

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж



УТВЕРЖДАЮ
Директор
С.А. Махновский
08.02.2023г

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ
МЕЖДИСЦИПЛИНАРНОГО КУРСА
МДК.02.02 Мобильные робототехнические комплексы
для обучающихся специальности
09.02.01 Компьютерные системы и комплексы**

Магнитогорск, 2023

ОДОБРЕНО

Предметно-цикловой комиссией «Информатика
и вычислительная техника»

Председатель Т.Б. Ремез

Протокол № 6 от «25» января 2023 г.

Методической комиссией МпК

Протокол № 4 от «8» февраля 2023 г.

Разработчик:

преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» Многопрофильный колледж Татьяна
Борисовна Ремез

Методические указания по выполнению практических и лабораторных работ разработаны на основе рабочей программы профессионального модуля 02 «Проектирование управляющих программ компьютерных систем и комплексов» МДК 02.02 «Мобильные робототехнические комплексы».

Содержание лабораторных работ ориентировано на подготовку обучающихся к освоению вида деятельности ВД 2 «Проектирование управляющих программ компьютерных систем и комплексов» программы подготовки специалистов среднего звена по специальности 09.02.01. Компьютерные системы и комплексы и овладению профессиональными компетенциями.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ	4
2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ	5
Лабораторное занятие №16. Изучение принципов объектно-ориентированного программирования	5
Лабораторное занятие №17. Тестирование двигателей постоянного тока	7
Лабораторное занятие №18. Тестирование двигателей с ШИМ-регуляцией и без неё	13
Лабораторное занятие №19 Сборка робота базовой модели	14
Лабораторное занятие №20. Подключение моторов и драйверов	18
Лабораторное занятие №21. Тестирование собранной колесной базы	25
Лабораторное занятие №22. Сборка верхней части робота	28
Лабораторное занятие №23. Подключение устройств обратной связи	32
Лабораторное занятие №24. Тестирование электронных компонентов робота	34
Лабораторное занятие №25. Программирование робота на заданное движение	36
Лабораторное занятие №26. Программирование робота на дополнительные действия	42
Лабораторное занятие №27. Управление роботом по каналу инфра красной связи	44
Лабораторное занятие №28. Управление роботом по каналу Bluetooth	52
Лабораторное занятие №29. Управление роботом через приложение	56
Лабораторное занятие №30. Движение робота по заданной траектории	61
Лабораторное занятие № 31. Оснащение робота датчиком	68
Лабораторное занятие № 32. Оснащение робота сервомотором	72
Лабораторное занятие № 33. Движение робота с объездом препятствий	77

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки студентов составляют лабораторные занятия.

Состав и содержание лабораторных работ направлены на реализацию действующего федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью лабораторных работ является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей программой ПМ.02. «Проектирование управляющих программ компьютерных систем и комплексов», МДК 02.02 «Мобильные робототехнические комплексы» предусмотрено проведение практических и лабораторных занятий.

В результате их выполнения, обучающийся должен:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- применять нормативные документы, определяющие требования к оформлению программного кода.

Содержание практических и лабораторных занятий ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению **профессиональными компетенциями:**

ПК 2.1. Проектировать, разрабатывать и отлаживать программный код модулей управляющих программ

ПК 2.2. Владеть методами командной разработки программных продуктов

ПК 2.3. Выполнять интеграцию модулей в управляющую программу

ПК 2.4. Тестировать и верифицировать выпуски управляющих программ

ПК 2.5. Выполнять установку и обновление версий управляющих программ (с учетом миграции - при необходимости)

А также формированию общих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности;

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях;

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях;

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности;

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

Выполнение обучающимися практических и лабораторных работ по ПМ.02. «Проектирование управляющих программ компьютерных систем и комплексов», МДК 02.02 «Мобильные робототехнические комплексы» направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;

- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;

- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;

- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Практические и лабораторные занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 2.2 Элементы робототехнических конструкций Лабораторное занятие №16.

Изучение принципов объектно-ориентированного программирования

Цель работы: изучить принципы ООП и их практическое применение.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации-производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Язык программирования в проекте Arduino, является объектно-ориентированным. Объекты — это сложные структуры, которые имеют оригинальное название, могут включать в себя как переменные, так и функции. Фактически, объекты созданы для удобного обращения к сложным структурам. Например, когда программно подключаем сервомотор, то подключаем сначала библиотеку (`#include <servo.h>`), а затем создаем объект `servomotor` (листинг 4.10). Естественно, что при этом управляющий контакт сервомотора должен быть физически подключен к порту Arduino, и на мотор подано электропитание (рис. 4.17).

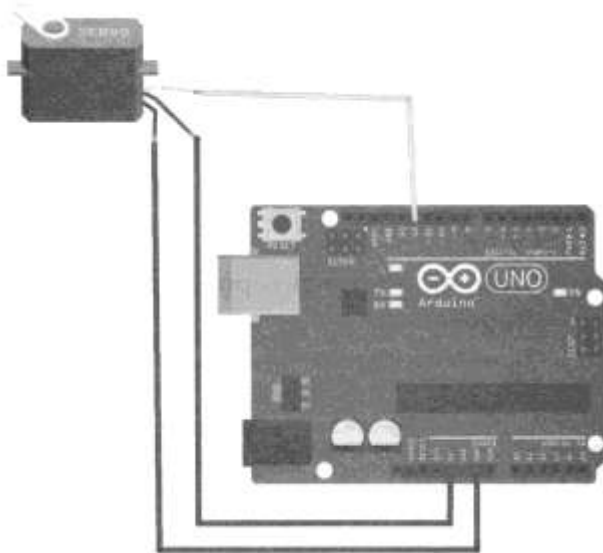


Рис. 4.17. Подключение сервомотора к контроллеру Arduino UNO

Листинг 4.10. Пример управления сервомотором

```
//Подключаем библиотеку для управления сервомоторами.
#include <Servo.h>
// Создаем объект сервомотор.
Servo servomotor;
void setup ()
{
// Присоединяем сервомотор к 12-му пину Arduino.
servomotor.attach(12);
}
void loop()
{
servomotor.write(10); // Поворачиваем вал сервомотора в положение 10°.
// Угол сервомотора увеличивается по часовой стрелке.
delay(500); // Задержка нужна, чтобы сервомотор успел повернуть.
servomotor.write(150); //Поворачиваем вал сервомотора в положение 150°.
delay(500);
// Задержка на поворот.
}
```

Разделение программы (внутренние библиотеки)

Несмотря на то, что в каталоге с программой для Arduino IDE может находиться только одна программа с расширением `.ino`, одноименная с каталогом, программу не обязательно «укладывать» в один файл. В рабочем каталоге с программой можно дополнительно создавать файлы с расширением `.h`, а затем подключать их к программе инструкцией `#include "new_fiie.h"`, где `new_fiie.h`— имя созданного дополнительного файла (оно может быть другим, но должно иметь расширение `.h`) (рис. 4.18).

При этом содержимое подключенного файла при сборке программы включается в тело основного файла программы.

Подключенные подобным образом файлы при открытии основной программы автоматически открываются на редактирование в среде Arduino IDE в отдельных вкладках, что удобно для редактирования. В своей практике я создаю подобные файлы для хранения библиотек функций. При этом для функций различной направленности создаю разные файлы. Таким образом, основной файл программы проекта освобождается от лишней, затрудняющей понимание, информации.

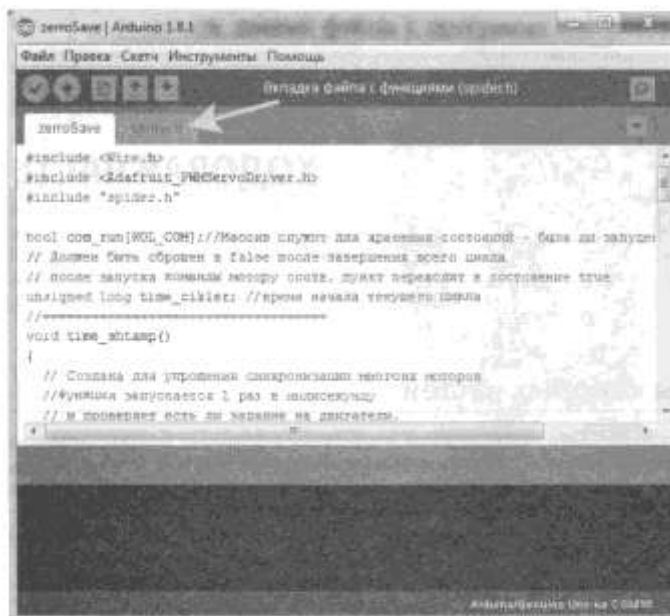


Рис. 4.18. Пример разделения программы на два файла: основной — servoSave.ino и дополнительный — spider.h

Задание.

1. Соберите схему, представленную на рис. 4.17
2. Создайте скетч управляющей программы, загрузите скетч в МК и проверьте работу сервопривода.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к

неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №17.

Тестирование двигателей постоянного тока

Цель работы: научиться подключать двигатели постоянного тока, а также управлять ими с помощью МК.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;

- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Ходовая часть — это комплекс узлов и агрегатов, с помощью которых возможно передвижение робота. Выбор ходовой части для робота, конечно, важный вопрос. При этом можно исходить либо из поставленных перед роботом задач, либо из имеющихся ресурсов, а если основная задача робота — это знакомство с робототехникой, то немаловажным критерием становится доступность деталей ходовой части и простота ее изготовления. Рассмотрим некоторые виды ходовых частей.

Ходовую часть робота можно организовать на основе ног (шагающий робот). Бывают они как двуногие, так и многоногие. Двуногие роботы применяются там, где требуется уподобить робота человеку, их тогда называют *человекоподобными*, или *андроидами*.

Двуногим роботам требуется большое количество высокоточных моторов, а это усложняет управление ими, поскольку вращение всех моторов нужно синхронизировать. Кроме того, потребуется научить робота держать равновесие, а при падении он должен уметь вставать на ноги.

Роботы, ходовые части которых состоят из 4-х и более ног, более устойчивы и могут не содержать механизмов балансировки, т. к. падение их почти невозможно. Ноги подобных роботов проще в реализации и управлении, да и содержат они по 2-3 сервопривода.

Гусеничная ходовая часть весьма широко распространена в робототехнике благодаря простой реализации и высокой проходимости роботов, построенных на ее основе.

Колесная ходовая часть с дифференциалом часто встречается в радиоуправляемых моделях машин. Как правило, в них задействовано два мотора постоянного тока: на дифференциал задних колес и на рулевую тягу.

Колесные ходовые части с мотором на каждое колесо бывают двух видов: с двумя моторными колесами и одним опорным роликом (рис. 5.8) и с четырьмя моторными колесами (рис. 5.9).

Существенных различий между ними нет, кроме того, что четырехколесный робот более мощный и устойчивый. Поэтому именно четырехколесная ходовая будет использована в базовой модели робота.

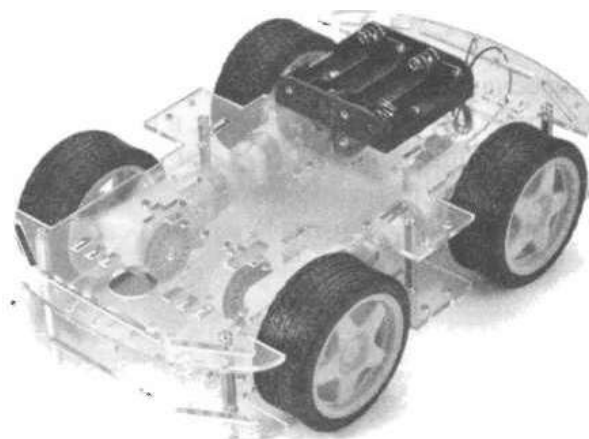
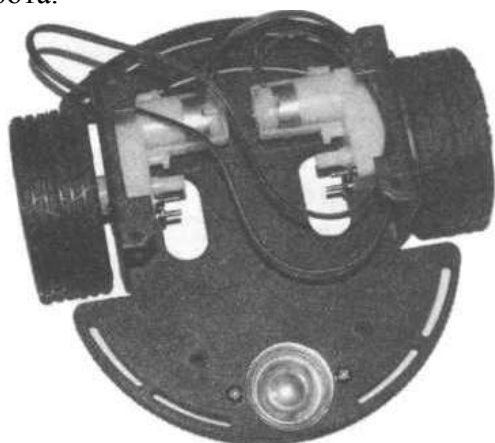


Рис. 5.8. Ходовая с 2-мя ведущими колесами Рис. 5.9. Ходовая с 4-мя ведущими колесами

После выбора структуры ходовой, следует определить, какие двигатели будет использовать наш робот. Конечно, исходить при этом надо из наличия соответствующих комплектующих и/или поставленных задач, поэтому вам предстоит выбор между двигателями постоянного тока с редуктором, сервомоторами постоянного вращения и шаговыми двигателями. Их различия представлены в табл. 5.1.

Таблица 5.1. Сравнительные параметры различных типов двигателей

Сравниваемые параметры	Двигатель постоянного тока с понижающим редуктором	Сервомотор постоянного вращения	Шаговый двигатель
Количество задействованных портов микроконтроллера для управления	2-3	1	4
Скорость реакции на поступившую команду, сек	~1/1000	~1/10	~1/1000
Скорость вращения (с учетом понижающего редуктора), об/мин	20-240	1-60	1-60
Стоимость (за единицу принята стоимость двигателя постоянного тока с редуктором), сравниваются двигатели, равные по мощности	1	5	3
Минимальное количество портов управления ходовой частью для проекта колесного робота с 4-мя ведущими колесами	4 (при отказе от управления мощностью и параллельном управлении двумя правыми и двумя левыми моторами)	2 (при параллельном управлении двумя правыми и двумя левыми моторами)	8 (при параллельном управлении двумя правыми и двумя левыми моторами)

Как можно видеть, наиболее быстрая реакция у двигателя постоянного тока, у него также и самая низкая стоимость. Сервомотор превосходит его за счет задействования меньшего количества портов Arduino. Шаговый двигатель при использовании в ходовой части не дает особых преимуществ, хотя если количество сделанных роботом оборотов колес нужно считать и строго дозировать, то это неплохой выбор.

Сервомоторами хорошо регулируется скорость движения робота, если скорость колес нужно менять постоянно и держать в точно заданных пределах. Скорость оборота колеса, вращаемого сервомотором, зависит только от установленного на порту значения, и при увеличении сопротивления (когда, например, робот движется в гору) практически не меняется, в то время как двигатель постоянного тока изменяет скорость своего вращения в зависимости от нагрузки.

Шаговым двигателем также можно держать постоянную скорость, но технически реализовать это несколько сложнее из-за большего количества задействованных портов Arduino. Так, роботу, имеющему 4 ведущих колеса, потребуется 16 портов управления, правда правые и левые колеса можно подключить к управлению параллельно, и тогда количество портов уменьшится до 8, что, однако, тоже избыточно.

Тем не менее, все три вида двигателей применяются в ходовых частях колесных роботов, и выбор зависит от требований, предъявляемых к роботу. Далее будет рассматриваться ходовая с двигателями постоянного тока, снабженными редукторами, т. к. строгих требований к скорости и мощности робота мы предъявлять не станем.

Драйверы двигателей

Платы Arduino, кроме специализированных, не поддерживают возможностей управления двигателями постоянного тока напрямую, и для этого приходится применять специальные

специализированные микросхемы, называемые *драйверами двигателей*. Наиболее распространенные из них приведены в табл. 5.2.

Таблица 5.2. Специализированные микросхемы драйверов двигателей

Характеристики	L293D	L293B	L298N
Максимальный ток на канал, А	0,6	1	3
Количество каналов	4	4	4
Встроенная диодная защита от паразитных токов	Есть	Нет	Нет

Здесь рассмотрены три микросхемы драйверов, которые подходят для управления направлением вращения и мощностью двигателей постоянного тока. Они же могут применяться и для управления шаговыми двигателями.

Микросхему L293D можно использовать без дополнительных доработок, более того— для повышения максимального тока эти микросхемы иногда применяются парами и снабжаются радиаторами. Микросхемы L298N и L293B должны быть установлены вместе с защитными диодами, которые предохраняют их от перегрузок.

Если вы не боитесь пайки, то можно приобрести две микросхемы L293 D, установить их друг на друга и спаять ножки, а к средним ножкам припаять дополнительный радиатор — например, кусок толстой медной проволоки. Теперь это изделие можно использовать в качестве драйвера для нашей модели (рис. 5.11).

Однако с целью сохранения простоты сборки в проектах будет использована схемная реализация на основе микросхемы L298N (рис. 5.12). Она довольно мощная, имеет хороший запас по максимальному току (3 А на канал), на плате уже установлены ограничительные диоды и имеется дополнительный стабилизатор напряжения на 5 В.

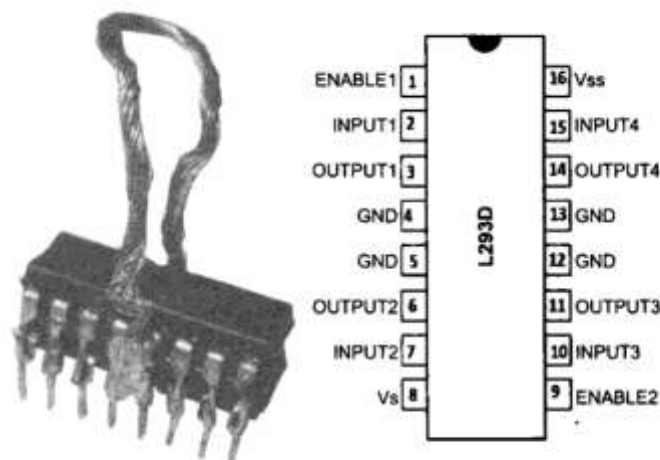


Рис. 5.11. Спаренная микросхема L293D с радиатором из проволоки (слева), назначение контактов микросхемы (справа)

В табл. 5.3 приведены значения сигналов на входах и показана соответствующая реакция двигателя.

Таблица 5.3. Значения сигналов на входах и соответствующая реакция двигателя

In 4	Enable 2	M1 (направление вращения условное)
0	0	Отключен
0	1	Отключен
1	1	Вал вращается по часовой стрелке
1	0	Отключен
1	ШИМ	Вал вращается по часовой стрелке (мощность зависит от величины положительного импульса ШИМ)

Задание

1. Соберите схему, приведенную на рис. 5.22. Перемычки на контактах ENA и ENB драйвера не убираем!

