ТЕХНОЛАБ

ОБРАЗОВАТЕЛЬНЫЙ РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(ИССЛЕДОВАТЕЛЬСКИЙ УРОВЕНЬ)

РЕСУРСНЫЙ НАБОР

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

(ИССЛЕДОВАТЕЛЬСКИЙ УРОВЕНЬ)

РЕСУРСНЫЙ НАБОР

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

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

ОБРАЗОВАТЕЛЬНЫЙ
РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(ИССЛЕДОВАТЕЛЬСКИЙ УРОВЕНЬ)
РЕСУРСНЫЙ НАБОР
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 (исследовательский уровень) ресурсный набор: от 14 лет / К. В. Ермишин, Д. Н. Каргин, А. А. Нагорный, А. О. Панфилов. — М. : Издательство «Экзамен», 2014. — 80 с.

ISBN 978-5-377-07825-8

Данное пособие предназначено для применения совместно с образовательным робототехническим модулем «Исследовательский уровень» ресурсный набор. 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 «Исследовательский уровень» ресурсный набор 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 «Исследовательский уровень» ресурсный набор.

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. 10. Тираж 1000 экз. Заказ №

ISBN 978-5-377-07825-8

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. Образовательный робототехнический модуль 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. Данное пособие описывает ряд моделей роботов, каждая из которых 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

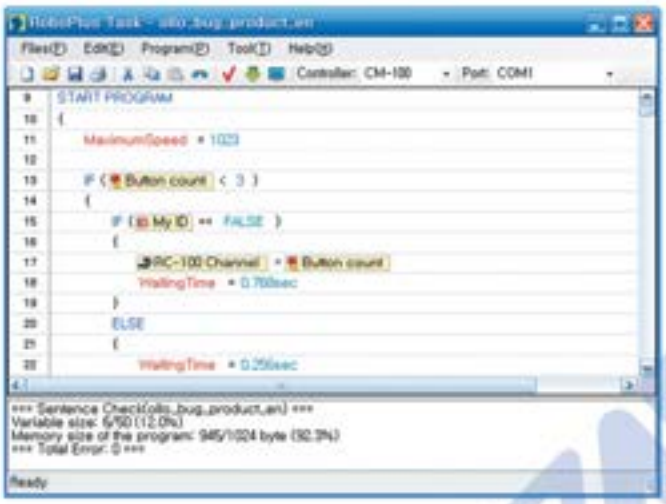
RoboPlus Manager

RoboPlus Motion

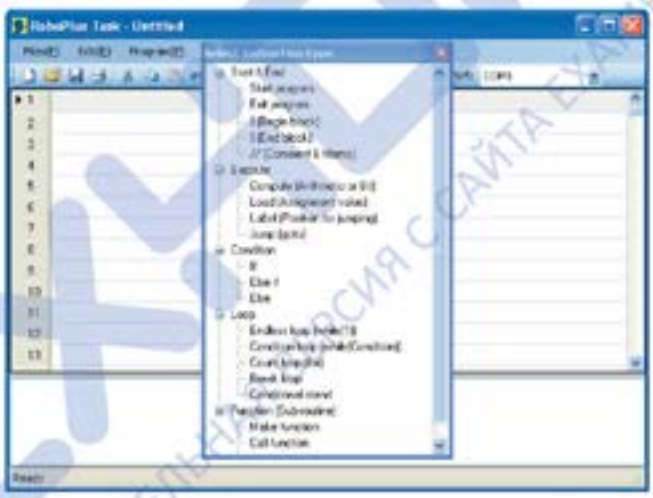
RoboPlus Terminal

Dynamixel Wizard

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

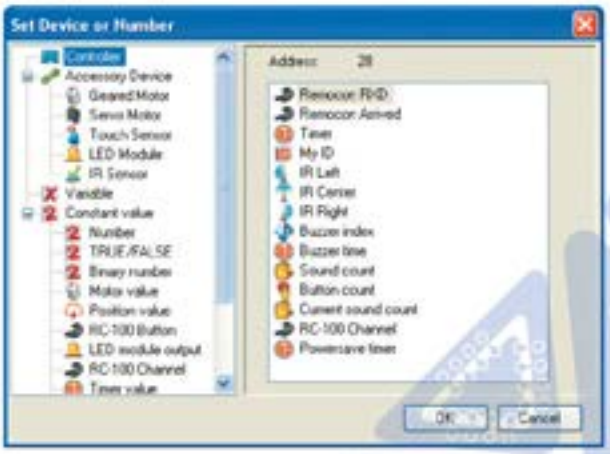


Программирование в RoboPlus Task осуществляется с помощью специализированного языка, подобного языку программирования С. Для удобства пользователя в RoboPlus в виде графических блоков реализованы базовые возможности набора, такие так: таймеры, блоки обработки данных с датчиков, блоки передачи данных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



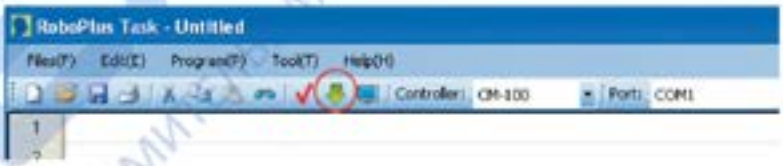
В языке среды RoboPlus все служебные слова, программные блоки, команды и комментарии располагаются в строках. Для того чтобы активировать строку, по ней нужно нажать дважды. После нажатия появится диалоговое окно, позволяющее выбрать различные команды языка.

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.

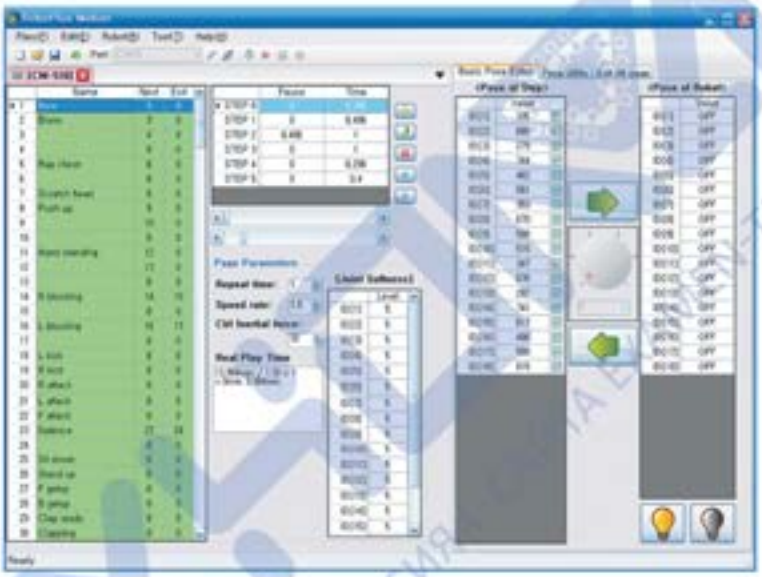
После того как программа определит тип программируемого контроллера, можно произвести компиляцию программы, тем самым проверить ее на наличие ошибок и подготовить к загрузке в программируемый контроллер.



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

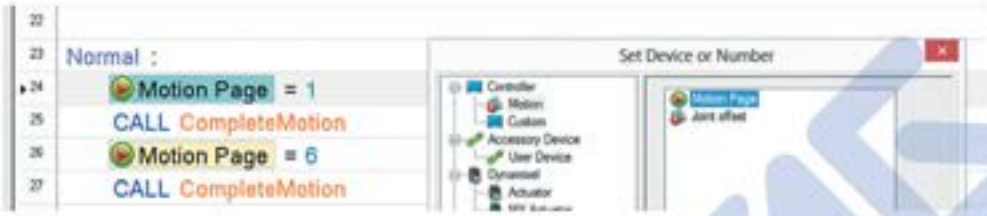
RoboPlus Manager – программа для настройки оборудования, входящего в состав робототехнических конструкторов ROBOTIS. С помощью данной программы RoboPlus обновляет собственные файлы и производит тестирование оборудования, подключенного к компьютеру в данный момент при помощи контроллера или специализированных 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.

