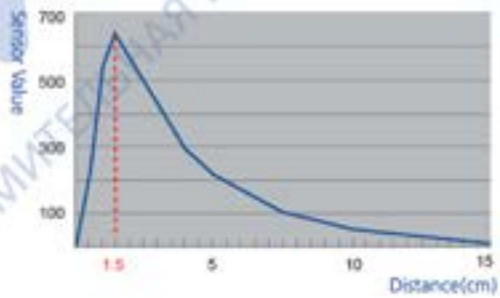
Отдельно рассмотрим функцию r_corner_turn, предназначенную для объезда препятствия справа. Данная функция сначала вызывает функцию forward_shortly, благодаря чему робот перемещается немного вперед, после чего вызывается функция right_turn для поворота направо. Поворот направо осуществляется до тех пор, пока расстояние до объекта станет не менее значения, задаваемого переменной threshold_4.

```
103: FUNCTION r_corner_turn
104: {
105:     CALL forward_shortly
106:     CALL right_turn
107:     WAIT WHILE ( PORT[5] <= threshold_4 )
108: }
```

Каждое из подобных условий определяется порогом срабатывания датчика, значения которого выбирается в зависимости от необходимого расстояния, на котором необходимо находиться относительно объекта при движении.

```
11:     // time parameters for turning a corner
12:     forward_time  = 0.150sec
13:     pivot_time  = 0.400sec
14:
15:     threshold_1  = 90
16:     threshold_2  = 150
17:     threshold_3  = 250
18:     threshold_4  = 250
```

Указанные выше значения переменных threshold могут быть подобраны опытным путем, а могут быть рассчитаны на основании характеристики датчика. Каждый датчик имеет собственную характеристику – зависимость выдаваемого значения от расчетной величины. Применяемые нами ИК-датчики имеют выходную характеристику, являющуюся зависимостью интенсивности отраженного света от расстояния до объекта.

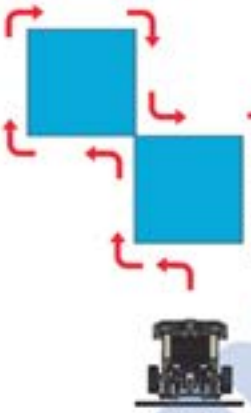


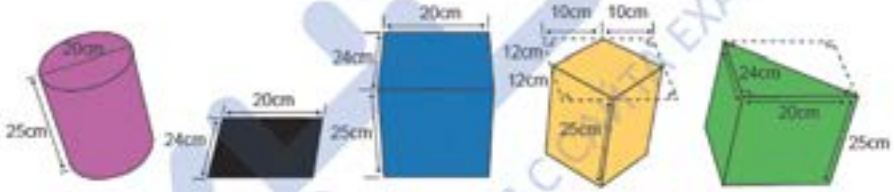
На основании приведенного графика можно подобрать такие значения переменной threshold, при которых мобильный робот будет обнаруживать объекты на заданном расстоянии и объезжать их, не приближаясь более чем положено.

Информация о характеристике датчика крайне важна при проектировании системы управления робота. Благодаря ей можно рассчитать точные перемещения исполнительного механизма, так и всего робота в целом.

Часть № 3. Подведение итогов

В рамках данной работы были изучены основы локальной навигации мобильных роботов и проведены эксперименты с реальной моделью робота. Методы и подходы, рассмотренные в процессе проведения эксперимента с роботом, являются достаточно общими и применимыми в любой другой аналогичной задаче.

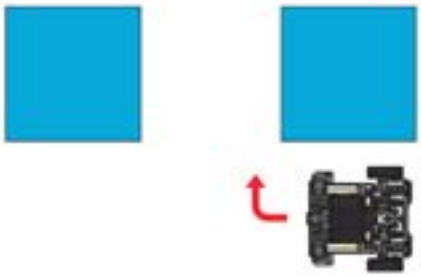




С помощью подобных алгоритмов робот может маневрировать в среде с различными объектами и объезжать препятствия произвольных габаритов. Несмотря на кажущуюся простоту и универсальность методов необходимо заранее оценивать условия применения мобильного робота.

Невозможно разработать систему управления на все случаи жизни, в одной ситуации робот сможет применяться успешно, а в другой ему может не хватить точности перемещений или собственной маневренности для избежания столкновения с объектом.

При разработке робота должны быть учтены все влияющие на него факторы и определены методы по их оценке и компенсации. Возможно, в одном из случаев необходимо будет изменить состав сенсорной системы робота и применить более технически совершенные датчики, а в другом случае может быть достаточно применения тех же самых технических средств, но другим образом.



В рассматриваемом во второй части примере в программе управления робота применялась функция r_corner_turn, предназначенная для объезда робота справа. Принцип работы ее заключался в небольшом перемещении вперед, за время задаваемое таймером, и последующем выполнении поворота.

При выезде передней части робота за габариты объекта датчик, подключенный к PORT[5], перестает видеть препятствие, и последующий поворот осуществляется роботом вслепую. Для того чтобы робот гарантированно выехал за габариты объекта перед выполнением поворота, предлагается установить второй ИК-датчик на борту робота, но уже сзади. Благодаря этому робот сможет продолжать прямолинейное движение ровно до тех пор, пока он полностью не проедет препятствующий движению объект.

```
FUNCTION r_corner_turn
{
    CALL forward
    WAIT WHILE ( PORT[4] > threshold_4 )
    CALL turn_right
    WAIT WHILE ( PORT[5] <= threshold_4 )
}
```

Данный пример иллюстрирует адаптивный подход к процессу разработки системы управления робота. К сожалению, не существует универсального решения на все случаи и нельзя рекомендовать один или несколько алгоритмов, способных решить любую задачу. Разработчик должен сам определить критерии, определяющие работоспособность проектируемой системы, и предложить способы их решения и дальнейшей реализации.

Лабораторная работа № 8

ЭКЗАМЕН ТЕХНОЛАБ

Принципы кодирования информации с помощью штрих-кодов

Лабораторная работа № 8
«Принципы кодирования информации с помощью штрих-кодов»

Информация является неотъемлемой частью современного высокотехнологичного мира. Существует множество различных способов хранения и передачи информации, и число их с каждым днем все больше растет. По мере роста объемов передачи информации совершенствуются способы ее архивации и хранения. Все чаще информация переводится в цифровой формат, поскольку именно он сочетает в себе возможность хранить максимально возможные объемы информации наиболее долго и с наивысшей степенью защиты.

Переведенная в цифровой формат информация хранится в двоичном коде, сохраненном с помощью одного из способов кодирования. Способ кодирования информации выбирается в зависимости от технологии ее записи и дальнейшего воспроизведения, от степени сжатия или в зависимости от необходимости особой защиты информации в виду ее ценности.

В качестве информации может выступать что угодно, например аудио- или видеоинформация, текстовая или графическая, также информацией являются команды и данные, которыми оперируют системы управления.

В данной работе предлагается рассмотреть довольно часто встречающийся способ кодирования информации – кодирование с помощью штриховых кодов или штрих-кодов. Кодирование штрих-кодами стало массово применяться наравне с появлением первых компьютеров, которые хранили рабочую информацию, управляющие программы на перфокартах – плотных картонных листах или пластинках с нанесенными черными или белыми отметками. Благодаря черным и белым отметкам побитно задается информация в цифровом формате, которая при считывании преобразуется в двоичный формат.

С помощью штрих-кодов кодируется информация о товарах в магазинах, информация о людях на карточках пропускных систем и многое другое. Для того чтобы считать информацию со штрих-кода, применяются сканеры, которые декодируют информацию в нужный для воспроизведения формат. Например, сканер штрих-кодов на кассе в магазине выдает информацию о наименовании товара, его стоимости и многое другое, штрих-код на пропуске офисного сотрудника может содержать информацию о его паспортных данных, должности и праве посещения различных производственных помещений. Аналогично тому, как информация со штрих-кодов считывается сканером на кассе магазина, информация с личной карточки считывается контрольно-пропускной системой в офисных зданиях.

В данной работе предлагается разработать собственную систему кодирования права доступа к охраняемому объекту. Для имитации охраняемого объекта предлагается использовать модель автоматизированного пропускного пункта, состоящего из сканера штрих-кодов и управляемого шлагбаума. В качестве сканера штрих-кода применяется массив ИК-датчиков, срабатывающий на отраженный от объекта свет.

Для того чтобы автоматизированный пропускной пункт открыл шлагбаум, необходимо, чтобы поднесенный к сканеру штрих-код нес в себе закодированную информацию – команду на подъем шлагбаума.

Команда на открытие шлагбаума задается в виде кодовой последовательности, данная кодовая последовательность жестко задается в программе управляющего контроллера. Для каждой такой команды генерируется штрих-код, в рассматриваемом нами случае применяется штрих-код, приведенный на рисунке ниже.

Штрих-код состоит из двух симметричных половинок, на одну из которых нанесена графическая информация, считываемая ИК-датчиками, вторая половинка сделана для удобства держать в руках.

Считывание информации со штрих-кода производится с помощью массива ИК-датчиков, которые срабатывают в зависимости от интенсивности отраженного света от каждой из полосок штрих-кода. Для того чтобы сканер считал информацию, необходимо поднести штрих-код к ИК-датчикам так, чтобы верхний и нижний габариты карточки совпали со сканером.

Если на сканер подана правильная информация, то шлагбаум открывается и загорается светодиод. Через некоторое время, если вблизи шлагбаума не обнаруживается объектов, препятствующих его опусканию, шлагбаум закрывается.

```
1: START PROGRAM
2: {
3:     CALL res_pos
4:
5:     // Переменной присваивается пороговое значение.
6:     td  = 50
```

Поскольку штрих-код состоит из черных и белых отметок, в качестве порога срабатывания можно взять достаточно небольшое значение. В рассматриваемом примере порог срабатывания ИК-датчиков определяется значением переменной td.

```
8:  ENDLESS LOOP
9:  {
10:     // Переменным присваиваются текущие значения ИК сенсоров.
11:     IR1  = ID[100]: IR Fire Data 1
12:     IR2  = ID[100]: IR Fire Data 2
13:     IR3  = ID[100]: IR Fire Data 3
14:     IR4  = ID[100]: IR Fire Data 4
15:     IR5  = ID[100]: IR Fire Data 5
16:     IR6  = ID[100]: IR Fire Data 6
17:     IR7  = ID[100]: IR Fire Data 7
18:
19:     // Формируется штрих-код: текущее значение ИК сенсора больше порогового - белая полоска, меньше - черная.
20:     IF ( IR1 > td && IR2 > td && IR3 < td && IR4 < td && IR5 < td && IR6 > td && IR7 < td )
21:     {
22:         CALL open
23:         Timer  = 5.120sec
24:         WAIT WHILE ( Timer  > 0.000sec )
25:         CALL close
26:     }
27: }
```

Программа управления в бесконечном цикле анализирует показания ИК-датчиков, и как только на вход семи ИК-датчиков поступает закодированная команда, происходит срабатывание пропускной системы. В этом случае вызывается функция open, отвечающая за подъем шлагбаума, который после непродолжительной задержки опускается вновь.

```
30: // Сброс начального и конечного положения привода.
31: FUNCTION res_pos
32: {
33:     init_pos = ID[1]: Present position
34:     goal_pos = init_pos + 300
35: }
```

```
38: FUNCTION open
39: {
40:     ID[1]: LED = TRUE
41:     ID[1]: Goal position = goal_pos
42: }
```

Функция open переводит сервопривод подъема шлагбаума из положения init_pos, задаваемого функцией res_pos, в положение goal_pos. При открытии шлагбаума загорается светодиод, сигнализирующий о работе сервопривода.

Закрытие шлагбаума осуществляется только в случае отсутствия под ним какого-либо объекта. Наличие объекта определяется с помощью ИК-датчика, подключенного к PORT[4].

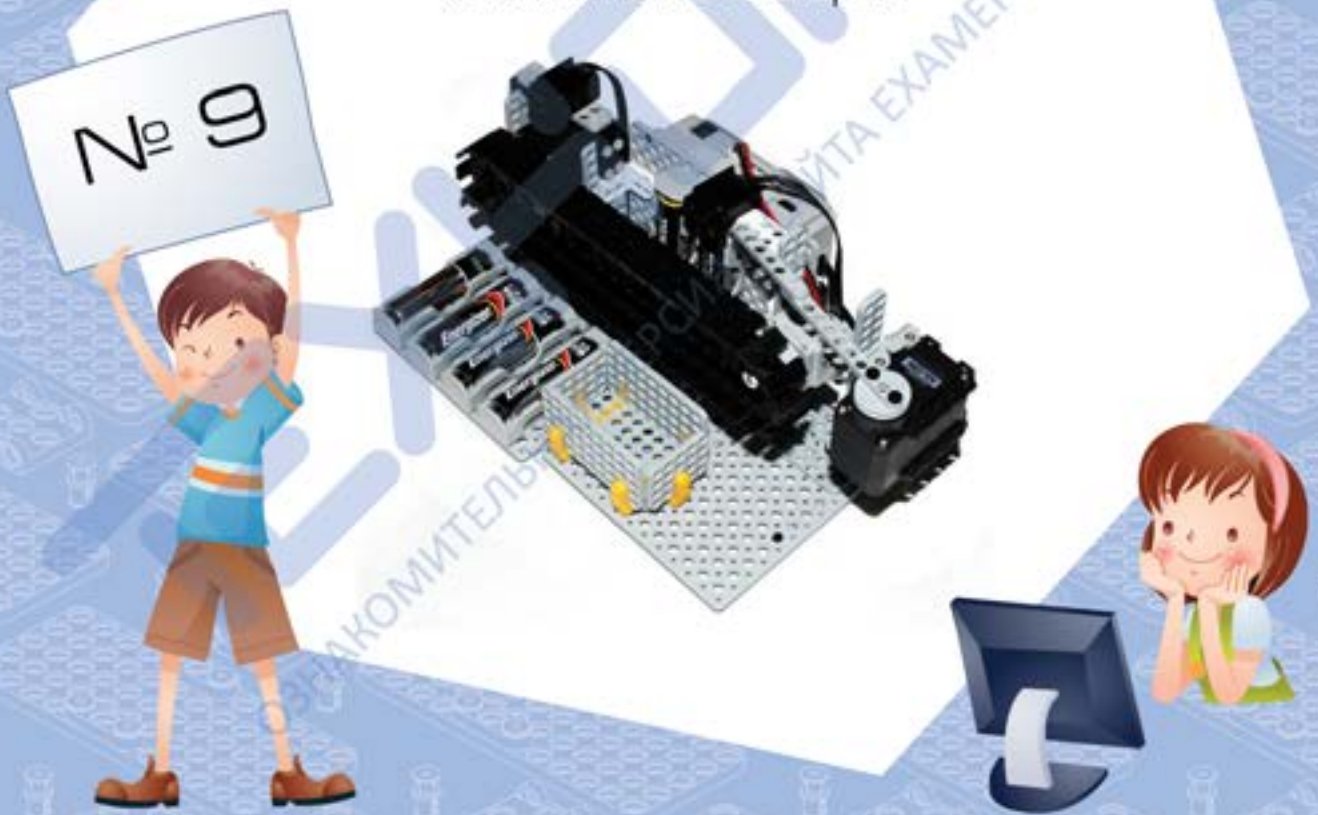
```
45: FUNCTION close
46: {
47:     WAIT WHILE ( PORT[4] > 5 )
48:     Buzzer index = Melody1
49:     Timer = 0.512sec
50:     WAIT WHILE ( Timer > 0.000sec )
51:     ID[1]: Goal position = init_pos
52:     ID[1]: LED = FALSE
53: }
```

Данная модель демонстрирует принцип работы сканеров штрих-кодов и способов кодирования информации. В случае необходимости можно закодировать любую другую команду, приводящую к открытию шлагбаума, ограничить количество попыток открытия пропускной системы и многое другое. Единственное, что остается неизменным, – это зарекомендовавший себя временем принцип, благодаря которому графическим способом кодируется информация.

Лабораторная работа № 9

ЭКЗАМЕН ТЕХНОЛАБ

Основы промышленной автоматизации



Лабораторная работа № 9
«Основы промышленной автоматизации»

Одним из практических применений робототехники является автоматизация производственных процессов, или, как это проще называется, промышленная автоматизация. Ранее нами уже неоднократно замечалось, что большинство автоматизированных механизмов можно отнести к робототехническим системам. Данное утверждение подтверждается тем, что, как правило, любой автоматизированный объект, содержащий подвижные узлы, приводится в движение приводами, которые в свою очередь управляются с помощью систем управления. Система управления приводами производственных механизмов – это наиболее часто встречающееся решение в промышленности, которое, несомненно, можно отнести к числу робототехнических решений.

Одна из наиболее часто решаемых задач в промышленной автоматизации – это согласованное управление различными частями производственных механизмов. Например, синхронное движение лент конвейеров, плавная остановка и торможение, подача инструмента в рабочую зону и многое другое. Любой подобный технологический процесс может быть описан в виде последовательности включения различных приводов.

В рамках данной работы предлагается разработать модель производственного участка, имитирующего работу упаковочной машины. Как только на конвейере появляется груз, срабатывает датчик наличия объекта на ленте и запускается привод конвейера. Как только груз, движущийся по ленте, достигает участка упаковки, срабатывает второй датчик и лента останавливается. С помощью сортировочного механизма груз снимается с ленты конвейера и попадает в специальную тару.

В конструкции модели производственного участка применяется два привода – привод движения ленты конвейера ID[2] и привод сортировочного механизма ID[1]. В начале ленты установлен датчик обнаружения объекта на конвейере, подключенный к PORT[6].

```
START PROGRAM
{
    CALL rev_pos
    ENDLESS LOOP
    {
        // Ожидание обнаружения объекта первым сенсором и запуск ленты.
        IF ( PORT[6] > 100 )
        {
            Timer = 1.280sec
            WAIT WHILE ( Timer > 0.000sec )
            ID[2]: Moving speed = CW:400

            // Ожидание обнаружения объекта вторым сенсором и остановка ленты.
            WAIT WHILE ( PORT[2] < 50 )
            ID[2]: Moving speed = CW:0
            Timer = 1.280sec
            WAIT WHILE ( Timer > 0.000sec )

            // Перемещение объекта в зону действия сбрасывающего устройства.
            ID[2]: Moving speed = CW:300
            WAIT WHILE ( PORT[2] > 20 )
            ID[2]: Moving speed = CW:0
            Timer = 1.280sec
            WAIT WHILE ( Timer > 0.000sec )

            // Сбрасывание объекта с ленты в ящик.
            ID[1]: Goal position = goal_pos
            Timer = 1.280sec
            WAIT WHILE ( Timer > 0.000sec )
            ID[1]: Goal position = init_pos
        }
    }
}
```

Конвейер приводится в движение после того, как на его ленте оказывается груз, движение конвейера осуществляется за счет привода ID[2], запущенного в режиме постоянного вращения. Движение ленты конвейера продолжается до тех пор, пока не срабатывает датчик PORT[2], свидетельствующий о том, что груз переместился в зону упаковки.

Для того чтобы груз, перемещаемый по ленте конвейера попал в упаковочную тару, он смещается к упаковочному механизму. Данное перемещение осуществляется за счет запуска конвейера на небольшой промежуток времени.

Упаковочный механизм приводится в движение сервоприводом и управляется с помощью двух значений – стартового положения и конечного. При переводе сервопривода из одного положения в другое специальный механизм сбрасывает груз с ленты конвейера в упаковочную тару, расположенную возле конвейера. После этого привод упаковочной машины переходит в стартовое положение.

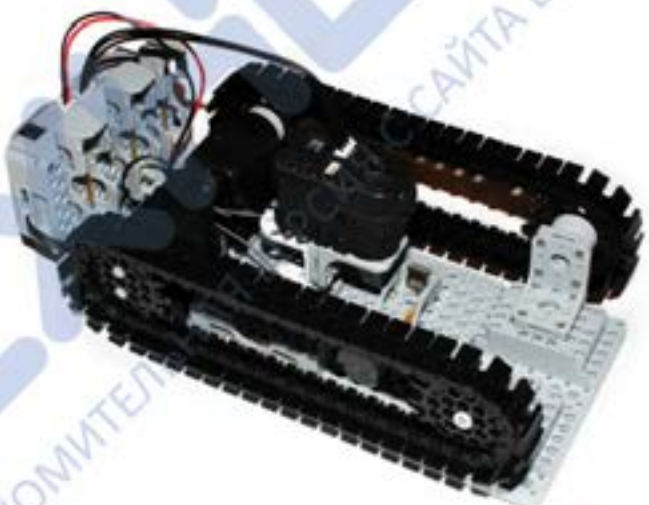
Данная модель является базовым примером, демонстрирующим работу большинства производственных участков. Почти на любом из производств можно встретить участок конвейерной линии для транспортировки каких-либо грузов. Обычно в промышленности применяются составные конвейерные линии, состоящие из нескольких секций, подобных данной модели. Таким образом, используя разработанную модель, можно смоделировать работу целого производственного участка, собрав его из нескольких конвейерных секций и оснастив дополнительными механизмами.

Несмотря на кажущуюся простоту подобное моделирование имеет большое значение. С помощью моделей реального оборудования можно разработать алгоритм управления всей производственной линией, не беспокоясь о возможности испортить дорогостоящее оборудование или продукцию.

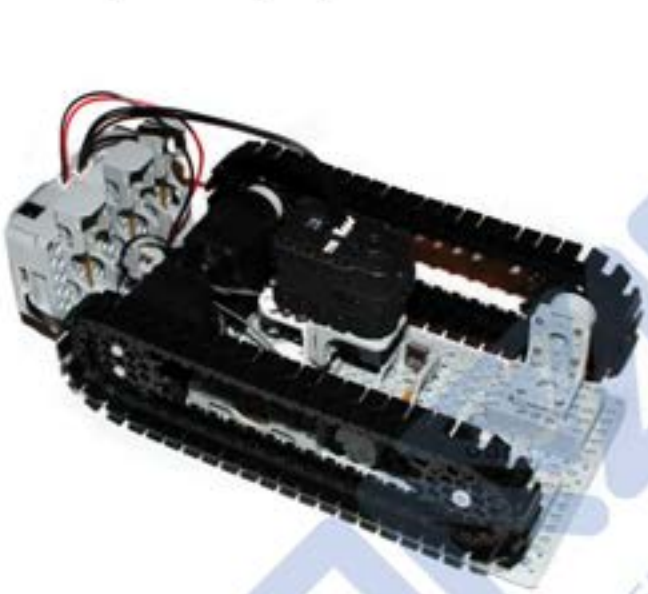
Лабораторная работа № 10

ЭКЗАМЕН ТЕХНОЛАБ

Основы синхронизации различных механизмов



Лабораторная работа № 10
«Основы синхронизации различных механизмов»



Применение роботизированных устройств и механизмов на производстве требует их точной и согласованной работы. Синхронизация работы производственных механизмов – крайне важная задача, обусловленная необходимостью координированной работы нескольких устройств одновременно. Обычно такая производственная необходимость вызвана требованиями по обработке продукции или одновременной подачи деталей и комплектующих на производственный участок.

Синхронизация механизмов может осуществляться с помощью каких-либо средств отсчета, например стартовой отметки положения промежуточного концевого выключателя и т.п. Такого рода синхронизация идентична начальной калибровке механизмов по базовым отметкам и может применяться только в тех случаях, когда требуется синхронизировать конечные положения каждого из объектов, а промежуточные положения не столь критичны.

Сложнее ситуация в случае с производственными процессами, в которых требуется постоянная согласованность работы устройств. В этом случае необходимо для управления такими механизмами применять системы с обратной связью, контролирующие рабочие параметры каждого из механизмов.

В рамках данной работы предлагается сконструировать модель производственного участка, состоящего из двух участков конвейерной линии, которые синхронизируются в случае наличия на них какого-либо груза. После синхронизации лент они доставляют грузы до места назначения одновременно.

Каждая из лент конвейера приводится в движение с помощью приводов ID[1] и ID[2], настроенных на работу в режиме постоянного вращения. Несмотря на то что приводам задается одинаковая скорость, каждый из них может двигаться со скоростью отличной от заданной. Это может быть вызвано неточностью работы системы управления или разной нагрузкой на привода, из-за чего снижается скорость их вращения. Движение конвейерных лент с разной скоростью может быть допустимо, но подача грузов на следующий производственный участок должна быть синхронной.