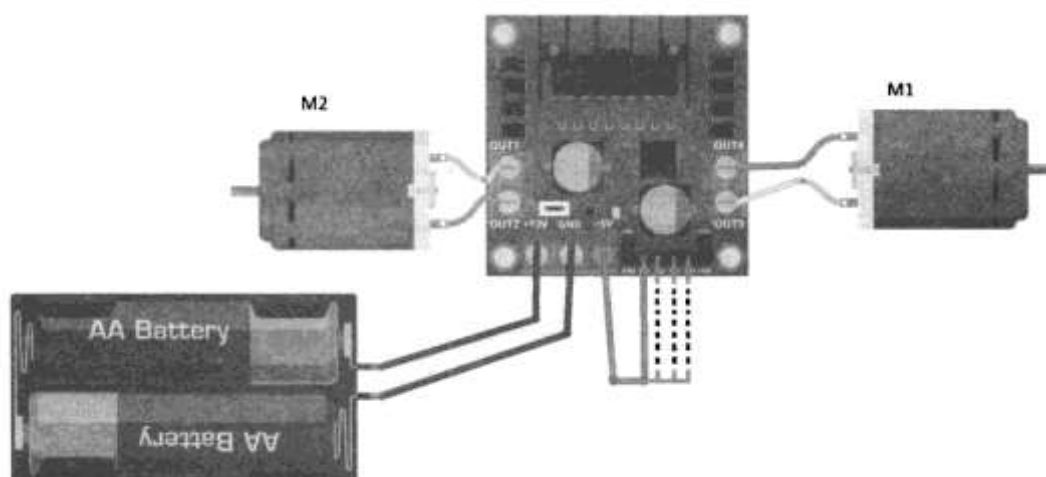


Рис. 5.22. Схема управления двигателями без контроллера Arduino

Двигатели подключаются к драйверу через винтовые зажимы, также подключается аккумуляторный бокс, а для контактов IN1-IN4 драйвера лучше использовать провода с готовыми клеммами Dupont. Подключая 5 вольт к различным контактам IN 1-IN4, проследите, как изменяется вращение двигателей M1 и M2 (используйте данные табл. 5.4¹).

Таблица 5.4. Значения сигналов на входах и соответствующая реакция двигателя

In 4	In 3	Enable 2	Состояние двигателя M1 (направление вращения условное)
0	0	0	Отключен
0	1	1	Вал вращается против часовой стрелки
1	0	1	Вал вращается по часовой стрелке
1	1	0	Отключен
1	1	1	Отключен (на обоих контактах двигателя напряжение питания)
0	0	1	Отключен (оба контакта двигателя заземлены)
1	0	ШИМ	Вал вращается по часовой стрелке (мощность зависит от величины положительного импульса ШИМ)
0	1	ШИМ	Вал вращается против часовой стрелки (мощность зависит от величины положительного импульса ШИМ)

¹ Контакты IN 1 и IN2 отвечают за работу двигателя M2, а IN3 и IN4 — двигателя M1.

2. Подключаем контроллер Arduino согласно схеме, приведенной на рис. 5.23.

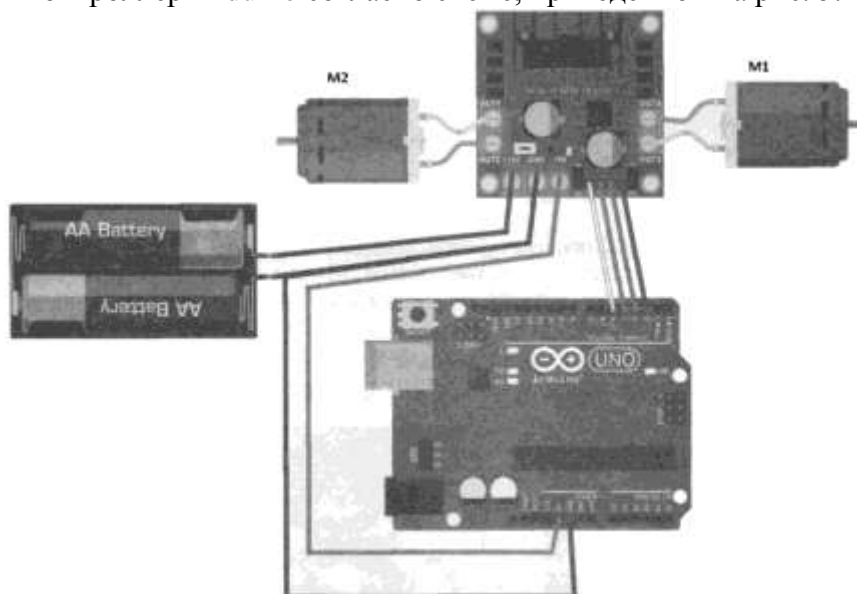


Рис. 5.23. Схема управления двигателями посредством Arduino (без регуляции ШИМ)

3. Можно использовать дополнительную плату Arduino Sensor shield V5.0 (схема на рис. 5.24). Для этого потребуются два гибких провода под винтовой зажим и четыре типа «мама-мама» для объединения информационных контактов драйвера и Arduino Uno.

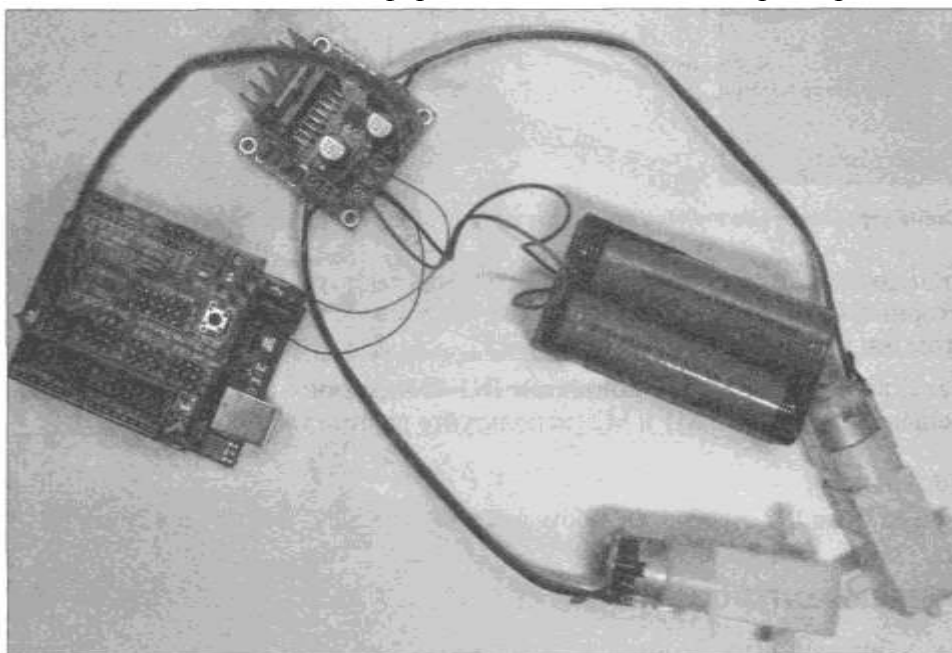


Рис. 5.24. Схема в сборе: Arduino UNO с установленным на нем Arduino Sensor shield V5.0, драйвер двигателей L298N, литиевые аккумуляторы в боксе, двигатели постоянного тока с редукторами

4. Создайте скетч тестовой программы, загрузите скетч в МК и проверьте работу двигателей. Логика работы тестовой программы (листинг 5.1) следующая:

В бесконечном цикле:

1. включаем оба двигателя.

2. Вращаем двигатель M2 1 сек в одну сторону:

```
digitalWrite(in1, LOW),  
digitalWrite (In2, HIGH)
```

и 1 сек в другую:

```
digitalWrite (In1, HIGH),  
digitalWrite(In2, LOW).
```

3. включаем двигатель M2.

Самостоятельно измените эту программу для управления двигателем M 1.

Листинг 5.1. Тестовая программа управления двигателями

//Создадим переменные для хранения номеров используемых пинов/портов Arduino.

```
int In1, In2, In3, In4;
```

```
//Настройка void setup() I
```

```
// Присвоим переменным номера пинов Arduino.
```

```
In1 = 5;
```

```
In2 = 4;
```

```
In3 = 3;
```

```
In4 = 2;
```

```
//Переведем данные пины/порты в режим вывода.
```

```
pinMode (In1, OUTPUT);
```

```
pinMode(In2, OUTPUT);
```

```
pinMode(In3, OUTPUT);
```

```
pinMode(In4, OUTPUT);
```

```
}
```

```
//Тело программы
```

```
void loop()
```

```
{ //Отключим оба двигателя.
```

```
digitalWrite(In1, LOW); //двигатель M2.
```

```
digitalWrite(In2, LOW);
```

```
digitalWrite(In3, LOW); //двигатель M1.
```

```
digitalWrite(In4, LOW);
```

```
delay(1000); // Ждем 1 сек.
```

```
//Выключим двигатель M2.
```

```
digitalWrite(In1, LOW); //двигатель M2.
```

```
digitalWrite(In2, HIGH);
```

```
delay(1000); // Ждем 1 сек.
```

```
// Вращаем двигатель M2 в другую сторону.
```

```
digitalWrite(In1, HIGH);
```

```
digitalWrite(In2, LOW);
```

```
delay(1000); // Ждем 1 сек.
```

```
}
```

Проделайте те же операции для двигателя M1 самостоятельно.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №18.

Тестирование двигателей с ШИМ-регуляцией и без неё

Цель работы: научиться подключать двигатели постоянного тока, а также управлять ими с помощью МК с ШИМ – регуляцией и без нее.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;

- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №19.

Сборка робота базовой модели

Цель работы: научиться выполнять сборку робота базовой модели, а также разрабатывать простейшую программу управления роботом.

Выполнив работу, Вы будете:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Широтно-импульсная модуляция

Рассмотрим подробнее, как работает широтно-импульсная модуляция. В режиме ШИМ в соответствующий порт командой **analogWrite (Порт, значение)** записывается определенное число. В зависимости от этого числа на выходе порта с частотой около 490 Гц генерируется последовательность импульсов, вид которой зависит от записанного числа. Эта зависимость показана на рис. 5.14. На время положительного фронта включается двигатель — соответственно, чем шире положительный фронт, тем дольше подключен двигатель, тем больше его мощность и скорость вращения.

Вращение в обе стороны

Если вал двигателя должен вращаться в обоих направлениях, нужно использовать схему, приведенную на рис. 5.15. Она позволяет подключать только два независимых двигателя, но параллельное подключение двигателей является допустимым.

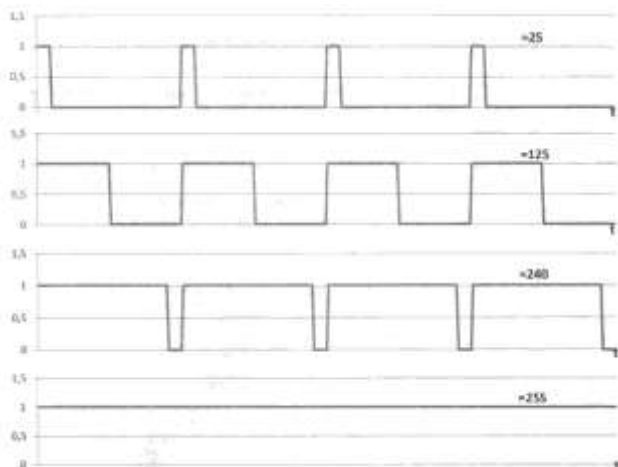


Рис. 5.14. Принцип работы ШИМ

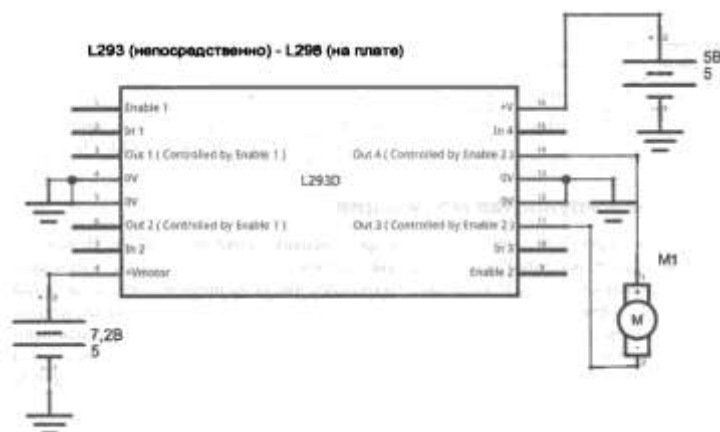


Рис. 5.15. Подключение двигателя для обеспечения двустороннего вращения

Задание.

1. Добавьте к схеме управления двигателями (рис. 5.23) возможность управления мощностью двигателей с помощью ШИМ (рис. 5.25), что позволит аппаратно управлять скоростью вращения вала двигателя и, соответственно, скоростью движения робота. Для этого потребуются два порта с аппаратным ШИМ, на плате они обозначены символом волнистой линии (~) — пусть это будут порты 6 и 9. Уберите переключки с контактов ENA и ENB драйвера и подсоедините эти контакты к портам 6 и 9 платы Arduino.

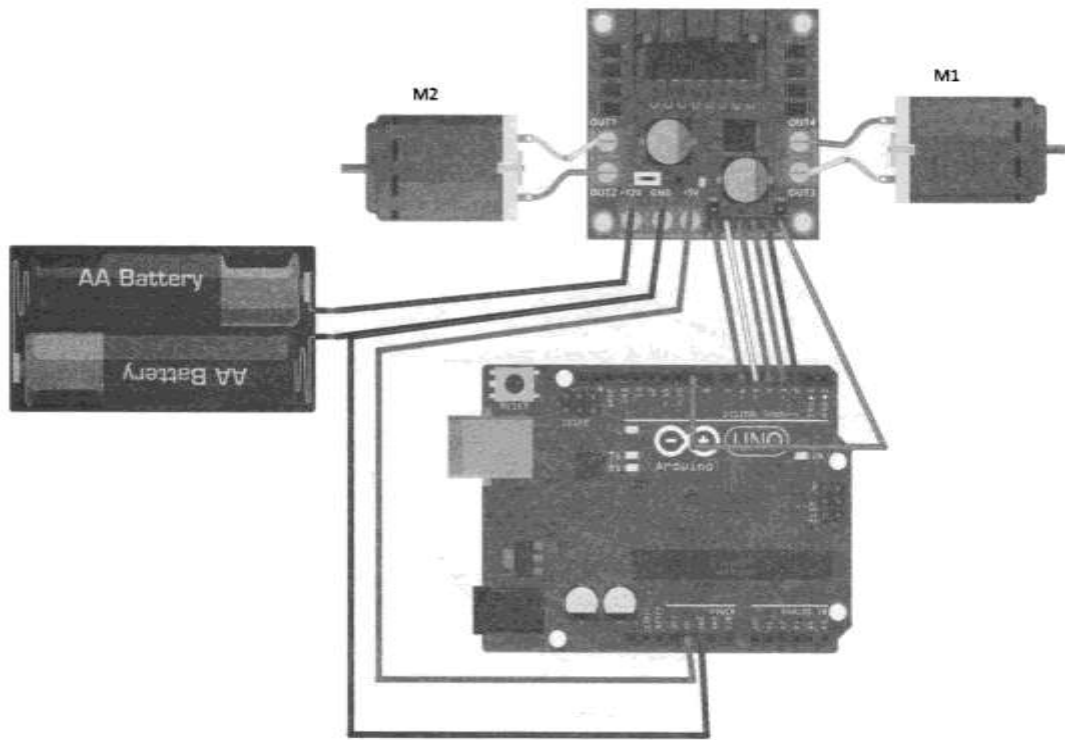


Рис. 5.25. Схема управления двигателями с регуляцией на основе ШИМ

Напомним, что значения ШИМ в Arduino могут изменяться от 0 до 255. На практике вал двигателя не сразу начнет вращаться — сначала двигатель станет гудеть и лишь по достижении определенного значения ШИМ начнет медленно увеличивать обороты, что связано с недостатком мощности для компенсации сил трения.

2. Создайте скетч с тестовой программой и загрузите ее в МК.

Логика работы тестовой программы управления двигателями с регуляцией на основе ШИМ (листинг 5.2) следующая:

В бесконечном цикле:

- Отключаем оба двигателя подачей на оба управляющих контакта низкого сигнала.
- Отключаем ШИМ (для чего присваиваем ШИМ значение 0).
- После секундной паузы даем команду вращения двигателю M2 путем подачи положительного сигнала на контакт IN2.
- Далее в цикле через 300 миллисекунд увеличиваем ширину положительного фронта ШИМ на контакте ENA от 0 до 255.
- Затем полностью отключаем ШИМ на ENA записью 0: `analogWrite (ENA, 0)`.
- Переключаем полярность питания на двигателе M2 и аналогично начинаем в цикле увеличивать ширину ШИМ.

Итогом работы программы будет поочередное ускоряющееся вращение двигателя M2 то в одну, то в другую сторону. Самостоятельно измените программу для управления вторым двигателем M1.

Листинг 5.2. Тестовая программа управления двигателями с регуляцией на основе ШИМ

//Создадим переменные для хранения номеров используемых пинов/портов Arduino.

`int In1, In2, In3, In4, ENA, ENB;`

//Настройка void setup() {

// Присвоим переменным номера пинов Arduino.