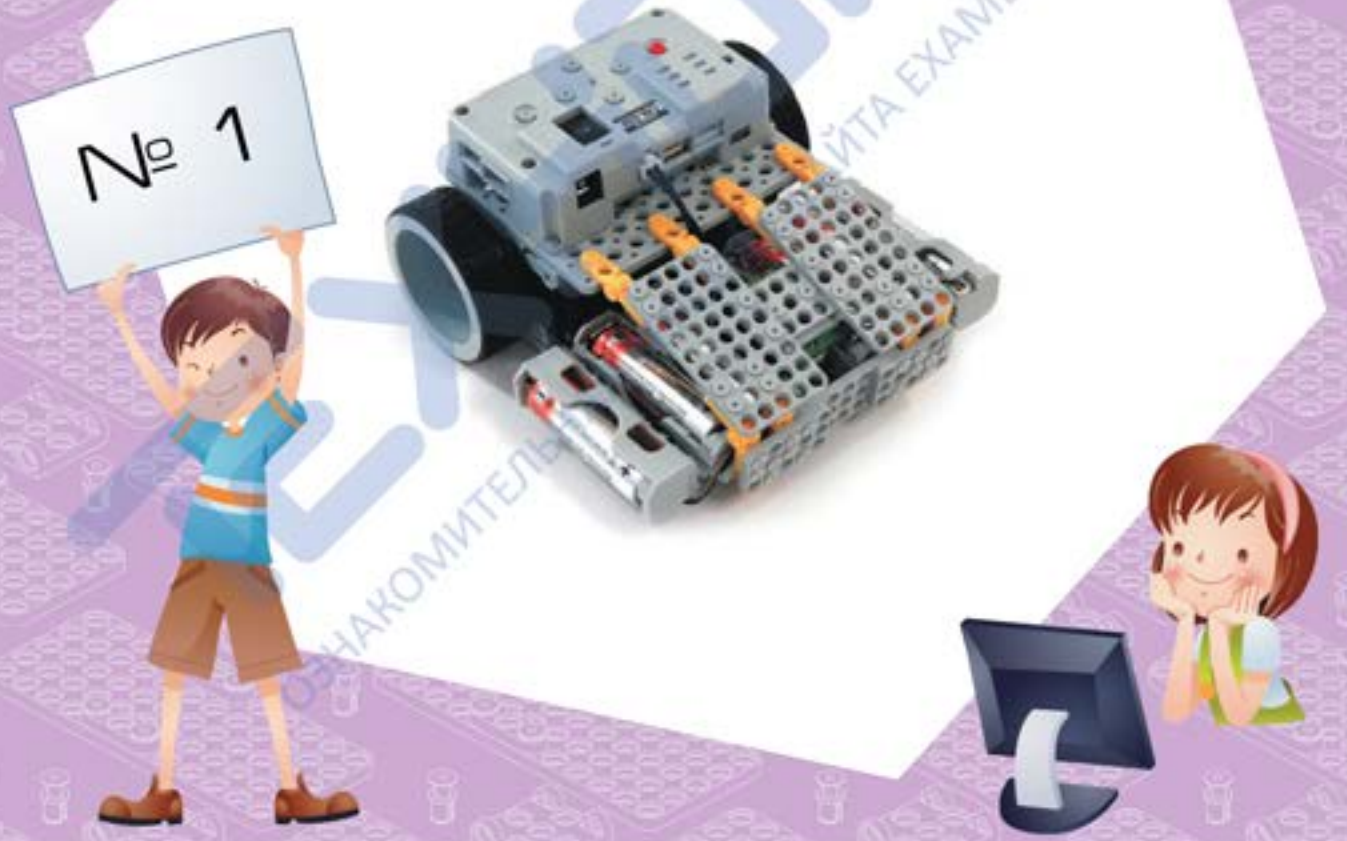
Dynamixel Wizard – программа, предназначенная для настройки и калибровки сервоприводов Dynamixel. С помощью данной программы для каждого из приводов можно задать ограничения скоростей вращения и углов поворота, а также получить код ошибки, препятствующей работе устройства.

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

Основы управления робототехническими системами



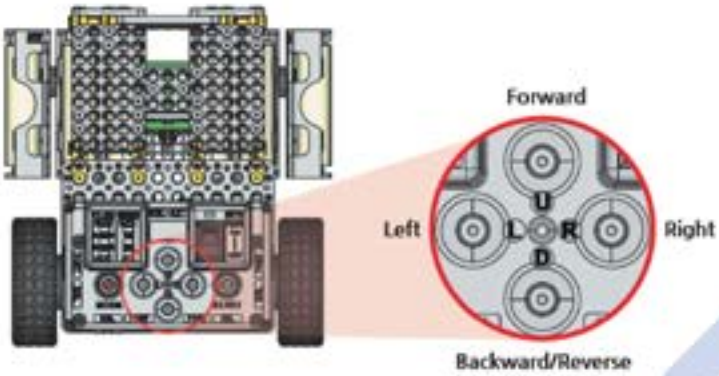
Лабораторная работа № 1
«Основы управления робототехническими системами»

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.

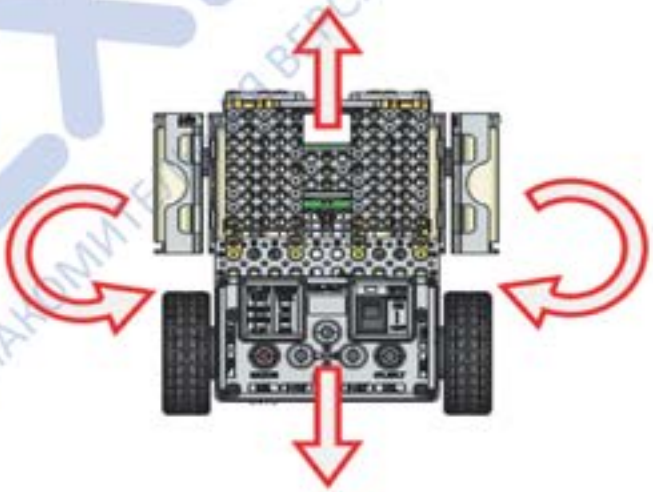


В рамках данной лабораторной работы предлагается сконструировать робота, управляемого с помощью команд, задаваемых кнопками. В качестве устройства, задающего команды, используется панель программируемого контроллера СМ-530. На панели контроллера СМ-530 располагается четыре кнопки управления – U (Up/ 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

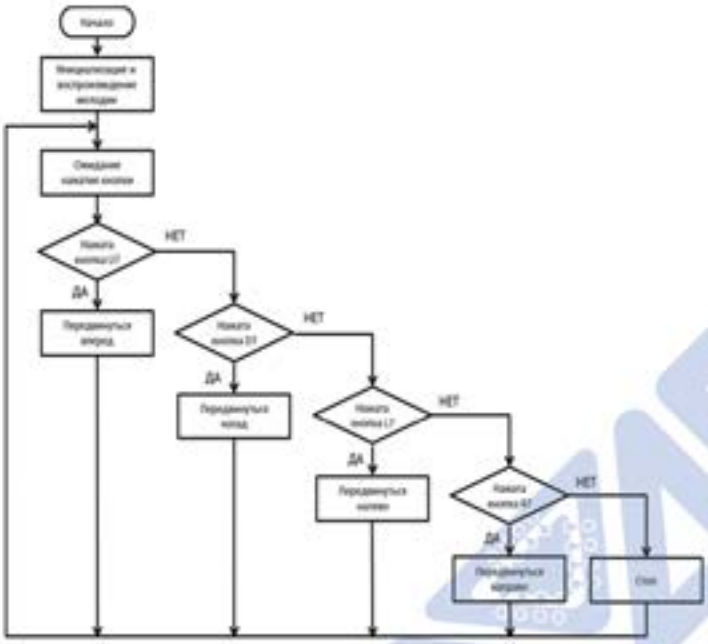
Рассмотрим простейший пример управления мобильным роботом с помощью четырех кнопок на панели контроллера СМ-530. Каждой кнопке контроллера присвоим функцию, выполняющую определенное действие:



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:     }
```

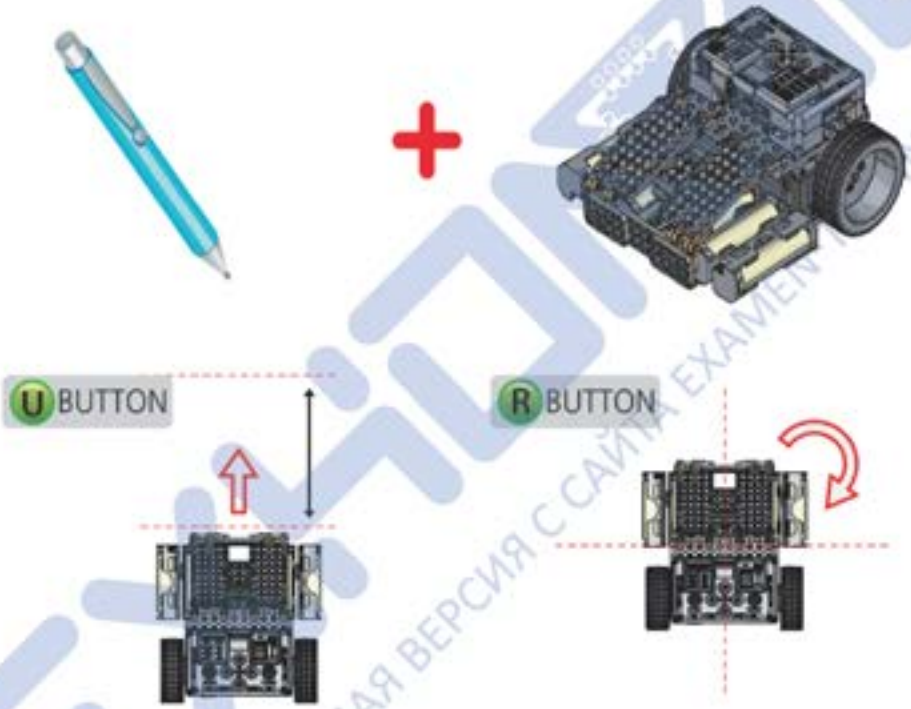
Каждая из функций, задающих движения робота, описывается отдельно. Рассмотрим состав подобной функции на примере функции reverse.

```
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:

```
START PROGRAM
{
    // L/R wheel speed (0 ~ 1023)
    // Modify speed until robot run straightly in "Straightness_check_mode"
    l_wheel_speed  = 400
    r_wheel_speed  = 400
    forward_time  = 1.000sec
    pivot_time  = 0.410sec
    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.

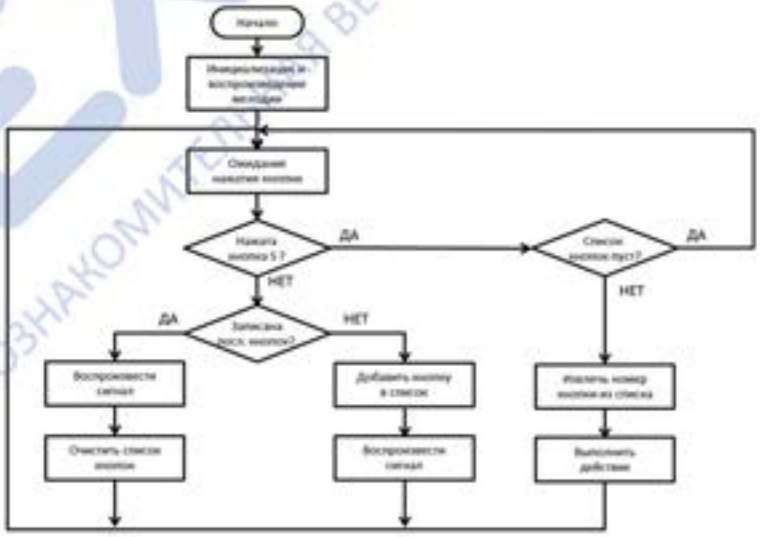
В случае если удастся добиться такой точности движения, мобильный робот будет обладать достаточно высокой маневренностью при собственных габаритах.

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 = 1
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.

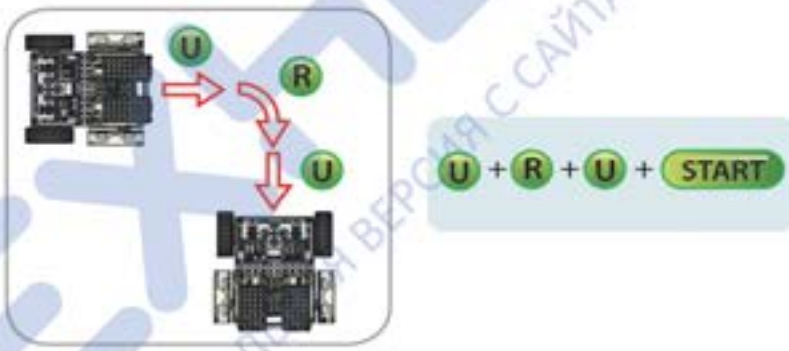
```
153: FUNCTION move
154: {
155:     IF ( move_type == U )
156:     {
157:         CALL standby
158:         CALL forward
159:     }
160:
161:     ELSE IF ( move_type == D )
162:     {
163:         CALL standby
164:         CALL reverse
165:     }
```

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.

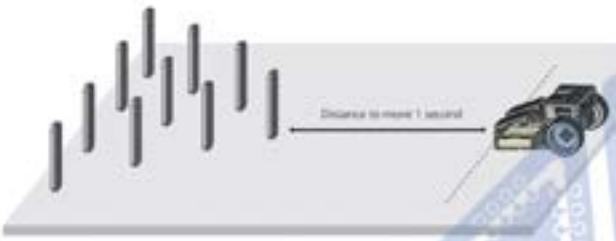
Программирование движений робота осуществляется с помощью кнопок U, D, L, R. В случае если необходимая последовательность команд задана, необходимо нажать на кнопку Start, после чего начнется процесс выполнения программы и робот начнет свое движение.



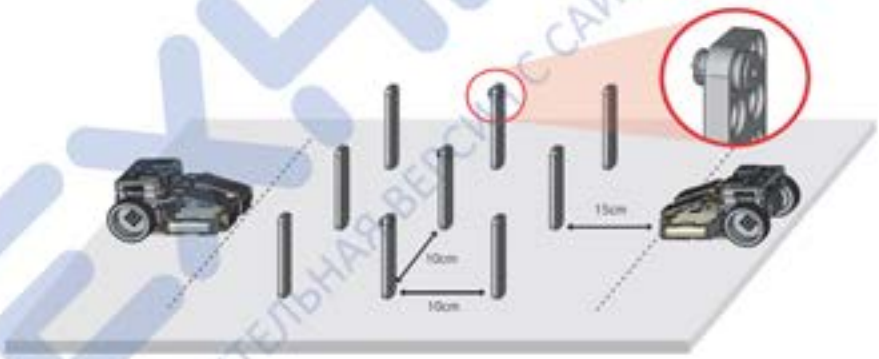
Часть № 3. Подведение итогов

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

Разработка робота, отслеживающего черную линию

Лабораторная работа № 2
«Разработка робота, отслеживающего черную линию»

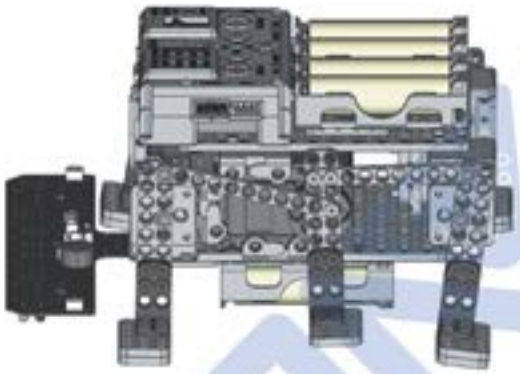
Мобильные роботы применяются для решения различных задач, но основное их предназначение – это перемещение в рабочей зоне. В зависимости от типа рабочей зоны и условий эксплуатации мобильного робота, роботы различаются по типу конструкций шасси. Наиболее часто встречаются колесные роботы, такие роботы 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.

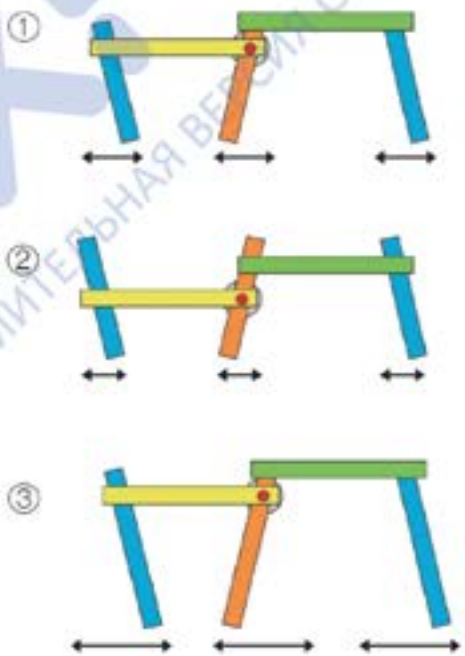
В рамках данной лабораторной работы предлагается рассмотреть конструктивные особенности роботов с шагающей кинематикой и исследовать процесс их движения вдоль линии. Поскольку в состав набора входят два сервомодуля, то существует единственный вариант конструкции шагающего шасси, которое можно разработать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

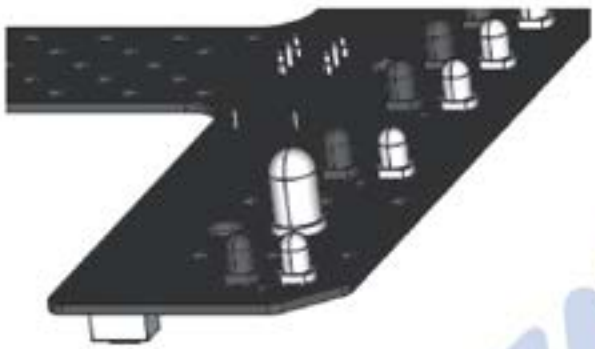
Синхронность движения ног робота-паука достигается за счет механической рычажной передачи. На корпусе робота закреплены ноги робота во вращающихся шарнирах, каждая из них соединена с рычагом, приводящим ее в движение с помощью привода.



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.

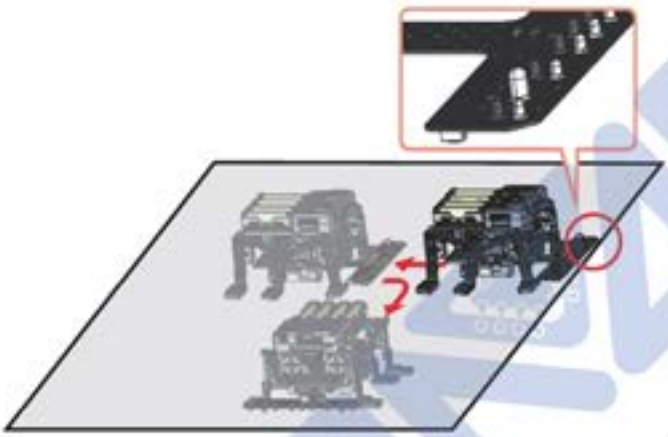


В рамках данной лабораторной работы предлагается разработать модель шестиногого робота-жука, ориентирующегося при движении с помощью ИК-датчиков. В качестве ИК-датчиков используется массив оптронов, состоящий из последовательно установленных светодиодов и фототранзисторов. С помощью подобного устройства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

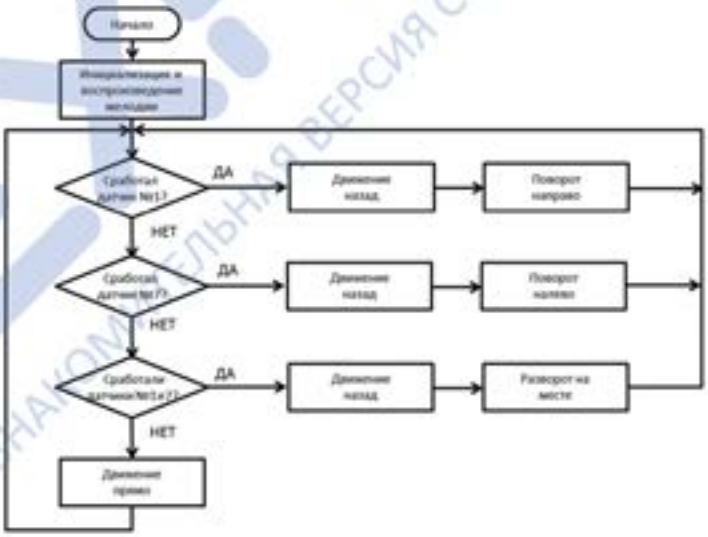
Примечание: для корректной работы ИК-датчиков очень важно осуществлять их калибровку на черный и белый цвет поверхности. Для калибровки массива ИК-датчиков поднимите робота на высоту порядка 5 см от пола и нажмите на кнопку Auto-set. После чего плавно проведите несколько раз над белой и черной поверхностью, 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: }
```