Синхронизация лент осуществляется при помощи сенсора AX-S1, обнаруживающего грузы на ленте. При обнаружении груза на одной из лент конвейера она останавливается, а вторая движется до тех пор, пока сенсор не обнаружит на ней груз тоже. В результате получается, что обе конвейерные линии останавливаются таким образом, что транспортируемые ими грузы располагаются параллельно друг другу. Благодаря этому они могут быть доставлены на следующий участок одновременно.

```
START PROGRAM
{

    CALL start

    ENDLESS LOOP
    {
        CALL detection_1
        CALL detection_2
        CALL detection_3

        IF ( det_1 == TRUE && det_2 == TRUE )
        {
            Timer = 1.920sec
            WAIT WHILE ( Timer > 0.000sec )
            CALL start

            Timer = 1.920sec
            WAIT WHILE ( Timer > 0.000sec )
            CALL stop_1
            CALL stop_2
            Buzzer index = Melody1
        }
    }
}
```

Программа управления данного производственного участка начинается с вызова функции start, приводящей в движение привода лент. Скорость движения каждой из лент может быть различной, например в зависимости от массы грузов или особенностей технологического процесса.

```
FUNCTION start
{
    Buzzer index = Melody1
    Timer = 1.024sec
    WAIT WHILE ( Timer > 0.000sec )
    ID[1]: Moving speed = CCW:250
    ID[2]: Moving speed = CW:325
}
```

После запуска конвейерной линии в бесконечном цикле анализируются показания ИК-датчиков универсального сенсорного модуля AX-S1. Универсальный сенсорный модуль AX-S1 содержит 3 ИК-датчика, расположенных справа, слева и по центру.

```
50: FUNCTION detection_1
51: {
52:     IF ( ID[100]: IR Left > 200 )
53:     {
54:         CALL stop_1
55:         det_1 = TRUE
56:     }
57:
58:     ELSE
59:     {
60:         det_1 = FALSE
61:     }
62: }

65: FUNCTION detection_2
66: {
67:     IF ( ID[100]: IR Right > 200 )
68:     {
69:         CALL stop_2
70:         det_2 = TRUE
71:     }
72:
73:     ELSE
74:     {
75:         det_2 = FALSE
76:     }
77: }
```

В случае если один из датчиков срабатывает на наличие груза на ленте, соответствующая лента конвейера останавливается. Если же срабатывает центральный датчик, то останавливаются оба конвейера, поскольку данный сигнал свидетельствует о наличии постороннего объекта между конвейерами, что может привести к аварийной ситуации.

Данная лабораторная работа примечательна тем, что в ней рассматривается один из методов синхронизации двух производственных механизмов с помощью внешнего устройства. В качестве внешнего устройства применяется универсальный сенсорный модуль AX-S1, представляющий собой несколько датчиков, расположенных в одном устройстве. Применение подобных устройств позволяет сэкономить количество портов программируемого контроллера, используемого для подключения датчиков. Благодаря этому, неиспользуемые порты управления можно применить при подключении каких-либо других внешних устройств, тем самым расширить функционал проектируемого робота.

Беспроводное управление роботами

ЭКЗАМЕН ТЕХНОЛАБ

Беспроводное управление роботами с помощью ZigBee

Беспроводное управление роботами с помощью ZigBee

Беспроводное управление по радиоканалу – наиболее часто встречающееся решение в робототехнике. Для гарантированной передачи данных между устройствами применяются различные протоколы передачи информации. В случае если совместно работающих устройств достаточно много, их объединяют в беспроводные сети и используют сетевые протоколы передачи данных. ZigBee – стандарт высокоуровневых протоколов беспроводной связи, применяемый массово при автоматизации промышленного оборудования, в системах автоматизации зданий и жилых помещений, в медицинском и телекоммуникационном оборудовании. Отличительная особенность стандарта ZigBee заключается в его высокой помехозащищенности, низком энергопотреблении и отсутствии необходимости получения частотного разрешения. Благодаря стандартизации и открытой спецификации различные производители электронных устройств могут разрабатывать собственные устройства совместимые с устройствами, использующими протокол ZigBee.

На сегодняшний день радиомодули ZigBee используются в системах с невысокой скоростью передачи данных на небольшие расстояния, но требующие гарантированной безопасности канала связи и точности доставки информации при крайне низком энергопотреблении. Низкое энергопотребление обусловлено функцией «спящий режим» во время простоев, но в отличие от модулей Bluetooth, время пробуждения радиомодулей ZigBee в разы меньше. Поэтому реакция устройств на передаваемые сигналы намного быстрее.

Для управления моделями роботов с помощью радиоканала на базе протокола ZigBee в робототехнических конструкторах ROBOTIS применяются специальные модули: ZIG-100/110A, ZIG2Serial, USB2Dynamixel.

Радиомодуль ZIG-110A предназначен для последовательной передачи данных с помощью ZigBee интерфейса от внешнего устройства к контроллеру робота и обратно. Данный модуль полностью совместим с программируемыми контроллерами CM-100 для наборов на базе конструкторов OLLO и CM-530, для наборов на базе конструкторов Bioloid.

Для того чтобы компьютер и робот могли осуществлять взаимный обмен информацией, используются устройства ZIG2Serial и USB2Dynamixel.

Модуль ZIG-100 представляет собой радиомодуль ZigBee, предназначенный для встраивания в различные устройства, например ZIG-100 может быть встроен в пульт ДУ RC-100, а также использоваться для беспроводного управления роботами с помощью ПК.

Устройство USB2Dynamixel – это универсальный преобразователь интерфейсов USB персонального компьютера в последовательный интерфейс, применяемый в сервоприводах, контроллерах и беспроводных модулях ROBOTIS.

Настройка и установка соединения по интерфейсу ZigBee

Для настройки и установки соединения между компьютером и роботом необходимо подключить к контроллеру и компьютеру модули ZIG-100 и ZIG-110A, а после их настроить.

Для этого необходимо выполнить следующие шаги:

1. Подключить модуль ZIG-110A к порту Communication Jack на контроллере CM-530 и к порту программирования для контроллера CM-100.
2. Подключить контроллер робота к компьютеру с помощью USB кабеля.
3. Включить питание контроллера.
4. Запустить RoboPlus и выбрать программу для настройки оборудования RoboPlus Manager.
5. Выбрать порт, к которому подключен контроллер (по умолчанию COM3) и нажать Connect для подключения к контроллеру.
6. Во вкладке Controller должны появиться все подключенные к контроллеру устройства. Каждое устройство, с которым работает контроллер, обладает ID номером. Модуль ZIG-110A распознается контроллером как устройство «My Remote ID» с ID 36. Модуль ZIG-100 распознается контроллером как устройство с именем «Remote ID» и идентификатором ID 34.

   Примечание: в случае применения контроллера CM-100 ID номера явным образом не доступны пользователю, поскольку к данному контроллеру подключается ограниченное количество устройств с заранее известными идентификационными номерами. Если же используется контроллер CM-530, необходимо указывать ID номера подключаемых к нему устройств.
7. Каждое из устройств имеет собственное значение Value. В рассматриваемом случае модуль с ID 34 имеет значение 63560, а значение модуля с ID 36 равно 63561.
8. Для того чтобы радиомодули с разными ID работали в паре, необходимо, чтобы значение внешнего модуля ZIG-100 равнялось значению модуля ZIG-110A, подключенного к контроллеру. Если выявлено несовпадение, то необходимо в графу Value внести требуемое значение и нажать кнопку Apply.

9. Подключение модуля ZIG2Serial к компьютеру осуществляется с помощью устройства USB2Dynamixel. Для совместной работы двух устройств необходимо установить ZIG2Serial в разъем COM-порта USB2Dynamixel и перевести переключатель выбора режимов в положение, соответствующее RS232.

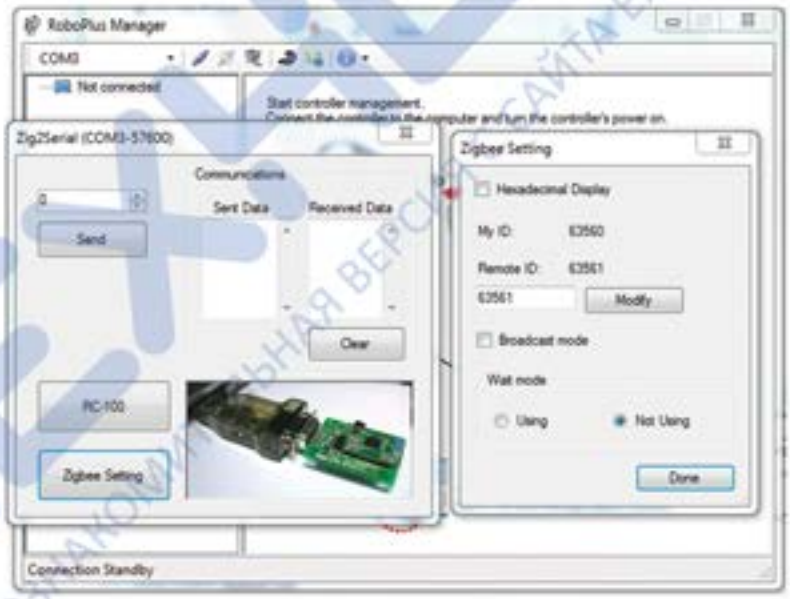
10. Подключите модуль USB2Dynamixel в USB порт компьютера. Дождитесь того, чтобы компьютер распознал подключенное внешнее устройство.

11. Запустите RoboPlus и RoboPlus Manager.

12. Выберите номер порта, к которому подключен USB2Dynamixel (по умолчанию COM3) и нажмите на кнопку Zig2Serial Management.

13. Войдите в меню ZigBee Setting (удерживайте кнопку в нажатом состоянии в течение 3 секунд). В появившемся меню необходимо ввести значение Value, соответствующее значению My Remote ID (в рассматриваемом примере значение, соответствующее ZIG-110A, равно 63561).

14. Проверьте соединение радиомодулей. Включите устройства, если настройки были произведены верно, то светодиоды модулей Zig2Serial и ZIG-110A, которые мигали, начнут гореть ровным светом.



15. Проверьте соединение радиомодулей. Запустите виртуальный пульт управления RC-100 (данная функция доступна для контроллеров CM-530).

16. Переведите контроллер CM-530 в режим PLAY с помощью кнопки MODE и нажмите на кнопку START.

17. С помощью открывшегося виртуального пульта управления можно путем нажатия кнопок U, L, R и D управлять движением робота. Для этого необходимо, чтобы в контроллер робота была записана одна из тестовых программ, которую можно скачать с официального сайта ROBOTIS или скачать из Help среды разработки RoboPlus.

Управление роботами

ЭКЗАМЕН ТЕХНОЛАБ

Управление роботами с помощью программной среды **LabView**

Управление роботами с помощью программной среды LabView

Дистанционное управление роботами или даже производственными механизмами довольно часто встречается в наше время. Несмотря на то что автономные системы управления все больше и больше занимают нишу управления робототехническими системами, системы ручного управления все еще играют значимую роль.

Дистанционное управление робототехническими системами в зависимости от поставленной задачи может сводиться к кардинально разным по реализации системам. Самое часто встречающееся в мобильной робототехнике решение – применение пультов дистанционного управления, в этом случае человек-оператор вручную с помощью пульта или джойстика управляет движением робота. В этом случае перемещение робота контролируется также вручную: оператор следует за роботом на некотором расстоянии или же оператор дистанционно наблюдает за окружающими пространством вокруг с помощью видеокамер, установленных на борту робота.

Также помимо ручного управления человеку-оператору обычно предоставляют функции наблюдения за ходом функционирования робототехнической системы. Такие решения наиболее часто встречаются в промышленности, в этом случае на рабочее место оператора поступает информация о состоянии технологического оборудования, информация и показания от датчиков, количество выпущенной продукции и многое другое. В подобных системах человеку-оператору очень редко предоставляется возможность управлять каким-либо механизмом, и все его функции сводятся к наблюдению за работой и остановке системы управления в аварийной ситуации.

В некоторых случаях системы управления роботом или робототехнической системой реализуют на удаленном высокопроизводительном сервере, который анализирует показания бортовых систем робота, производит необходимые расчеты и передает роботу управляющий сигнал, по которому тот осуществляют свою дальнейшую работу.

Примером последней системы может быть лабораторный стенд для управления роботами на базе программной среды LabView. С помощью программной среды LabView на удаленном сервере или обычном компьютере можно создать программу управления роботом, которая возьмет на себя самые ресурсоемкие функции, например анализ показаний датчиков, планирование маршрута, распознавание графической и видеоинформации. Благодаря этому можно существенно упростить бортовую систему управления роботом, оставив за ней простейшие функции реагирования на поступающие от удаленного компьютера команды.

Такой подход позволяет создавать сложные системы управления для достаточно простых роботов, разрабатываемых даже на базе робототехнических конструкторов. Данное пособие содержит рекомендации по основам разработки систем управления роботами с использованием программной среды LabView. В качестве модели для отработки алгоритмов управления предлагается использовать роботов из образовательных модулей «Профессиональный уровень» или «Исследовательский уровень». Этих роботов объединяет общее свойство, они принадлежат к одному модельному ряду Bioloid корейской компании Robotis, поэтому в их состав входят одинаковые аппаратные средства: программируемый контроллер CM-530, сервопривода Dynamixel, ИК-датчики и многое другое.

Благодаря применению программной среды LabView для роботов серии Bioloid можно разработать сложные системы управления, использующие все ресурсы удаленного компьютера. Связь робота с управляющим компьютером осуществляется с помощью беспроводного канала ZigBee. Применение беспроводного интерфейса дает возможность разрабатывать роботов, не привязанных к пульту управления с ИК-каналом, ограничивающим их мобильность. Роботы с беспроводным радиоинтерфейсом могут работать на удаленном расстоянии от управляющей станции, в других помещениях и при наличии каких-либо преград между ними. Таким образом, совместное использование возможностей робототехнических наборов и программной среды LabView открывают перед разработчиками новые возможности по проектированию и эксплуатации роботов в образовательном и соревновательном процессе.

Управление базовым робототехническим набором с помощью программной среды LabView

Робототехнический модуль обладает достаточно широким функционалом – на базе данного руководства можно проводить лабораторные работы по изучению основ проектирования роботов и разработки для них систем управления. Роботы, разрабатываемые из образовательного робототехнического модуля, могут применяться в робототехнических соревнованиях и исследовательской деятельности.

Одной из основ исследовательской деятельности является проведение экспериментов, сбор и анализ данных. Процесс исследовательской деятельности неотъемлем от процесса проектирования и разработки каких-либо новых и инновационных решений, поэтому он крайне важен. Программная среда LabView является одним из универсальных инструментов автоматизации исследовательской деятельности и лабораторных экспериментов. С помощью LabView можно анализировать результаты измерений сенсорных устройств, производить сложные преобразования и вычисления. Поэтому знакомство с программной средой LabView является важным этапом процесса изучения робототехники.

Для ознакомления с процессом применения LabView для управления роботами предлагается воспользоваться специальной базовой программой, представляющей собой дистанционный пульт управления человека-оператора мобильного робота. Данная программа позволяет управлять мобильным роботом на базе робототехнического набора аналогичным образом – с помощью ИК-пульта управления RC-100, производить сбор данных и многое другое. В данном разделе мы кратко остановимся на описании данной программы и возможностей по ее применению.

Базовая программа представляет собой проект в программной среде LabView и располагается на диске для преподавателя в разделе «LabView/Базовая программа». Данная программа может применяться в качестве эмулятора ИК-пульта для всех роботов серии Bioloid.

Требования:

1) Операционная система ПК Windows XP/Vista/7 (32/64bit)
2) Наличие установленной программной среды LabView (версия не позднее 9.0)
3) Наличие установленного программного пакета Microsoft Visual C++ 2005 re-distribution package
4) Наличие установленной программной среды RoboPlus
5) Наличие интегрированных в программную среду LabView библиотек для управления оборудованием – Dynamixel SDK и ZigBee SDK
6) Наличие установленных драйверов LabView NI VISA (версия не позднее 5.3)
7) Наличие устройств USB2Dynamixel, Zig2Serial, комплекта ZigBee для контроллера CM-530.

Описание базовой программы

Данная программа представляет собой библиотеку виртуальных инструментов на базе LabView 2012, адаптированную для работы с внешним устройством, в нашем случае роботом серии Bioloid. Данная программа разработана для работы с Bioloid Stem, но применима для работы с любыми роботами серии Bioloid.

Для работы с базовой программой необходимо осуществить запуск проекта, в результате чего на экране компьютера появится лицевая панель программы.

Лицевая панель содержит ряд элементов управления, предназначенных для перехода к различным режимам работы программы.

«Перейти к работе с набором» – меню, переводящее к средствам управления роботом.

«Руководство пользователя» – меню, вызывающее руководство по работе с базовой программой.

«Описание набора» – меню, вызывающее руководство по работе с базовым робототехническим набором.

«Выход» – завершение работы программы.

Работа с базовой программой начинается с перехода в меню «Перейти к работе с набором». В результате этого пользователю предлагается ознакомиться с различными способами работы с устройствами, подключенными к компьютеру. На выбор предлагается два варианта – управление контроллером CM-530 и сервоприводами Dynamixel посредством USB интерфейса, а также беспроводное управление роботом с помощью ZigBee интерфейса.

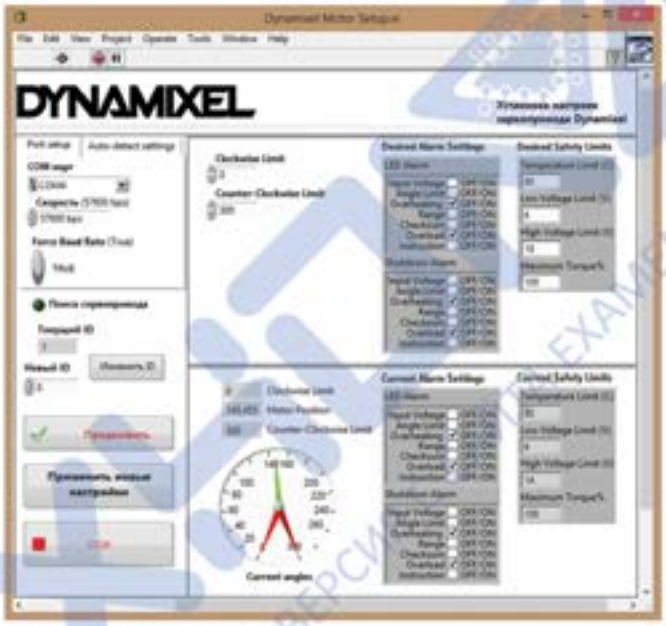
При переходе в меню «Знакомство с USB2Dynamixel и Dynamixel SDK» пользователю предлагается воспользоваться одним из режимов работы – «Проводной режим» для управления сервоприводами с помощью проводного канала связи или «Беспроводной режим» для управления сервоприводами с помощью интерфейса ZigBee.

Работа в режиме «Проводной режим»

Для взаимодействия с сервоприводами в проводном режиме необходимо подключить сервопривод (или несколько сервоприводов) к адаптеру USB2Dynamixel, а сам адаптер необходимо перевести в положение TTL. Для подключения адаптера необходимо использовать кабель с ответвлением для питания сервоприводов (+9.6В) или же питание приводов можно осуществить от контроллера CM-530.

1) Установка настроек сервопривода

Данный виртуальный инструмент позволяет произвести настройку сервопривода перед использованием. В настройку сервопривода входят параметры инициализации (первоначального состояния/положения), а также параметры сигналов и критических состояний.



Порядок работы

1. Убедиться в правильности подключения сервопривода Dynamixel к адаптеру USB2Dynamixel.
2. Выбрать соответствующий COM-порт, указать скорость, а во вкладке Auto-detect settings указать параметры автоматического поиска сервопривода: диапазон ID и время поиска. Для продолжения работы нажать «Продолжить».
3. Дождаться того, что загорится индикатор «Поиск сервопривода». Как только сервопривод будет найден, на индикаторе «Текущий ID» отобразится значение ID сервопривода, что свидетельствует о том, что сервопривод готов к настройке.
4. Установить необходимые настройки, а после нажать на «Применить новые настройки», после чего выбранные настройки будут применены к сервоприводу.
5. Нажатие кнопки STOP осуществляет возврат к окну выбора виртуального элемента (vi).

2) **Обмен данными с сервоприводом**

С помощью данного виртуального элемента осуществляется обмен данными с сервоприводом по последовательному интерфейсу. Данный виртуальный элемент предназначен для отправки команд сервоприводу и получения ответных сообщений.

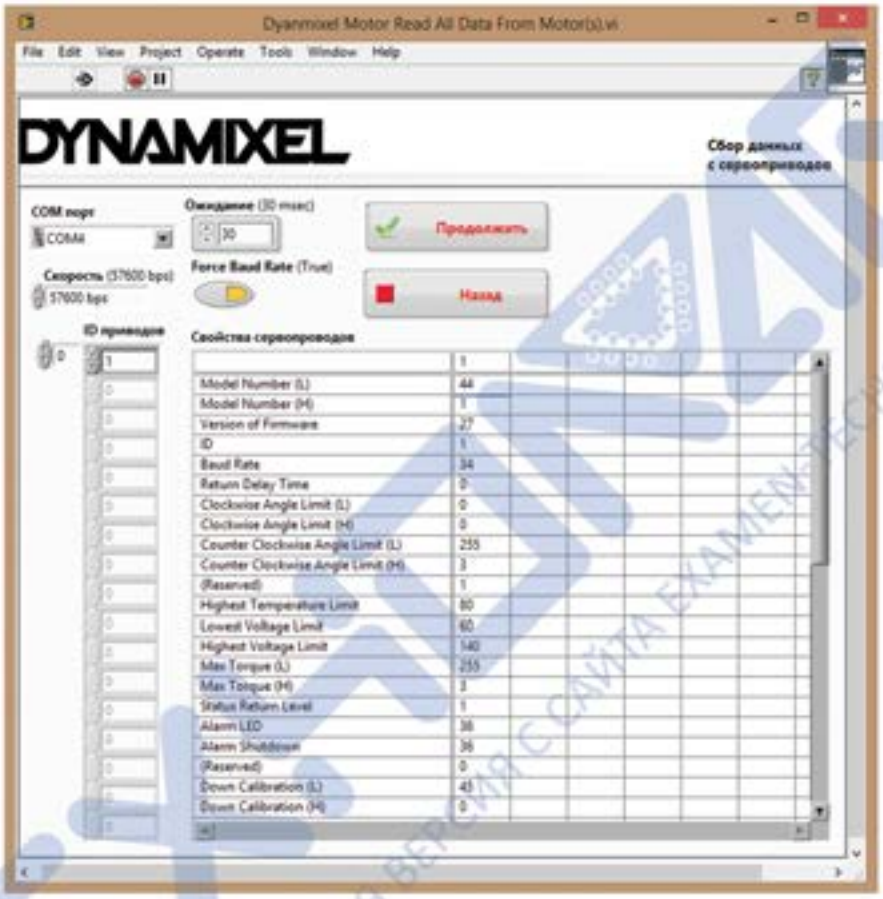
Порядок работы

1. Необходимо убедиться в правильности подключения сервопривода Dynamixel к адаптеру USB2Dynamixel.
2. Выбрать соответствующий COM-порт и указать скорость, после чего нажать кнопку «Продолжить».
3. Указать ID сервопривода, адрес и выбранное действие нажать «Отправить команду». В окне «Результаты» отобразится результат выполнения заданной команды.
4. В случае завершения работы с данным виртуальным элементом нажать кнопку STOP.

Примечание: подробное описание команд и их адресация содержатся в руководстве пользователя для каждого из типов сервоприводов Dynamixel.

3) Сбор данных от сервоприводов

Данный виртуальный элемент предназначен для запроса данных, описывающих состояние сервопривода и вывода их на экран пользователя. Программа предназначена для работы как с одним сервоприводом, так и с группой сервоприводов.

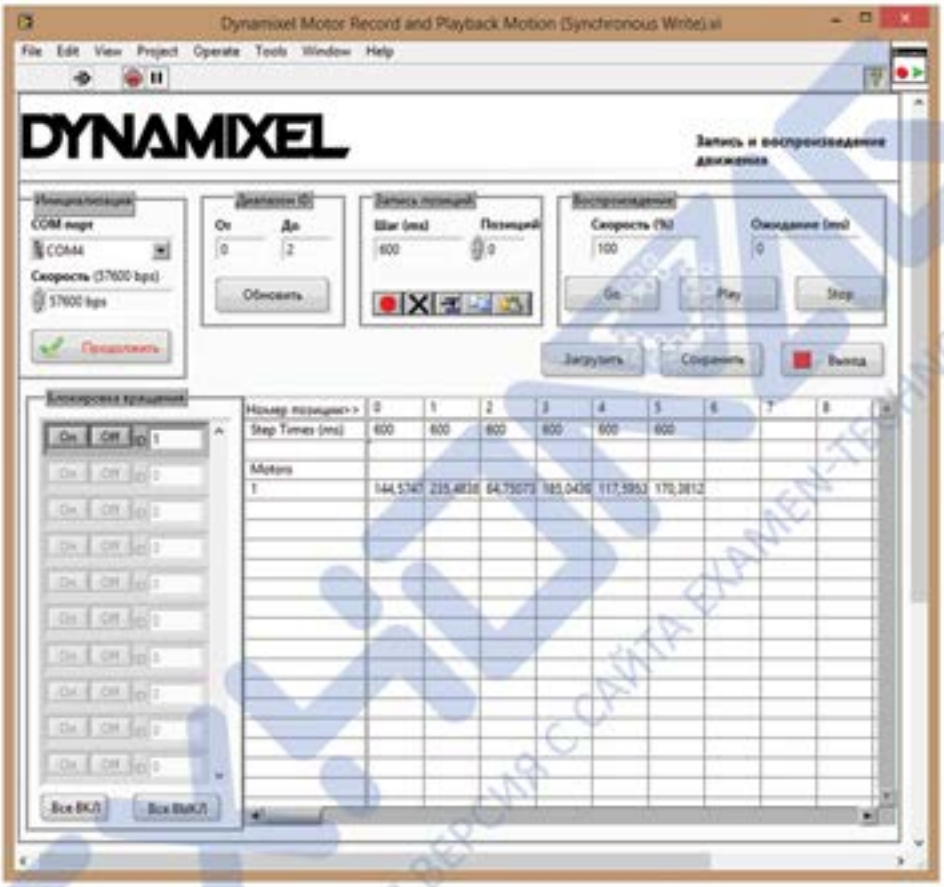


Порядок работы

1. Необходимо убедиться в правильности подключения сервопривода Dynamixel к адаптеру USB2Dynamixel.
2. Выбрать соответствующий COM-порт и указать скорость, а также в массив ID приводов необходимо ввести ID необходимых сервоприводов. После чего нажать кнопку «Продолжить». В окне «Свойства сервоприводов» отображаются все свойства выбранных сервоприводов и их текущие значения.
3. В случае завершения работы с данным виртуальным элементом нажать кнопку «Назад».