`In1 = 5; //Двигатель M2`

`In2 = 4; //Двигатель M2`

`In3 = 3; //Двигатель M1`

`In4 = 2; //Двигатель M1`

`ENA=6; //Двигатель M2`

`ENB=9; //Двигатель M1`

//Переведем данные пины/порты в режим вывода.

```

pinMode(In1, OUTPUT);
pinMode(In2, OUTPUT);
pinMode(In3, OUTPUT);
pinMode (In4, OUTPUT) ;
pinMode(ENA, OUTPUT);
pinMode(ENB, OUTPUT);
}
//Тело программы
void loop() {
//Отключим оба двигателя.
digitalWrite(In1, LOW); //двигатель M2.
digitalWrite(In2, LOW);
digitalWrite(In3, LOW); //двигатель M1.
digitalWrite(In4, LOW);
analogWrite(ENA, 0); //ШИМ двигателя M2 полностью выключен.
analogWrite(ENB,0); //ШИМ двигателя M1 полностью выключен.
delay(1000); // Ждем 1 сек.
//Выключим двигатель M2.
digitalWrite(In1, LOW); //двигатель M2.
digitalWrite(In2, HIGH);
for(int i=0;i<255;i++)
{
analogWrite(ENA, 1); delay(300); // Ждем 0.3 сек.
}
analogWrite (ENA, 0); // ШИМ двигателя M2 полностью выключен.
// Вращаем двигатель M2 в другую сторону.
digitalWrite(In1, HIGH);
digitalWrite(In2, LOW);
for(int i=0; i<255;i++)
{
analogWrite(ENA,i); delay(300); // Ждем 0.3 сек.
}
analogWrite(ENA,0); // ШИМ двигателя M2 полностью выключен.
delay(1000); // Ждем 1 сек.
Прделайте те же операции для двигателя M1 самостоятельно.
}

```

3. Регулирование скорости вращения без использования аппаратного ШИМ

Регулировать скорость вращения колес можно и без использования аппаратного ШИМ. Для этого вернем схему в состояние, показанное на рис. 5.23-5.24 (не забудьте установить переключки на контакты ENA и ENB драйвера), и загрузим программу, приведенную в листинге 5.3.

Отличие этой программы от предыдущей в том, что значение ШИМ на контактах ENA и ENB всегда максимально (на них постоянное напряжение 5 вольт), а скорость вращения двигателей регулируется программно, путем краткосрочного включения или отключения вращения двигателей в заданном направлении.

В бесконечном цикле для левого двигателя это выглядит так:

- включаем двигатель M2 подачей низкого сигнала на контакты IN 1 и IN2, ждем 1 секунду.
- В цикле for по переменной i от 0 до long_c (в программе это 100) начинаем отключать и включать вращение двигателя в одном направлении, при этом время, когда двигатель выключен, задается командой delay (long_c - i), а когда включен— delay (i). Так как значение переменной i в цикле каждый повтор увеличивается, то время, в течение которого двигатель выключен, постепенно уменьшается, а время работы двигателя увеличивается. Это приводит к плавному увеличению скорости

вращения вала.

- Затем та же процедура повторяется для вращения в обратную сторону.
- Теперь измените программу для управления двигателем М1 самостоятельно.

Листинг 5.3. Тестовая программа управления скоростью вращения двигателей без использования регулирования на основе аппаратного ШИМ

```
//Создадим переменные для хранения номеров используемых пинов/портов Arduino.
```

```
int In1, In2, In3, In4;
```

```
//Настройка void setup()
```

```
{
```

```
// Присвоим переменным номера пинов Arduino.
```

```
In1 = 5; //Двигатель М2
```

```
In2 = 4; //Двигатель М2
```

```
In3 = 3; //Двигатель М1
```

```
In4 = 2; //Двигатель М1
```

```
//Переведем эти пины/порты в режим вывода.
```

```
pinMode(In1, OUTPUT);
```

```
pinMode (In2, OUTPUT);
```

```
pinMode (In3, OUTPUT);
```

```
pinMode(In4, OUTPUT);
```

```
}
```

```
//Тело программы
```

```
void loop()
```

```
{
```

```
int i;
```

```
int long_c;
```

```
//Остановить оба двигателя.
```

```
digitalWrite(In1, LOW); //двигатель М2.
```

```
digitalWrite(In2, LOW);
```

```
digitalWrite(In3, LOW); //двигатель М1.
```

```
digitalWrite(In4, LOW);
```

```
delay(1000) ; // Ждем 1 сек.
```

```
long_c = 100;
```

```
for (i = 0; i <= long_c; i++)
```

```
{
```

```
//Остановить двигатель М2.
```

```
digitalWrite(In1, LOW); //двигатель М2.
```

```
digitalWrite(In2, LOW);
```

```
delay(long_c - i) ; // Ждем (long_c-i)*0.001 сек.
```

```
//Вращаем двигатель М2.
```

```
digitalWrite(In1, LOW); //двигатель М2.
```

```
digitalWrite(In2, HIGH); delay(i); // Ждем (i*0.001) сек.
```

```
}
```

```
delay(3000) ;
```

```
// Вращаем двигатель М2 в другую сторону.
```

```
for (i = 0; i <= long_c; i++)
```

```
{
```

```
//Остановить двигатель М2 .
```

```
digitalWrite(In1, LOW); //двигатель М2.
```

```
digitalWrite(In2, LOW);
```

```
delay(long_c - i); // Ждем (long_c-i)*0.001 сек.
```

```
//Вращаем двигатель М2.
```

```
digitalWrite(In1, HIGH); //двигатель М2.
```

```
digitalWrite(In2, LOW);
```

```
delay(i); // Ждем (i*0.001) сек.//
```

```
}  
delay(3000);  
// Проделайте те же операции для второго двигателя M1 самостоятельно.  
}
```

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №20.

Подключение моторов и драйверов

Цель работы: научиться монтировать моторы и элементы управления на базовую основу робота.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Сборка робота — процесс не сложный, но требует внимания, так как неправильное подсоединение проводов может привести к порче электронных компонентов. А неправильное

крепление двигателей — например, сильная затяжка винтов, может привести к их заклиниванию.

Базовая конструкция робота содержит 4 мотора постоянного тока с редукторами и колесами под эти редукторы, корпусом из пластика, необходимыми крепежными элементами и соответствующей управляющей логикой (рис. 6.1-6.4).

Робот состоит из двух уровней:

- нижний: двигатели, колеса, драйвер двигателей;
- верхний: источники электропитания, выключатель, плата Arduino, голова с датчиком на сервомоторе.

Задание. Выполните сборку и подключение устройств нижнего уровня базовой конструкции

Если корпус изготовлен методом резки лазером из листов акрилового стекла, покрытого специальной защитной пленкой, то на пленке, как правило, остаются черные разводы от продуктов горения. Пленка также может мешать соединению деталей, поэтому перед сборкой остатки пленки следует удалить, поддев острым лезвием (рис. 6. 1).

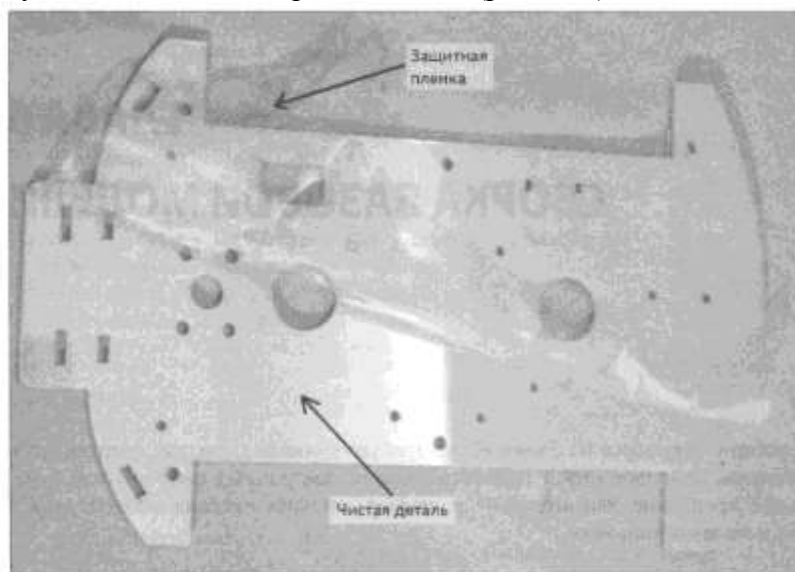


Рис. 6.1. Деталь робота из акрилового стекла с отделяемой пленкой

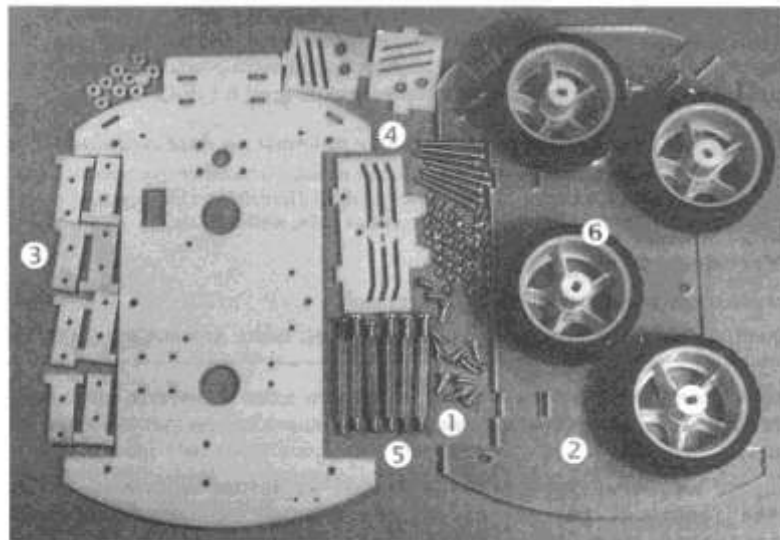


Рис. 6.2. Набор конструктивных деталей: винты и гайки (1); верх корпуса и низ (2); Т-образные элементы для установки двигателей в корпус (3); элементы для установки вращающейся «головы» с датчиком расстояния (4); стойки (5); колеса (6)

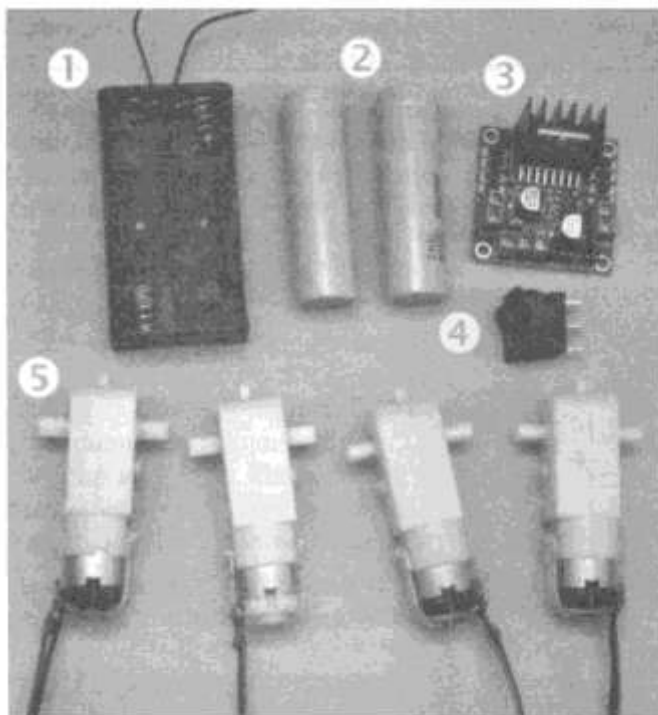


Рис. 6.3. Питание и двигатели: бокс для аккумуляторов формата 18650 (1); литиевые аккумуляторы формата 18650 (2); драйвер двигателей (3); выключатель подачи электропитания (4); моторы с редукторами (5)

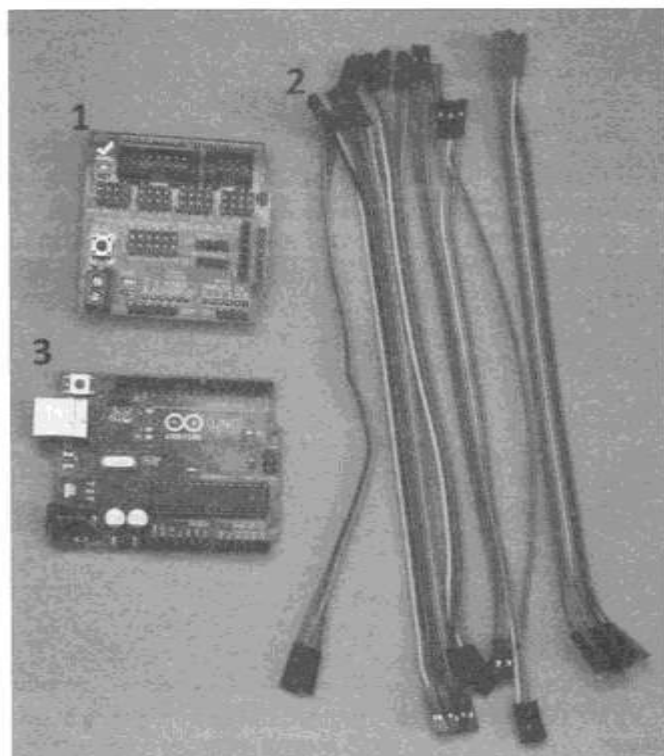


Рис. 6.4. Управляющая логика и провода: плата Arduino Sensor Shield v5.0 (1); провода с клеммами dupont (2); плата Arduino Uno (3)

Элементы питания

В качестве элементов питания в проекте используются два литиевых аккумулятора формата 18650 (см. рис. 6.3). Они подключаются последовательно и дают напряжение питания 7,2-8,4 вольта в зависимости от степени заряда. Но можно использовать и шесть батареек или аккумуляторов формата АА (рис. 6.5), при этом будет получено напряжение 7,2-9 вольт.

Двигатели

Перед установкой двигателей на ходовую часть следует припаять к контактным площадкам двигателей провода (рис. 6.6). Лучше использовать для этого гибкие многожильные провода разного цвета. Провода должны быть достаточной длины, чтобы не только достать до платы драйвера двигателей, но и быть аккуратно уложенными. Оставьте для них длину 12-15 см, при монтаже к плате драйвера лишнее можно будет отрезать.

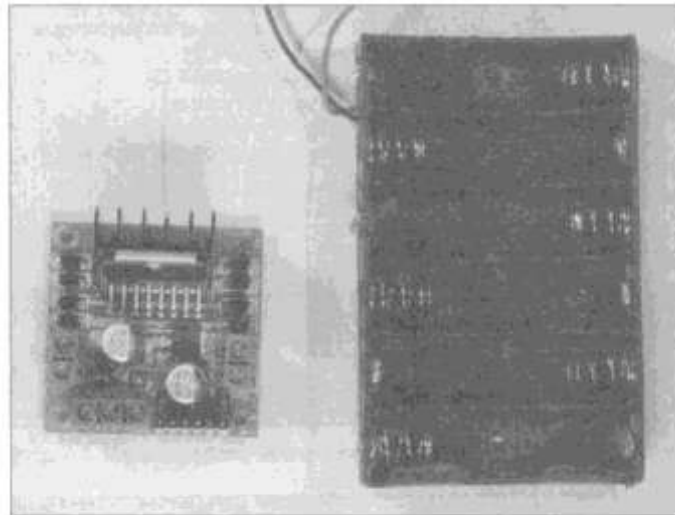


Рис. 6.5. Плата драйвера двигателей (слева) и бокс для шести батарей/аккумуляторов формата АА (справа)

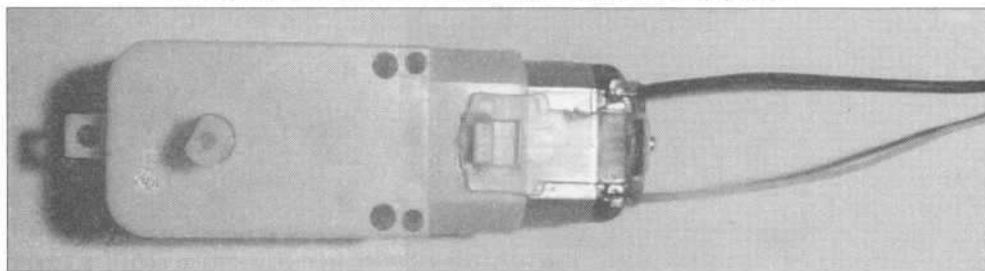


Рис. 6.6. Двигатель с припаянными проводами

При пайке, если контакты двигателя сильно окислились и не поддаются лужению, их следует немного почистить острым ножом — это ускорит процесс лайки. Но не перестарайтесь, контакты можно сломать!

При сборке рекомендуется при креплении проводов к двигателю использовать именно пайку, но если она недоступна, можно снять изоляцию с конца провода, конец без изоляции тщательно скрутить и продеть в ушко контакта двигателя, после чего продетый конец обмотать вдоль той неизолированной части провода, которая осталась до ушка контакта.

Для исключения электромагнитных наводок, которые возникают при работе двигателей и могут стать причиной сбоев в работе электроники робота, следует между контактами каждого электрического двигателя впаять керамический конденсатор (рис. 6.7). Электромагнитные наводки могут влиять на радиоприем — например, если поднести радиоприемник к работающему двигателю, то ничего, кроме помех, слышно не будет. Конденсатор же, благодаря своим свойствам, поглощает подобные высокочастотные скачки электрического напряжения. Для напряжения 5-12 В и используемых маломощных моторов достаточно конденсатора емкостью 100 нФ.

Для установки двигателей в корпус робота применяются Т-образные элементы (рис. 6.8), которые устанавливаются снизу корпуса в пазы (рис. 6.9). Каждый двигатель можно закрепить как на одном, так и на двух Т-образных элементах (предпочтительнее все же на двух). Этапы установки двигателя на двух Т-образных элементах показаны на рис. 6.8-6.11. Двигатели, как можно видеть, крепятся двумя болтами в специальные крепежные отверстия Т-образного элемента.