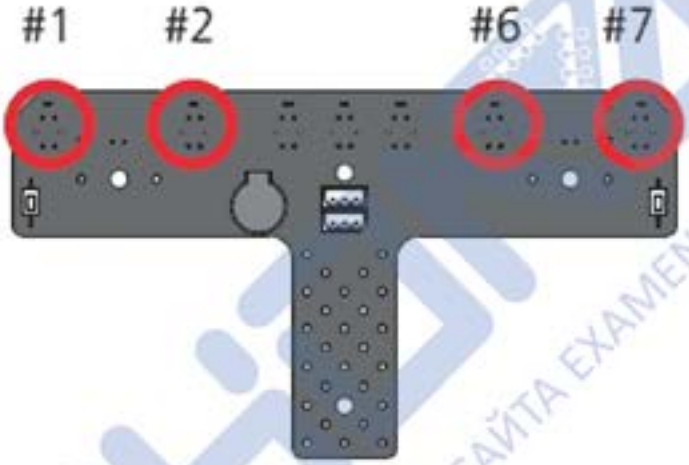
Управление каждым из приводов в отдельности сводится к его вращению с заданной скоростью в течение определенного промежутка времени. В зависимости от скорости вращения и времени работы определяется перемещение робота.

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.

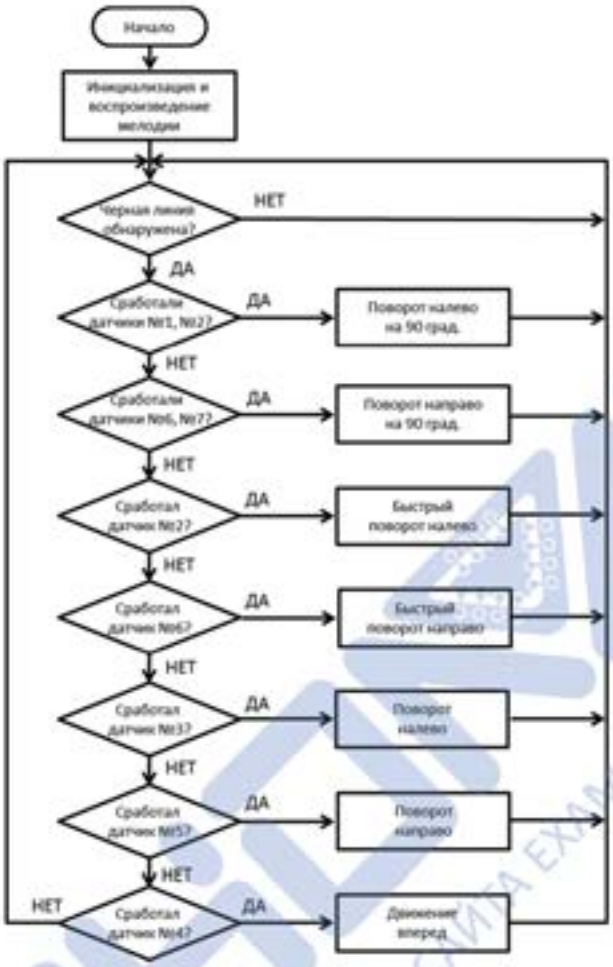
Часть № 2. Использование ИК-датчиков для управления движением робота вдоль линии