4) Запись и воспроизведение движения

Данный виртуальный элемент предназначен для программирования вращения сервоприводов. С помощью данной программы можно пошагово задать очередность положений каждого из сервоприводов.



Порядок работы

1. Необходимо убедиться в правильности подключения сервопривода Dynamixel к адаптеру USB2Dynamixel.
2. Выбрать соответствующий COM-порт, указать скорость обмена данными, после чего нажать кнопку «Продолжить».
3. Указать диапазон ID для поиска подключенных сервоприводов. Нажать кнопку «Обновить».
4. После обнаружения сервопривода его ID отобразится в списке «Блокировка вращения», а также в таблице.

5. С помощью нажатия кнопки Record фиксируется текущее положение сервопривода. Изменив положение сервопривода, снова нажмите Record. После чего установите индикатор позиций в положение 0 и нажмите Go, а затем Play. Сервопривод повторит запрограммированные движения.

6. Можно сохранять файлы с записью движения в формате dsq, а затем снова их воспроизводить. Для этого воспользуйтесь кнопками «Загрузить» и «Сохранить».

7. В случае завершения работы нажмите кнопку «Выход».

Работа в режиме «Беспроводной режим»

Данный виртуальный элемент применяется для управления сервоприводом с помощью беспроводного интерфейса ZigBee.

Для работы с данной программой необходимо, чтобы в программируемый контроллер CM-530 была установлена программа CM_530_zigbee_dxl.tsk.

Порядок работы с программой

1. Необходимо установить требуемый режим работы сервоприводов с помощью программной среды RoboPlus.

2. Необходимо включить робота Bioloid, подключить модуль ZIG-100 к компьютеру. Убедиться, что модули ZIG-100 и ZIG-110A установили между собой соединение (красные лампочки на них должны начать гореть ровным светом)

3. Запустить программу, выбрать COM-порт (по умолчанию COM3) и нажать кнопку «Подключиться». В случае успешного подключения к порту загорится лампочка.

4. Выбрать либо режим постоянного вращения сервоприводов или режим контроля положения (в зависимости от режима работы).

5. С помощью устройства управления «Данные» можно изменять значение скорости вращения или положения сервоприводов.
6. В случае завершения работы необходимо нажать кнопку «Остановить сервопривод» и затем Stop.

На этом знакомство с устройством USB2Dynamixel и Dynamixel SDK можно считать законченным. Вернувшись к начальному окну программы можно перейти к знакомству с Zig2Serial и ZigBee SDK.

При переходе в меню «Знакомство с ZIG2Serial и ZigBee SDK» пользователю предоставляется возможность дистанционного управления роботом и сбора показаний его датчиков.

Для работы с данной программой необходимо, чтобы в программируемый контроллер CM-530 была установлена программа CM_530_zigbee_main.tsk.

В появившемся окне на выбор предлагаются следующие меню:

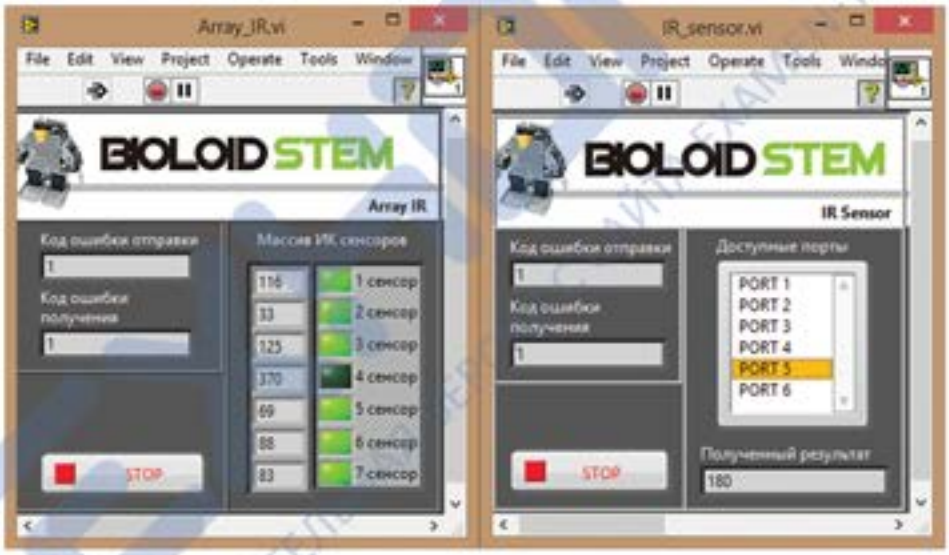
«Управление роботом» – программа эмуляции ИК-пульта дистанционного управления RC-100.

«Опрос датчиков» – программа для сбора показаний датчиков, установленных на роботе.

«Выход» – меню выхода из программы.

При переходе к программе «Опрос датчиков» появляется окно, позволяющее выбрать одну из программ опроса датчиков.

В зависимости от выбранной программы появляется один из виртуальных элементов, отображающих данные, поступающие от различных сенсорных устройств.



Программа «Управление роботом» полностью повторяет ИК-пульт RC-100 и дает возможность управлять роботом так, как будто у пользователя в руках реальный джойстик. Для управления роботом нужно подключить устройства ZigBee к контроллеру CM-530 и персональному компьютеру. Подключение устройства ZigBee к контроллеру CM-530 осуществляется путем включения модуля ZIG100 в разъем Communication Jack программируемого контроллера. Подключение этого же модуля к персональному компьютеру осуществляется с помощью устройств USB2Dynamixel и ZIG2Serial. Необходимо установить модуль ZIG100 в устройство ZIG2Serial и с помощью USB2Dynamixel подключить его к персональному компьютеру. Настройку данного соединения необходимо выполнить в соответствии с рекомендациями раздела «Беспроводное управление роботами с помощью ZigBee» данного учебного пособия.

Для работы с роботом включите контроллер CM-530 и с помощью кнопки MODE переведите его в режим PLAY. Запустите контроллер путем нажатия кнопку START.

На виртуальном пульте управления выставьте значение COM-порта, соответствующее соединению по интерфейсу ZigBee, далее нажмите Connect. В случае удачного подключения устройств друг к другу загорится зеленая лампочка.

С помощью виртуального пульта управления можно управлять роботом в ручном режиме. Управление возможно с помощью компьютерной мышки, в этом случае пользователь должен нажимать на соответствующие кнопки виртуального пульта. Также возможно управление с помощью клавиатуры, когда пользователь управляет направлением движения робота с помощью нажатий клавиш на клавиатуре персонального компьютера.

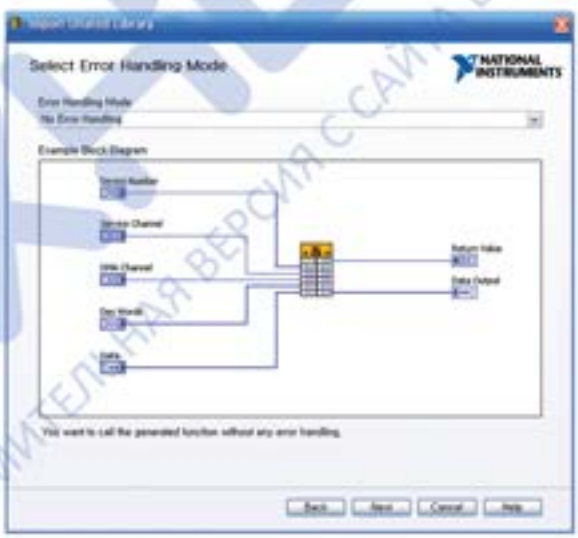
Для завершения работы с набором необходимо нажать на кнопку STOP, выключить программируемый контроллер CM-530 и в случае необходимости отключить модули беспроводного интерфейса ZigBee.

Данный раздел демонстрирует возможности применения программной среды LabView для управления роботами. С помощью базовой программы пользователь может наглядно ознакомиться с функционалом программной среды LabView, оценить ее возможности и, быть может, придумать собственное решение одной из робототехнических задач. Для дальнейшего освоения технологии разработки систем управления роботами пользователю предлагается ознакомиться с руководством по разработке базовых программ управления в программной среде LabView.

Разработка систем управления роботами и устройствами

ЭКЗАМЕН ТЕХНОЛАБ

Разработка систем управления роботами и устройствами с помощью **LabView**



Разработка систем управления роботами и устройствами с помощью LabView

Применение различных подходов к решению поставленных задач, а также использования различных инструментов для их реализации является отличительной особенностью профессионального подхода к проектной и исследовательской деятельности. Использование программной среды LabView позволяет расширить горизонты применения образовательных робототехнических модулей в процессе обучения учащихся робототехнике.

Данный раздел предназначен для демонстрации основ применения LabView при управлении роботами серии Bioloid. В данном разделе описывается процесс разработки программ управления в среде LabView и программ для управляющего контроллера CM-530. Программированию контроллера CM-530 уделяется отдельное внимание, поскольку в данном случае его программы качественно отличаются от рассматриваемых ранее. Теперь на программу контроллера CM-530 возлагается задача приема команд управления от удаленного компьютера и передачи ему требуемой информации, в то время как все алгоритмические задачи решаются в программной среде LabView.

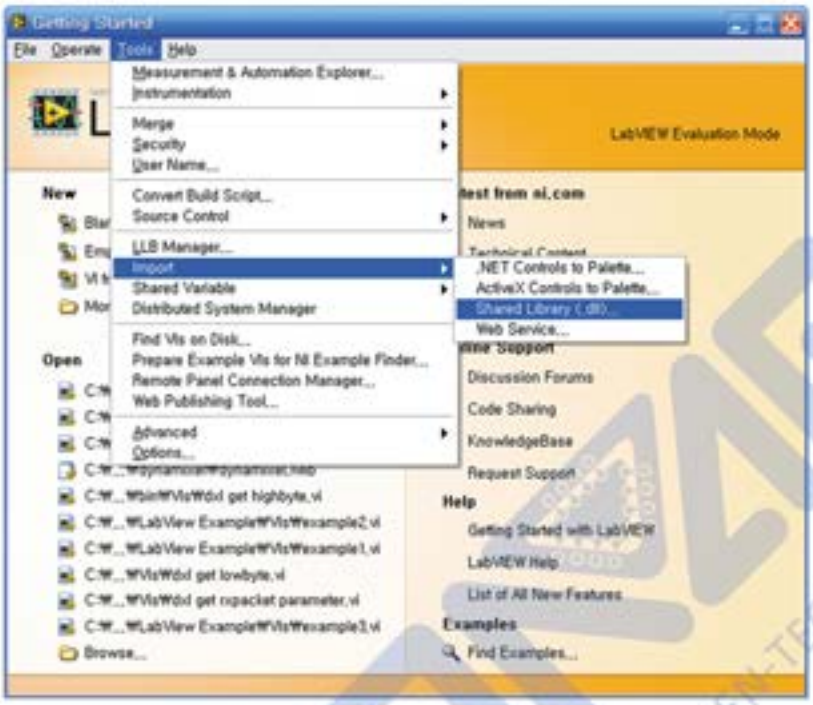
Описание библиотеки Dynamixel SDK

Библиотека Dynamixel SDK представляет собой стандартную библиотеку для работы с различными сервоприводами серии Dynamixel. В данной библиотеке реализован инструментарий для работы с сервоприводами, используя который пользователь может разрабатывать собственные системы управления.

Импортирование библиотеки в LabView

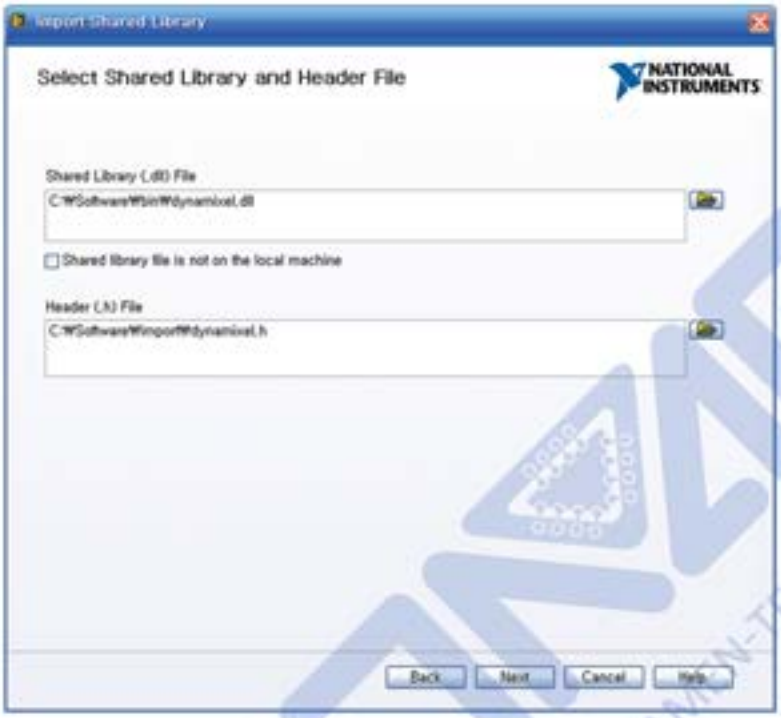
Для того чтобы использовать стандартные функции управления сервоприводами Dynamixel, необходимо импортировать библиотеку Dynamixel SDK в программную среду LabView. Данную библиотеку можно скачать с сайта корейского производителя ROBOTIS, также она расположена в каталоге LabView диска для преподавателя.

1) Выберите Tools – Import – Shared Library (.dll) в стартовом окне LabView.



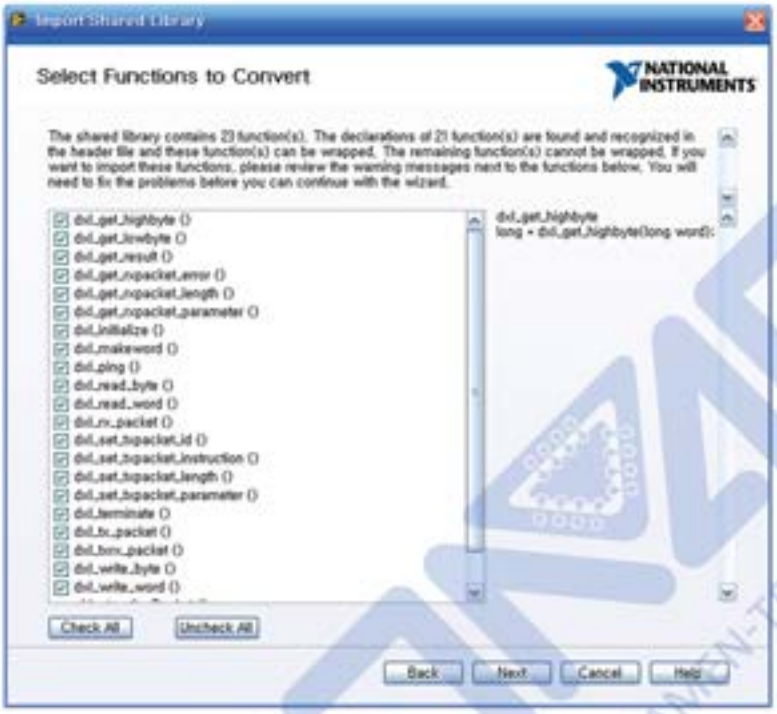
2) В появившемся окне выберите «Create VIs for a shared library» и нажмите кнопку Next.

3) Выберите dynamixel.dll и dynamixel.h в каталоге Dynamixel SDK и нажмите Next.

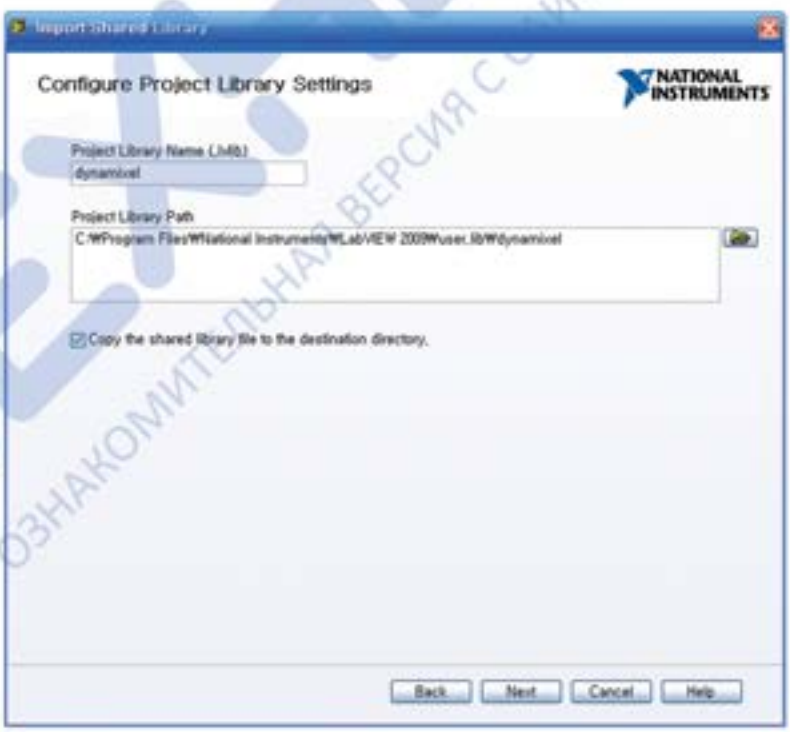


4) Появившееся окно оставить без изменений и для перехода далее нажать Next.

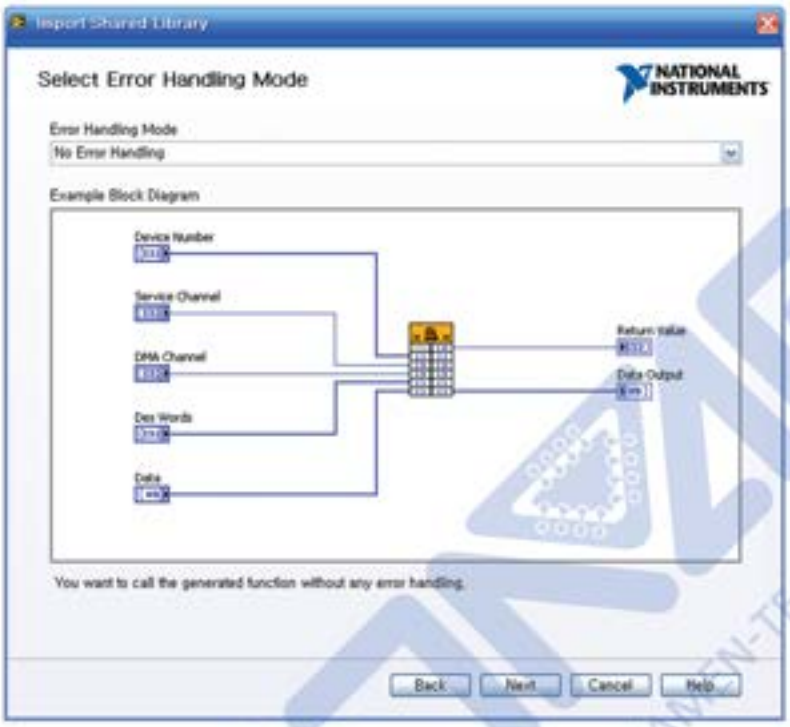
5) В появившемся окне проверить: все ли функции выбраны. Нажать Next для перехода к следующему этапу.



6) Ничего не изменяя перейти к следующему меню, нажав Next.

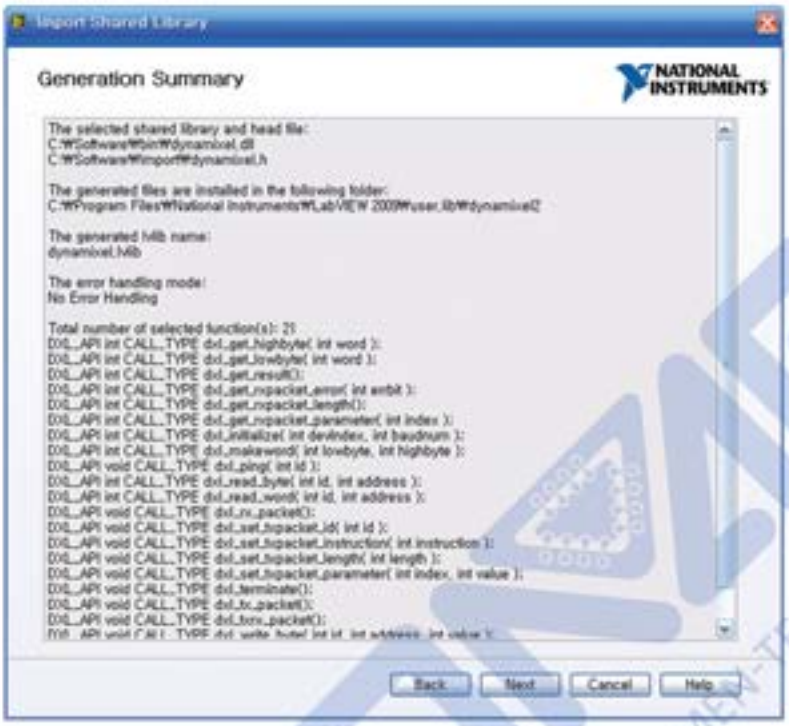


7) Ничего не изменяя перейти к следующему меню, нажав Next.

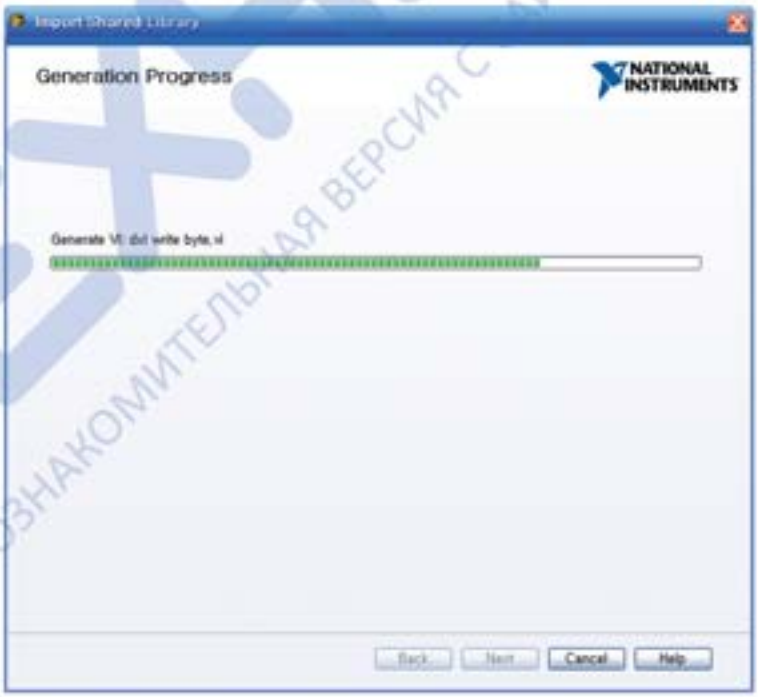


8) Ничего не изменяя перейти к следующему меню, нажав Next.

9) Ничего не изменяя перейти к следующему меню, нажав Next.



10) Дождаться завершения процесса создания виртуальных инструментов (vi). По завершению процесса перейти к следующему меню, нажав Next.



По завершении процесса создания виртуальных элементов появляется палитра виртуальных инструментов в разделе User Libraries.