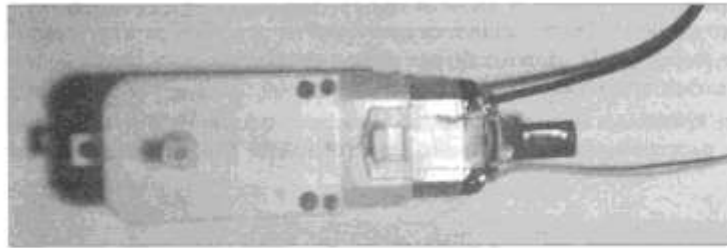


Рис. 6.7. Двигатель с припаянными проводами и конденсатором

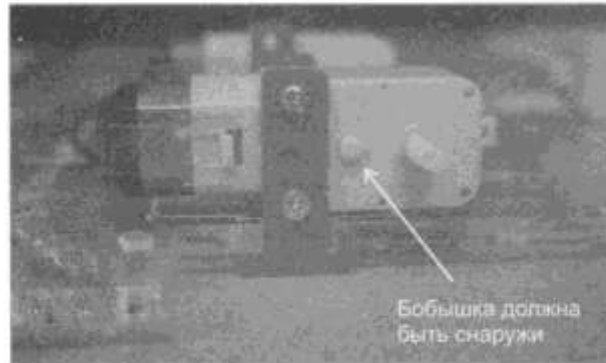


Рис. 6.8. Подготовка двигателя к монтажу: на двигатель установлен внешний T-образный элемент и вставлены 3-мм винты длиной 30 мм

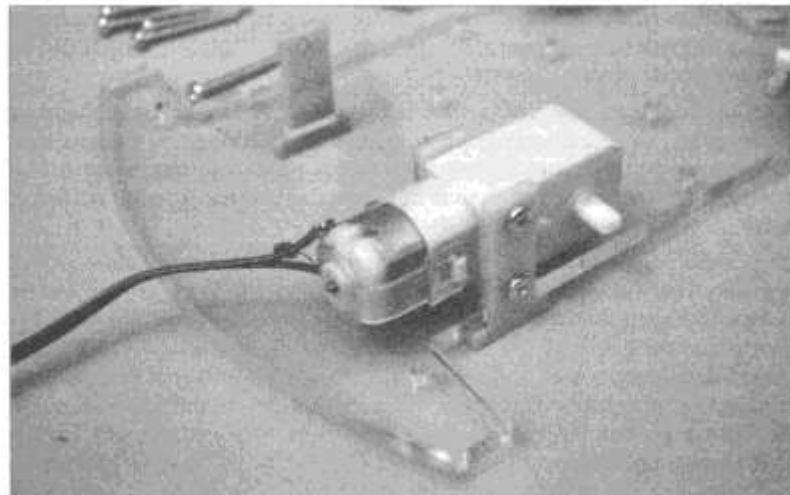


Рис. 6.9. Установка двигателя на T-образные крепления в корпус робота



Рис. 6.10. Фиксируем винты

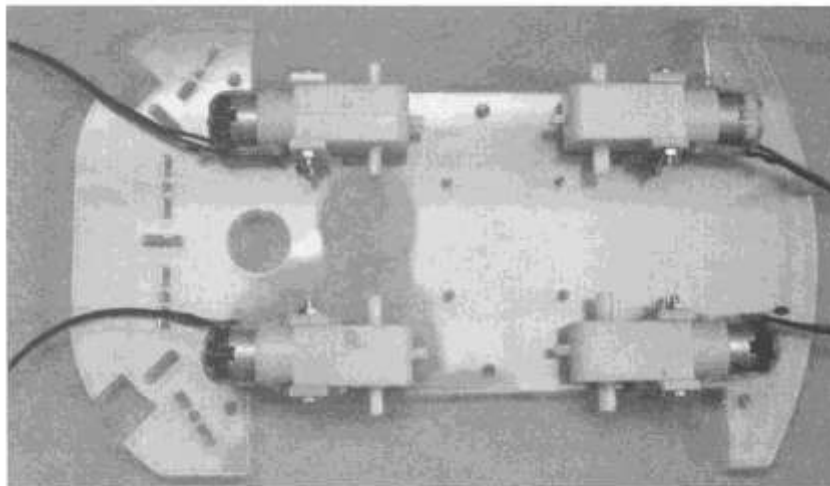


Рис. 6.11. Все двигатели смонтированы

Драйвер двигателей

После установки двигателей следует установить на плате корпуса робота плату, содержащую драйвер двигателей (типа L298N). Предварительно вставьте в отверстия платы драйвера винты и снизу наденьте на них пластиковые шайбы, иначе можно повредить плату драйвера (рис. 6.12).

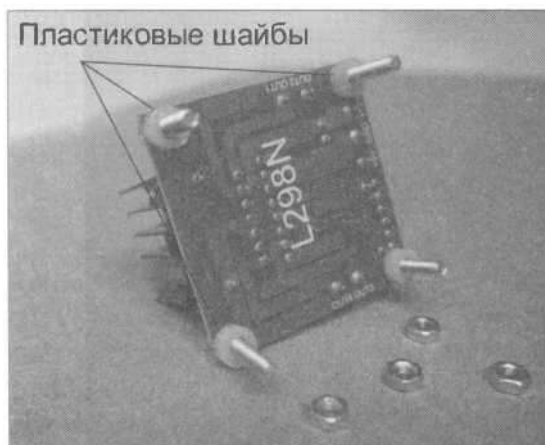


Рис. 6.12. Плата драйвера со вставленными винтами длиной 14 мм и пластиковыми шайбами

В плате корпуса робота просверлены специальные отверстия для крепления платы драйвера (рис. 6.13) — устанавливать плату драйвера следует именно на них.

Плата драйвера крепится четырьмя винтами в эти отверстия на корпусе (рис. 6.14). После установки платы переверните корпус и затяните гайки (рис. 6.15).

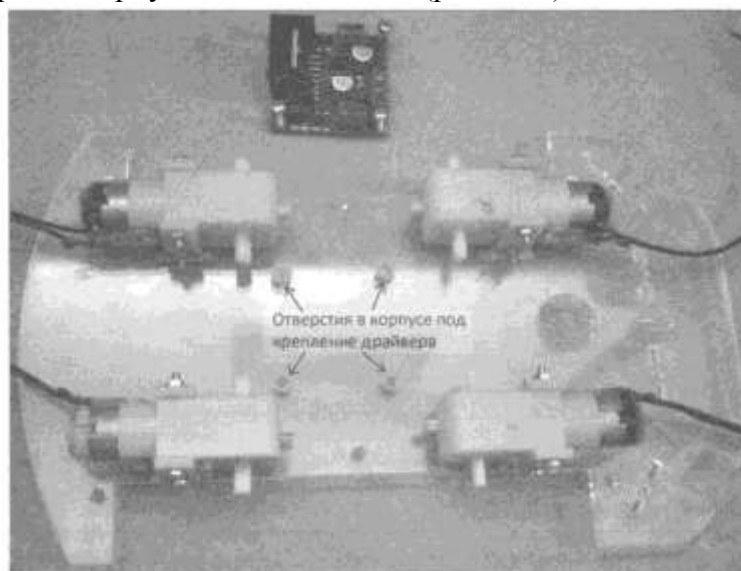


Рис. 6.13. Расположение отверстий для крепления драйвера на корпусе робота

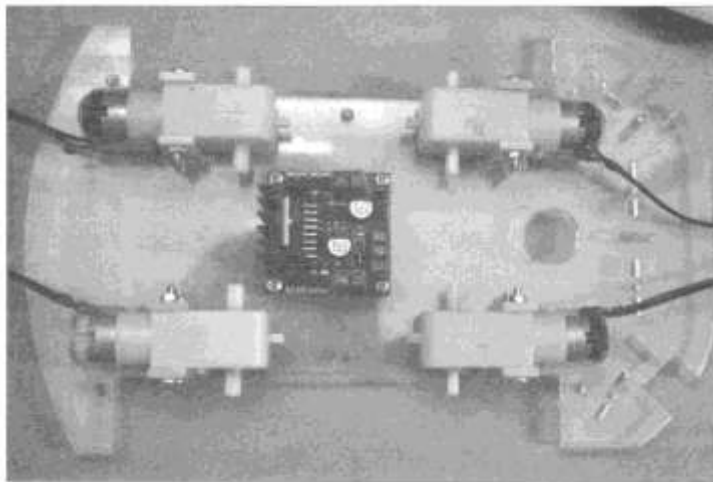


Рис. 6.14. Монтаж платы драйвера: вид сверху

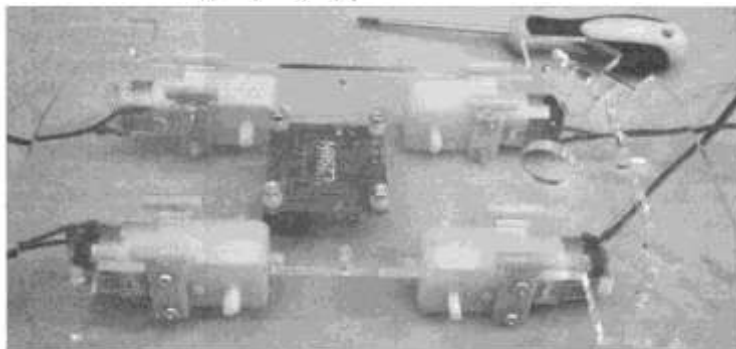


Рис. 6.15. Монтаж платы драйвера: вид снизу

Пояснения к установке и настройке платы драйвера L298N представлены на рис. 6.16.

Соблюдайте порядок подключения!

Важно соблюдать указанный на рис. 6.16 порядок подключения! При монтаже НЕЛЬЗЯ менять местами двигатели и контакты управления!

Контакты OUT1 и OUT2 подключаем к двигателям правой стороны робота, OUT3 и OUT4 — к двигателям его левой стороны. Клемма + 12V подключается к основному электропитанию робота, клемма GND («земля») — к отрицательному выводу источника питания (в дальнейшем для всех используемых приборов контакты GND должны быть объединены). Клемма +5V будет использована для электропитания платы Arduino UNO, датчиков и других маломощных, но требующих стабилизированного электропитания устройств, от нее потребуются вывести провод. Перемычки ENA и ENB включают или отключают двигатели, а также при помощи широтно-импульсной модуляции через них можно управлять скоростью вращения колес, но во втором издании книги мы их использовать не будем и оставим с установленными перемычками.

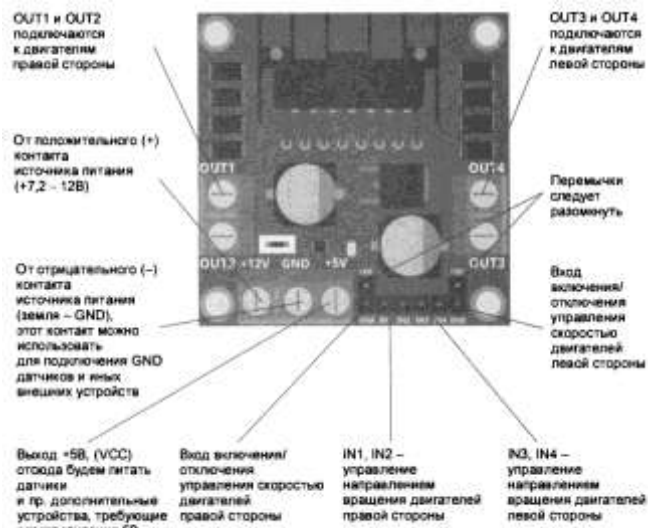


Рис. 6.16. Вид платы драйвера L298N с пояснениями

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. демонстрация собранной части робота;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №21.

Тестирование собранной колесной базы

Цель работы: научиться выполнять тестирование базовой основы робота с помощью программы, загруженной в МК.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

При установке двигателей, как показано на рис. 6.11, передний и задний двигатели каждой стороны перевернуты относительно друг друга. Это потребует перекрестного соединения проводов от них при подключении к драйверу: верхний провод переднего двигателя соединяется с нижним проводом от заднего и наоборот. При этом есть вероятность ошибки, и прежде чем приступить к сборке верхнего уровня робота, требуется провести тщательную проверку и исключить любые ошибки.

Такую проверку для уверенности лучше расписать по шагам:

- 1) Прежде всего убедитесь, что установлены перемычки ENA и ENB, разрешающие

включение двигателей.

2) Соберите схему, показанную на рис. 5.22. Но подключите к каждому выходу драйвера по паре двигателей, как только что описано. При этом контакты IN1-IN4 должны быть свободны — никаких напряжений на них подавать не надо, т. к. контакты драйвера уже шунтированы через резисторы на «землю» (GND).

3) Подайте питание от блока питания на плату драйвера: «плюс» — на + 12V, «минус» — на GND. Должен загореться светодиод.

4) Отдельным проводом (удобно использовать провод от тестера) кратковременным прикосновением подайте +5V от платы драйвера на IN 1 — правые двигатели должны начать вращаться. Если что-то пошло не так, проверьте соединения и контакты. Двигатели также должны вращаться в одном направлении. Если вращение происходит в разных направлениях — проверьте правильность объединения правых двигателей в пару и объедините правильно.

5) Кроме того, обратите внимание, что при подаче «плюса» на IN 1 колеса должны вращаться назад, а при подаче «плюса» на IN2 — вперед. Если происходит обратное — поменяйте местами провода, подводимые к драйверу исследуемой пары двигателей.

6) Так же проверьте и настройте левые двигатели. При подаче «+5V» на IN4 левые колеса должны вращаться вперед, а при подаче «плюса» на IN3 — назад.

Контакты IN1, IN2, IN3, IN4 управляют подачей электропитания на соответствующие по номерам выводы OUT1, OUT2, OUT3, OUT4. Например, подача логического нуля на IN1 подключит OUT1 на «землю» (GND), а подача логической единицы на IN2 подключит OUT2 к положительному контакту электропитания (обозначен на рис. 6.16 как + 12V) — это вызовет вращение подключенного двигателя. Если же на IN1 подать логическую единицу, а на IN2 — ноль, то вал двигателя начнет вращаться в противоположную сторону.

Провода от двигателей соединяются с платой драйвера через винтовой зажим (рис. 6.17).

Учтите также, что каждый выход L298N будет управлять сразу парой двигателей, подключенных параллельно, и, перепутав полярность, можно получить колеса, вращающиеся в разные стороны! Для соблюдения полярности рекомендую к контактным площадкам, расположенным ближе к корпусу, припаять по черному проводу, а к верхним — по белому. Контактные площадки передних и задних двигателей перевернуты относительно друг друга (это видно на рис. 6.15), поэтому следует при подключении к драйверу черный и белый провода от разных двигателей одной стороны объединить (рис. 6.18).

Порядок подключения проводов от первой пары двигателей к контактам OUT1- OUT2 и OUT3-OUT4, конечно, имеет значение. В нашем случае:

1) верхний контакт переднего правого двигателя и нижний контакт заднего правого двигателя следует подключить к OUT1;

2) нижний контакт переднего правого двигателя и верхний контакт заднего правого — к OUT2;

3) верхний контакт переднего левого двигателя и нижний контакт заднего левого двигателя следует подключить к OUT4; нижний контакт переднего левого двигателя и верхний контакт заднего левого — к OUT3.

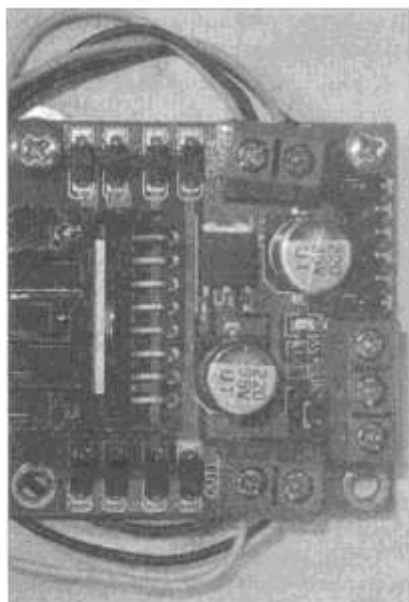


Рис. 6.17. Провода от двигателей присоединены к плате драйвера

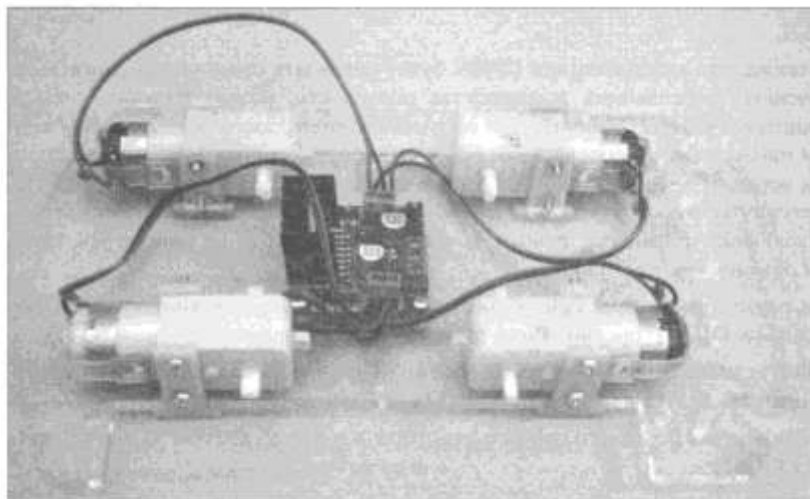


Рис. 6.18. Двигатели робота подключены к драйверу

Важно!

Обратите внимание — контактные площадки всех двигателей должны быть направлены к центру ходовой части робота, а бобышки (см. рис. 6.8) смотреть наружу.

Чтобы провода не болтались, удобно прикрепить их к корпусу клеем (клеевым пистолетом) либо стяжками.

После того, как полярность двигателей проверена, и в случае ошибки исправлена, можно приступать к сборке верхнего уровня.

Перед этим установим в контакты ENI-EN4 драйвера провода со съемными клеммами («мама-мама»), а в винтовые зажимы + 12, GND и 5V зажем провода длиной примерно 15 см (рис. 6.19).

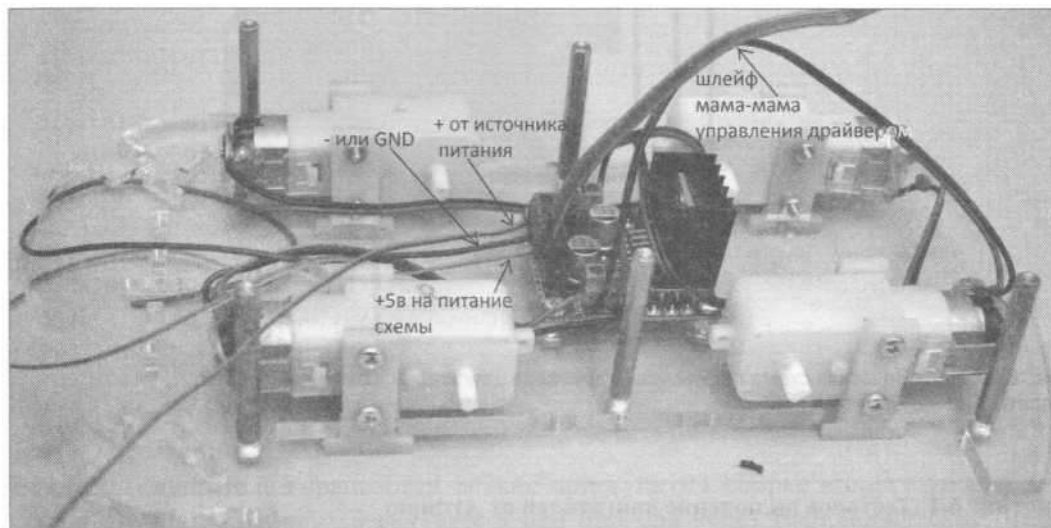


Рис. 6.19. Нижний уровень корпуса робота в сборе с установленными проводами и стойками

Задание. Протестировать работоспособность сборки, присоединив к драйверу плату Arduino по схеме, приведенной на рис. 6.20.