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: }
```

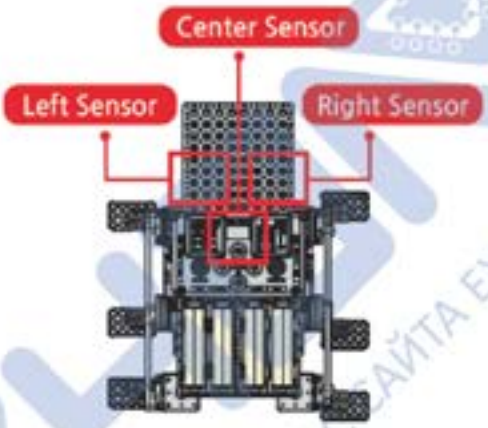
Функции, описывающие движения робота, представляют собой набор инструкций, задающих скорости приводов робота. Это обусловлено тем, что время действия каждой из функций задается алгоритмически за счет перехода от одного условия к другому. Таким образом, робот совершает определенный маневр только во время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

Разработка робота для движения вдоль линии

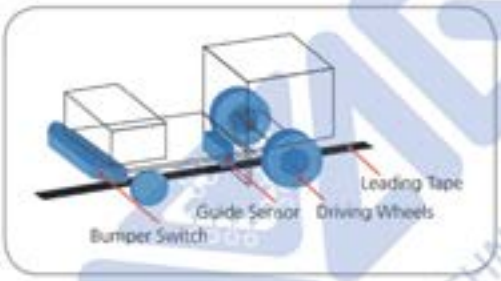
Лабораторная работа № 3
«Разработка робота для движения вдоль линии»

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.

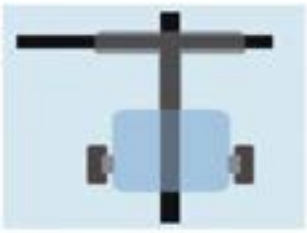
Например, одной из достаточно жестко формализованных задач может быть перемещение объектов внутри производственного помещения. Как правило, при перевозке грузов на складах или в производственных цехах роботы преодолевают один и тот же маршрут постоянно. Соответственно данный маршрут заранее известен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



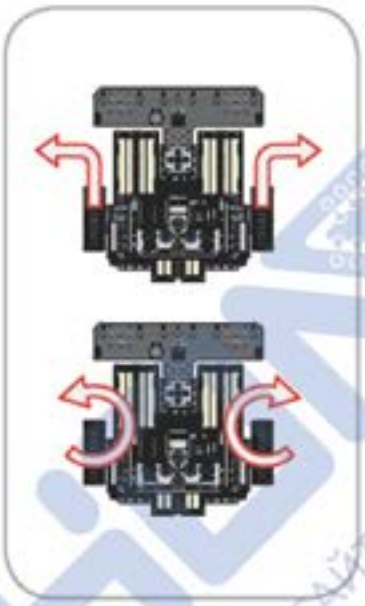
Мобильные роботы, передвигающиеся в цехах вдоль линии, подобно роботам из предыдущих лабораторных работ оснащаются различными сенсорными устройствами для восприятия окружающей обстановки: ИК-датчиками, камерами, датчиками безопасности и т.п. Но в отличие от рассматриваемых ранее роботов, реальные роботы 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.

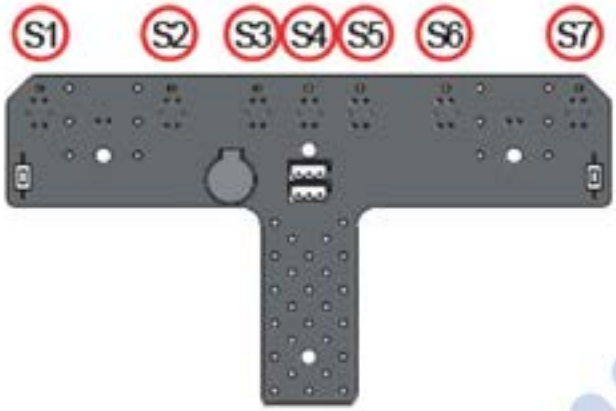


Становится очевидным, что методы движения вдоль линии, представляющей собой замкнутую траекторию, не совсем приемлемы в подобном случае. С примерами различных алгоритмов движения вдоль линии можно ознакомиться в предыдущих работах, но сразу же можно сделать вывод о том, что ни один из них не учитывает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



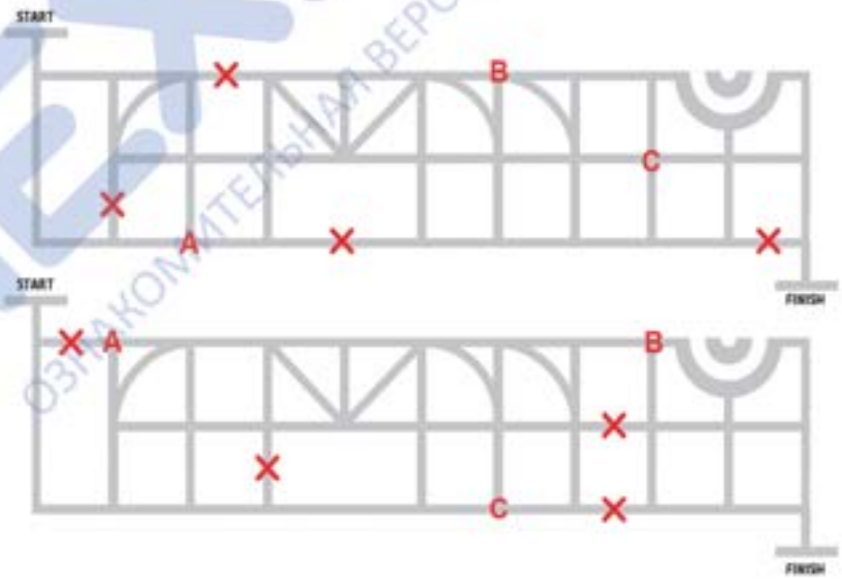
Для того распознавания пересечений направляющих линий может применяться множество различных способов, например в подобных целях достаточно часто применяются камеры. Но обработка изображений требует большой производительности бортового компьютера робота, поэтому такие решения не всегда применимы. В данной 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.

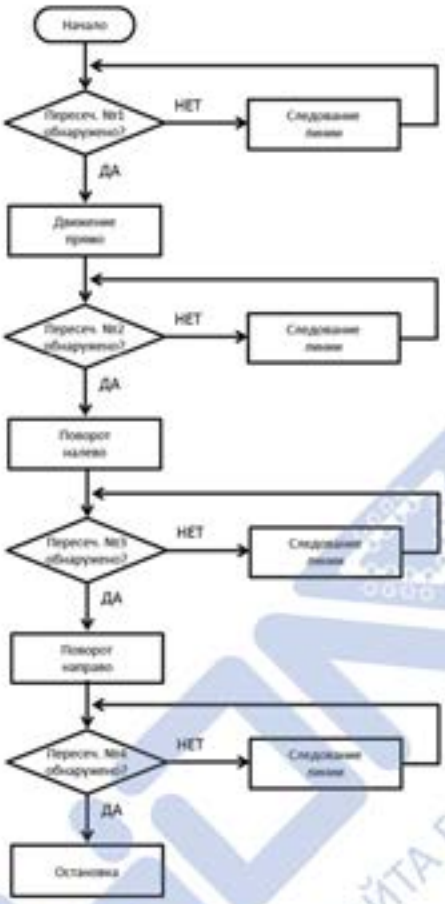
Часть № 1. Выполнение маневров вблизи перекрестков

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 l_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.