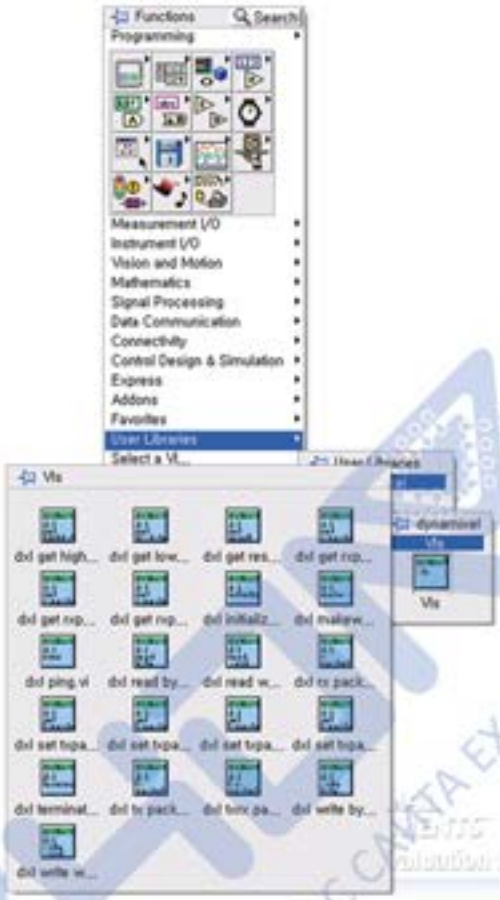


Таблица 1. Описание функций палитры виртуальных инструментов Dynamixel

Функция	Описание	Параметры входные	Параметры выходные
dxl_initialize	Инициализация и установка связи с подключенными устройствами	**devIndex** – число подключенных в настоящее время устройств **baudnum – значение скорости в бодах (см. Таблица 2).**	1 – нет ошибки 0 – есть ошибка
dxl_terminate	Отключение сеанса связи с устройствами	-	-

dxl_set_txpacket_id	ID сервопривода	ID – идентиф. номер сервопривода	-
dxl_set_txpacket_instruction	Входные командные коды в пакет инструкций	**instruction – одна из команд** (см. Таблица 3)	-
dxl_set_txpacket_instruction	Вводимый параметр в пакет инструкций	**index – номер параметра value – Значение параметра в диапазоне 0 – 255**	-
dxl_set_txpacket_length	Установка длины пакета инструкций	**length – длина пакета инструкций**	-
dxl_get_rxpacket_length	Проверка длины пакета статуса	-	Значение длины пакета статуса
dxl_get_rxpacket_parameter	Проверка параметра пакета статуса	**index – номер параметра для проверки**	Значение заданного параметра
dxl_get_rxpacket_error	Фиксирует ошибки, включенные в пакет статуса	**-errbit** – флаг бита для фиксации ошибки (см. Таблица 4)	1 – есть ошибка 0 – нет ошибки
dxl_tx_packet	Передача пакета инструкций на Dynamixel	-	-
dxl_rx_packet	Получение пакета статуса из буфера драйвера. После вызова результат должен быть проверен с помощью dxl_get_result	-	-
dxl_txrx_packet	Одновременное выполнение функций dxl_tx_packet dxl_tx_packet	-	-
dxl_get_result	Выдает результат проверки пакетов	-	(см. Таблица 5)
dxl_ping	Проверка Dynamixel с заданным ID. Результат проверяется функцией dxl_get_result	**id – ID сервопривода для проверки**	-

dxl_write_byte	Запись информационного байта в Dynamixel. Результат может быть получен с помощью dxl_get_result	**id –** ID Dynamixel **для записи информации** **address** – значение адреса **value** – записываемое значение	-
dxl_write_word	Запись двух информационных байтов в Dynamixel. Результат может быть получен с помощью dxl_get_result	**id –** ID Dynamixel **для записи информации** -**address** – значение адреса -**value** – записываемое значение	-
dxl_read_byte	Считывание одного информационного байта в Dynamixel. Результат может быть получен с помощью dxl_get_result	**id –** ID Dynamixel **для записи информации** **address** – значение адреса	-
dxl_read_word	Считывание двух информационных байтов в Dynamixel. Результат может быть получен с помощью dxl_get_result	**id –** ID Dynamixel **для записи информации** **address** – значение адреса	-
dxl_makeword	Преобразует двухбитный тип данных в тип WORD	**Lowbyte** – младший байт, **Highbyte** – старший байт	Слово данных в формате WORD
dxl_get_highbyte	Извлечение старшего байта из переменной типа WORD	**word – данные для извлечения старшего бита**	Возвращает старший байт
dxl_get_lowbyte	Извлечение младшего байта из переменной типа WORD	**word – данные для извлечения старшего бита**	Возвращает младший байт

Вышеуказанная таблица палитры виртуальных инструментов демонстрирует собой реализацию протокола управления сервоприводами Dynamixel.

Таблица 2. Соответствие передаваемых значений и скоростей

Address	Set BPS	Goal BPS	Error
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

Вышеуказанная таблица задает соответствие скоростей передачи данных по последовательному интерфейсу, задаваемых значений и значений погрешности.

Таблица 3. Командные коды

№	Имя	Содержание
1	INST_PING	Не выполняется. Используется, когда контроллер готов к передаче статусного пакета
2	INST_READ	Команда чтения данных из Dynamixel
3	INST_WRITE	Команда записи данных в Dynamixel
4	INST_REG_WRITE	Команда идентичная команде WRITE_DATA, но для функционирования необходимо получение команды ACTION.
5	INST_ACTION	Команда активации работы
6	INST_RESET	Команда возврата Dynamixel в начальное состояние, идентичное заводскому
131	INST_SYNC_WRITE	Команда используется для одновременного контроля нескольких Dynamixel

В данной таблице задается перечень командных кодов пакетов инструкций. С помощью данных кодов определяется назначение пакета.

Таблица 4. Описание ошибок и соответствующие им значения команд

№	Имя	Содержание
1	ERRBIT_VOLTAGE	If the Error-impressed voltage is deviated from the set movement voltage range in the control table, it is set to 1.
2	ERRBIT_ANGLE	If the writing of Goal Position value is deviated from the range between CW Angle Limit ~ CCW Angle Limit, it is set to 1.
4	ERRBIT_OVERHEAT	If the internal temperature of Dynamixel is deviated from the set movement temperature range in the control table, it is set to 1.
8	ERRBIT_RANGE	When the command is deviated from the using range, it is set to 1.
16	ERRBIT_CHECKSUM	When Checksum of the transmitted Instruction Packet is not matching, it is set to 1.
32	ERRBIT_OVERLOAD	When the set torque cannot control the current load, it is set to 1.
64	ERRBIT_ INSTRUCTION	If an undefined instruction is transmitted, or action command is transmitted without reg_write command, it is set to 1.

Данная таблица содержит перечень сообщений об ошибках в процессе управления сервоприводами.

Таблица 5. Расшифровка результатов проверки

Значение	Сообщение
COMM_TXSUCCESS	Success of Instruction packet Transmission
COMM_RXSUCCESS	Success of Status Packet Reception
COMM_TXFAIL	Failure of Instruction Packet Transmission by Error
COMM_RXFAIL	Failure of Status Packet Reception by Error
COMM_TXERROR	Problems in Instruction packet
COMM_RXWAITING	Status packet is not arrived yet.
COMM_RXTIMEOUT	Relevant Dynamixel is not responding.
COMM_RXCORRUPT	Problems in Status packet

Данная таблица содержит перечень сообщений об ошибках в процессе управления сервоприводами.

Программирование и управление роботами серии **Bioloid**

ЭКЗАМЕН ТЕХНОЛАБ

Программирование и управление роботами серии **Bioloid** с помощью базовых средств **RoboPlus** и программной среды **LabView**

Программирование и управление роботами серии Bioloid с помощью базовых средств RoboPlus и программной среды LabView

Данный раздел содержит ряд примеров, направленных на наглядную демонстрацию процесса разработки программ управления роботами серии Bioloid в базовой для данных наборов среде программирования RoboPlus. Для демонстрации возможностей программной среды LavView, наряду с программированием роботов, разрабатывается программа управления для персонального компьютера, предназначенная для дистанционного управления роботами и отображения на экране компьютера рабочей информации.

Информация в данном разделе предназначена для предварительного ознакомления пользователями, в совершенстве освоившим процесс разработки роботов на базе оборудования из робототехнических наборов серии Bioloid, знакомых с программными средами RoboPlus и LabView.

Часть 1. Беспроводная передача данных от персонального компьютера к роботу с помощью интерфейса ZigBee

Основная задача данного примера – иллюстрация процесса передачи данных между персональным компьютером с установленной программной средой LabView и программируемым контроллером CM-530.

Для работы с данным примером необходимо наличие установленной на персональный компьютер программной среды LabView и установленной библиотеки ZigBee SDK. Данная библиотека должна располагаться в разделе functions – User libraries – Zigbee – Vis.

Программа контроллера CM-530 должна быть настроена на прием данных по последовательному интерфейсу. В качестве передаваемых данных в примере используется набор команд, передаваемых от пульта управления.

Разработка программы в среде программирования LabView

Принцип работы данной программы заключается в установлении соединения по беспроводному последовательному интерфейсу между персональным компьютером и роботом, подключении и отключении COM-портов, инициализации оборудования в соответствии с выбранным портом и передачи текстовых кодов команд.

1) Создать новую блок-диаграмму в рабочем пространстве LabView. Поместить в нее две структуры Case Structure, Round LED-индикатор с именем ZGB_Connected, Square LED-индикатор с именем TxD_LED.

2) Создать кнопку OK Button с названием ZGB_Connect и соединить ее со входом Case Selector первой структуры. Установить настройку этой кнопки Mechanical Action в позицию Switch Until Released. Поместить внутрь первой структуры в окне True структуру Flat Sequence с тремя окнами (Frame).

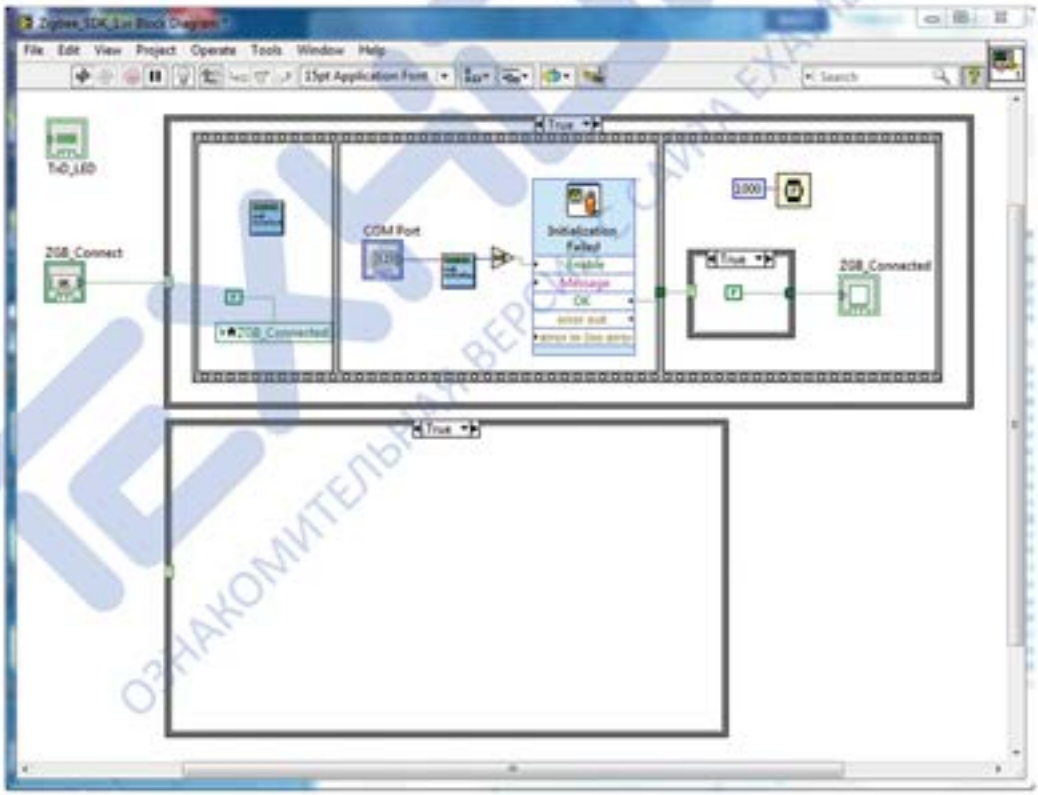
3) Поместить внутрь первого окна функцию zgb_terminate.vi из библиотеки ZigBee. Данная функция необходима для закрытия сессии работы с COM-портом, в случае когда предыдущая сессия не была корректно завершена.

4) Поместить локальную переменную ZGB_Connected внутрь первого окна и присвоить ей значение FALSE.

5) Поместить во второе окно функцию zgb_initialize.vi, вход которой необходимо соединить с numeric control COM Port. Выход данной функции сравнить со значением «0», отправим на вход Enable express-vi Display Msg.

6) Установить в окне Message to Display параметр Initialization Failed. Данное окно предназначено для инициализации соединения с COM-портом, к которому подключен модуль ZIG-100 с помощью ZIG2Serial и USB2Dynamixel. В случае успешной инициализации соединения функция zgb_initialize.vi возвращает значение 1, а в случае неуспешной – значение «0» и сообщение о неуспешной инициализации.

7) В последнем окне поместить структуру Case Structure и соединить ее вход с выходом OK express-vi Display Msg. В окно True поместить константу False, а в окно False – константу True и соединить их с индикатором ZGB_Connected. При завершении данного этапа блок-диаграмма должна выглядеть как на рисунке.



8) Создать кнопку OK Button и дать ей название Transmit. Установить настройку этой кнопки Mechanical Action в позицию Switch Until Released.

9) На вход второй структуры Case Structure, созданной в п.1, необходимо подать значение переменной ZGB_Connected с помощью локальной переменной.

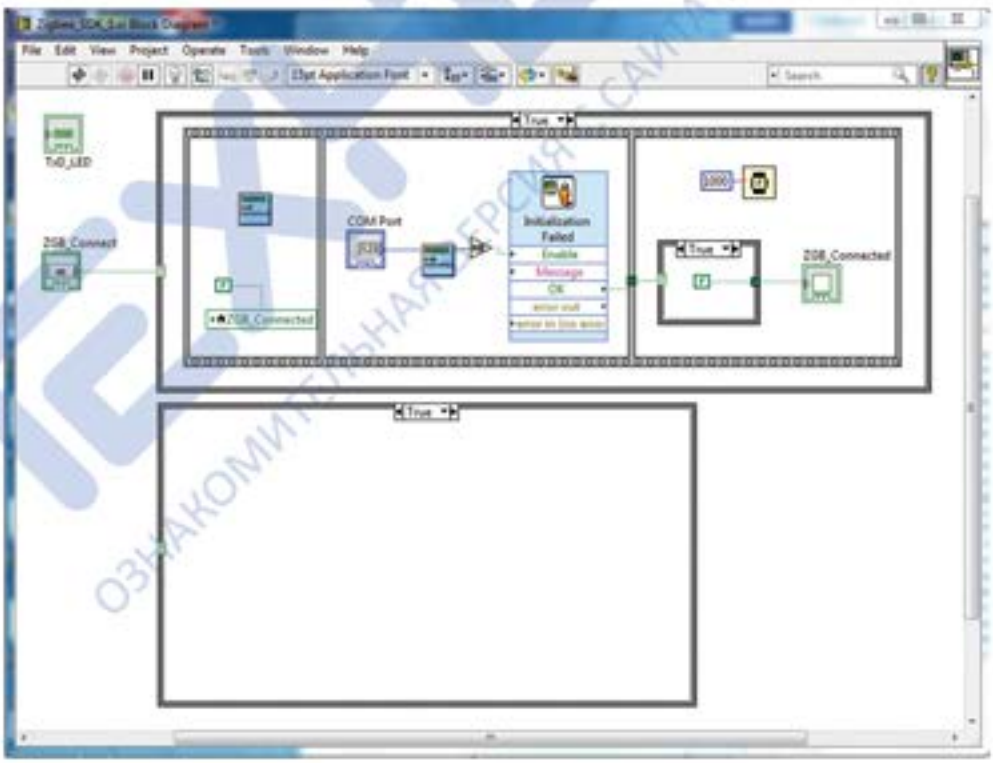
10) Поместить внутрь окна True цикл While Loop, а внутрь данного цикла поместить структуру Case Structure, вход которой необходимо соединить с кнопкой Transmit.

11) Внутрь структуры Case Structure поместить функцию zgb_tx_data.vi, вход которой необходимо соединить с numeric control data to transmit. Выход структуры необходимо сравнить со значением «0», отправив на вход Enable express-vi Display Msg, расположенный вне структуры.

12) В окне Message to Display необходимо установить сообщение Transmission Failed. Также внутри окна True текущей структуры Case Structure необходимо присвоить локальной переменной TxD_LED значение true. Данный этап реализует механизм отправки сообщений с помощью интерфейса ZigBee.

13) В окне False текущей структуры Case Structure необходимо поместить еще одну структуру Case Structure и подать ей на вход значение переменной TxD_LED.

14) Поместить в окне True структуры Case Structure функцию zgb_tx_data.vi, с поданным на вход значением «0». Реализация данного этапа необходима для фильтрации повторной отправки команд. При завершении данного этапа блок-диаграмма должна выглядеть как на рисунке.



15) Полученную блок-диаграмму необходимо целиком поместить в цикл While Loop.

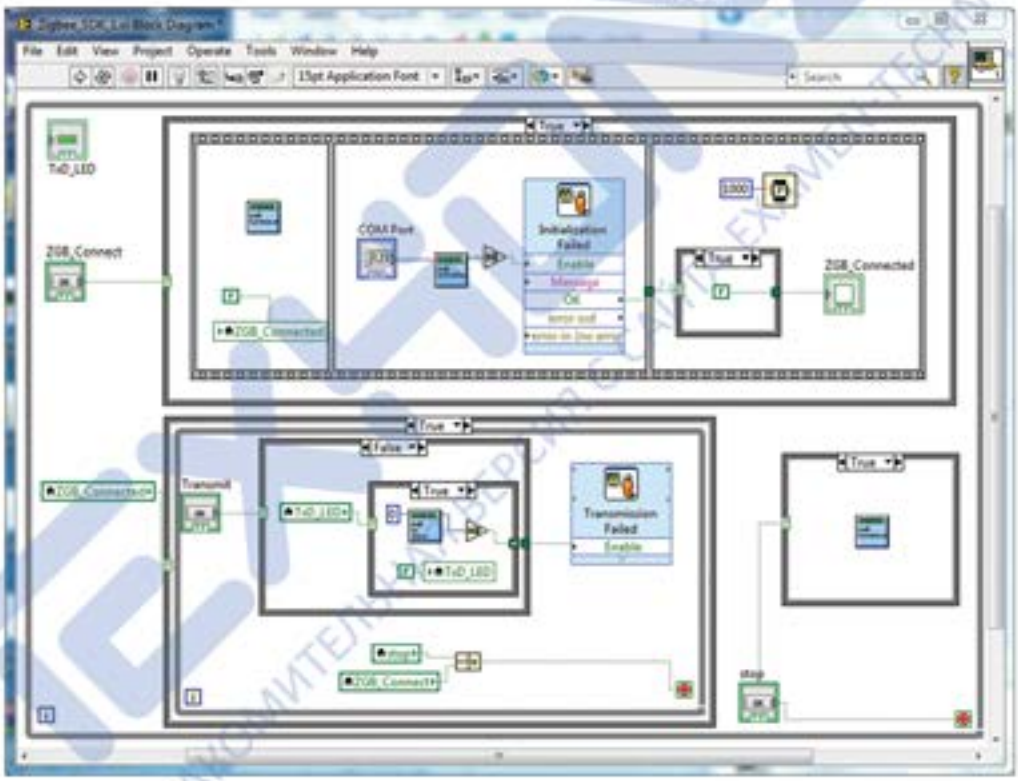
16) Создать кнопку stop и установить ее настройку в позицию Switch When Released.

17) Соединить кнопку stop и Loop Condition последнего созданного цикла While Loop. Также внутри текущего цикла While Loop расположить структуру Case Structure, а в ее окне True разместить функцию zgb_terminate.vi.

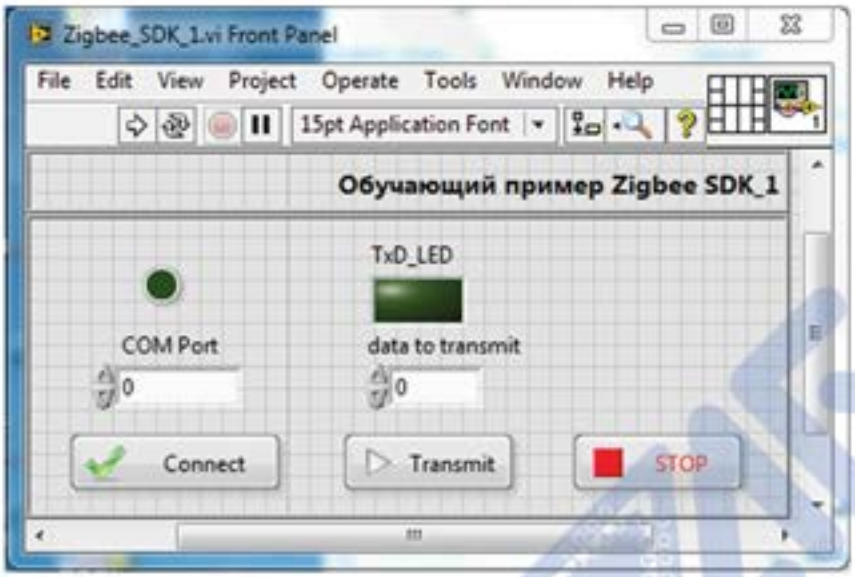
18) Соединяем вход структуры Case Structure с кнопкой stop.

19) В созданном цикле While Loop необходимо объединить локальные переменные stop и ZGB_Connect с помощью логической операции OR. Полученный результат необходимо соединить с Loop Condition цикла While Loop.

20) Разрабатываемая блок-диаграмма готова. Результаты работы должны выглядеть как на рисунке ниже.



Для удобства применения данной программы необходимо настроить ее лицевую панель на подобии того, как это сделано на рисунке ниже.



Порядок работы с программой:

1) Включить робота Bioloid и настроить беспроводное соединение по интерфейсу ZigBee.

2) Убедиться в том, что программируемый контроллер CM-530 запрограммирован соответствующим образом.

3) Запустить программу и осуществить подключение к COM-порту. В случае необходимости номер COM-порт можно задавать в соответствующем окне.

4) В окне data to transmit ввести код команды в текстовом формате.

Если в контроллере робота установлена соответствующая программа и робот собран верно, то в зависимости от поданной команды робот будет выполнять одно из заданных действий.

Данная программа дублирует пульт дистанционного управления RC-100, поэтому для ее использования совместно с роботом необходимо всего лишь модифицировать программу ручного управления роботом так, чтобы она воспринимала команды, поступающие по последовательному интерфейсу.

Модификация программы контроллера CM-530

Для того чтобы роботом серии Bioloid можно было управлять дистанционным образом, необходимо модифицировать его программу управления. Суть всех модификаций сводится к тому, что каждое из действий робота должно выполняться вследствие вызова с помощью команды от удаленного компьютера.

```
26:   ENDLESS LOOP
27:   {
28:       WAIT WHILE ( Remocon Arrived == FALSE )
29:       ReceiveData = Remocon RXD
```

Передаваемое значение ReceivedData сравнивается с всеми возможными командами, запрограммированными в контроллере CM-530. В результате определяется код заданной операции KeyValue.

```
31:       KeyValue = ReceiveData & U+D+L+R
```

В зависимости от текущего кода операции, т.е. от значения переменной KeyValue, выполняется одно из условий.

```
32:       IF ( KeyValue == U )
33:       {
34:           CALL standby
35:           Buzzer time = Play Melody
36:           Buzzer index = Melody0
37:           CALL forward
38:       }
39:
40:       ELSE IF ( KeyValue == D )
41:       {
42:           CALL standby
43:           Buzzer time = Play Melody
44:           Buzzer index = Melody0
45:           CALL reverse
46:       }
47:
48:       ELSE IF ( KeyValue == L )
49:       {
50:           CALL standby
51:           Buzzer time = Play Melody
52:           Buzzer index = Melody1
53:           CALL pivot_left
54:       }
```

Поскольку программа LabView пересылает текстовые команды, в виде кода операции, необходимо определить коды операций, задаваемых нажатиями клавиш пульта RC-100.

Код операции можно определить в меню панели управления, в настройках пульта RC-100. При выборе какой-либо кнопки пульта управления, а также их комбинаций, в окне ниже отображается текстовое значение кода операции.

Часть 2. Сбор показаний массива ИК-датчиков

Основная задача данного примера – иллюстрация процесса сбора и анализа данных с помощью программы, разработанной в среде программирования LabView. В данной работе предлагается рассмотреть процесс сбора данных с массива ИК-дальномеров, часто применяемого в наборах серии Bioloid.

Данная работа демонстрирует одно из основных предназначений LabView – сбор и анализ результатов в рамках проведения эксперимента.

Разработка программы в среде программирования LabView

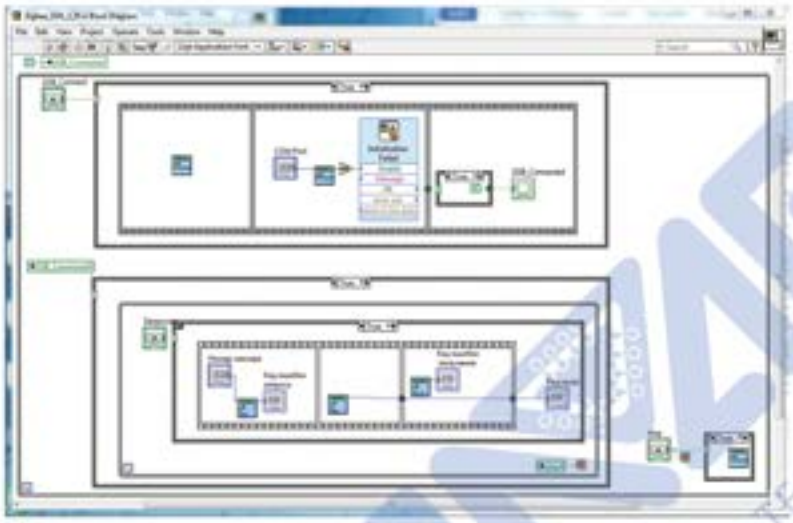
Программа, разрабатываемая в данной части, основывается на блок-диаграмме из предыдущей, поскольку они содержат общую часть, касающуюся передачи данных по COM-порту.

Принцип работы программы заключается в передаче на удаленный компьютер по беспроводному каналу показаний одного из ИК-датчиков из массива.