Для тестирования сначала присоедините провода к плате драйвера двигателей. Используйте для этого провода разного цвета. После соединения запишите для себя на бумаге, какого цвета провод к какому контакту платы драйвера подключен, — это поможет избежать путаницы. Провода можно паять или использовать клеммы. Если вы еще не очень хорошо паяете, то рекомендую использовать провода с клеммами, которые быстрее в монтаже. Присоединив провода к плате драйвера, подключите, как показано на рис. 6.20, плату Arduino и загрузите в нее программу из листинга 6.1.

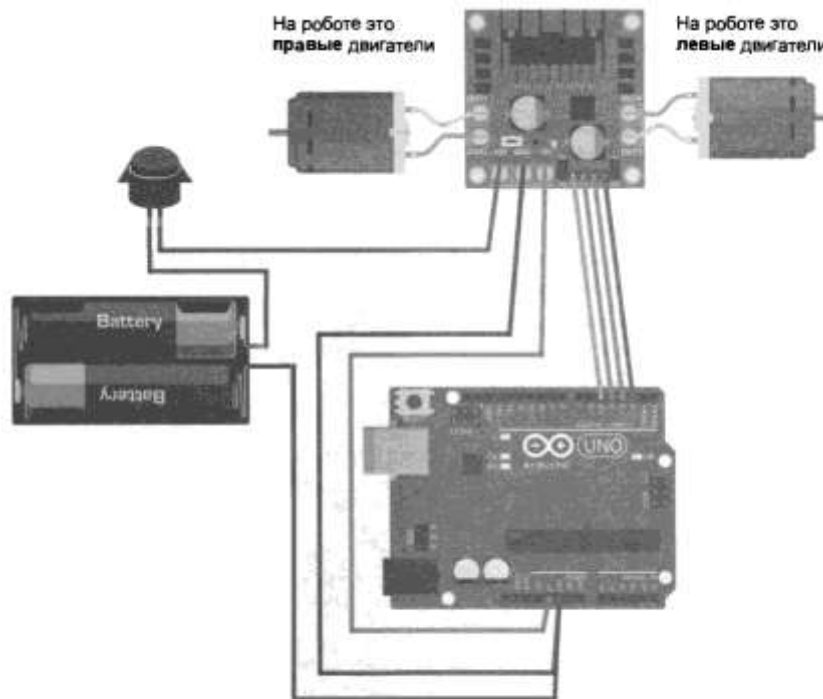


Рис. 6.20. Электрическая схема подключения драйвера двигателей, самих двигателей и платы Arduino Uno

Листинг 6.1. Тестовое включение двигателей от Arduino

//Создадим переменные для хранения номеров используемых пинов/портов Arduino.

```
int In1, In2, In3, In4;
```

```
//Настройка void setup() {
```

```
// Присвоим переменным номера пинов Arduino.
```

```
In1 = 2;
```

```
In2 = 3;
```

```
In3 = 4;
```

```
In4 = 5;
```

```
//Переведем эти пины/порты в режим вывода.
```

```
pinMode(In1, OUTPUT);
```

```
pinMode(In2, OUTPUT);
```

```
pinMode(In3, OUTPUT);
```

```
pinMode(In4, OUTPUT);
```

```
}
```

```
//Тело программы
```

```
void loop () {
```

```
//Остановить все двигатели.
```

```
digitalWrite(In1, LOW); //Правые двигатели на работе.
```

```
digitalWrite(In2, LOW);
```

```
digitalWrite(In3, LOW); //Левые двигатели на работе.
```

```
digitalWrite(In4, LOW); delay(1000); // Ждем 1 сек.
```

```
//Включить-выключить поочередно левые и правые двигатели
```

```
digitalWrite(In1, HIGH); //Выключили один правый двигатель на работе.
```

```
delay(1000); // Ждем 1 сек.
```

```
digitalWrite(In1, LOW); //Выключили
```

```
digitalWrite(In2, HIGH); //Включили другой правый двигатель на работе.
```

```
delay(1000); // Ждем 1 сек.
```

```
digitalWrite(In2, LOW); //Выключили
```

```
digitalWrite(In3, HIGH); //Включили один левый двигатель на работе.
```

```
delay(1000); // Ждем 1 сек.
```

```
digitalWrite(In3, LOW); //Выключили
```

```
digitalWrite(In4, HIGH); //Включили другой левый двигатель на работе.
```

```
delay(1000); // Ждем 1 сек.
```

```
digitalWrite(In4, LOW); //Выключили
}
}
```

Если валы двигателей вращаются, можно приступать к сборке второго уровня. Отсоедините плату Arduino, а ведущие к ней провода через технологические отверстия пластины корпуса выведите наверх, т. к. они должны будут позже подсоединены к плате Arduino, которая стационарно монтируется на втором уровне робота.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы (демонстрация правильной работы собранной части робота);
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №22. Сборка верхней части робота

Цель работы: научиться выполнять сборку верхней части робота по алгоритму.

Выполнив работу, Вы будете:

уметь:

- применять стандартные алгоритмы в соответствующих областях;
- применять нормативные документы, определяющие требования к оформлению программного кода;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Задание. Выполните сборку верхней части робота по инструкции.

Крепление верхней части корпуса осуществляется с помощью бронзовых стоек, закрепляемых на винты (см. рис. 6.19). Установим стойки, временно отложим собранную нижнюю часть робота в сторону и подготовим необходимые элементы сборки верхнего уровня (рис. 6.21).

Прикрепим плату Arduino винтами к корпусу, как показано на рис. 6.22, не забыв проложить под плату изолирующие шайбы. Закрепим бокс для аккумуляторов, воспользовавшись для этого винтами с потайной головкой (иначе потом возникнут трудности с установкой в него аккумуляторов). Вместо винтов для крепления бокса можно использовать двусторонний скотч.

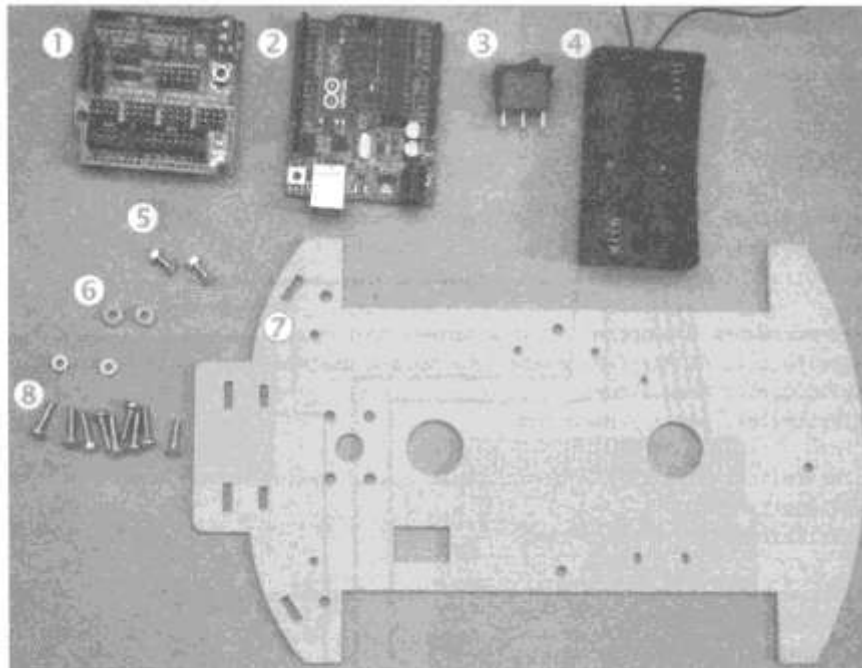


Рис. 6.21. Минимальный набор для верхнего уровня робота: плата Arduino Sensor Shield v5.0 (1); плата Arduino Uno (2); выключатель (3); бокс для аккумуляторов (4); 3-мм винты с потайными головками (5); изолирующие шайбы (6); верх корпуса (7); 3-мм винты и гайки для крепления платы Arduino и корпуса к стойкам (8)

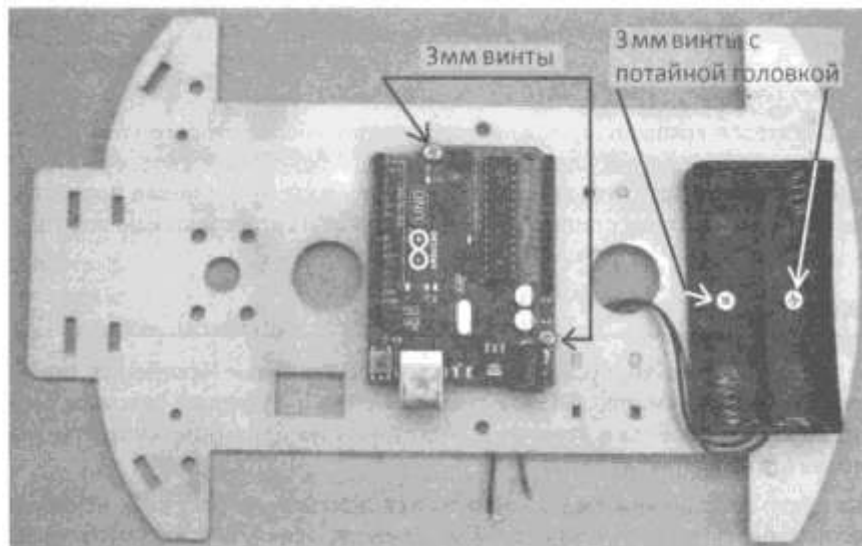


Рис. 6.22. Этап установки платы Arduino и бокса для аккумуляторов

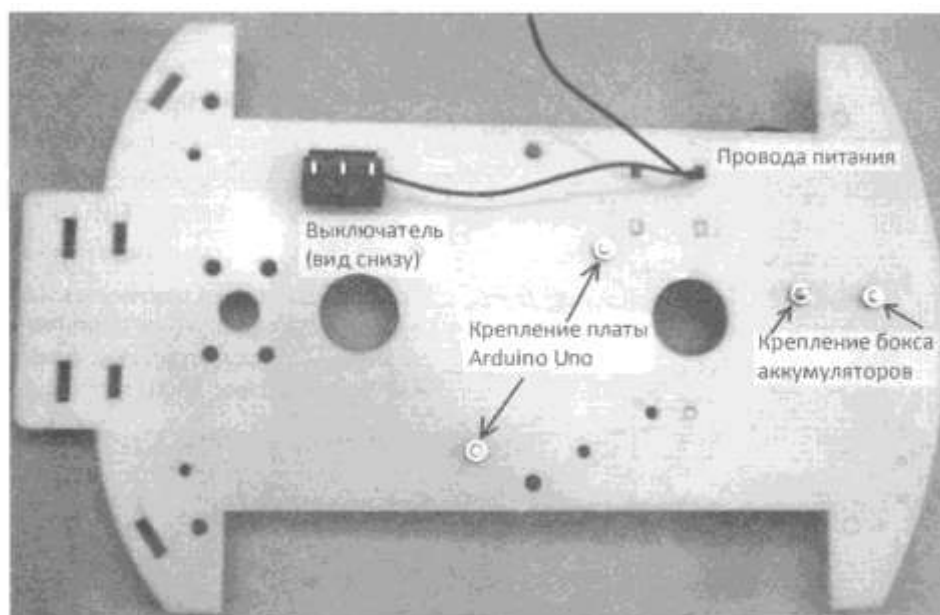


Рис. 6.23. Верх корпуса с установленным выключателем, боксом и Arduino Uno (вид снизу)

На следующем этапе нужно установить выключатель в прямоугольный паз (рис. 6.23). Выключатель имеет подпружиненный механизм-защелку и не будет выпадать. Если ваш выключатель трехконтактный, то средний контакт— общий, он замыкается с левым или правым контактом в зависимости от положения выключателя. Для включения/выключения робота потребуются два контакта: средний и один из крайних. Настоятельно рекомендую припаять провода питания к выключателю и только при невозможности пайки использовать скрутку. На рис. 6.24 правый контакт присоединен к красному проводу (+) от бокса аккумуляторов, а средний — к проводу, который мы ранее присоединили к клемме 12V драйвера. Шлейф от контактов EN1-EN4 драйвера и провода, присоединенные к GND и 5V, следует продеть в технологические отверстия верха корпуса для использования на втором уровне.

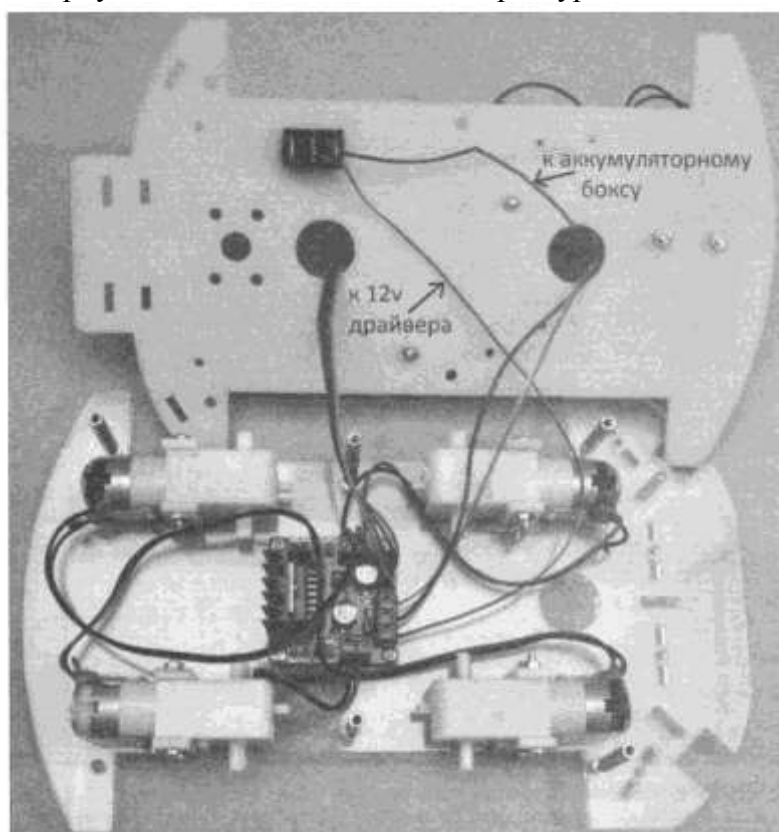


Рис. 6.24. Подготовка верхней части корпуса робота к установке (прикреплены и выведены провода)

Установим верхнюю часть корпуса на стойки нижней части и закрепим их винтами (рис.

6.25). Далее осторожно, чтобы не повредить ножки, установим плату Arduino Sensor Shield v5.0 на Arduino Uno. Подсоединим выведенные вверх провода к Sensor Shield.

Стабилизатор +5 В на плате Arduino Uno недостаточно мощный, чтобы питать от него все датчики и серводвигатель «головы» робота. Поэтому поступим следующим образом. Нестабилизированное напряжение 7-12 В от аккумуляторов подадим через выключатель на вход драйвера двигателей (см. рис. 6.20). В драйвере двигателей есть существенно более мощный стабилизатор на +5 В. От него и будем запитывать плату Arduino Uno, плату Sensor Shield, а через нее сервомотор «головы» и все внешние датчики робота.

На плате Sensor Shield (рис. 6.25):

- + джампер SEL платы оставляем замкнутым;
- + под винтовой разъем GND зажимаем отрицательный провод от аккумуляторов и GND от драйвера двигателей (средний провод);
- + под винтовой разъем VCC зажимаем провод от 5V драйвера двигателей.

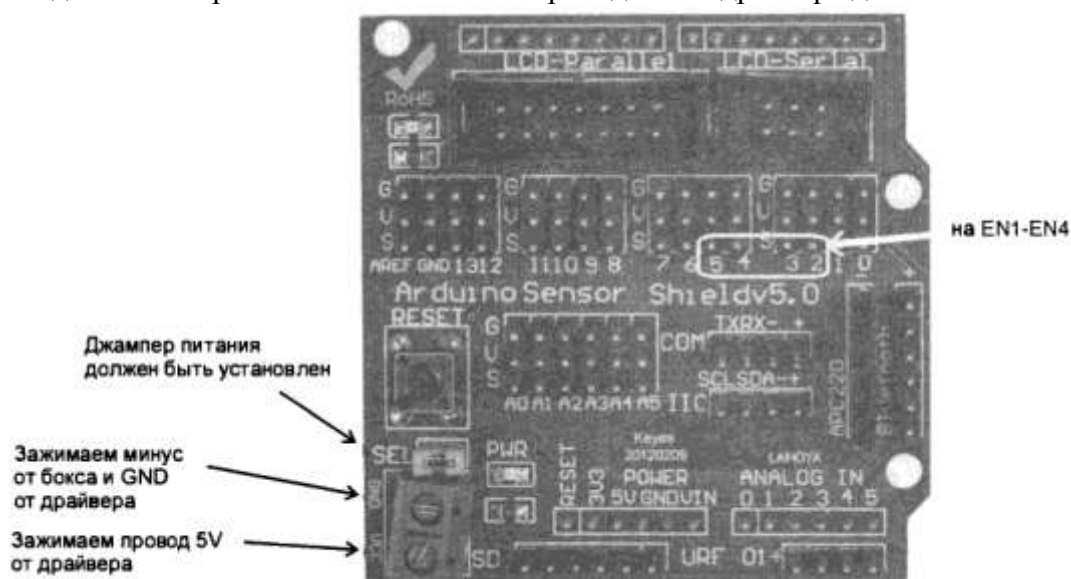


Рис. 6.25. Плата Arduino Sensor Shield v5.0 с пояснениями по закреплению проводов

Электрическая схема подключения драйвера двигателей, самих двигателей и платы Arduino Uno была приведена на рис. 6.20. Общий вид всех соединений приведен на рис. 6.24. Плата Sensor Shield не показана, чтобы не усложнять схему. Корпус робота в сборе (пока еще без колес) показан на рис. 6.26.