```
165: FUNCTION l_node_l_turn
166: {
167:     CALL l_node_forward
168:     CALL pivot_left
169: }
```

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.

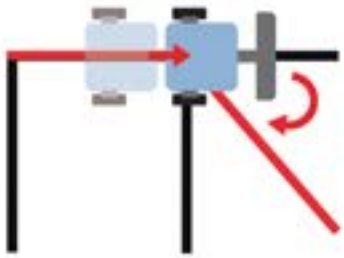
Для повышения точности работы программы в функции l_node_detect реализован механизм защиты от ложных срабатываний. Одно и то же условие проверяется дважды после программируемой задержки.

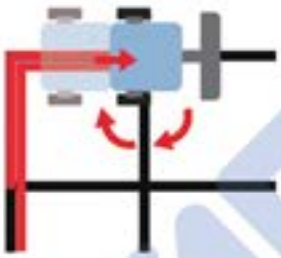
```
322:        // If situadon is the same after "node_judging_time", robot considers that it detects a left node
323:        High-resolution Timer = node_judging_time
324:        CALL precise_timer_standby
325:
326:        CALL detect_black
327:        IF ( black_1 > 0 && black_2 > 0 && black_3 > 0 )
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.

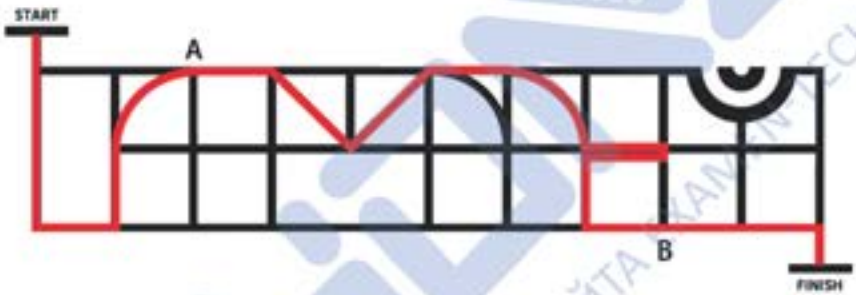
Часть № 2. Выполнение сложных маневров

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 l_node_forward
50:     CALL l_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.

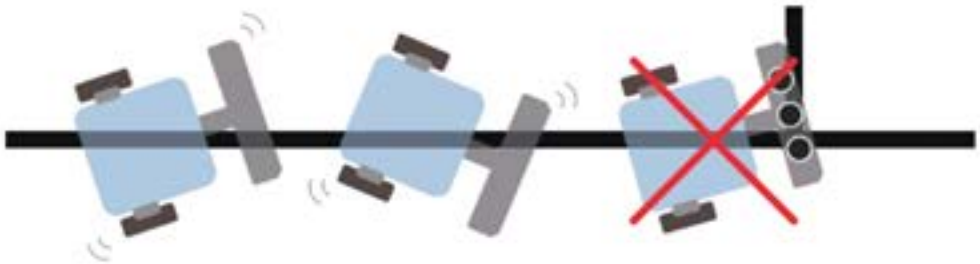
В случае же если робот отклонился от линии достаточно сильно, необходимо скорректировать его положение максимально быстро, поэтому скорость его движения увеличивается.

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.

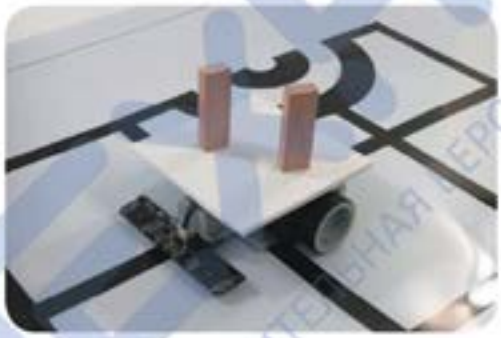


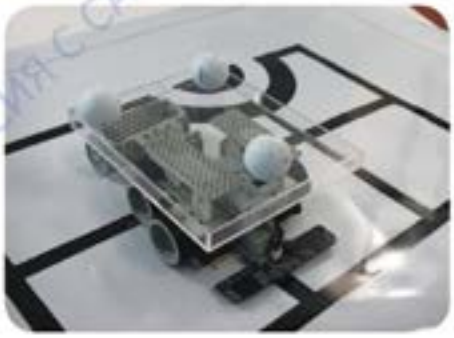
Соблюдение скоростного режима – это не просто требование безопасности движения, это в первую очередь требование, позволяющее минимизировать ошибки работы управляющей программы робота. Стоит помнить, что скорость движения робота существенным образом влияет на качество распознавания узловых точек маршрута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

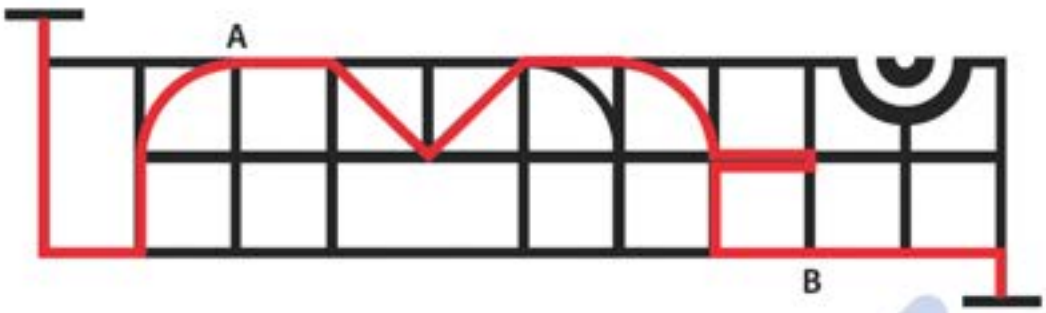
Данная лабораторная работа наиболее важная среди работ, рассмотренных ранее. Это обусловлено не только сложностью преподносимого материала, но и затронутыми важными проблемами, такими как обеспечение точности и качества работы робототехнических систем.



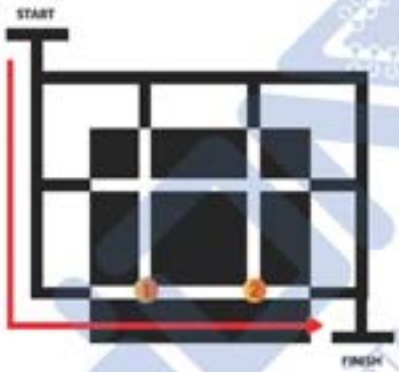


Цель данной работы продемонстрировать, что не только внешние факторы влияют на качество работы того или иного робота. Существенное влияние на процесс выполнения поставленной задачи может оказывать сам робот, функционирующий на основании программы управления.

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.