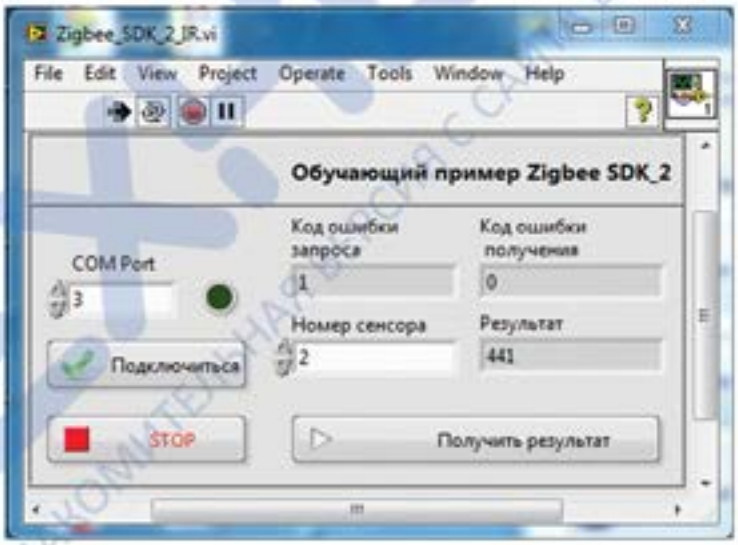
1) Для разработки программы необходимо открыть блок-диаграмму, созданную в предыдущем разделе.
2) Очистить цикл While Loop от содержимого.
3) Внутри цикла While Loop необходимо создать структуру Case Structure и элемент управления (control) с именем «Запрос».
4) Соединить элемент управления «Запрос» со входом структуры Case Structure.
5) В окне True используемой структуры расположить элемент Flat Sequence с тремя окнами (Frame).
6) Поместить внутрь первого окна функцию zgb_tx_data.vi из библиотеки ZigBee. Данная функция предназначена для передачи контроллеру CM-530 номера ИК-датчика, показания которого необходимо передать.
7) Для задания номера ИК-датчика необходимо создать элемент управления с именем «Номер сенсора» и индикатор с именем «Код ошибки запроса». Соединить оба этих элемента со входом и выходом функции zgb_tx_data.vi.
8) Поместить внутрь второго окна функцию zgb_rx_data.vi из библиотеки ZigBee. Данная функция предназначена для получения данных от контроллера CM-530.
9) Создать индикатор с именем «Результат» и соединить его с выходом функции, проведя соединение через третье окно Flat Sequence.
10) В третье окно элемента Flat Sequence необходимо поместить функцию zgb_rx_check.vi, предназначенную для вывода результатов.

11) Создать индикатор с именем «Код ошибки получения» и соединить его с функцией zgb_rx_check.vi.

12) В результате блок-диаграмма должна приобрести вид идентичный рисунку ниже.



Для удобства применения данной программы необходимо настроить ее лицевую панель на подобии того, как это сделано на рисунке ниже.



Порядок работы с программой

1) Включить робота Bioloid и настроить беспроводное соединение по интерфейсу ZigBee.

2) Убедиться в том, что программируемый контроллер CM-530 запрограммирован соответствующим образом.

3) Запустить программу и осуществить подключение к COM-порту. В случае необходимости номер COM-порт можно задавать в соответствующем окне.

4) В окне «Номер сенсора» ввести номер ИК-датчика из массива и нажать кнопку «Получить результат».

Модификация программы контроллера CM-530

Для того чтобы программируемый контроллер робота реагировал на запросы, поступающие от удаленного компьютера, его программа управления должна быть скорректирована соответствующим образом.

```
START PROGRAM
{
    ENDLESS LOOP
    {
        WAIT WHILE ( Remocon Arrived == FALSE )
        ReceiveData = Remocon RXD

        IF ( ReceiveData == 1 )
        {
            Remocon TXD = ID[100], IR Fire Data 1
        }
        IF ( ReceiveData == 2 )
        {
            Remocon TXD = ID[100], IR Fire Data 2
        }
        IF ( ReceiveData == 3 )
        {
            Remocon TXD = ID[100], IR Fire Data 3
        }
        IF ( ReceiveData == 4 )
        {
            Remocon TXD = ID[100], IR Fire Data 4
        }
        IF ( ReceiveData == 5 )
        {
            Remocon TXD = ID[100], IR Fire Data 5
        }
        IF ( ReceiveData == 6 )
        {
            Remocon TXD = ID[100], IR Fire Data 6
        }
        IF ( ReceiveData == 7 )
        {
            Remocon TXD = ID[100], IR Fire Data 7
        }


    }
}
```

Код управляющей программы должен содержать перечень условий, определяющих номер одного из датчиков, с которого необходимо считать значения. Номер каждого из условий должен соответствовать одному из номеров, передаваемых по команде от удаленного компьютера.

Часть 3. Сбор показаний массива ИК-датчиков

Основная задача данного примера – иллюстрация возможностей автоматического сбора данных с мобильного робота и передачи их на удаленный компьютер. Подобные решения очень востребованы в реальных задачах, т.к. обычно процесс сбора и архивации информации происходит автоматически или циклично, а не по отдельным запросам.

Получение своевременной информации о состоянии технической системы дает возможность системе управления скорректировать рабочий режим и выбрать оптимальный алгоритм работы.

Разработка программы в среде программирования LabView.

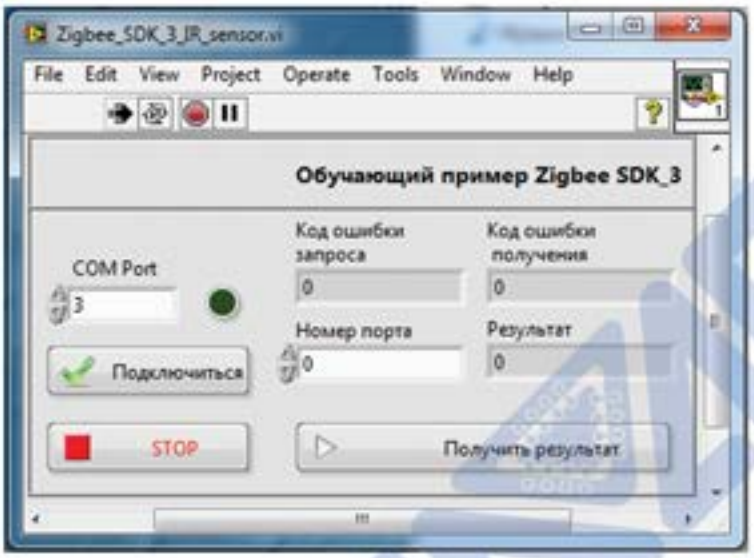
Программа, разрабатываемая в данной части, основывается на блок-диаграмме из предыдущей, поскольку они содержат общую часть, касающуюся передачи данных по COM-порту.

Принцип работы программы заключается в передаче на удаленный компьютер по беспроводному каналу показаний одного из ИК-датчиков, подключенных к одному из портов контроллера СМ-530.

Отличие данной программы от предыдущей заключается в том, что в данной программе осуществляется непрерывный объем данными между контроллером СМ-530 и удаленным компьютером. Для этого необходимо внести изменения в разработанную ранее блок-диаграмму.

Для автоматизации процесса считывания данных необходимо поместить функцию zgb_rx_data.vi в параллельный циклу запроса цикл While Loop. В итоге при нажатии на кнопку «Получить результат» будет проводиться опрос выбранного порта до тех пор, пока не будет выбран другой порт.

Лицевая панель программы не претерпела существенных изменений по сравнению с предыдущей частью. Рекомендуется настроить лицевую панель аналогичным образом, как показано на рисунке ниже.



Порядок работы с программой

1) Включить робота Bioloid и настроить беспроводное соединение по интерфейсу ZigBee.
2) Убедиться в том, что программируемый контроллер CM-530 запрограммирован соответствующим образом.
3) Запустить программу и осуществить подключение к COM-порту. В случае необходимости номер COM-порт можно задавать в соответствующем окне.
4) В окне «Номер порта» ввести номер порта, к которому подключен ИК-датчик и нажать кнопку «Получить результат».
5) Результаты измерений датчика будут отображаться в окне «Результат».

Модификация программы контроллера CM-530

Для того чтобы программируемый контроллер робота реагировал на запросы, поступающие от удаленного компьютера, его программа управления должна быть скорректирована соответствующим образом.

```
1: START PROGRAM
2: {
3:     ENDLESS LOOP
4:     {
5:         WAIT WHILE ( Remocon Arrived == FALSE )
6:         ReceiveData = Remocon RXD
7:
8:         IF ( ReceiveData == 1 )
9:         {
10:            LOOP WHILE ( Remocon RXD == 1 )
11:            {
12:                Remocon TXD = PORT[1]
13:            }
14:        }
15:        IF ( ReceiveData == 2 )
16:        {
17:            LOOP WHILE ( Remocon RXD == 2 )
18:            {
19:                Remocon TXD = PORT[2]
20:            }
21:        }
22:        IF ( ReceiveData == 3 )
23:        {
24:            LOOP WHILE ( Remocon RXD == 3 )
25:            {
26:                Remocon TXD = PORT[3]
27:            }
28:        }
29:        IF ( ReceiveData == 4 )
30:        {
31:            LOOP WHILE ( Remocon RXD == 4 )
32:            {
33:                Remocon TXD = PORT[4]
34:            }
35:        }
36:        IF ( ReceiveData == 5 )
37:        {
38:            LOOP WHILE ( Remocon RXD == 5 )
39:            {
40:                Remocon TXD = PORT[5]
41:            }
42:        }
43:        IF ( ReceiveData == 6 )
44:
45:            LOOP WHILE ( Remocon RXD == 6 )
46:            {
47:                Remocon TXD = PORT[6]
48:            }
49:        }
50:    }
51: }
```


Код управляющей программы должен содержать перечень условий, определяющих номер одного из портов, с которого необходимо считать значения. Номер каждого из условий должен соответствовать одному из номеров, передаваемых по команде от удаленного компьютера.

Часть 4. Управление сервоприводами Dynamixel

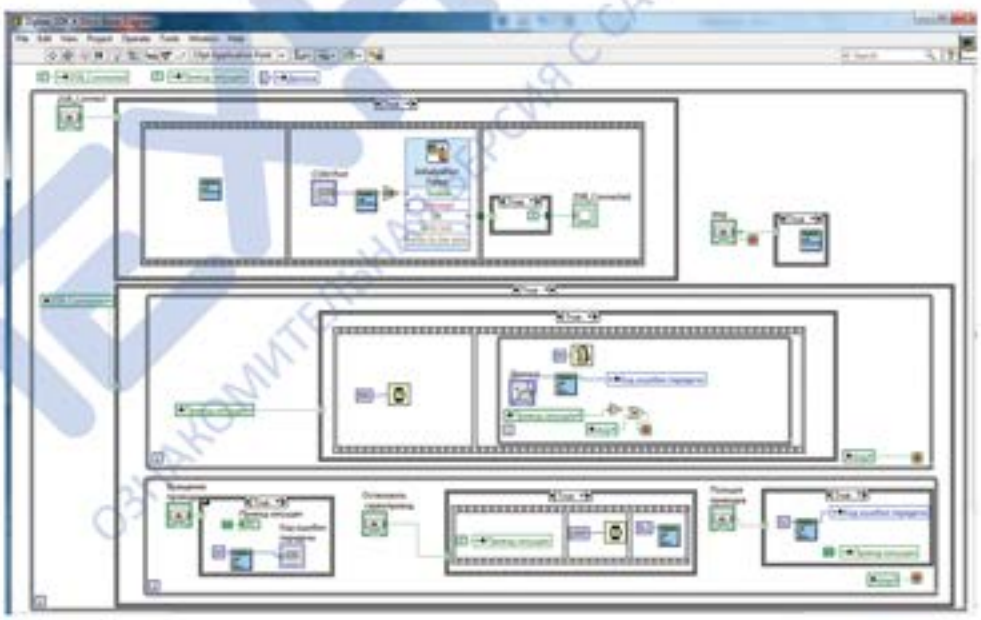
Основная задача данного примера – иллюстрация процесса управления сервоприводами с помощью последовательного протокола. В данной работе предлагается рассмотреть процесс управления сервоприводами Dynamixel с помощью программы, разработанной в программной среде LabView.

Данный пример демонстрирует процесс управления сервоприводами в различных режимах – в режиме управления скоростью вращения выходного вала и в режиме отработки и удержания заданного положения.

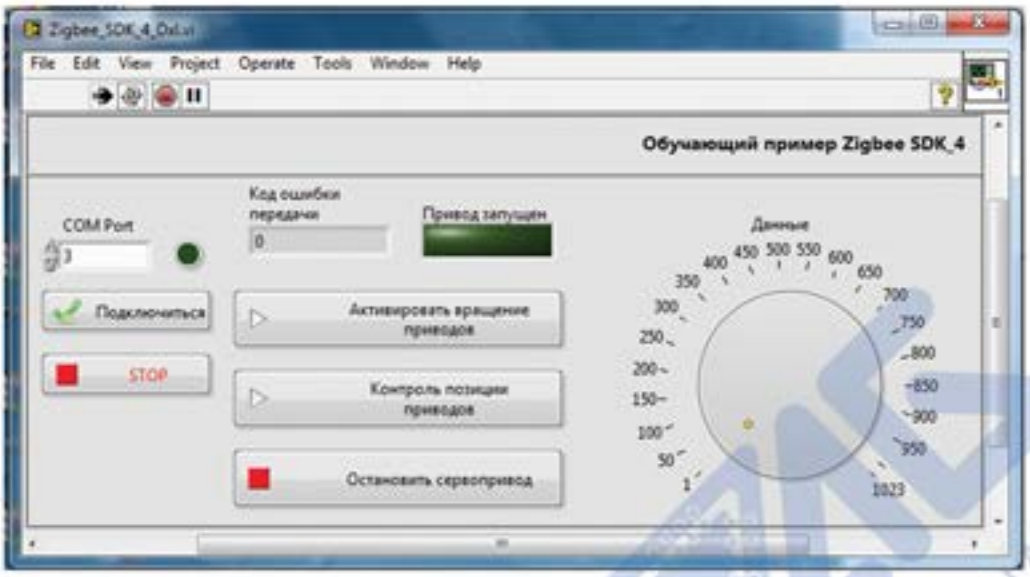
Разработка программы в среде программирования LabView.

Данная программа является завершающей для цикла из четырех работ и обобщает в себе наработки каждой из предыдущих. Также как и в предыдущих частях, программа разрабатывается на основе предыдущей, путем ее усовершенствования. Особенность данной работы в том, что она обобщает в себе два различных режима управления сервоприводами – режим постоянного вращения с заданной скоростью (Wheel), а также режим позиционного управления (Joint).

Поскольку сервопривод может работать в одном из двух вышеуказанных режимов, в блок-диаграмме необходимо добавить два параллельно работающих цикла While Loop. В результате выбора одного из режимов работы на привод поступают данные в диапазоне от 0 до 1023. В режиме Wheel данные представляют собой скорость вращения привода. В режиме Joint данные задают угол поворота привода.



Поступающие данные на привод задаются с помощью специального control элемента, располагаемого на лицевой панели программы.



Порядок работы с программой

1) Установить требуемый режим работы сервоприводов с помощью программы, написанной для контроллера CM-530 в среде RoboPlus.

2) Включить робота Bioloid и наладить канал связи по интерфейсу ZigBee. Убедиться в правильности настроек по свечению светодиодов модулей.

3) Запустить программу, выбрать COM-порт (по умолчанию COM3) и нажать кнопку «Подключиться». В случае успешного подключения к COM-порту загорится лампочка.

4) Выбрать один из режимов работы приводов.

5) Изменяя положение модуля управления «Данные», задать требуемое значение.

6) При завершении работы нажать кнопку «Остановить сервопривод» и затем Stop.

Модификация программы контроллера СМ-530

В отличие от предыдущих примеров в данном случае программа контроллера должна не только реагировать на изменяющиеся входные значения, но и сама функционировать в двух различных режимах.

Программа алгоритмически представляет собой бесконечный цикл, в котором анализируются приходящие по COM-порту данные. Первые пришедшие данные классифицируются как команды, задающие один из режимов работы сервопривода – режим постоянного вращения с помощью функции forward или режим управления положением с помощью функции position.

```
1:  START PROGRAM
2:  {
3:      Buzzer time  = Play Melody
4:      Buzzer index  = Melody0
5:      ENDLESS LOOP
6:      {
7:          WAIT WHILE ( Remocon Arrived  ==  FALSE )
8:          ReceiveData  = Remocon RXD
9:
10:         KeyValue  = ReceiveData  &  U+D+L+R+1+2+3+4+5+6
11:         IF ( KeyValue  ==  1 )
12:         {
13:             Buzzer time  = Play Melody
14:             Buzzer index  = Melody0
15:             CALL forward
16:         }
17:         IF ( KeyValue  ==  2 )
18:         {
19:             Buzzer time  = Play Melody
20:             Buzzer index  = Melody0
21:             CALL position
22:         }
23:     }
24: }
```

Данные, поступающие в контроллер СМ-530, распознаются функциями forward и position как управляющие значение, т.е. как величины задания скорости вращения сервопривода и его целевое положение.

```
25: FUNCTION forward
26: {
27:     LOOP WHILE ( Remocon RXD != 6 )
28:     {
29:         l_wheel_speed = Remocon RXD
30:         r_wheel_speed = Remocon RXD
31:         ID[1]: Moving speed = CCW:0 + l_wheel_speed
32:         ID[2]: Moving speed = CW:0 + r_wheel_speed
33:     }
34:     ID[1]: Moving speed = CCW:0
35:     ID[2]: Moving speed = CW:0
36: }
```

```
37: FUNCTION position
38: {
39:     LOOP WHILE ( Remocon RXD != 6 )
40:     {
41:
42:         ID[1]: Goal position = Remocon RXD
43:         ID[2]: Goal position = Remocon RXD
44:     }
45:     ID[1]: Goal position = 1
46:     ID[2]: Goal position = 1
47: }
```

Аналогичным образом можно задавать различные скорости вращения и целевые положения для любого из сервоприводов, подключенных к контроллеру СМ-530.

Приложение

Данное приложение представляет собой сопроводительную информацию, предназначенную для использования совместно с методическими указаниями для преподавателя с целью применения при подготовке к проведению занятий. Также данные материалы могут быть полезны при проектировании роботов и могут использоваться учащимися как совместно с инструкциями по сборке, так и отдельно в процессе проектирования.

Профессиональное проектирование – сложный и многогранный процесс, сочетающий в себе сбор и анализ информации, исследовательскую деятельность и процесс разработки. Первая часть данных методических указаний содержит инструкции по разработке различных роботов и робототехнических устройств, поясняющие базовые основы применения каких-либо элементов. Данный раздел предназначен для расширения горизонтов проектирования и применения комплектующих робототехнического набора в собственных разработках.

В данном приложении содержится краткое техническое описание основных и наиболее значимых комплектующих, входящих в состав образовательных наборов на базе роботов серии Bioloid. В данном разделе описываются сервопривода, контроллеры, датчики и прочие устройства, применение которых востребовано при разработке роботов.

Сервопривод Dynamixel

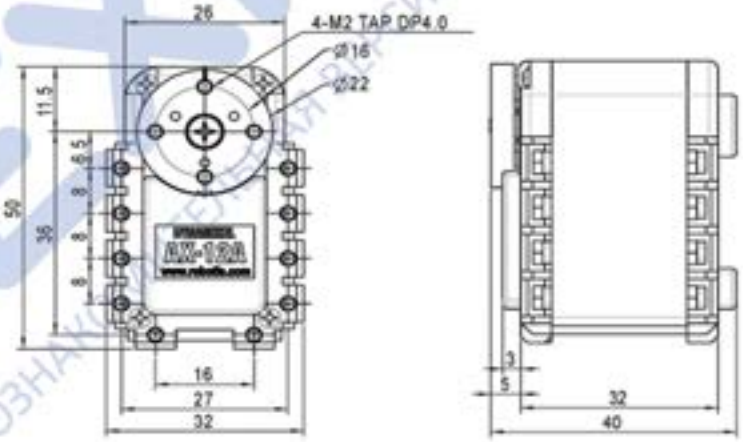
ЭКЗАМЕН ТЕХНОЛАБ

Сервопривод **Dynamixel**

Сервопривод Dynamixel

Введение

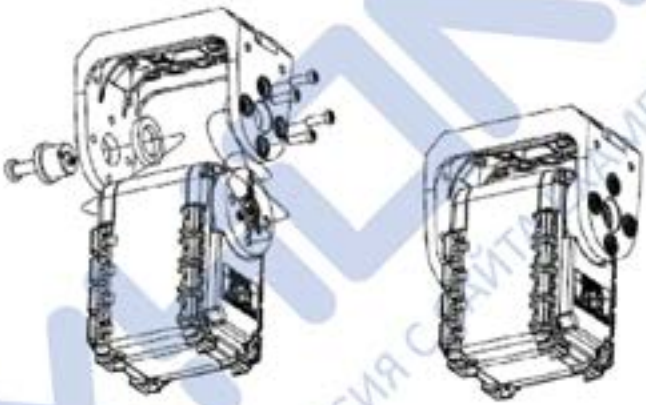
Сервопривод Dynamixel – это уникальное устройство корейской компании ROBOTIS, представляющее собой привод на базе двигателя постоянного тока, совмещенный с прецизионным редуктором и платой управления. Плата управления сервопривода обеспечивает его связь с устройствами управления, принимает управляющие команды, а также передает сообщения обратной связи о текущем состоянии привода, его скорости и положения.



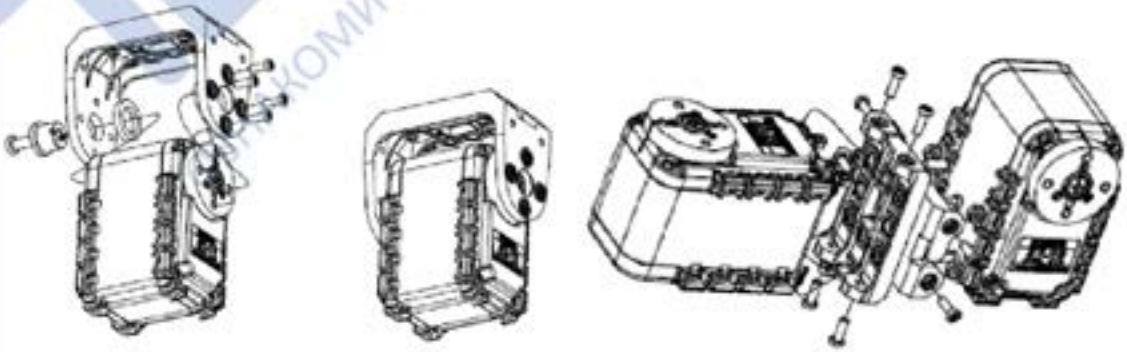
Несмотря на свои небольшие габариты сервопривод развивает достаточно большой крутящий момент, а также может выдерживать высокие нагрузки, возникающие вследствие различных реакций механизма. По сравнению с обычными модельными сервомашинками, сервопривода Dynamixel обладают рядом конкурентных преимуществ:

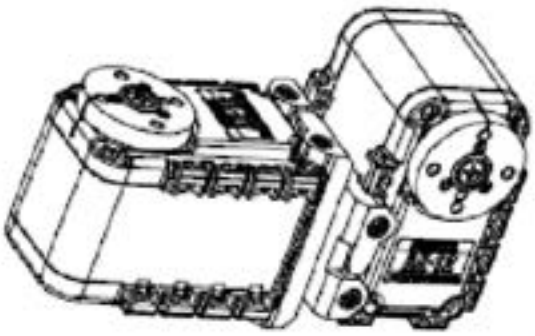
1) Высокая точность с разрешающей способностью 1024.
2) Наличие обратной связи по нагрузке, скорости и положению.
3) Наличие индикации аварийного состояния и текущего статуса.
4) Последовательный интерфейс для управления с помощью пакетов команд.
5) Наличие подшипникового узла на выходном валу привода, что дает возможность применять сервопривод при наличии действующих внешних сил.
6) Многообразие различных крепежных элементов, позволяющих применять привода в различных конструкциях.

Поскольку сервопривода наиболее часто применяются для передачи вращения или поворота механизмов робота на какой-либо угол, для крепления Dynamixel наиболее часто применяются скобы.



Для соединения приводов друг с другом или для установки их в роботе или каком-либо другом механизме, применяются торцевые скобы. Закрепив скобу на одном из торцов сервопривода, становится возможным скреплять сервопривода друг с другом или с конструктивными элементами робота.