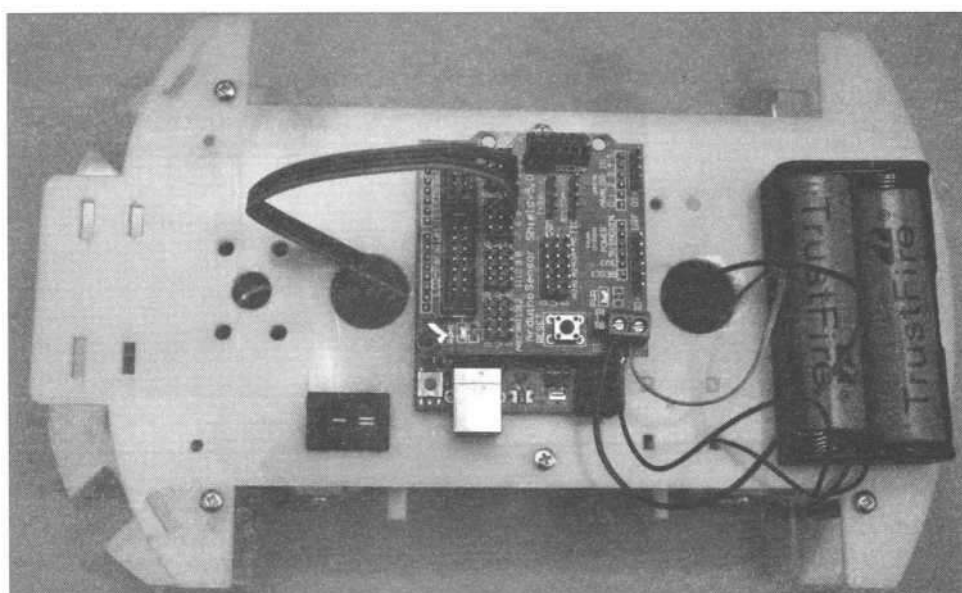


Рис. 6.26. Корпус робота в сборе (осталось установить колеса)

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы (демонстрация собранной части робота);
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №23.

Подключение устройств обратной связи

Цель работы: научиться подключать устройства обратной связи, а также управлять ими с помощью МК.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

- 1.Компьютер с лицензионным программным обеспечением для программирования МК.
- 2.Мультимедиа проектор.
- 3.Робототехнический набор

Теоретические сведения.

Рассмотрим, как можно дать роботу возможность подавать сигналы в ответ на определенные события — например, начало поворота, остановку и т. д. Для этого можно применить светодиод или зуммер, присоединенные к контактам (D2-D 13 или A0-A3) платы Arduino робота.

Таким образом можно оснастить робот простейшими устройствами, которые помогут сигнализировать о его состоянии. Такими устройствами могут быть светодиод и пьезозуммер – световая и звуковая индикация.

Светодиод (рис. 6.27) светится, если через него протекает ток от анода к катоду. Значит, для его использования на анод светодиода следует подать положительное напряжение, а катод заземлить (на минус). При этом светодиод будет нормально светиться лишь тогда, когда к нему приложено напряжение, соответствующее номинальному (от 1,4 до 3 вольт— зависит от типа), если же напряжение будет больше, он очень быстро выйдет из строя. Для того чтобы светодиод светил долго, применяется токоограничивающий резистор.

Простейшая схема соединения светодиода показана на рис. 6.28. Длинная ножка светодиода — анод, короткая — катод. Длинная ножка через резистор сопротивлением 200 Ом будет подключена к контакту D6 платы Arduino, а короткая ножка посажена на «землю» (GND). Обратите внимание, что если поменять ножки светодиода местами, он не будет излучать свет.

Схема подключения зуммера представлена на (рис. 6.31).

Укладка проводов

Когда к роботу полностью подведено питание и управление, он скрывается под ворохом незакрепленных проводов. Болтающиеся провода, как правило, за что-нибудь цепляются, что приводит к их отсоединению или обрыву. Во избежание таких проблем провода нужно объединять и укладывать. Это удобно делать при помощи маленьких стяжек или изоленты. Можно также использовать термоклей — для этого на корпус переносится капля термоклея, а затем в нее утапливаются провода. Если потом провода потребуется освободить, то термоклей нужно нагреть до 200 градусов, например, феном.

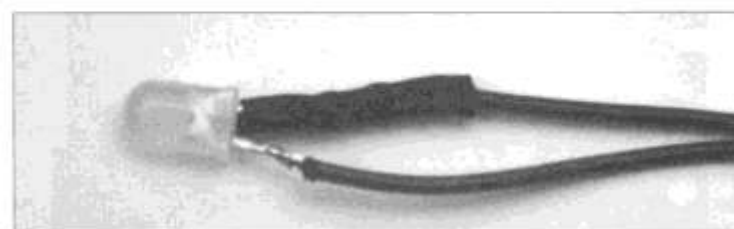
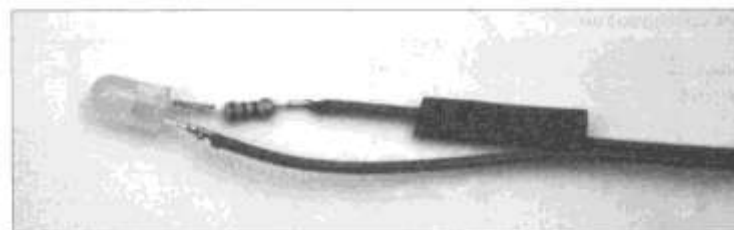
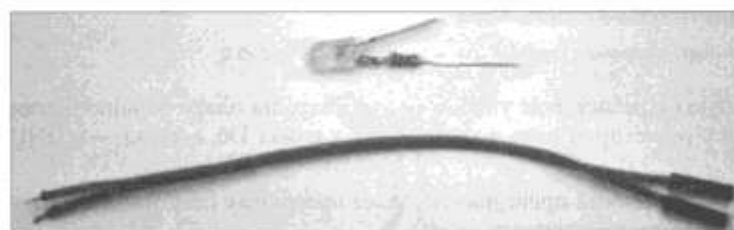
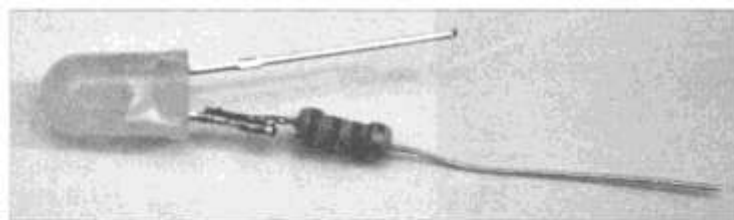
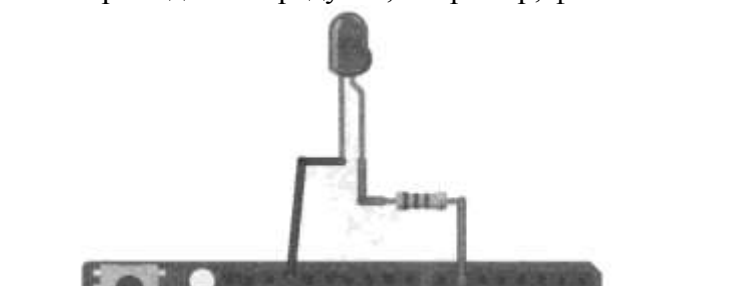
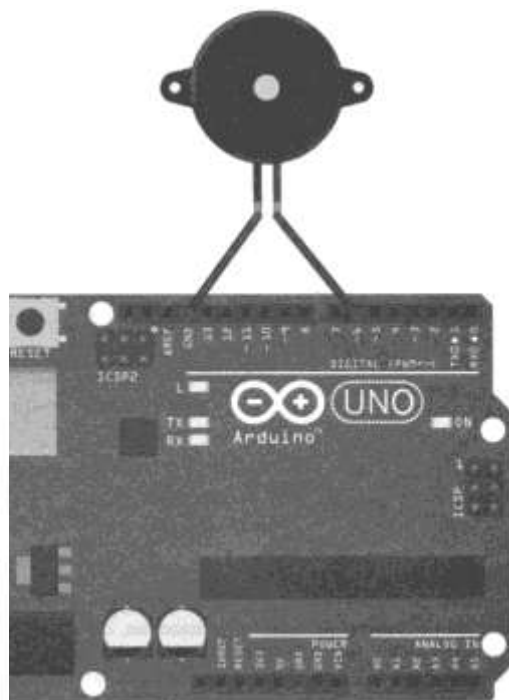


Рис. 6.29. Пайка светодиода к проводам



Подключение зуммера к Arduino

Задание. Выполните подключение устройств световой и звуковой индикации робота согласно инструкции.

Для подключения светодиода к проводам используют пайку: нужно обрезать длинную ножку и подпаять к ней резистор сопротивлением 200 Ом (рис. 6.29, а), затем подготовить провода с клеммами «мама» с одной стороны, а с другой стороны клеммы отрезать, зачистить и залудить кончики (рис. 6.29, б), далее подпаять провода к светодиоду (рис. 6.29, в) и

изолировать контакт с резистором, надев на него термотрубку — она сначала широкая, но после нагрева сильно стягивается, что очень удобно (рис. 6.29, г).

Полученный светодиод с резистором удобно монтировать на плату Arduino Sensor Shield v5.0: контакт с резистором надо подсоединить к ножке D6, а катод — к GND (рис. 6.30).

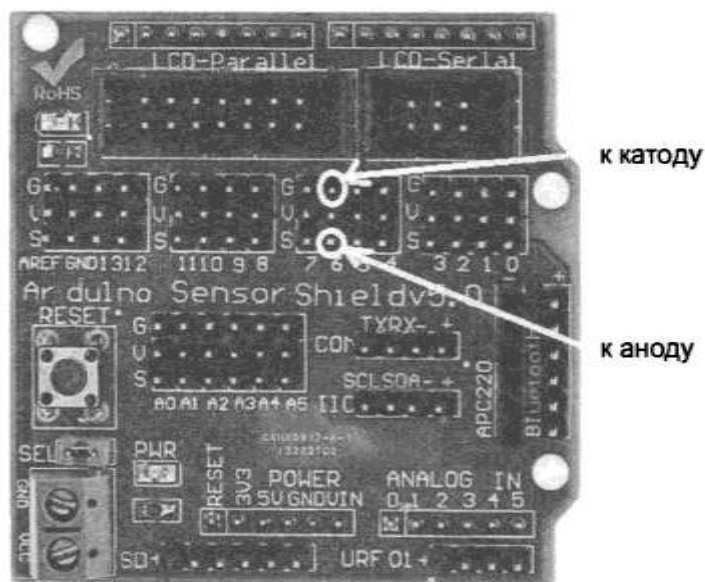


Рис. 6.30. Подключение подготовленного светодиода к Arduino Sensor Shield v5.0

В результате выполнения работы должно быть получено готовое устройство:

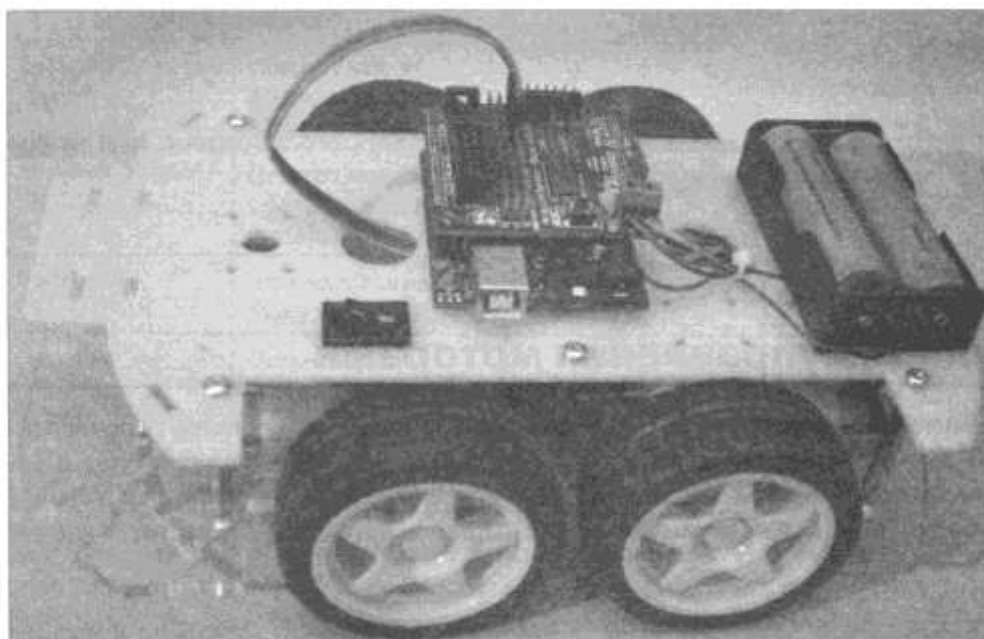


Рис. 6.32. Базовая модель робота в сборе

После окончания сборки робота следует сделать его контрольный осмотр, проверить, нет ли оторванных проводов, надеть колеса (рис. 6.32) и приступить к созданию первой программы, которая научит робота пользоваться моторами, совершать прямолинейное движение и повороты.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы (демонстрация собранного робота);
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №24.

Тестирование электронных компонентов робота

Цель работы: научиться выполнять тестирование электронных компонентов робота с помощью программы, загруженной в МК.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Задание. Выполните тестирование устройств световой и звуковой индикации, установленных на роботе.

1. Для **проверки работы светодиода**, загрузите в робота приведенную программу (листинг 6.2). Алгоритм работы: Светодиод 10 раз мигает, затем на 1 секунду гаснет.

Листинг 6.2. Мигаем светодиодом

//Создадим переменные для хранения номеров используемых пинов/портов Arduino.

```
int In1, In2, In3, In4, InDiod;
```

```
//Настройка void setup() {
```

```
// Присвоим переменным номера пинов Arduino.
```

```
InDiod = 6;
```

```
pinMode(InDiod, OUTPUT);
```

```
digitalWrite(InDiod, HIGH);
```

```
In1 = 2;
```

```
In2 = 3;
```

```
In3 = 4;
```

```

In4 = 5;
//Переведем эти пины/порты в режим вывода.
pinMode(In1, OUTPUT);
pinMode(In2, OUTPUT);
pinMode(In3, OUTPUT);
pinMode(In4, OUTPUT);
}
//Тело программы
void loop () {
for (int i = 0; i < 10; i++)
(
digitalWrite(InDiod, HIGH);
delay (200) ;
digitalWrite(InDiod, LOW);
delay(200);
}
delay(1000) ;
}

```

- Для **тестирования работы зуммера** можно использовать программу из листинга 6.3. Контакт зуммера (в данном случае это ножка 6) переводим в режим вывода, а затем командой `tone` (номер пина, частота) заставляем его издавать писк. Команда `noTone` (номер пина) отменяет звук. Зуммер, прерываясь, гудит на частоте 1000 Гц в течение 2 секунд, затем на 1 секунду замолкает.

Листинг 6.3. Тестирование зуммера

//Создадим переменные для хранения номеров используемых пинов/портов Arduino.

```

int In1,
In2,
In3,
In4,
InZ;
//Настройка
void setup() {
// Присвоим переменным номера пинов Arduino.
InZ = 6;
pinMode (InZ, OUTPUT);
In1 = 2;
In2 = 3;
In3 = 4;
In4 = 5;
// Переведем эти пины/порты в режим вывода.
pinMode(In1, OUTPUT);
pinMode(In2, OUTPUT);
pinMode(In3, OUTPUT);
pinMode(In4, OUTPUT);
}
//Тело программы
void loop() {
for (int i =0; i < 10; i++)
{
tone(InZ, 1000);
delay(100) ;
noTone(InZ);
delay(100);
}
}

```



```
}  
delay(1000);  
}
```

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №25.

Программирование робота на заданное движение

Цель работы: научиться разрабатывать программу управления роботом, обеспечивающую заданное движение робота.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации-производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Переменные и функции управления моторами

Для управления моторами надо включить в программу четыре целочисленные глобальные переменные (табл. 7.1)— они будут отвечать за включение и отключение моторов, а также за

направление вращения колес. Значения, записанные в них, будут являться номерами выводов Arduino, которыми потребуется управлять.

Таблица 7.1. Переменные управления моторами

Переменная	Назначение: номер вывода Arduino — контакт драйвера
motor_L1	2 — IN4
motor_L2	3 — IN3
motor_R1	4 — IN1
motor_R2	5 — IN2

Создадим также базовую функцию `setup_motor_system` — для присвоения глобальным переменным значений с номерами портов Arduino и перевода используемых портов в режим вывода данных.

Функции движений

Для управления роботом нам также потребуются функции, управляющие его движением. Дадим им названия и закрепим за ними определенные действия (табл. 7.2).

Таблица 7.2. Функции движений

Функция	Назначение	Выставляемые значения на портах			
		motor_L1	motor_L2	motor_R1	motor_R2
<code>forward()</code>	Включает движение вперед	1	0	1	0
<code>forward_left()</code>	Поворот налево с блокировкой левых колес	0	0	1	0
<code>forward_right()</code>	Поворот направо с блокировкой правых колес	1	0	0	0
<code>backward()</code>	Включает движение назад	0	1	0	1
<code>_stop()</code>	Остановка	0	0	0	0

Первая поездка

Для проверки работоспособности собранного робота нужно записать в него простую программу с определенной последовательностью движений. Созданные нами функции только включают моторы с заданными параметрами, но не позволяют управлять длительностью процесса. Поэтому нам следует позаботиться о том, чтобы после включения какой-либо функции, — например, `forward()`, прошло некоторое время до включения следующей. За это время робот, обрабатывая поданную на него команду, каким-то образом сместится. Естественно, что от времени, в течение которого выполняется движение, зависит величина самого движения: если это поворот, то увеличение времени увеличит угол поворота, а если это движение вперед, то увеличится пройденное расстояние.

Алгоритм

Рис. 7.1 Демонстрирует небольшой линейный (в нем нет условных операторов) алгоритм тестовых движений робота. Согласно этому алгоритму, робот начинает двигаться, совершает повороты и проезды, а после выполнения всех команд начинает все сначала.



Рис. 7.1. Линейный алгоритм тестовых движений робота

Задание 1 . В соответствии с приведенным на рис. 7.1 алгоритмом, создайте скетч с тестовой программой (листинг 7.1), которая осуществляет в цикле несколько видов движений. Ее цель — проверить, что все сделано правильно, и колеса вращаются в нужном направлении.