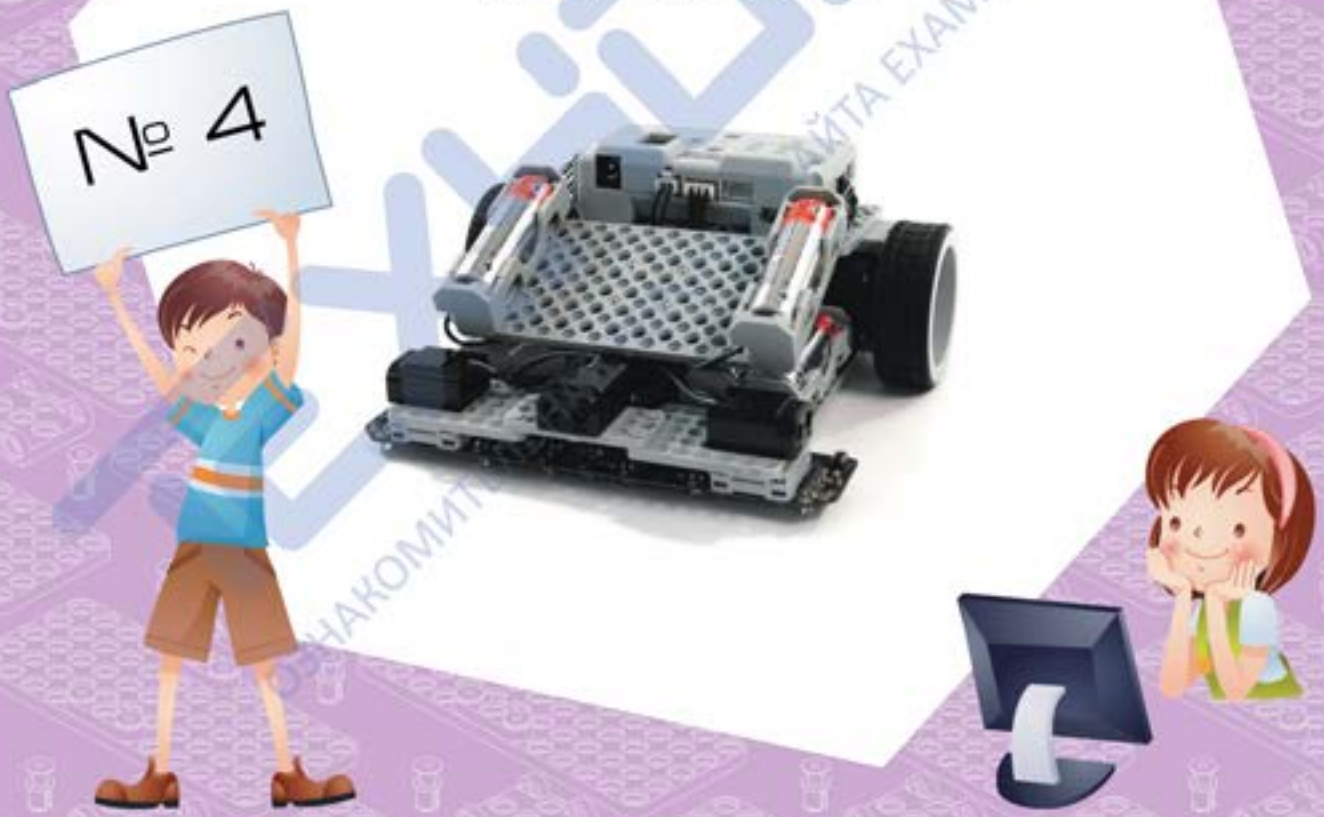


Разработчик робототехнической системы должен предусмотреть все возможные варианты применения своего проектного решения. Чем больше всевозможных влияющих факторов будет учтено на стадии проектирования, тем точнее и качественнее будет функционировать робот.

Лабораторная работа № 4

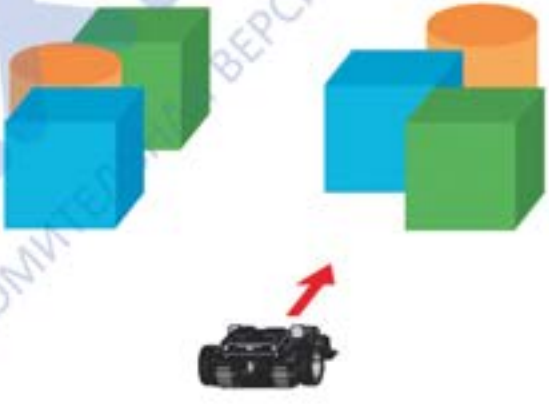
ЭКЗАМЕН ТЕХНОЛАБ

Разработка робота, маневрирующего среди препятствий



Лабораторная работа № 4
«Разработка робота, маневрирующего среди препятствий»

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.

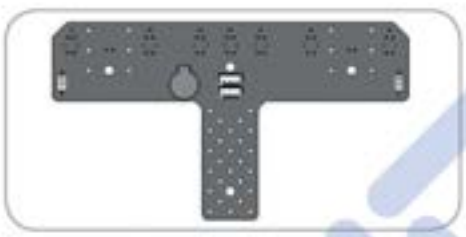
Предлагается рассмотреть ряд приемов, которые могут быть использованы разработчиками как метод усовершенствования системы управления мобильного робота, благодаря чему она сможет выполнять более широкий спектр задач.

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

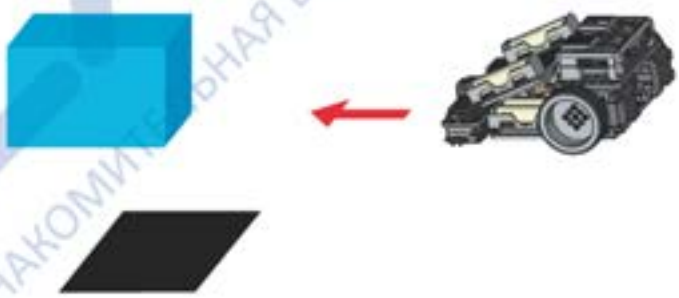
Достаточно часто при движении мобильных роботов по пересеченной местности возникают ситуации, в которых робот не может преодолеть то или иное препятствие на своем пути.

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.

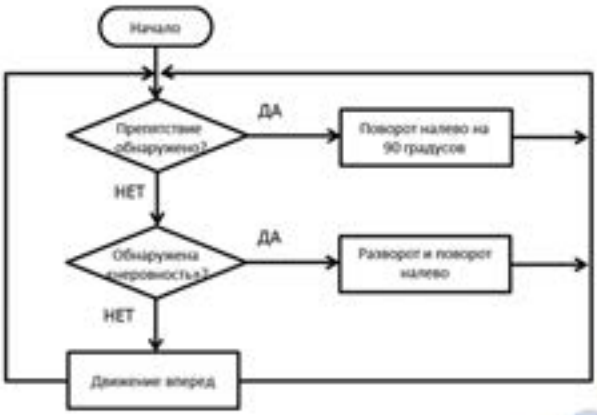
В рамках данной части предлагается рассмотреть модель робота, анализирующего наличие препятствий на собственном пути, а также оценку возможности дальнейшего перемещения. Под этим понимается анализ рабочей поверхности, например поиск обвалов и ям на пути и т.п.



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.



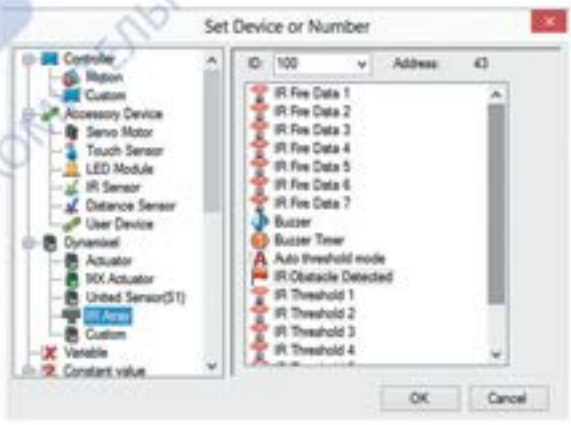
Алгоритм движения робота достаточно прост – робот движется прямолинейно и если он обнаруживает препятствие у себя на пути, он совершает поворот налево, если же на пути робота встречается обрыв или область черного цвета («неровность» поверхности), робот разворачивается и едет в противоположную сторону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



По большому счету программа сводится к единственному бесконечному циклу, анализирующему показания ИК-датчика, подключенного к PORT[6], а также срабатыванию 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.