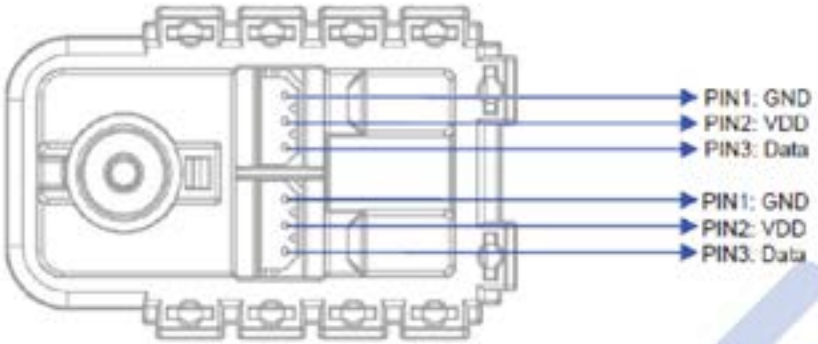




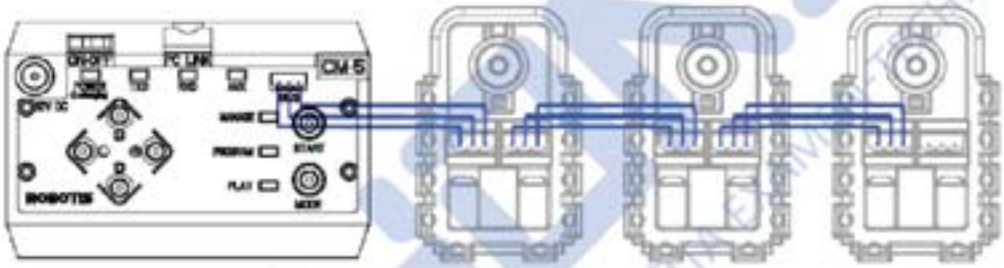
Основные технические характеристики сервопривода AX-12A:

1) Масса – 55 г.
2) Передаточное отношение редуктора – 1/254.
3) Точность (разрешающая способность) – 0,35 угловых градусов.
4) Рабочий диапазон – 300 угловых градусов в режиме отработки положения.
5) Наличие режима постоянного вращения.
6) Потребляемое напряжение питания 7 – 10 В.
7) Рекомендуемое напряжение питания 9,6 В.
8) Потребляемый ток 900 мА.
9) Управление посредством полудуплексного асинхронного последовательного интерфейса.
10) Управление посредством цифровых пакетов специализированного протокола.
11) Возможность использовать в качестве сетевого устройства с идентификационным номером из диапазона 0-253.
12) Наличие обратной связи по положению, скорости, нагрузки, напряжению и др.

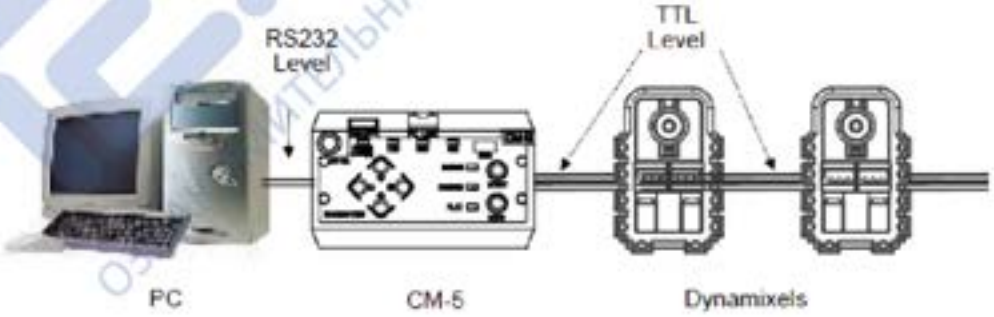
Подключение сервоприводов Dynamixel осуществляется с помощью трехпроводных шлейфов, содержащих информационный канал, канал питания и землю.



Между собой и устройствами управления сервопривода соединяются последовательным образом. Таким образом, каждый из приводов может управляться по последовательной шине и к одному порту устройства управления можно подключить несколько приводов одновременно.



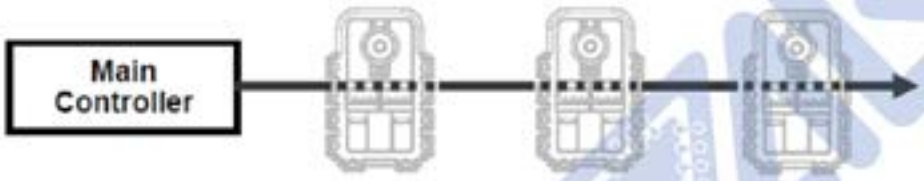
Управление сервоприводами осуществляется с помощью TTL-логики полудуплексного интерфейса UART. Поскольку интерфейс UART является универсальным асинхронным последовательным интерфейсом, то с помощью различных преобразующих устройств последовательную сеть сервоприводов Dynamixel возможно подключить к управляющим контроллерам или персональному компьютеру.



Руководство по применению

Соблюдайте правильность подключения разъемов сервоприводов Dynamixel во избежание их повреждения.

Последовательное подключение сервоприводов дает возможность организовывать сети из различного количества Dynamixel. Помните, что каждый сеанс связи устройства управления с одним из сервоприводов занимает определенное время. Поскольку же система управления роботом должна постоянно обновлять информацию о состоянии приводов, процесс опроса сети может занять достаточно много времени, что существенно скажется на быстродействии системы управления.

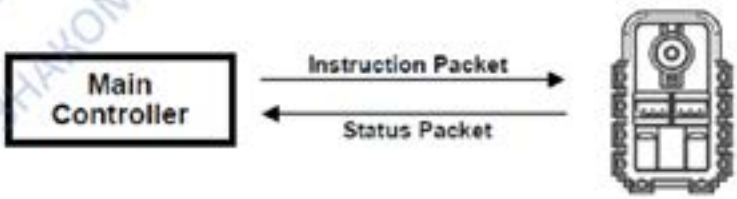


При разработке робота конфигурируйте сети с минимально возможным количеством сервоприводов Dynamixel. В случае если по проекту число приводов превышает допустимое значение, определяемое производительностью управляющего контроллера, рекомендуется разделять сервопривода на различные подсети, управляемые разными контроллерами.

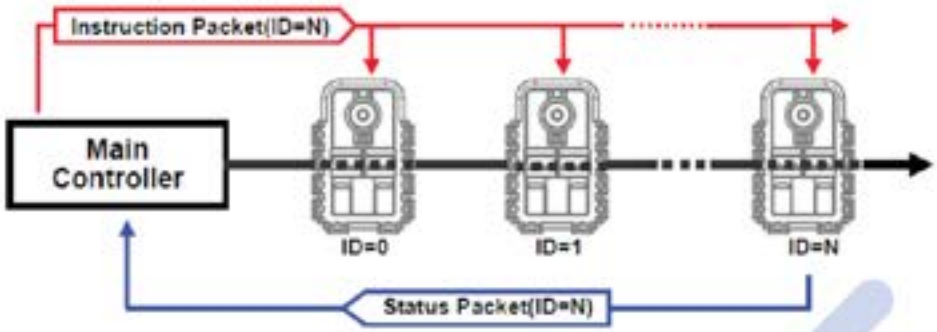
При разработке программы управления особое внимание рекомендуется уделять информации о состоянии сервопривода. Сервопривода Dynamixel в качестве обратной связи могут передавать сообщения о собственном состоянии и ошибках, возникших в процессе работы. Своевременное использование подобной информации дает возможность системе управления оперативно отреагировать на аварийную ситуацию и скорректировать алгоритм управления или остановить робота.

Управление сервоприводами Dynamixel

Управление сервоприводами Dynamixel осуществляется с помощью команд передаваемых от управляющего устройства по последовательному интерфейсу. Взаимосвязь сети сервоприводов и управляющего устройства осуществляется в две стороны – как на передачу команд или инструкций «Instruction Packet», так и на передачу сообщений о состоянии сервопривода «Status Packet».



Управляющий контроллер рассылает пакеты с инструкциями сервоприводам с определенным ID, а те в ответ высылают пакет с текущим статусом. Подобный процесс происходит асинхронным образом для каждого из сервоприводов.



Важно соблюдать порядок в нумерации ID номерами сервоприводов в сети. Каждый сервопривод в сети должен иметь свой уникальный идентификационный номер. В случае совпадения ID номеров устройств в сети это неизбежно приведет к возникновению коллизий передачи данных и потери управляющих команд, что может привести к возникновению аварийных ситуаций при работе робота.

Протокол передачи данных Dynamixel

Передача управляющих и статусных пакетов между управляющими устройствами и Dynamixel осуществляется с помощью асинхронного последовательного протокола (идентичного UART), содержащего 8 бит данных и 1 бит контроля четности. Контроль четности или паритета представляет собой наиболее простой метод оценки правильности передачи данных между устройствами, с его помощью можно определить возникновение одиночной ошибки в передаваемом сообщении.

Пакет передаваемых инструкций (Instruction Packet) выглядит следующим образом:

0XFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	...	PARAMETER N	CHECK SUM

0XFF 0XFF

Стартовые байты, определяющие начало нового пакета, с помощью которых осуществляется синхронизация передаваемых сообщений.

ID

Идентификационный номер сервопривода Dynamixel. Каждый привод должен иметь собственный ID в диапазоне, всего доступно до 254 идентификационных номеров. Диапазон значений данного байта от 0X00 до 0XFD. Помимо уникальных идентификационных номеров зарезервирован специальный ID (Broadcasting ID) для широковещательной связи управляющего контроллера со всеми сервоприводами сети. В случае применения широковещательной связи связь осуществляется в одном направлении – от контроллера к сервоприводам, т.е. на каждый из сервоприводов сети поступает управляющий пакет.

LENGTH

Длина Instruction Packet вычисляется по формуле «Число параметров N + 2».

INSTRUCTION

Инструкция, передаваемая сервоприводу.

PARAMETR

Используется в случае передачи дополнительной информации помимо инструкций.

CHECK SUM

Контрольная сумма, значение, рассчитываемое согласно следующей формуле CHECK SUM = ~(ID + LENGTH + INSTRUCTION + PARAMETR_1 + ... + PARAMETR_N). Данное значение используется для проверки целостности полученных после передачи данных.

Пакет с передаваемой информацией (Status Packet) о состоянии сервопривода Dynamixel выглядит следующим образом:

0XFF	0XFF	ID	LENGTH	ERROR	PARAMETER1	PARAMETER2	...	PARAMETER N	CHECK SUM

0XFF 0XFF

Стартовые байты, определяющие начало нового пакета, с помощью которых осуществляется синхронизация передаваемых сообщений.

ID

Идентификационный номер сервопривода Dynamixel. Каждый привод должен иметь собственный ID в диапазоне, всего доступно до 254 идентификационных номеров. Диапазон значений данного байта от 0X00 до 0XFD. Помимо уникальных идентификационных номеров зарезервирован специальный ID (Broadcasting ID) для широковещательной связи управляющего контроллера со всеми сервоприводами сети. В случае применения широковещательной связи связь осуществляется в одном направлении – от контроллера к сервоприводам, т.е. на каждый из сервоприводов сети поступает управляющий пакет.

LENGTH

Длина Instruction Packet вычисляется по формуле «Число параметров N + 2».

ERROR

Данный байт содержит информацию о коде ошибки, передаваемой от сервопривода к управляющему контроллеру. Коды ошибок и их описание приводятся в таблице ниже.

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	Overload Error	Set to 1 if the specified maximum torque can't control the applied load.
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect.
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range.
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

PARAMETR

Используется в случае передачи дополнительной информации помимо инструкций.

CHECK SUM

Контрольная сумма, значение, рассчитываемое согласно следующей формуле CHECK SUM = ~(ID + LENGTH + INSTRUCTION + PARAMETR_1 + ... + PARAMETR_N). Данное значение используется для проверки целостности полученных после передачи данных.

Вся передаваемая с помощью протокола информация – управляющие инструкции и статусные сообщения перечислены в Control table. Сервопривод управляется с помощью передаваемых данных типа WR (write) и выдает данные типа RD (read).

Control Table

	Address	Item	Access	Initial Value
EEPROM Area	0(0X00)	Model Number(L)	RD	12(0x0C)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	1(0x01)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
	7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
	8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
	9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
	15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	Alarm LED	RD,WR	4(0x04)
	18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Down Calibration(L)	RD	?
	21(0X15)	Down Calibration(H)	RD	?
	22(0X16)	Up Calibration(L)	RD	?
	23(0X17)	Up Calibration(H)	RD	?
RAM Area	24(0X18)	Torque Enable	RD,WR	0(0x00)
	25(0X19)	LED	RD,WR	0(0x00)
	26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
	27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
	28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
	29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
	30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
	31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
	32(0X20)	Moving Speed(L)	RD,WR	0
	33(0X21)	Moving Speed(H)	RD,WR	0
	34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
	35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
	36(0X24)	Present Position(L)	RD	?
	37(0X25)	Present Position(H)	RD	?
	38(0X26)	Present Speed(L)	RD	?
	39(0X27)	Present Speed(H)	RD	?
	40(0X28)	Present Load(L)	RD	?
	41(0X29)	Present Load(H)	RD	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46(0x2E)	Moving	RD	0(0x00)
	47(0x2F)	Lock	RD,WR	0(0x00)
	48(0x30)	Punch(L)	RD,WR	32(0x20)
	49(0x31)	Punch(H)	RD,WR	0(0x00)

Адреса, приведенные в таблице Control table, могут применяться в качестве данных, передаваемых в пакетах, над которыми могут выполняться различные операции.

Помимо данных с помощью последовательного интерфейса Dynamixel передаются инструкции, приведенные в таблице ниже:

Instruction	Function	Value	Number of Parameter
PING	No action. Used for obtaining a Status Packet	0x01	0
READ DATA	Reading values in the Control Table	0x02	2
WRITE DATA	Writing values to the Control Table	0x03	2 ~
REG WRITE	Similar to WRITE_DATA, but stays in standby mode until the ACION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by the REG_WRITE instruction	0x05	0
RESET	Changes the control table values of the Dynamixel actuator to the Factory Default Value settings	0x06	0
SYNC WRITE	Used for controlling many Dynamixel actuators at the same time	0x83	4~

С помощью данных инструкций задаются операции, которые выполняются с передаваемыми по протоколу данными.

Пример пакетов передачи данных между устройством управления и сервоприводом Dynamixel.

Пример № 1. Установка ID=1 для сервопривода Dynamixel.

В данном примере пакет данных отправляется сервоприводу методом широковещательной передачи, поэтому в байт ID записано значение 0XFE. Данный метод следует применять, когда идентификационный номер присваивается сервоприводу с неизвестным до этого ID.

Общая длина пакета составляет 4 байта, в числе которых: 3 служебных байта и 1 байт с данными.

Пример № 2. Чтение данных от сервопривода Dynamixel.

В данном примере устройство управления обращается к сервоприводу с ID = 1. В качестве команды применяется инструкция READ_DATA, описываемая передаваемым значением 0X02. В качестве демонстрации работы данной функции производится считывание температуры сервопривода. Данное значение хранится в RAM по адресу 0X2B, которое передается в виде данных в размере одного байта 0X01.

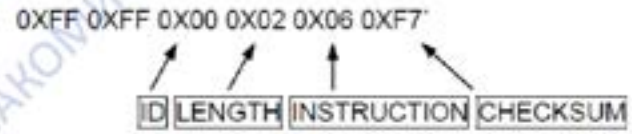
Ответное сообщение содержит 1 байт данных, представляющих собой измеренное значение температуры сервопривода 0X20, что равно 32 градусам по шкале Цельсия.

Пример № 3. Получение сообщения о состоянии Dynamixel.

Запрос пакета статусного сообщения осуществляется с помощью команды PING, описываемой соответствующим байтом со значением данных 0X01. Общая длина посылки составляет два байта и состоит из байта ID и байта инструкции.

В ответ на отправленный запрос сервопривод высылает ответное сообщение, содержащее байт со значением кода ошибки.

Пример № 4. Сброс текущего состояния Dynamixel.



С помощью функции RESET осуществляется сброс текущего состояния сервопривода Dynamixel к базовым настройкам. Если данный сервопривод имел какой-либо идентификационный номер, находился в состоянии ошибки или аварии, все соответствующие регистры будут обнулены.

Возвращаемый статусный пакет имеет следующий вид:

В данном разделе были рассмотрены основы передачи данных по асинхронному последовательному интерфейсу сервоприводов Dynamixel. С помощью данного протокола пользователь может самостоятельно управлять сервоприводом, посылать на него управляющие команды и считывать текущее состояние.

Сервоприводу Dynamixel являются уникальным в своем роде устройством, сочетающим техническую оснащенность на уровне профессионального промышленного оборудования наряду с простотой комплектующих, традиционно входящих в различные робототехнические конструкторы. Применение сервоприводов Dynamixel в процессе изучения робототехники позволяет сделать серьезный шаг от игрового процесса к профессиональной разработке роботов и проведения исследовательской деятельности.

Универсальный сенсорный модуль AX-S1

ЭКЗАМЕН ТЕХНОЛАБ

Универсальный сенсорный модуль **AX-S1**

Универсальный сенсорный модуль AX-S1

Введение

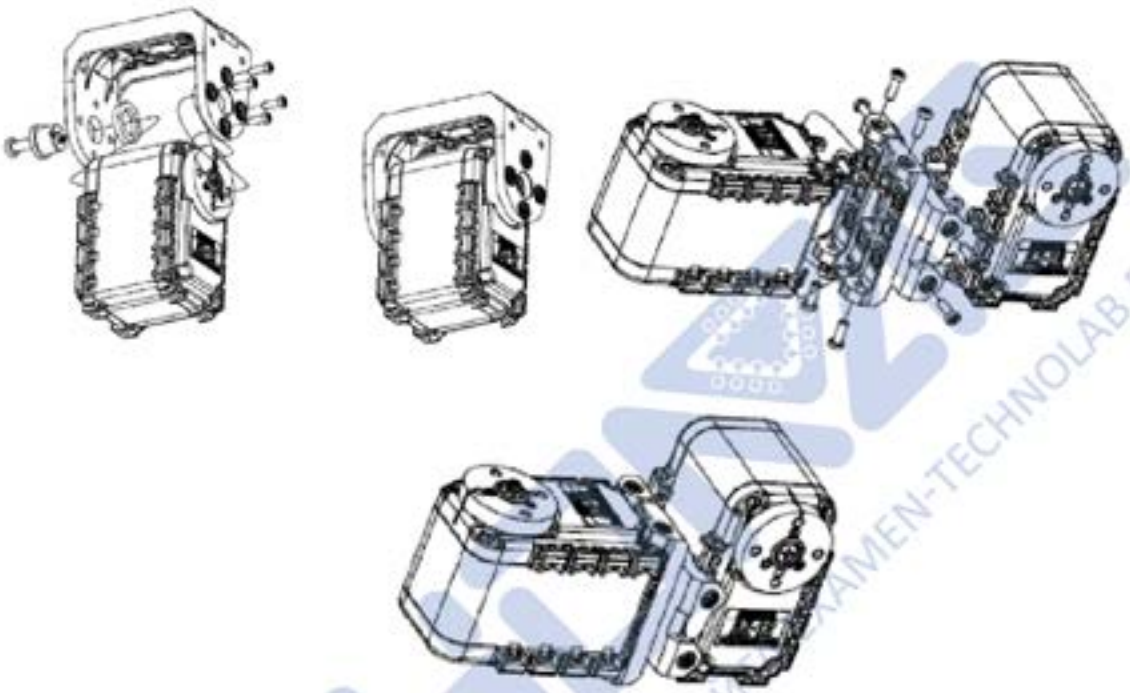
Универсальный сенсорный модуль AX-S1 является уникальным изделием корейской компании ROBOTIS, сочетающим в себе датчик звука, ИК-приемник, ИК-датчик, датчик освещенности, а также динамик. Также отличительной особенностью данного устройства является то, что передача данных и управляющих команд осуществляется по асинхронному последовательному протоколу, идентичному тому, что применяется в сервоприводах Dynamixel. Благодаря этому данное устройство можно применять в сетях, содержащих сервопривода, датчики и другие устройства.

Таким образом, применение модуля AX-S1 наряду с применением сервоприводов Dynamixel позволяет существенно повысить функционал робота за счет того, что применение подобных сетевых устройств не ограничено количеством портов ввода/вывода программируемого контроллера. Используя сервопривода Dynamixel и сенсорные устройства модуля AX-S1, пользователь может разрабатывать различных роботов, содержащих множество степеней подвижности и сенсорных устройств.

Отличительные особенности сенсорного модуля AX-S1:

1) Высокая точность измерений с разрешающей способностью 1024.
2) Наличие обратной связи по всем измеряемым параметрам на основе последовательного интерфейса.
3) Наличие режима предупреждения выхода измеряемых параметров за пределы допустимого диапазона.
4) Наличие ИК-модуля, позволяющего передавать данные между устройствами AX-S1.
5) Множество различных устройств встроены в общий корпус и управляются посредством единого протокола.

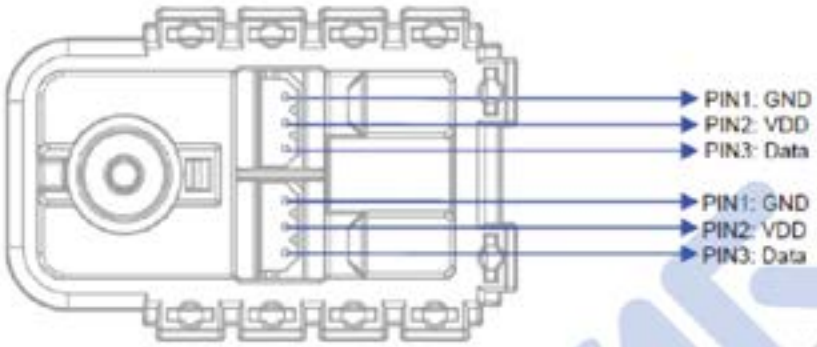
Устройство AX-S1 обладает такими же габаритными размерами и креплением, как и сервопривода Dynamixel. Благодаря этому сенсорный модуль можно крепить с помощью стандартных скоб, применяемых для крепления сервоприводов Dynamixel, сходящих в состав роботов Bioloid.



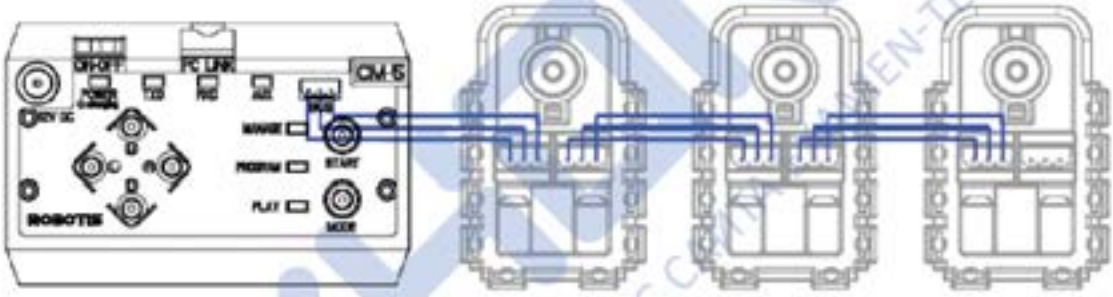
Основные технические характеристики устройства AX-S1:

1) Масса – 37 г.
2) Разрежающая способность – 1024 (10 бит).
3) Потребляемое напряжение питания 7 – 10 В.
4) Рекомендуемое напряжение питания 9,6 В.
5) Рабочая температура -5 … +85 градусов шкалы Цельсия.
6) Управление посредством цифровых пакетов специализированного протокола.
7) Возможность использовать в качестве сетевого устройства с идентификационным номером из диапазона 0 – 253.
8) Наличие встроенного ИК-модуля для приема и передачи сигналов.
9) Наличие обратной связи от встроенных устройств – датчик, температуры, микрофон, ИК-датчик.

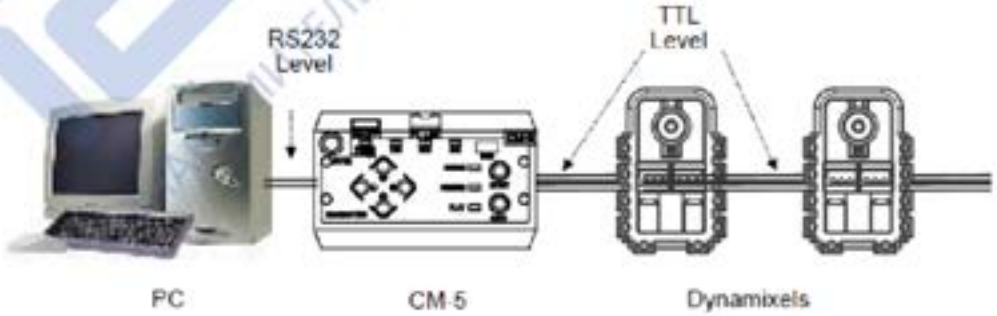
Подключение устройства AX-S1 осуществляется с помощью трехпроводных шлейфов, содержащих информационный канал, канал питания и землю.



Между собой и устройствами управления AX-S1 соединяются последовательным образом. Таким образом, каждый из модулей может управляться по последовательной шине и к одному порту устройства управления можно подключить несколько модулей одновременно.

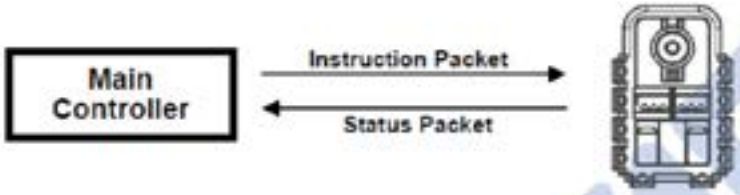


Управление устройствами AX-S1 осуществляется с помощью TTL-логики полудуплексного интерфейса UART. Поскольку интерфейс UART является универсальным асинхронным последовательным интерфейсом, то с помощью различных преобразующих устройств последовательную сеть AX-S1 возможно подключить к управляющим контроллерам или персональному компьютеру.