Листинг 7.1. Тестовая программа движений робота

```

// Объявляем переменные для хранения состояния двух моторов
int motor_L1, motor_L2;
int motor_R1, motor_R2;
// Функция инициализации управления моторками.
void setup_motor_system(int L1, int L2, int R1, int R2)
{
// Заносятся в переменные номера контактов (пинов) Arduino
// Для левых
motor_L1=L1;
motor_L2=L2;
// и правых моторов робота.
motor_R1=R1;
motor_R2=R2;
// Переводятся указанные порты в состояние вывода данных.
pinMode (motor_L1, OUTPUT);
pinMode (motor_L2, OUTPUT);
pinMode(motor_R1, OUTPUT);
pinMode(motor_R2, OUTPUT);
}
// движение вперед.
void forward()
{
// Если двигатель будет работать не в ту сторону,
// поменять на нем контакты местами.

```

Если ваш робот совершает повороты и едет не так, как задумано в программе, не спешите его разбирать, включите в программе все команды, кроме движения вперед, и проверьте, какие колеса вращаются неправильно. Для этих колес поменяйте местами контакты на плате Arduino Sensor Shield, а если это проблематично, то измените значения переменных `motor_L1`, `motor_L2`, `motor_R1`, `motor_R2`.

Например, если колеса правой стороны вращаются в нужную сторону, а левой — в обратную, то следует поменять местами контакты на выводах 2 и 3, если же наоборот, то на выводах 5 и 4.

```

digitalWrite(motor_L1, HIGH);
digitalWrite(motor_L2,LOW);
digitalWrite(motor_R1, HIGH);
digitalWrite(motor_R2,LOW);
}
// Поворот налево с блокировкой левых колес.
void forward_left()
{
// блокировка вращения левых колес.
digitalWrite(motor_L1, LOW);
digitalWrite(motor_L2, LOW) ;
// правые колеса вращаются.
digitalWrite(motor_R1, HIGH);
digitalWrite(motor_R2, LOW);
}
// Поворот направо с блокировкой правых колес.
void forward_right()
// левые колеса вращаются.
digitalWrite(motor_L1, HIGH);
digitalWrite(motor_L2, LOW) ;
// блокировка вращения правых колес.
digitalWrite(motor_R1, LOW);
digitalWrite(motor_R2, LOW);
}
// Включаем движение назад.
void backward()
{
// Смена направления вращения двигателей.
digitalWrite(motor_L2, HIGH);
digitalWrite(motor_L1, LOW);
digitalWrite(motor_R2, HIGH);
digitalWrite(motor_R1, LOW);
}
void _stop()
(
// Блокировка всех колес.
digitalWrite(motor_L2, LOW);
digitalWrite(motor_L1,LOW);
digitalWrite(motor_R2, LOW);
digitalWrite(motor_R1, LOW);
}
// Функция инициализации, выполняется один раз.
void setup ()
{
// Переменные - номера контактов (пинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2,3,4,5);
// Двигатели остановлены.
stop();
}
// Основная программа.
void loop ()
{
// Пример движения робота задан жестко в программе

```

```

// и повторяется в цикле. forward(); /
/ едет вперед 1 секунду. delay(1000);
forward_left();
// поворачивает налево 0,5 секунд. delay(500);
forward(); // едет вперед 0,5 секунд. delay(500);
forward_right(); // поворачивает направо 0,5 секунд. delay(500);
__ stop();
delay(500);
backward(); // движется назад 0,8 секунд. delay(800);
}

```

Задание 2. Разделите программу на два файла, разметив функции управления моторами и тестовую езду в различные скетчи.

Все функции управления моторами (листинг 7.2, а) поместим в файл motor.h (он должен быть создан в папке с текущей программой) и подключим его к основной программе инструкцией #include "motor.h". Файл этот можно создать в редакторе Блокнот.

Листинг 7.2, а. *Все функции управления моторами.* Вспомогательный файл motor.h

// Объявляем переменные для хранения состояния двух моторов.

```
int motor_L1, motor_L2;
```

```
int motor_R1, motor_R2;
```

// Функция инициализации управления моторками.

```
void setup_motor_system(int L1, int L2, int R1, int R2)
```

```
{
```

// Заносятся в переменные номера контактов (пинов) Arduino.

```
motor_L1=L1;
```

```
motor_L2=L2;
```

// Для левых и правых моторов работа.

```
motor_R1=R1;
```

```
motor_R2=R2;
```

// Переводятся указанные порты в состояние вывода данных.

```
pinMode (motor_L1, OUTPUT);
```

```
pinMode (motor_L2, OUTPUT);
```

```
pinMode(motor_R1, OUTPUT);
```

```
pinMode (motor_R2, OUTPUT);
```

```
}
```

// движение вперед. void forward()

```
{
```

// Если двигатель будет работать не в ту сторону,

// поменять на нем контакты местами.

```
digitalWrite(motor_L1, HIGH);
```

```
digitalWrite(motor_L2,LOW);
```

```
digitalWrite(motor_R1, HIGH);
```

```
digitalWrite(motor_R2,LOW);
```

// Поворот налево с блокировкой левых колес.

```
void forward_left()
```

```
{
```

// блокировка вращения левых колес.

```
digitalWrite(motor_L1, LOW);
```

```
digitalWrite(motor_L2, LOW);
```

// правые колеса вращаются.

```
digitalWrite(motor_R1, HIGH);
```

```
digitalWrite(motor_R2, LOW);
```

```
}
```

// Поворот направо с блокировкой правых колес. void forward_right()

```
{
```

```

// левые колеса вращаются.
digitalWrite(motor_L1, HIGH);
digitalWrite(motor_L2, LOW) ;
// блокировка вращения правых колес.
digitalWrite(motor_R1, LOW);
digitalWrite(motor_R2, LOW);
}
// Вешаем движение назад. void backward()
{
//Смена направления вращения двигателей.
digitalWrite(motor_L2, HIGH);
digitalWrite(motor_L1,LOW);
digitalWrite(motor_R2, HIGH);
digitalWrite(motor_R1,LOW);
}
void _stop()
{
// Блокировка всех колес.
digitalWrite(motor_L2, LOW);
digitalWrite(motor_L1, LOW);
digitalWrite(motor_R2, LOW);
digitalWrite(motor_R1,LOW);
}

```

В этом случае основная программа будет очень короткой (листинг 7.2, б).

Листинг 7.2, б. *Тестовая программа движений робота.*

```

Основной файл программы listng_7_2.ino
#include "motor.h" // Подключаем все функции управления моторами
// Функция инициализации, выполняется один раз.
void setup ()
{
// Переменные - номера контактов (пинов) Arduino.
// Для левых и правых моторов машинки. setup_motor_system(2,3, 5, 4) ;
// Двигатели остановлены.
stop();
)
// Основная программа. void loop ()
{
// Пример движения робота задан жестко в программе
// и повторяется в цикле.
forward();
// едет вперед 1 секунду.
delay(1000);
forward_left();
// поворачивает налево 0,5 секунд.
delay(500);
forward();
// едет вперед 0,5 секунд.
delay(500);
forward_right();
// поворачивает направо 0,5 секунд.
delay(500);
stop();
delay(500);
}

```

```
backward();
// движется назад 0,8 секунд.
delay(800);
}
```

Как теперь будет выглядеть наша программа в среде Arduino IDE, показано на рис. 7.2.



Рис. 7.2. Демонстрация деления программы на два файла: а — integ_7_2 и б — motor.h

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №26.

Программирование робота на дополнительные действия

Цель работы: научиться разрабатывать программу управления роботом, обеспечивающую индикацию состояний робота.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации-производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;

- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

- 1.Компьютер с лицензионным программным обеспечением для программирования МК.
- 2.Мультимедиа проектор.
- 3.Робототехнический набор

Задание.

Модифицируйте основную программу таким образом, чтобы при остановке загорался светодиод, подключенный к контроллеру Arduino (установлен на управление от 6-го контакта МК). Для этого нужно перевести контакт 6 в режим вывода и подавать на него 1, чтобы светодиод загорелся, и 0 — чтобы погас (листинг 7.3).

Листинг 7.3. Робот движется и сигнализирует светодиодом

```
#include "motor.h"
int diod_pin = 6;
// Функция инициализации, выполняется один раз.
void setup()
{
// Переменные - номера контактов (пинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2, 3, 5, 4);
//переводим пин светодиода в режим вывода.
pinMode(diod_pin, OUTPUT);
//гасим светодиод.
digitalWrite(diod_pin, 0) ;
// Двигатели остановлены.
stop();
}
// Основная программа.
void loop()
{
// Пример движения робота задан жестко в программе
// и повторяется в цикле.
forward(); // едет вперед 1 секунду.
delay(1000) ;
forward_left(); // поворачивает налево 0,5 секунд.
delay(500) ;
forward(); // едет вперед 0,5 секунд.
delay(500);
forward_right(); // поворачивает направо 0,5 секунд.
delay(500);
//Выключаем светодиод. digitalWrite(diod_pin, 1);
```



```

    stop();
    delay(500);
    //Гасим светодиод.
    digitalWrite(diod_pin, 0);
    backward();
    // движется назад 0,8 секунд.
    delay(800);
}

```

Самостоятельно добавьте в программу индикацию короткими звуковыми сигналами.

Таким образом, робот научился исполнять команды, позволяющие ему двигаться и поворачивать, а также сигнализировать при определенных ситуациях. Таким образом, нами создана универсальная программа, на основании которой довольно просто сделать программу движений любой сложности, чередуя этапы включения определенных двигательных функций и времени, в течение которого они работают.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №27.

Управление роботом по каналу инфракрасной связи

Цель работы: научиться разрабатывать программу управления роботом по каналу инфракрасной связи.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;

-понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Управление по IR-каналу самое простое, и реализовать его можно, имея только IR-приемник (рис. 8.2), а в качестве пульта управления воспользоваться любым IR-пультом — например, от телевизора. Но IR-прием работает только в пределах прямой видимости, что делает управление роботом не очень эффективным.

Готовый набор IR-компонентов для использования с Arduino-роботами представлен на рис.



Рис. 8.3. IR-пульт (слева), плата IR-приемника (справа сверху), соединительные провода (справа внизу)

8.3. Подойдет нам также и любой инфракрасный пульт от телевизора или другой бытовой техники.

Для приема инфракрасного сигнала требуется IR-приемник. Он может быть смонтирован на плате (рис. 8.4, а)— в этом случае у нас будет возможность визуально следить за приемом сигнала по миганию индикатора. Можно также использовать IR-приемник без платы (рис. 8.4, б). Следует обратить внимание, что контакты платы IR-приемника и контакты IR-приемника без платы не совпадают.

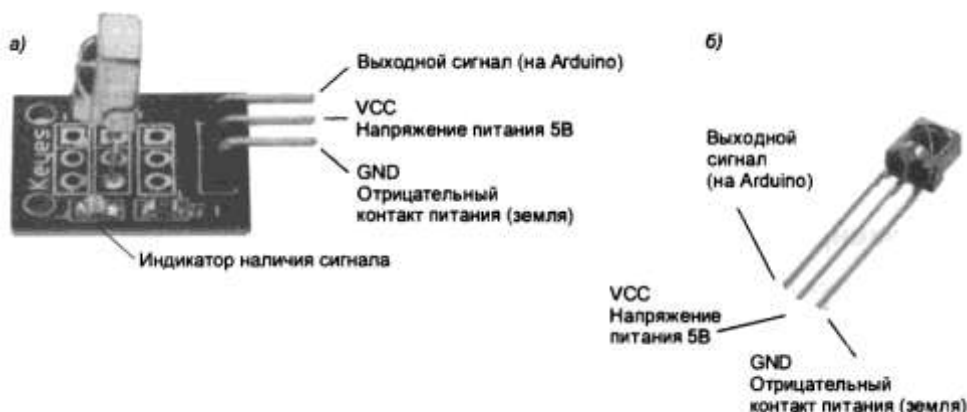


Рис. 8.4. а — плата IR-приемника; б — IR-приемник без платы

Схема подключения IR-приемника, уже установленного на плате, приведен на рис. 8.5. Питание подано от платы Arduino, но можно подать его и от другого источника стабилизированного электропитания 5 В. На плате приемника (см. рис. 8.4, а) левый контакт обозначен символом «-» — это «земля» (GND), правый контакт обозначен символом «S» — это сигнальный контакт (Data), средний контакт на плате не обозначен — это напряжение питания (Vcc = 5 В).

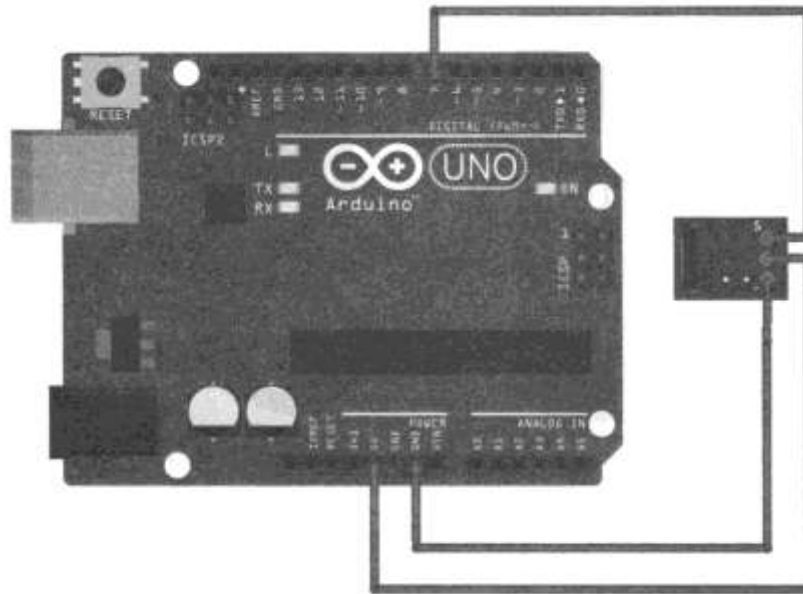


Рис. 8.5. Подключение IR-приемника к Arduino UNO

Располагаться IR-приемник должен таким образом, чтобы быть в прямой видимости от пульта управления. На роботе это место не должно ничем перекрываться. Общий вид робота в сборе, оснащенного IR-приемником, приведен на рис. 8.6.

В случае использования IR-приемника без платы рекомендуется для уменьшения помех и ложного срабатывания установить дополнительный фильтр питания (рис. 8.7).



Рис. 8.6. Робот, оснащенный IR-приемником

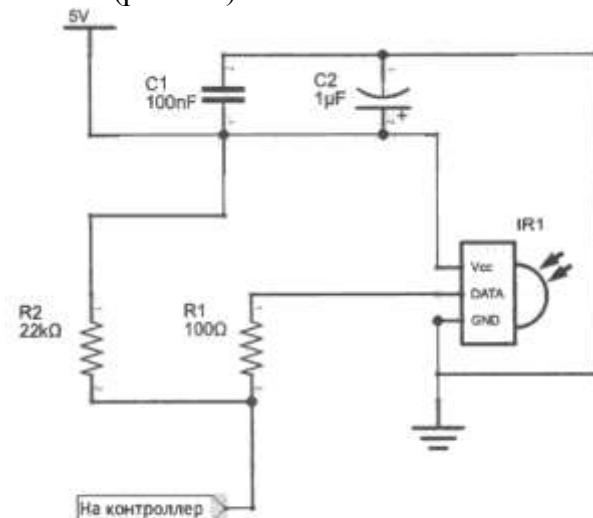


Рис. 8.7. Фильтр помех для IR-приемника

С Arduino IDE уже поставляется библиотека для работы с IR-приемником, но она не содержит примеров. Чтобы посмотреть и попробовать типичные примеры работы с IR-приемником, нужно установить расширенную библиотеку. Для этого следует выполнить ряд действий:

Зайти на компьютере в папку, куда установлена программа Arduino (обычно это C:\Program Files (x86)\Arduino или C:\Program Files\Wduino).

В этой папке зайти в папку Libraries и удалить из нее папку RobotIRremote.

С сайта <https://github.com/z3t0/Arduino-roremote> скачать ZIP-архив, содержащий обновленную библиотеку IRRemote.h.

Через меню Arduino IDE Скetch / Подключить подключить полученную библиотеку.

Если примеры работы с IR-приемником вам не интересны, можно использовать и встроенную библиотеку. При этом следует удалить файлы IRemoteTools.cpp и IRemoteTools.h из папки Libraries\RobotIRremote\src, иначе компиляция программы заканчивается ошибкой.

После установки расширенной библиотеки у вас появятся примеры для управления роботом с помощью IR-пульта.

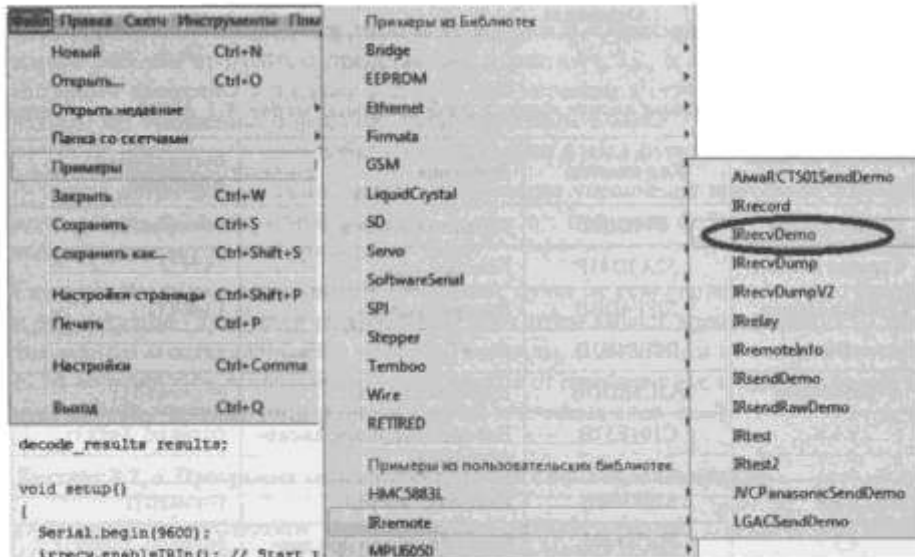


Рис. 8.8. Выбор примера программы для тестирования IR-приемника

Выберите пример IRrecvDemo (рис. 8.8) — этот пример также приведен в листинге 8.1. Проверьте, к какому контакту у вас подключена сигнальная нога IR-приемника (в примере на рис. 8.5 это контакт 7). Теперь нужно запустить программу, открыть порт для просмотра кодов кнопок пульта и дополнить полученными кодами табл. 8.1, содержащую перечень функций движения робота и их назначение. Для пульта, изображенного на

рис. 8.3, коды кнопок приведены в табл. 8.2.