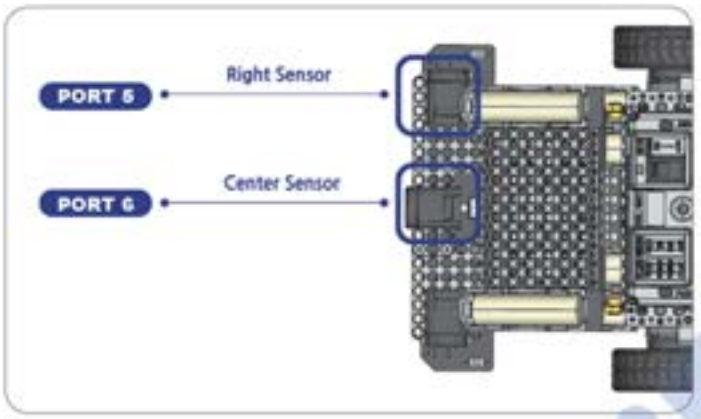
Таким образом, используя даже такие простые средства, как в данном наборе можно смоделировать и исследовать процесс применения мобильного робота в произвольной рабочей зоне. В качестве задания для закрепления результатов можно рассмотреть процесс объединения двух задач воедино – задачи следования по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

Часть № 2. Объезд препятствия по периметру

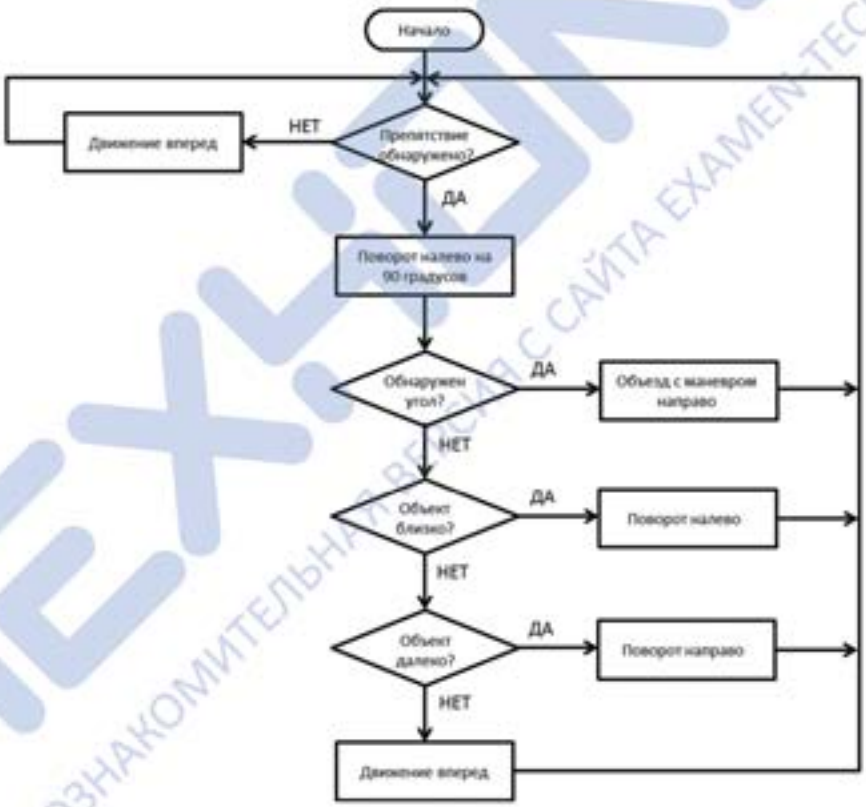
Ранее нами рассматривались задачи объезда препятствий в процессе движения робота по маршруту. В предыдущих работах акцентировалось внимание на алгоритмическую составляющую – поиск препятствия и принятие решения о маневре, а под препятствием понимался объект, который объезжался роботом за один маневр.

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.

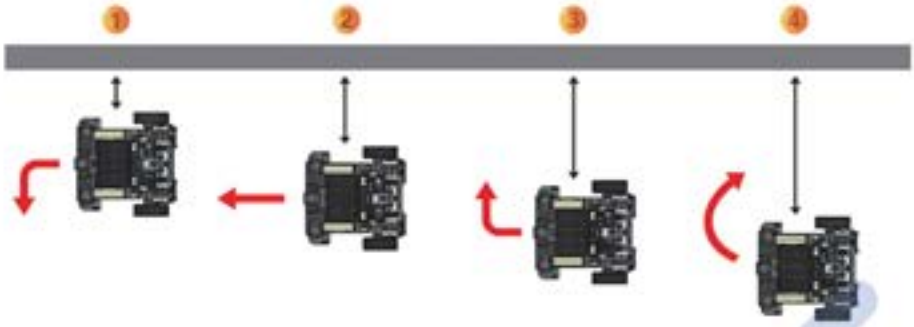
Для того чтобы мобильный робот мог обнаруживать препятствие в процессе его объезда, необходимо расположить один из ИК-датчиков сбоку. В этом случае, подъехав к объекту и начав маневр по его объезду, система управления робота будет постоянно контролировать расстояние до объекта.



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: }
```

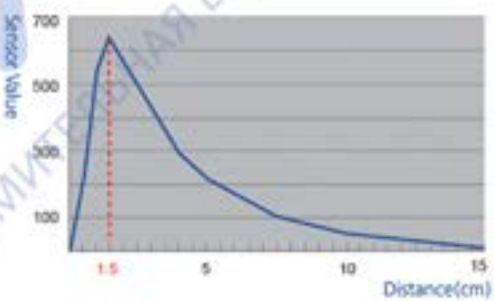
Отдельно рассмотрим функцию r_corner_turn, предназначенную для объезда препятствия справа. Данная функция сначала вызывает функцию forward_shortly, благодаря чему робот перемещается немного вперед, после чего вызывается функция right_turn для поворота направо. Поворот направо осуществляется до тех пор, 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.

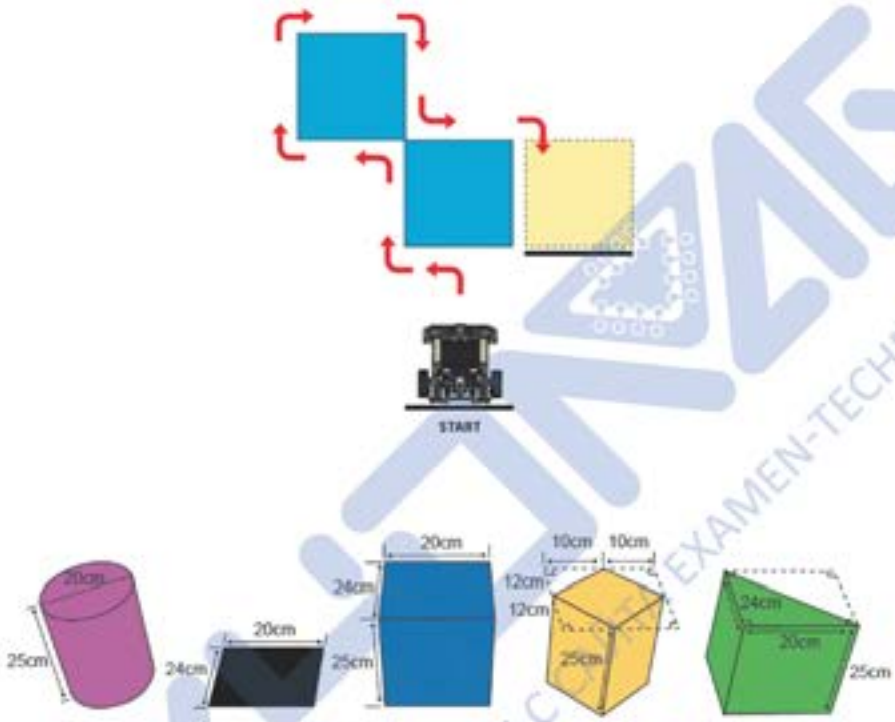


На основании приведенного графика можно подобрать такие значения переменной threshold, при которых мобильный робот будет обнаруживать объекты на заданном расстоянии и объезжать их, не приближаясь более чем положено.

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.

Для заметок

Для заметок

Для заметок

Учебно-методическое издание

Ермишин Константин Владимирович
Каргин Дмитрий Николаевич
Нагорный Алексей Александрович
Панфилов Алексей Олегович

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

ОБРАЗОВАТЕЛЬНЫЙ
РОБОТОТЕХНИЧЕСКИЙ МОДУЛЬ

(ИССЛЕДОВАТЕЛЬСКИЙ УРОВЕНЬ)
РЕСУРСНЫЙ НАБОР
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

(ИССЛЕДОВАТЕЛЬСКИЙ УРОВЕНЬ)

РЕСУРСНЫЙ НАБОР

ЭКЗАМЕН ТЕХНОЛАБ

Издательство ЭКЗАМЕН®

www.examen-technolab.ru

Артикул ТР-0621-МПР

ISBN 978-5-377-07825-8

9 785377 078258

14+ ЛЕТ