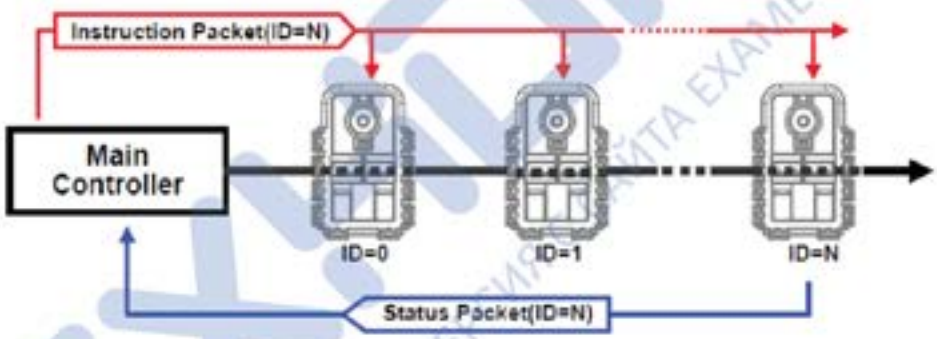
Управление универсальным сенсорным модулем AX-S1.

Управление модулем AX-S1 осуществляется с помощью команд, передаваемых от управляющего устройства по последовательному интерфейсу. Взаимосвязь сети AX-S1 и управляющего устройства осуществляется в две стороны – как на передачу команд или инструкций «Instruction Packet», так и на передачу сообщений о состоянии сервопривода «Status Packet».



Управляющий контроллер рассылает пакеты с инструкциями модулям AX-S1 с определенным ID, а те в ответ высылают пакет с текущим статусом. Подобный процесс происходит асинхронным образом для каждого из устройств в отдельности.

В целом управление модулем AX-S1 полностью идентично управлению сервоприводами Dynamixel. Данные устройства работают на базе общего интерфейса и управляются с помощью одинакового протокола.



Важно соблюдать порядок в нумерации ID номерами модулей AX-S1 в сети. Каждый модуль в сети должен иметь свой уникальный идентификационный номер. В случае совпадения ID номеров устройств в сети это неизбежно приведет к возникновению коллизий передачи данных и потери управляющих команд, что может привести к возникновению аварийных ситуаций при работе робота.

Протокол передачи данных AX-S1

Передача управляющих и статусных пакетов между управляющими устройствами и AX-S1 осуществляется с помощью асинхронного последовательного протокола (идентичного UART), содержащего 8 бит данных и 1 бит контроля четности. Контроль четности или паритета представляет собой наиболее простой метод оценки правильности передачи данных между устройствами, с его помощью можно определить возникновение одиночной ошибки в передаваемом сообщении.

Пакет передаваемых инструкций (Instruction Packet) выглядит следующим образом:

0XFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	...	PARAMETER N	CHECK SUM

0XFF 0XFF

Стартовые байты, определяющие начало нового пакета, с помощью которых осуществляется синхронизация передаваемых сообщений.

ID

Идентификационный номер модуля AX-S1. Каждый привод должен иметь собственный ID в диапазоне, всего доступно до 254 идентификационных номеров. Диапазон значений данного байта от 0X00 до 0XFD. Помимо уникальных идентификационных номеров зарезервирован специальный ID (Broadcasting ID) для широковещательной связи управляющего контроллера со всеми сервоприводами сети. В случае применения широковещательной связи связь осуществляется в одном направлении – от контроллера к сервоприводам, т.е. на каждый из сервоприводов сети поступает управляющий пакет.

LENGTH

Длина Instruction Packet, вычисляется по формуле «Число параметров N + 2».

INSTRUCTION

Инструкция, передаваемая модулю AX-S1.

PARAMETR

Используется в случае передачи дополнительной информации помимо инструкций.

CHECK SUM

Контрольная сумма, значение, рассчитываемое согласно следующей формуле CHECK SUM = ~(ID + LENGTH + INSTRUCTION + PARAMETR_1 + ... + PARAMETR_N). Данное значение используется для проверки целостности полученных после передачи данных.

Пакет с передаваемой информацией (Status Packet) о состоянии модуля AX-S1 выглядит следующим образом:

OXFF	OXFF	ID	LENGTH	ERROR	PARAMETER1	PARAMETER2	...	PARAMETER N	CHECK SUM

0XFF 0XFF

Стартовые байты, определяющие начало нового пакета, с помощью которых осуществляется синхронизация передаваемых сообщений.

ID

Идентификационный номер модуля AX-S1. Каждый привод должен иметь собственный ID в диапазоне, всего доступно до 254 идентификационных номеров. Диапазон значений данного байта от 0X00 до 0XFD. Помимо уникальных идентификационных номеров зарезервирован специальный ID (Broadcasting ID) для широковещательной связи управляющего контроллера со всеми AX-S1 сети. В случае применения широковещательной связи связь осуществляется в одном направлении – от контроллера к AX-S1, т.е. на каждый из модулей сети поступает управляющий пакет.

LENGTH

Длина Instruction Packet вычисляется по формуле «Число параметров N + 2».

ERROR

Данный байт содержит информацию о коде ошибки, передаваемой от AX-S1 к управляющему контроллеру. Коды ошибок и их описание приводятся в таблице ниже.

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	0	
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Inout Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

PARAMETR

Используется в случае передачи дополнительной информации помимо инструкций.

CHECK SUM

Контрольная сумма, значение, рассчитываемое согласно следующей формуле CHECK SUM = ~(ID + LENGTH + INSTRUCTION + PARAMETR_1 + ... + PARAMETR_N). Данное значение используется для проверки целостности полученных после передачи данных.

Вся передаваемая с помощью протокола информация – управляющие инструкции и статусные сообщения перечислены в Control table. Модуль AX-S1 управляется с помощью передаваемых данных типа WR (write) и выдает данные типа RD (read).

	Address	Item	Access	Initial Value
Control Table / EEPROM Area	0(0X00)	Model Number(L)	RD	13(0x0D)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	100(0x64)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	(Reserved)	RD,WR	255(0xFF)
	7(0X07)	(Reserved)	RD,WR	3(0x03)
	8(0X08)	(Reserved)	RD,WR	255(0xFF)
	9(0X09)	(Reserved)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	100(0x64)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	(Reserved)	RD,WR	255(0XFF)
	15(0X0F)	(Reserved)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	(Reserved)	RD,WR	4(0x04)
	18(0X12)	(Reserved)	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Obstacle Detected Compare Value	RD,WR	32(0x20)
	21(0X15)	Light Detected Compare Value	RD,WR	32(0x20)
	22(0X16)	(Reserved)	RD,WR	32(0x20)
	23(0X17)	(Reserved)	RD	3(0x03)
RAM Area	24(0X18)	(Reserved)	RD,WR	0(0x00)
	25(0X19)	(Reserved)	RD,WR	0(0x00)
	26(0X1A)	Left IR Sensor Data	RD	?
	27(0X1B)	Center IR Sensor Data	RD	?
	28(0X1C)	Right IR Sensor Data	RD	?
	29(0X1D)	Left Luminosity	RD	?
	30(0X1E)	Center Luminosity	RD	?
	31(0X1F)	Right Luminosity	RD	?
	32(0X20)	Obstacle Detection Flag	RD	?
	33(0X21)	Luminosity Detection Flag	RD	?
	34(0X22)	(Reserved)	RD,WR	0
	35(0X23)	Sound Data	RD,WR	?
	36(0X24)	Sound Data Max Hold	RD,WR	?
	37(0X25)	Sound Detected Count	RD,WR	?
	38(0X26)	Sound Detected Time(L)	RD,WR	?
	39(0X27)	Sound Detected Time(H)	RD,WR	?
	40(0X28)	Buzzer Index	RD,WR	?
	41(0X29)	Buzzer Time	RD,WR	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46(0x2E)	IR Remocon Arrived	RD	0(0x00)
	47(0x2F)	Lock	RD,WR	0(0x00)
	48(0x30)	IR Remocon RX Data 0	RD	?
	49(0x31)	IR Remocon RX Data 1	RD	?
	50(0x32)	IR Remocon TX Data 0	RD,WR	?
	51(0x33)	IR Remocon TX Data 1	RD,WR	?
	52(0x34)	Obstacle Detected Compare	RD,WR	?
	53(0x35)	Light Detected Compare	RD,WR	?

Адреса, приведенные в таблице Control table, могут применяться в качестве данных, передаваемых в пакетах, над которыми могут выполняться различные операции.

Помимо данных с помощью последовательного интерфейса AX-S1 передаются инструкции, приведенные в таблице ниже:

Instruction	Function	Value	Number of Parameter
PING	No action. Used for obtaining a Status Packet	0x01	0
READ DATA	Reading values in the Control Table	0x02	2
WRITE DATA	Writing values to the Control Table	0x03	2 ~
REG WRITE	Similar to WRITE_DATA, but stays in standby mode until the ACION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by the REG_WRITE instruction	0x05	0
RESET	Changes the control table values of the Dynamixel actuator to the Factory Default Value settings	0x06	0
SYNC WRITE	Used for controlling many Dynamixel actuators at the same time	0x83	4 ~

С помощью данных инструкций задаются операции, которые выполняются с передаваемыми по протоколу данными.

Пример пакетов передачи данных между устройством управления модулем AX-S1.

Пример № 1. Установка ID=1 для модуля AX-S1.

В данном примере пакет данных отправляется AX-S1 методом широковещательной передачи, поэтому в байт ID записано значение 0XFE. Данный метод следует применять, когда идентификационный номер присваивается AX-S1 с неизвестным до этого ID.

Общая длина пакета составляет 4 байта, в числе которых: 3 служебных байта и 1 байт с данными.

Пример № 2. Чтение данных от модуля AX-S1.

В данном примере устройство управления обращается к модулю с ID = 1. В качестве команды применяется инструкция READ_DATA, описываемая передаваемым значением 0X02. В качестве демонстрации работы данной функции производится считывание температуры AX-S1. Данное значение хранится в RAM по адресу 0X2B, которое передается в виде данных в размере одного байта 0X01.

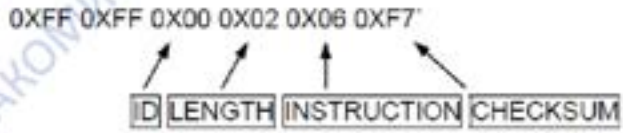
Ответное сообщение содержит 1 байт данных, представляющих собой измеренное значение температуры модуля AX-S1 0X20, что равно 32 градусам по шкале Цельсия.

Пример № 3. Получение сообщения о состоянии модуля AX-S1.

Запрос пакета статусного сообщения осуществляется с помощью команды PING, описываемой соответствующим байтом со значением данных 0X01. Общая длина посылки составляет два байта и состоит из байта ID и байта инструкции.

В ответ на отправленный запрос AX-S1 высылает ответное сообщение, содержащее байт со значением кода ошибки.

Пример № 4. Сброс текущего состояния модуля AX-S1.



С помощью функции RESET осуществляется сброс текущего состояния модуля AX-S1 к базовым настройкам. Если данный AX-S1 имел какой-либо идентификационный номер, находился в состоянии ошибки или аварии, все соответствующие регистры будут обнулены.

Возвращаемый статусный пакет имеет следующий вид:

Пример № 5. Запрос показаний ИК-датчика модуля AX-S1.

Instruction Packet	Instruction = READ_DATA, Address = 0x1C, DATA = 0x01
Communication	->[Dynamixel]:FF FF 64 04 02 1C 01 78 (LEN:008) <-[Dynamixel]:FF FF 64 03 00 21 77 (LEN:007)

С помощью функции READ_DATA считывается один байт данных по адресу 0X1C. По указанному адресу хранятся показания правого ИК-датчика, входящего в состав модуля AX-S1.

Возвращаемое сообщение содержит байт данных со значением 0X21.

Пример № 6. Запрос показаний датчика освещенности.

Instruction Packet	Instruction = READ_DATA, Address = 0x1E, DATA = 0x01
Communication	->[Dynamixel]:FF FF 64 04 02 1E 01 76 (LEN:008) <-[Dynamixel]:FF FF 64 03 00 00 98 (LEN:007)

С помощью функции READ_DATA считывается один байт данных по адресу 0X1E. По указанному адресу хранятся показания правого ИК-датчика, входящего в состав модуля AX-S1.

Возвращаемое сообщение содержит байт данных со значением 0X00.

Пример № 7. Считывание показаний измерения микрофоном амплитуды внешних звуков.

Instruction Packet	Instruction = READ_DATA, Address = 0x23, DATA = 0x01
Communication	->[Dynamixel]:FF FF 64 04 02 23 01 71 (LEN:08) <-[Dynamixel]:FF FF 64 03 00 7E 1A (LEN:007)

С помощью функции READ_DATA считывается один байт данных по адресу 0X23. По указанному адресу хранятся показания правого ИК-датчика, входящего в состав модуля AX-S1.

Возвращаемое сообщение содержит байт данных со значением 0X7E.

Пример № 8. Считывание количества зафиксированных звуков.

Instruction Packet	Instruction = READ_DATA, Address = 0x25, DATA = 0x01
Communication	->[Dynamixel]:FF FF 64 04 02 25 01 6F (LEN:008) <-[Dynamixel]:FF FF 64 03 00 02 96 (LEN:007)

С помощью функции READ_DATA считывается один байт данных по адресу 0X25. По указанному адресу хранятся показания правого ИК-датчика, входящего в состав модуля AX-S1.

Возвращаемое сообщение содержит байт данных со значением 0X02.

Пример № 9. Воспроизведение звука заданной нотной тональности определенное время.

Instruction Packet	ID=100, Instruction = WRITE_DATA, Address = 0x28, DATA = 0x05, 0xFF
Communication	->[Dynamixel]:FF FF 64 05 03 28 05 FF 67 (LEN:009) <-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

С помощью функции WRITE_DATA записываются два байта данных по адресу 0X28. Первый байт представляет собой номер ноты, а второй – продолжительность ее воспроизведения.

В данном разделе были рассмотрены основы передачи данных по асинхронному последовательному интерфейсу, используемому универсальным сенсорным модулем AX-S1 и сервоприводами Dynamixel. С помощью данного протокола пользователь может самостоятельно управлять устройствами, посылать на них управляющие команды и считывать текущее состояние.

Универсальный сенсорный модуль AX-S1 является уникальным в своем роде устройством, сочетающим техническую оснащенность на уровне профессионального промышленного оборудования, наряду с простотой комплектующих, традиционно входящих в различные робототехнические конструкторы. Применение подобных устройств в процессе изучения робототехники позволяет сделать серьезный шаг от игрового процесса к профессиональной разработке роботов и проведения исследовательской деятельности.

Преобразователь интерфейсов USB2Dynamixel

ЭКЗАМЕН ТЕХНОЛАБ

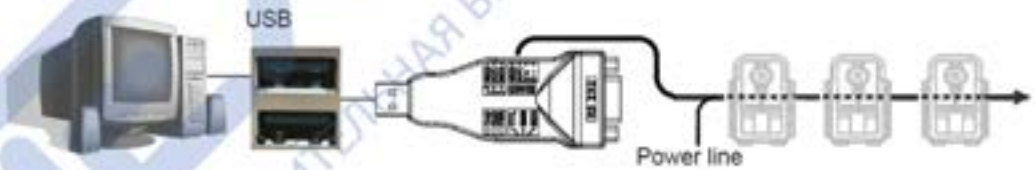
Преобразователь интерфейсов **USB2Dynamixel**

Преобразователь интерфейсов USB2Dynamixel

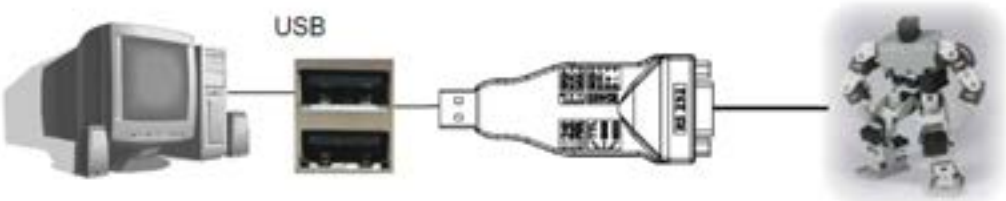
Введение

Преобразователь USB2Dynamixel – устройство, предназначенное для согласования интерфейсов TTL (UART), RS232, RS485 друг с другом и с USB интерфейсом персонального компьютера.

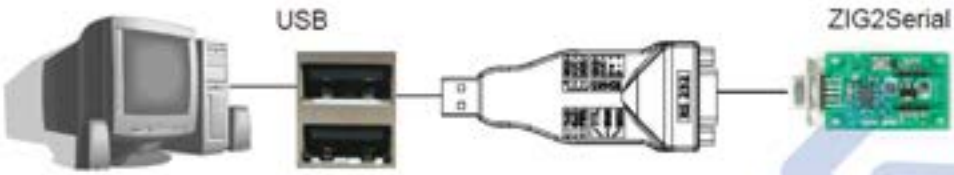
С помощью USB2Dynamixel возможно подключение напрямую к компьютеру любых устройств, функционирующих на базе протокола Dynamixel. Например, для управления сервоприводами Dynamixel с помощью компьютера достаточно подключить сеть сервоприводов к преобразователю, подать на них питание и подключить преобразователь USB2Dynamixel к компьютеру. Получившееся соединение распознается компьютером как виртуальный COM-порт, что без труда позволяет управлять сервоприводами и роботами на их базе с помощью программ, разработанных на компьютере и в различных средах программирования.



Также с помощью USB2Dynamixel можно осуществлять программирование роботов Bioloid, в которых применяется устаревшая модификация управляющего контроллера CM2, CM5, CM510, CM530.



Преобразователь USB2Dynamixel может также применяться для того, чтобы согласовывать работу различных устройств робототехнических наборов Bioloid с персональным компьютером. В частности, с помощью USB2Dynamixel осуществляется подключение ZIGBee передатчиков к компьютеру, благодаря чему становится возможным дистанционно управлять роботами из программной среды LabView.



Описание портов USB2 Dynamixel

Преобразователь USB2Dynamixel содержит набор портов для подключения различных устройств и согласования их интерфейсов.

Для подключения к компьютеру применяется стандартный USB – порт, распознаваемый компьютером как виртуальный COM-порт. Сбоку USB2Dynamixel расположены два порта – 4pin (4P) и 3pin (3P), предназначенные для подключения сервоприводов Dynamixel и устройств, работающих на базе их протокола.

Справа USB2Dynamixel содержит набор светодиодов, отображающих режим работы устройства и панель с переключателями режима.

Трехпроводный порт (3P) предназначен для подключения устройств, входящих в робототехнические наборы серии Bioloid – сервопривода Dynamixel, универсальный сенсорный модуль AX-S1, массив ИК-датчиков и прочее. Данный порт поддерживает последовательные сети на основе TTL (UART).

Четырехпроводный порт (4P) предназначен для подключения сервоприводов старшего модельного ряда, работающих на базе интерфейса RS-485.

Последовательный порт (9pin) применяется для подключения внешних устройств, работающих на базе интерфейса RS-232.

Подключение к USB2 Dynamixel сервоприводов Dynamixel серии AX

Подключение сервоприводов осуществляется к порту 3P, к которому также подключаются все устройства, работающие на основе TTL интерфейса.

Необходимо перевести переключатель режимов работы USB2Dynamixel в крайнее левое положение.

При помощи трехпроводного шлейфа необходимо подключить устройство к порту 3P преобразователя USB2Dynamixel.

При объединении сервоприводов Dynamixel в последовательную сеть каждое из них соединяется с помощью трехпроводных шлейфов последовательно друг с другом.

Преобразователь USB2Dynamixel не передает напряжение питания на подключенные к нему устройства, поэтому для запитывания сетевых устройств необходимо подать питание на последний элемент сети.

Подключение к USB2Dynamixel сервоприводов Dynamixel серии DX/RX

Сервопривода серии DX, RX функционируют на основе интерфейса RS-485, поэтому их подключение необходимо осуществлять к порту 4P преобразователя USB2Dynamixel.

Необходимо перевести переключатель режимов работы USB2Dynamixel в центральное положение.

При помощи четырехпроводного шлейфа необходимо подключить устройство к порту 4P преобразователя USB2Dynamixel.

При объединении сервоприводов Dynamixel в последовательную сеть, каждое из них соединяется с помощью четырехпроводных шлейфов последовательно друг с другом.

Преобразователь USB2Dynamixel не передает напряжение питания на подключенные к нему устройства, поэтому для запитывания сетевых устройств необходимо подать питание на последний элемент сети.

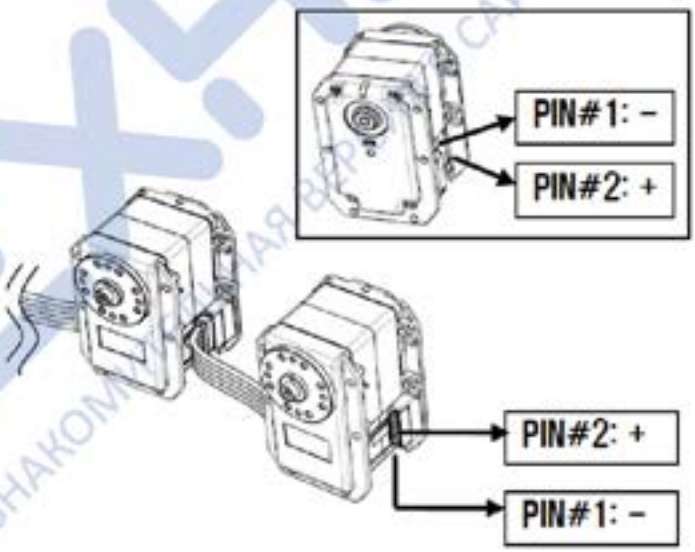
Подача напряжения питания на сервопривода, подключенные к USB2Dynamixel

Преобразователь USB2Dynamixel не осуществляет подачу питания на сервопривода, поскольку на него самого подается питание от USB порта компьютера. Поэтому необходимо осуществлять подачу напряжения питания на сервопривода из вне.

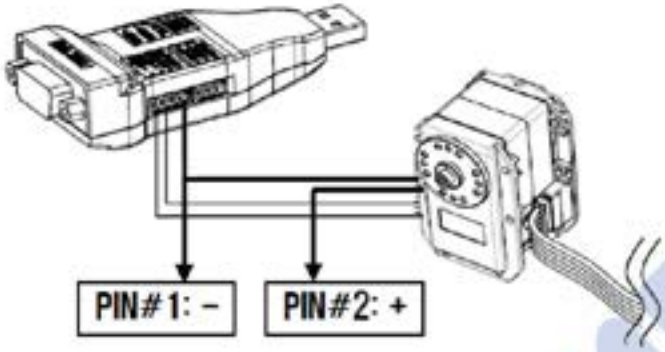
В предыдущих разделах было отмечено, что сервопривода бывают двух типов с соединительными разъемами типа 3P и 4P. В каждом из таких разъемов специально зарезервировано по 2 контакта для подачи питания. Контакт №1 представляет собой «землю» GND, а контакт №2 предназначен для подачи напряжения питания.

4 Pin Cable			3 Pin Cable		
Pin No.	Signal	Pin Figure	Pin No.	Signal	Pin Figure
1	GND		1	GND	
2	NOT Connected		2	NOT Connected	
3	DATA + (RS-485)		3	DATA (TTL)	
4	DATA - (RS-485)				

Для того чтобы подать напряжение питания на сеть сервоприводов, необходимо запитать контакты №1 и №2 крайнего сервопривода сети, у которого есть один свободный порт.



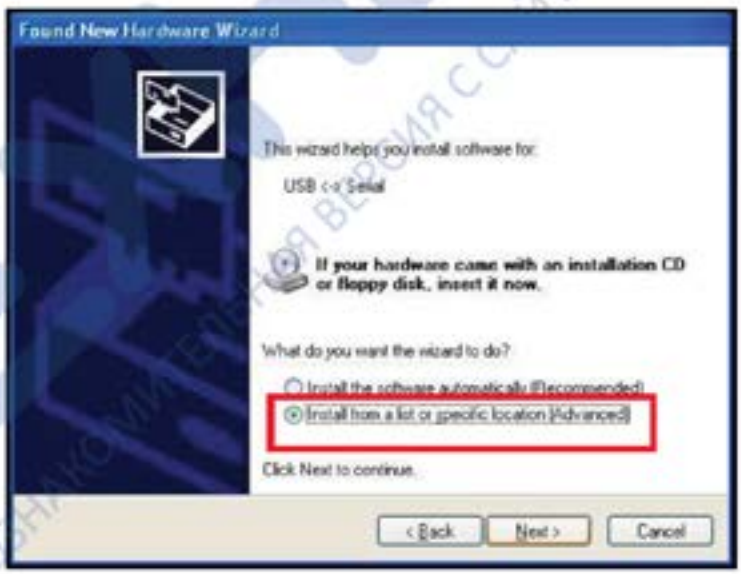
В случае если питание невозможно подать на один из сервоприводов, это можно сделать с помощью преобразователя USB2Dynamixel согласно схеме приведенной ниже.



Установка драйверов преобразователя USB2 Dynamixel

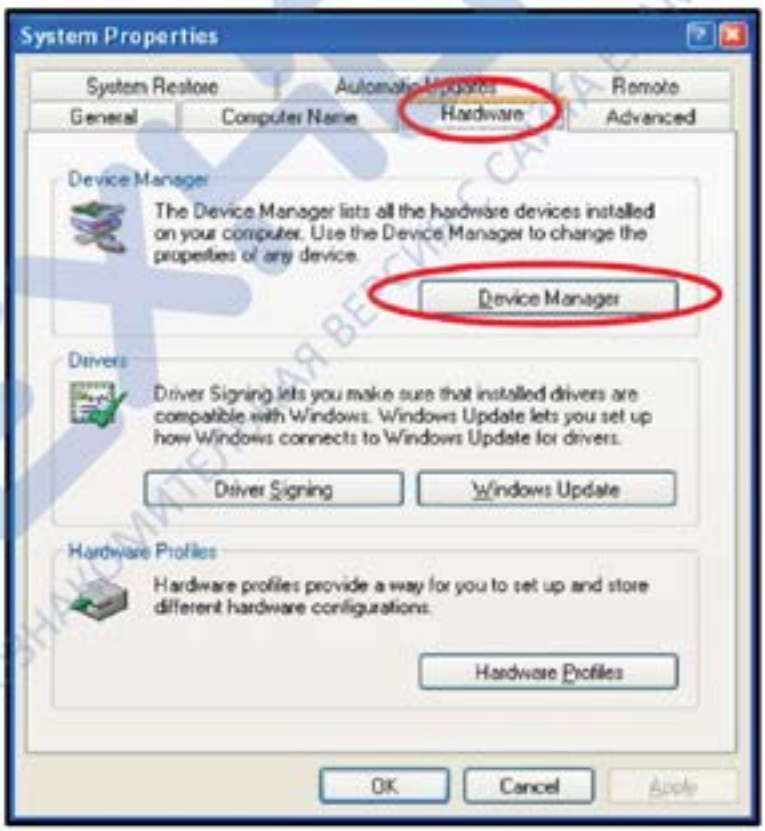
Для работы с преобразователем USB2Dynamixel необходимо, чтобы на персональном компьютере были установлены его драйвера. Драйвера устройства можно скачать с сайта производителя, также установка драйверов осуществляется автоматически при установке программного обеспечения RoboPlus.

Для установки драйвера необходимо подключить преобразователь USB2Dynamixel к USB порту персонального компьютера. В появившемся окне мастера установки оборудования необходимо выбрать одну из указанных строк.

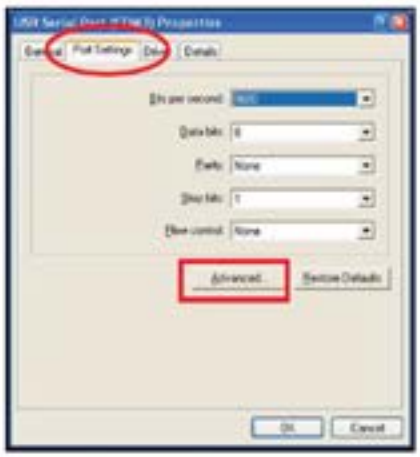


При установке драйвера необходимо указать путь к его файлам, которые можно загрузить с сайта производителя. По окончании процесса установки на экране появится соответствующее окно.

Для дальнейшей работы с преобразователем USB2Dynamixel необходимо произвести его настройку. Настройки производятся в разделе «Оборудование», находящимся в разделе «Система» папки «Панель управления».



Во вкладке «Port (COM&LPT)» необходимо выбрать установленный USB Serial Port и осуществить его настройку.



После установки драйверов и выполнения настроек параметров последовательного порта преобразователь USB2Dynamixel полностью готов к работе.

Комплект беспроводных модулей ZigBee

ЭКЗАМЕН ТЕХНОЛАБ

Комплект беспроводных модулей **ZigBee**

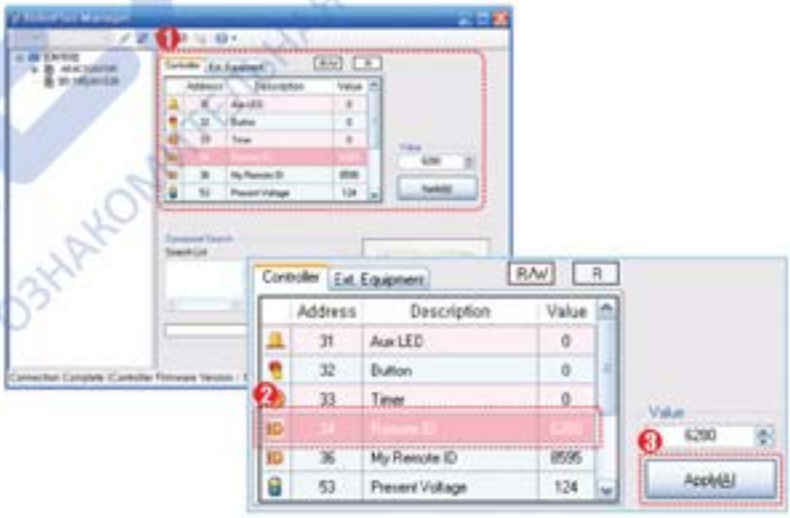
Комплект беспроводных модулей ZigBee

Комплект беспроводных модулей ZigBee состоит из трех устройств – два модуля ZIG-100/ZIG-110A представляют собой пару устройств, осуществляющих передачу данных друг другу, третий модуль ZIG2Serial предназначен для передачи данных на персональный компьютер.

Модули ZIG-100/ZIG-110A

Данные модули предназначены для подключения к устройствам, между которыми необходимо организовать канал беспроводной передачи данных. С сопряженными устройствами модули ZIG-100/ZIG-110A взаимодействуют с помощью UART интерфейса.

Для того чтобы между модулями установился канал связи, необходимо осуществить их настройку в программной среде RoboPlus. В окне программы RoboPlus Manager каждый из модулей определяется с соответствующим адресом и значением. Для обеспечения канала связи между устройствами необходимо присвоить каждому из устройств одинаковое значение.



Подобная настройка соединений одинакова как для модулей ZIG-100, так и для модулей ZIG-110A. Если же одним из устройств в цепочке передачи данных является персональный компьютер, то к нему ZigBee модули подключаются с помощью устройства ZIG2Serial. В этом случае необходимо осуществить такую же настройку соединения, как и в рассматриваемом ранее случае.

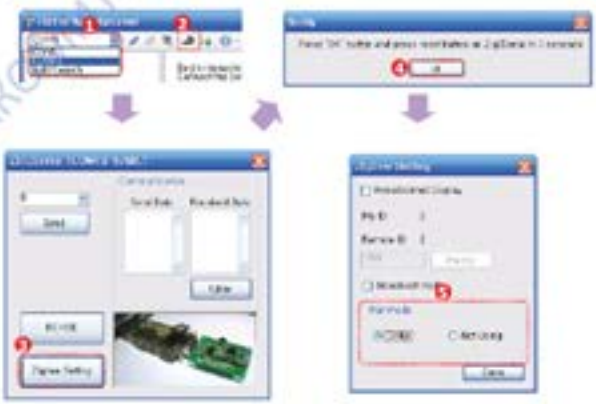
Передача данных с помощью ZigBee радиоканала может осуществляться по трем различным схемам: «1:1» - передача информации между парой модулей; «1:N» – передача информации от одного модуля группе модулей; «N:N» – передача информации между группой модулей.

Передача данных по схеме «1:1»

Данные передаются между двумя устройствами, настроенными на работу в паре. При подаче питания на каждый из модулей они переходят в рабочий режим, о чем свидетельствует включение светодиодов. После этого между ними можно пересылать данные.

Передача данных по схеме «1: N»

Модуль ZIG-100 может быть настроен для работы в режиме ожидания. В этом случае модуль игнорирует попытки передачи информации до тех пор, пока он не будет включен. При включении модуля он устанавливает взаимосвязь с первым обнаруженным модулем.



Передача данных по схеме «N: N»

Модули ZigBee могут быть настроены для работы в режиме широковещательной передачи. В этом случае одна и та же информация передается между всеми модулями.

Для того чтобы перевести модуль ZIG-110A в широковещательный режим, необходимо установить его текущее значение равным 4. При переводе в широковещательный режим модуля ZIG-100 его статус индицируется с помощью выходов №7, 8 его порта.

Схема подключения модуля ZIG-100

1) GND – линия «земля» модуля ZigBee (0 В).
2) VCC – линия питания модуля ZigBee (2,7 – 3,6 В).
3) LED – линия отображающая статус текущего соединения.
4) RESET – линия сброса настроек ZigBee модуля.
5) RXD – линия принимаемых данных ZigBee модуля.
6) TXD – линия передаваемых данных ZigBee модуля.
7) CHANNEL_SEL1 – линия статуса широковещательного режима.
8) CHANNEL_SEL2 – линия статуса широковещательного режима.

Схема подключения модуля ZIG-110A.

1) RXD – линия принимаемых данных ZigBee модуля.

2) TXD – линия передаваемых данных ZigBee модуля.

3) VCC – линия питания (2,7 – 3,6 В).

4) GND – линия «земля» (0 В).

Модуль ZIG2Serial

Данный модуль предназначен для подключения ZigBee радиомодулей к персональному компьютеру и служит для преобразования интерфейса в виртуальный COM-порт, идентифицируемый операционной системой ПК.

Подключение модуля к компьютеру осуществляется с помощью преобразователя USB2Dynamixel.

Pin No.	Signal
1	NC
2	TXD [RS-232]
3	RXD [RS-232]
4	NC
5	GND
6	NC
7	NC
8	NC
9	USB power [5V]

Модуль ZIG2Serial подключается к последовательному порту преобразователя, а тот в свою очередь подключается к USB-порту компьютера. При таком подключении обеспечивается преобразование интерфейсов с помощью USB2Dynamixel, так и подача питания на модуль ZIG2Serial.

Модуль ZIG2Serial содержит разъем для подключения модуля беспроводной передачи данных ZIG-100, с помощью которого на модуль подается питание и передаются данные. При подключении модулей важно соблюдать их взаимную ориентацию, так чтобы при установке ZIG-100 его антенна была направлена в противоположную сторону от разъема последовательного порта.

Краткое описание модуля ZIG2Serial

Масса: 11, 12 г

Габариты: 56x34x12 мм

Напряжение питания: 4,5 – 5,5 В

Базовая скорость передачи данных: 57600 бит/с.

1) Communication Status Display LED

   Светодиод, отображающий текущий статус передачи данных, включен во время передачи данных.

2) ZIG-100 Connector

   Порт для подключения модуля ZIG-100.

3) ZIG-100 Status Display LED

   Светодиод, отображающий текущий статус модуля ZIG-100. Выключен, в случае если модуль не подключен. Включен, в случае если модуль присоединен и готов к передаче данных. Мигает, в случае если модуль не обнаружен.

4) Reset Switch

   Переключатель для принудительной перезагрузки модуля.

5) Power Connector

   Порт для подачи внешнего питания на модуль. Для работы модуля требуется напряжение питания уровня 5 В, которые могут быть также получены с помощью преобразователя USB2Dynamixel от USB порта компьютера.

6) RS232 Connector

   Порт для подключения к COM-порту компьютера по интерфейсу RS-232.

Программируемый контроллер СМ-530

ЭКЗАМЕН ТЕХНОЛАБ

Программируемый контроллер **СМ-530**

Программируемый контроллер СМ-530

Введение

Программируемый контроллер СМ-530 представляет собой универсальный блок управления роботами серии Bioloid. Помимо управляющего микропроцессора данный блок содержит панель с кнопками управления, набор портов для соединения со внешними устройствами, а также ряд сенсорных устройств для диагностики рабочего состояния.

Масса СМ-530: 54 г

Микропроцессор: STM32F103RE

Рабочее напряжение: 6 – 15 В

Рекомендованное напряжение: 11,1 В

Потребляемый ток в режиме программирования: 50 мА

Максимальный входной ток: 300 мА

Общий максимальный ток: 10 А

Диапазон рабочих температур: -5 ... +70 градусов шкалы Цельсия

Наличие кнопок управления: 5 шт. + 1 шт. (Reset)

Наличие встроенного микрофона: 1 шт.

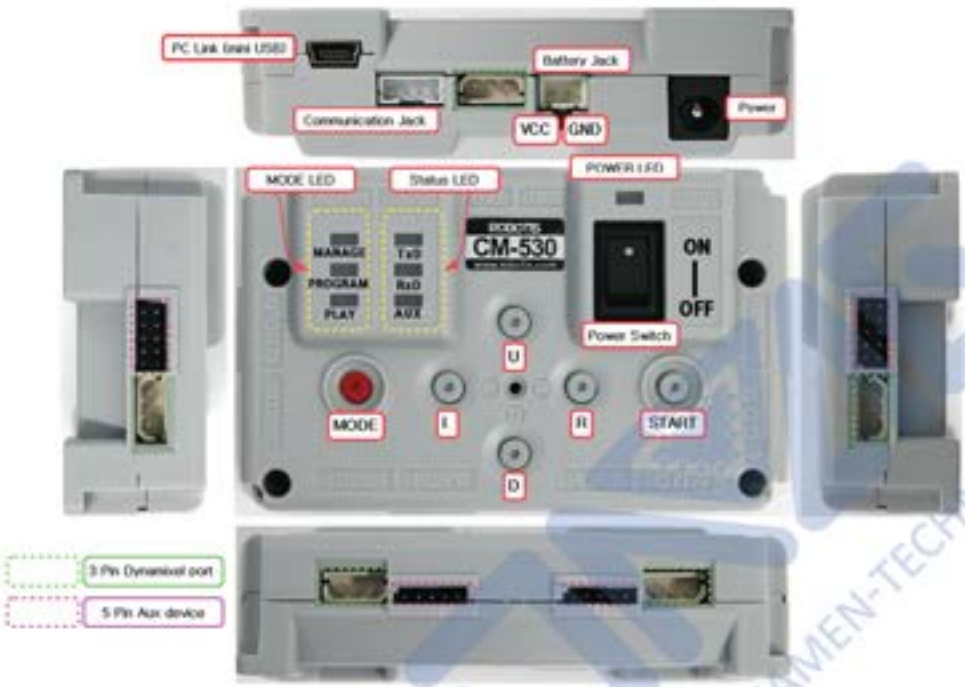
Наличие датчика напряжения: 1 шт.

Наличие портов типа 5 pin: 6 шт.

Наличие портов типа 3 pin: 5 шт.

Наличие светодиодного дисплея для отображения справочной информации.

Спецификация портов программируемого контроллера СМ-530



PC Link (mini USB)

Используется для подключения к USB порту компьютера. Данный порт содержит в себе преобразователь интерфейсов, поэтому распознается компьютером как виртуальный COM-порт. Порт предназначен для программирования контроллера СМ-530.

Communication Device Connection Jack

Порт предназначен для связи со внешними устройствами по последовательному интерфейсу.

Battery Jack

Порт предназначен для подключения внешнего источника питания.

Power Jack

Порт предназначен для подключения питания от сети или для подзарядки робота с помощью устройства SMPS.

Power LED

Светодиод для индикации состояния программируемого контроллера «Вкл/Выкл».

Power Switch

Переключатель «Вкл/Выкл» для подачи питания на робота.

MODE Button

Кнопка управления рабочим режимом программируемого контроллера CM-530. Может применяться для переключения между группами запрограммированных режимов работы – режим дистанционного или автоматического управления, демонстрационный режим или режим ручного управления. Каждый из режимов может быть запрограммирован пользователем в отдельности.

START Button

Кнопка запуска робота. Предназначена для запуска работы робота по загруженной в память программе и в установленном режиме (MODE).

U / L / D / R Button

Кнопки управления роботом в ручном режиме. Функции кнопок могут быть заданы в программе. Кнопки могут применяться как для задания рабочего режима робота, так и для его непосредственного управления в процессе работы.

AX/MX Series Bus Port

Порт типа 3P для подключения сервоприводов с целью управления ими по последовательному интерфейсу. Порт также может применяться для подключения любых других устройств, поддерживающих данный интерфейс и протокол.

Peripheral Devices Connection Port

Порт типа 5P для подключения внешних устройств, таких как датчики, кнопки управления и т.п.

Mode Display LED

Панель светодиодов для индикации режима работы программируемого контроллера CM-530. Панель содержит три светодиода:

Manage – светодиод, отображающий заданный в данный момент режим работы контроллера CM-530. Загорается во время передачи команд устройствам и сервоприводам Dynamixel и при отладке программ контроллера в среде программирования RoboPlus.

Program – светодиод, отображающий режим редактирования программы робота в редакторе RoboPlus Motion.

Play – светодиод, отображающий режим работы программируемого контроллера по заданной программе.

Status Display LED

Панель светодиодов для отображения текущего статуса программируемого контроллера СМ-530. Панель содержит три светодиода:

TxD – отображение процесса передачи данных.

RxD – отображение процесса приема данных.

AUX – светодиод, статус которого можно задавать из пользовательской программы.

Описание порта 5P программируемого контроллера СМ-530

Данный порт предназначен для подключения к программируемому контроллеру СМ-530 внешних устройств. В качестве подключаемых устройств могут применяться различные аналоговые датчики и устройства, управляемые дискретными сигналами.

Данный порт содержит пять рабочих линии, среди которых – две питающие линии, две линии отводятся под дискретные сигналы и одна под вход АЦП.

Технические характеристики порта 5P следующие:

1) OUT 1 : 3.3 V – дискретный выход с максимальным током 0,3 А.
2) VCC (5 V) – линия питания внешнего устройства.
3) ADC – вход аналого-цифрового преобразователя.
4) GND – линия «земля».
5) OUT 2 : 3.3 V – дискретный выход с максимальным током 0,3 А.

Описание порта 3P программируемого контроллера СМ-530

Данный порт предназначен для подключения сервоприводов Dynamixel по последовательному TTL интерфейсу. Также помимо сервоприводов данный порт поддерживает подключение других устройств, работающих на базе протокола Dynamixel.

Данный порт содержит три рабочие линии, среди которых: две линии питающие и одна линия применяется для цифровой передачи данных.

Технические характеристики порта 3P следующие:

1) GND – линия «земля».
2) VDD – линия питания.
3) DATA – канал передачи данных.

Описание порта Communication Device Connection Jack контроллера СМ-530

Данный порт предназначен для подключения внешних устройств, взаимодействующих с программируемым контроллером СМ-530 как с виртуальным последовательным портом.

Технические характеристики порта следующие:

1) GND – линия «земля» 0 В.
2) VDD – линия питания 5 В.
3) RXD – линия приема данных.
4) TXD – линия передачи данных.

Подключение питания к контроллеру СМ-530

Подключение сетевого питания к программируемому контроллеру СМ-530

осуществляется с помощью порта «Power Jack». К данному порту подключается сетевой адаптер SMPS, преобразующий переменное сетевое напряжение в постоянное напряжение, питающее контроллер.

При подаче питания и включении контроллера должен загореться светодиод «Power LED».

Подача питания осуществляется с помощью переключателя «Power Switch» при переключении из режима OFF в режим ON .

Для запуска программы контроллера необходимо перевести CM-530 в режим «Play» с помощью кнопки MODE Button. Для запуска контроллера в работу необходимо нажать кнопку START Button.

Для завершения работы контроллера необходимо отключить питание с помощью переключателя «Power Switch».

Подключение контроллера CM-530 к компьютеру

Подключение программируемого контроллера CM-530 к компьютеру осуществляется с помощью кабеля USB-mini USB через порт «PC Link (mini USB)».

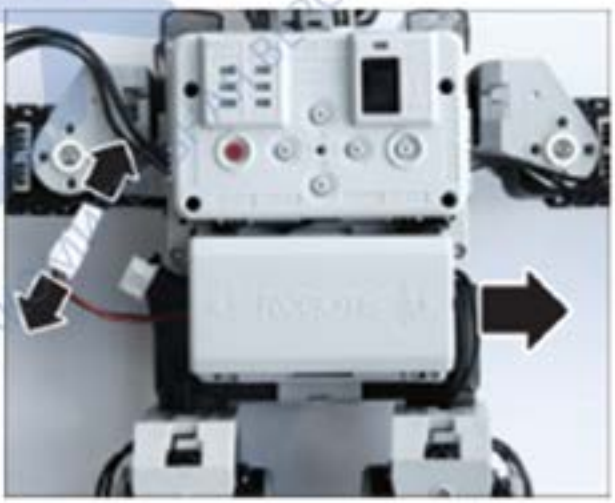
Подключение модулей беспроводной связи ZIGBee

Подключение модулей ZigBee или Bluetooth к программируемому контроллеру СМ-530 осуществляется с помощью порта «Communication Device Connection Jack».

Зарядка батареи контроллера СМ-530

Контроллер СМ-530 и все контроллеры младшей серии, входящие в состав роботов Bioloid, содержат датчик уровня напряжения питающей батареи. В случае падения уровня заряда батареи контроллер воспроизводит предупредительный сигнал, что означает необходимость подзарядки аккумуляторной батареи или смены комплекта батареек AA на новый.

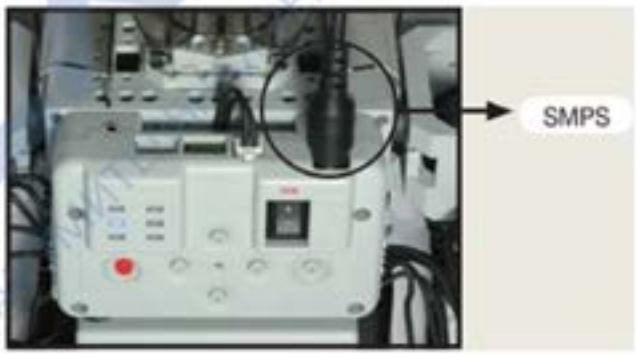
Для того, чтобы подключить батарею к зарядному устройству ее нужно в первую очередь отключить от робота.



Аккумуляторную батарею необходимо подключить к зарядному устройству, а его с помощью сетевого адаптера можно подключить к сети для подачи питания.

При включении зарядного устройства в сеть должна загореться красная лампочка зарядного устройства. В случае окончания процесса зарядки должна загореться зеленая лампочка.

В случае необходимости робот серии Bioloid может работать, подключенным к сети. В этом случае необходимо присоединить сетевой адаптер к контроллеру СМ-531 робота и осуществить запуск.



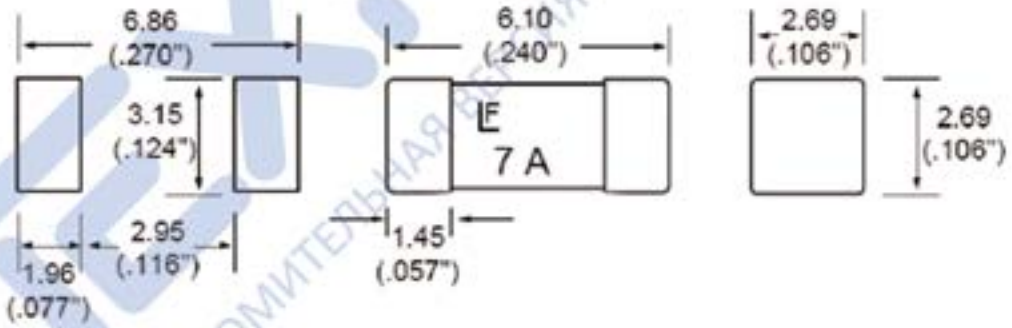
Для поддержания роботов Bioloid в работоспособном состоянии максимально длительное время необходимо следить за зарядом аккумуляторной батареи и не допускать ее полного разряда. Всегда, когда контроллер СМ-530 воспроизводит предупреждающий сигнал, необходимо производить подзарядку аккумуляторной батареи.

Для защиты электрических цепей программируемого контроллера СМ-530 и других комплектующих роботов Bioloid применяются плавкие предохранители.

Плавкий предохранитель защищает внутренние цепи робота Bioloid от превышения допустимого значения потребляемого тока, как вследствие перегрузок, так и в случае возникновения короткого замыкания.

Одной из причин возникновения перегрузок по току является «заклинивание» сервоприводов во время работы, торможение робота во время движения, например при столкновении с объектами и др.

Короткое замыкание в цепях роботов Bioloid может возникнуть при замыкании между линиями 3P и 4P шлейфов, соединяющих различные устройства робота между собой. Это может возникнуть из-за перетирания изоляции линий вследствие многократной эксплуатации робота.



В случае выхода из строя плавкого предохранителя его всегда можно заменить на резервный.

Замена плавкого предохранителя должна осуществляться при выключенном питании программируемого контроллера СМ-530 и полностью отключенного от сети. Помимо этого необходимо отключить от контроллера аккумуляторную батарею и все внешние устройства робота.

После выполнения всех вышеописанных подготовительных мероприятий можно с помощью пинцета осуществить замену плавкого предохранителя.

Учебно-методическое издание

Ермишин Константин Владимирович
Каргин Дмитрий Николаевич
Нагорный Алексей Александрович
Панфилов Алексей Олегович

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

ОБРАЗОВАТЕЛЬНЫЙ
РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(ПРОФЕССИОНАЛЬНЫЙ УРОВЕНЬ)
от 14 ЛЕТ

Издательство ***«ЭКЗАМЕН»***
«ЭКЗАМЕН-ТЕХНОЛАБ»

Гигиенический сертификат
№ РОСС RU. АЕ51. Н 16466 от 25.03.2013 г.

Главный редактор *Л. Д. Лаппо*
Корректоры *Н. С. Садовникова, С. С. Гаврилова, Е. В. Григорьева*
Дизайн обложки
и компьютерная верстка *А. А. Винокуров*

107045, Москва, Луков пер., д. 8.
E-mail: по общим вопросам: robo@examen-technolab.ru;
www.examen-technolab.ru
по вопросам реализации: sale@examen-technolab.ru
тел./факс +7 (495) 641-00-19 (многоканальный)

ТЕХНОЛАБ
ОБРАЗОВАТЕЛЬНЫЙ РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ
(ПРОФЕССИОНАЛЬНЫЙ УРОВЕНЬ)

www.examen-technolab.ru

Артикул ТР-0541-МП

ISBN 978-5-377-07625-4

14+
ЛЕТ