Задание. Создайте скетч программы ИК управления роботом, предварительно получив коды кнопок пульта

Листинг 8.1. Проверка кодов кнопок пульта

```
// Подключение библиотеки IRremote.h.
#include <IRremote.h>
// Подключаем глобально библиотеку из Arduino IDE.
#include "IRremote.h"
// Подключаем локально библиотеку из текущего каталога
// Порт Arduino для приемника. int RECV_PIN = 7;
// Создаем объект IR-приемник.
IRrecv irrecv(RECV_PIN);
// Создаем структуру результата приема данных IR-канала.
decode_results results; void setup()
// Устанавливаем скорость порта связи с ПК.
Serial.begin(9600);
// Запуск IR-приемника.
irrecv.enableIRIn();
}
void loop() {
// Если пришли данные.
if (irrecv.decode(&results))
{
// Послать полученные данные на ПК в 16-м представлении.
Serial.println(results.value, HEX);
// Готов к приему. irrecv.resume();
}
delay(100);
}
```

Таблица 8.1. Функции движения робота и их назначение

№ п.п.	Функция	Назначение
1.	<code>forward()</code>	Движение вперед
2.	<code>forward_left()</code>	Поворот налево с блокировкой левых колес. Поворот применяется при движении вперед, при переключении от движения вперед к этому виду поворота не требуется остановка
3.	<code>forward_right()</code>	Поворот направо с блокировкой правых колес. Поворот применяется при движении робота вперед, при переключении от движения вперед к этому виду поворота не требуется остановка
4.	<code>left()</code>	Поворот налево на месте. Правые колеса вращаются вперед, левые назад
5.	<code>right()</code>	Поворот направо на месте. Правые колеса вращаются назад, левые вперед
6.	<code>backward()</code>	Движение назад
7.	<code>backward_left()</code>	Поворот налево с блокировкой левых колес. Поворот применяется при движении задним ходом, при переключении от движения назад к этому виду поворота не требуется остановка
8.	<code>backward_right()</code>	Поворот направо с блокировкой правых колес. Поворот применяется при движении задним ходом, при переключении от движения назад к этому виду поворота не требуется остановка
9.	<code>_stop()</code>	Остановка

Таблица 8.2. Соответствие кнопок пульта, изображенного на рис. 8.3, действиям робота

Название кнопки пульта	Код кнопки	Действие	Вызываемая функция
Стрелка вверх	511DBB	Двигаться вперед	<code>forward()</code>
Стрелка влево	52A3D41F	Поворот влево	<code>left()</code>
Стрелка вправо	20FE4DBB	Поворот вправо	<code>right()</code>
«Ок»	D7E84B1B	Стоп	<code>_stop()</code>
Стрелка вниз	A3C8EDDB	Двигаться назад	<code>backward()</code>
« 1 »	C101E57B	Поворот влево при движении вперед	<code>forward_left()</code>
« 2 »	97483BFB	Двигаться вперед	<code>forward()</code>
« 3 »	F0C41643	Поворот вправо при движении вперед	<code>forward_right()</code>
« 4 »	9716BE3F	Поворот влево	<code>left()</code>
« 5 »	3D9AE3F7	Стоп	<code>_stop()</code>
« 6 »	6182021B	Поворот вправо	<code>right()</code>
« 7 »	8C22657B	Поворот влево при движении назад	<code>backward_left()</code>
« 8 »	488F3CBV	Двигаться назад	<code>backward()</code>
« 9 »	449E79F	Поворот вправо при движении вперед	<code>backward_right()</code>
« 0 »	1BC0157B	Стоп	<code>_stop()</code>
« * »	32C6FDF7	Стоп	<code>_stop()</code>
« # »	3EC3FC1B	Стоп	<code>_stop()</code>

Программа для робота предусматривает, что данные принимаются с ИР-датчика, коды, генерируемые пультом при нажатии той или иной кнопки, взяты из табл. 8.2, принятые коды анализируются на совпадение, и при совпадении запускается та или иная функция (см. табл. 8.1).

Текст программы приведен в листингах 8.2, а и б. Обратите внимание, что содержимое рабочей программы представлено в листинге 8.2, а, а функции управления моторами вынесены в листинг 8.2, б— они перешли в отдельный файл с именем `motor.h`.

В тексте приводимых далее листингов функции управления моторами подключаются локально инструкцией `#include "motor.h"`, при этом файл `motor.h` должен лежать в одном каталоге с примером.

Следует учесть, что если кнопки не нажаты, пульт не генерирует никакого сигнала, и эту ситуацию требуется обработать. Робот отслеживает время последнего нажатия кнопки и останавливается через 0,3 секунды, если кнопка не нажата повторно. Если коды кнопок вашего пульта отличаются от приведенных в программе, их следует скорректировать (они записаны после ключевых слов case).

Листинг 8.2, а. *Программа управления роботом с помощью IR-пульта*

```
// Подключение библиотеки IRremote.h.
#include <IRremote.h>
// Подключаем глобально библиотеку из Arduino IDE.
#include "IRremote.h"
// Подключаем локально библиотеку из текущего каталога #include "motor.h"
// Функции управления моторками.
// Порт для IR-приемника.
int RECV_PIN = 7;
// Создание IR-приемника.
IRrecv irrecv(RECV_PIN);
// Переменная для хранения кодов кнопок, получаемых от IR-приемника.
decode_results results;
// Хранит время последнего нажатия кнопки. unsigned long _time;
// Опишем коды кнопок макроподстановки:
#define FORWARD 0x5//DBB #define LEFT 0x52A3D41F #define RIGHT 0x20FE4DBB
#define STOP 0xD7E84B1B #define BACKWARD 0xA3C8EDDB #define FORWARDLEFT
0xC101E57B #define FORWARD2 0x97483BFB #define FORWARDRIGHT 0xF0C41643 #define
LEFT2 0x9716BE3F #define STOP2 0x3D9AE3F7 #define RIGHT2 0x6182021B #define
BACKWARDLEFT 0x8C22657B #define BACKWARD2 0x488F3CB8 #define BACKWARDRIGHT
0x449E79F #define STOP3 0x1BC0157B #define STOP4 0x32C6FDF7 #define STOP5 0x3EC3FC1B
// Инициализация.
void setup()
{
// Переменные - номера контактов (цинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2, 3, 4, 5);
stop(); // Двигатели остановлены.
// Запуск IR-приемника.
irrecv.enableIRIn();
_time = millis();
}
// Основная программа.
void loop()
{
if (irrecv.decode(&results))
{
_time = millis();
switch (results.value) {
// Вперед case FORWARD:
forward(); break;
// Назад case BACKWARD:
backward(); break;
// Влево case LEFT:
left(); break;
// Вправо case RIGHT:
right(); break;
// Вперед case FORWARD2:
forward(); break;
```

```

// Назад case BACKWARD2:
backward(); break;
// Влево case LEFT2:
left (); break;
// Вправо case RIGHT2 :
right(); break;
// Прямо и влево case FORWARDLEFT:
forward_left(); break;
// Прямо и вправо case FORWARDRIGHT:
forward_right(); break;
// Назад и влево case BACKWARDLEFT:
backward_left(); break;
// Назад и вправо case BACKWARDRIGHT:
backward_right(); break;
// Стоп case STOP:
stop();
break; case STOP2:
stop();
break; case STOP3:
stop();
break; case STOP4:
stop();
break; case STOP5:
stop();
break;
>
irrcv.resume();
}
// Если никакая клавиша не нажата более 0,3 сек., то остановка.
if ( ( millis () -_time) >300) (_stop() );

```

Листинг 8.2, б. *Содержимое модифицированного файла motor.h*

```

// Объявляем переменные для хранения состояния двух моторов.
int motor_L1,
motor_L2;
int motor_R1,
motor_R2;
// Функция инициализации управления моторами.
void setup_motor_system(int L1, int L2, int R1, int R2)
{
// Заносятся в переменные номера контактов (пинов) Arduino. motor_L1 = L1; motor_L2 = L2;
// Для левых и правых моторов работа. motor_R1 = R1; motor_R2 = R2;
// Переводятся указанные порты в состояние вывода данных.
pinMode (motor_L1, OUTPUT) ;
pinMode (motor_L2, OUTPUT);
pinMode (motor_R1, OUTPUT);
pinMode (motor_R2, OUTPUT);
}
// движение вперед. void forward()
{
// Если двигатель будет работать не в ту сторону,
// поменять на нем контакты местами.
digitalWrite(motor_L1, HIGH);
digitalWrite(motor_L2, LOW);

```

```

digitalWrite(motor_R1, HIGH);
digitalWrite(motor_R2, LOW);
}
// Поворот налево с блокировкой левых колес. void forward_left()
{
// блокировка вращения левых колес.
digitalWrite(motor_L1, LOW);
digitalWrite(motor_L2, LOW);
// правые колеса вращаются.
digitalWrite(motor_R1, HIGH);
digitalWrite(motor_R2, LOW);
}
// Поворот направо с блокировкой правых колес. void forward_right()
{
// левые колеса вращаются.
digitalWrite (motor_L1, HIGH);
digitalWrite(motor_L2, LOW);
// блокировка вращения правых колес.
digitalWrite(motor_R1, LOW);
digitalWrite(motor_R2, LOW);
}
// Поворот налево на месте.
void left()
{
// левые колеса вращаются назад.
digitalWrite (motor_L1, LOW) ;
digitalWrite (motor_L2, HIGH);
// правые колеса вращаются вперед .
digitalWrite(motor_R1, HIGH) ;
digitalWrite (motor_R2, LOW);
}
// Поворот направо на месте. void right()
{
// левые колеса вращаются вперед.
digitalWrite(motor_L1, HIGH);
digitalWrite(motor_L2, LOW);
// правые колеса вращаются назад.
digitalWrite(motor_R1, LOW);
digitalWrite (motor_R2, HIGH);
}
// Выключаем движение назад.
void backward()
{
// Смена направления вращения двигателей.
digitalWrite(motor_L2, HIGH);
digitalWrite(motor_L1, LOW);
digitalWrite(motor_R2, HIGH);
digitalWrite(motor_R1, LOW);
}
// Смена направления вращения двигателей.
digitalWrite(motor_L2, LOW);
digitalWrite(motor_L1, HIGH);
digitalWrite(motor_R2, HIGH);

```



```

digitalWrite (motor_R1, LOW);
}
// Выключаем движение назад. void backward_right()
{
// Смена направления вращения двигателей .
digitalWrite(motor_L2, HIGH);
digitalWrite(motor_L1, LOW);
digitalWrite(motor_R2, LOW);
digitalWrite (motor_R1, HIGH);
}
void _stop()
{
// Блокировка всех колес.
digitalWrite(motor_L2, LOW);
digitalWrite (motor_L1, LOW);
digitalWrite(motor_R2, LOW);
digitalWrite(motor_R1, LOW);
}

```

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №28.

Управление роботом по каналу Bluetooth

Цель работы: научиться разрабатывать программу управления роботом по каналу Bluetooth.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;

- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Для связи с роботом по Bluetooth потребуется установить на него контроллер Bluetooth — тогда управлять роботом можно будет со смартфона на операционной системе Android. В магазине приложений для Android-устройств Google Play Маркет есть достаточное количество бесплатных приложений, реализующих функции управления колесными роботами (рис. 8.9).

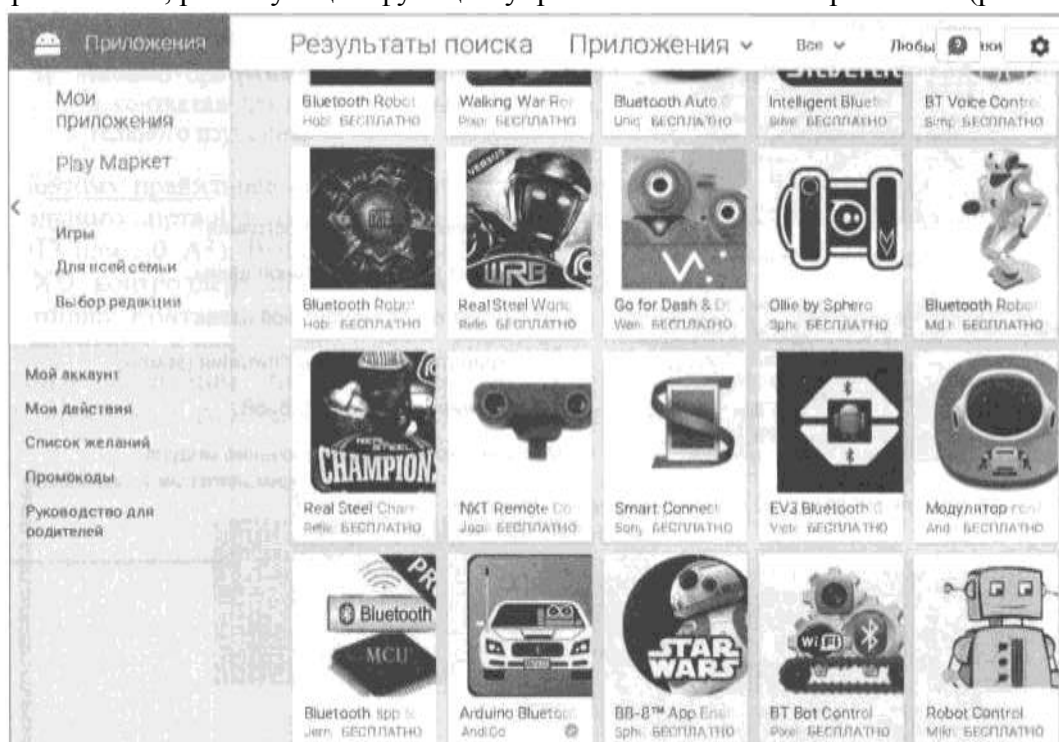


Рис. 8.9. Программы управления роботами на Google Play Маркет (для смартфонов на Android)

Подбор элементной базы

В качестве Bluetooth-контроллера робота можно применить недорогие модули HC-05, HC-06 (рис. 8.10) или SPP-C (рис. 8.11). Питание 5 В подается на контакты VCC и GND. Для работы на передачу и прием используются контакты RXD и TXD, на плате Arduino робота потребуется задействовать для работы с этим Bluetooth - контроллером два свободных порта. Если хотите установить контроллеру имя, отличное от имени по умолчанию, контроллер следует перепрограммировать. При этом для модуля HC-05 (имя по умолчанию hc-05) может потребоваться еще один порт. После смены имени порт можно будет освободить.

Вариант Bluetooth -модуля SPP-C (см. рис. 8.11) удобен тем, что не требует применения пайки для изменения имени и других настроек. После подключения по умолчанию ему присваивается имя vt04-A. Подключается он аналогично HC-05, но есть небольшое отличие в синтаксисе написания программ: при использовании AT-команд между командой и параметром не ставится знак =. Например, команда присвоения имени модулю для HC-05 выглядит так:

AT+NAME=ROBOPES

а для SPP-C и HC-06 так:

AT+NAMEROBOPES

По этой команде модулю будет присвоено имя ROBOPEP.

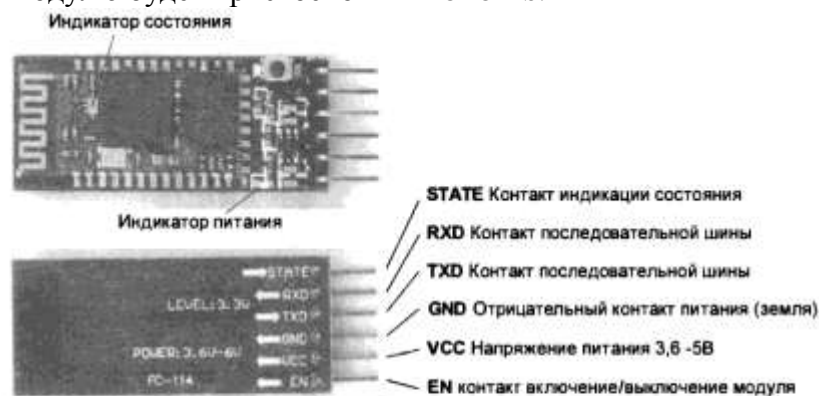


Рис. 8.10. Модуль HC-05/HC-06 приемопередачи по Bluetooth

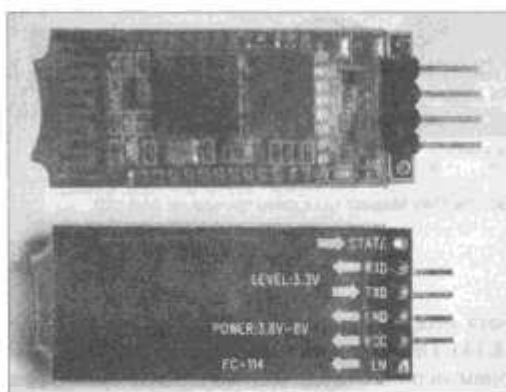


Рис. 8.11. Модуль SPP-C приемопередачи по Bluetooth. Назначение контактов то же, что и на рис. 8.10

За этим небольшим исключением все написанное далее одинаково для рассмотренных модулей. Еще стоит отметить, что HC-06 визуально не отличается от HC-05, но в HC-06 отсутствует ряд режимов работы.

Подключение к Arduino

Обычно Bluetooth-контроллер подключается на стандартные входы последовательного порта D0 и D1 — на плате Arduino они обозначены как RX и TX. Но эти же входы используются при программировании робота, поэтому каждый раз при проведении процедуры программирования вам придется отсоединять Bluetooth-контроллер, иначе он будет мешать обмену информацией между компьютером и Arduino.

Программирование робота по Bluetooth

Можно программировать робота и по Bluetooth, подключив Bluetooth-контроллер к контактам D0 и D1 контроллера Arduino, однако правильнее использовать специальную библиотеку эмуляции последовательного порта и подключить контроллер на любые свободные входы Arduino (D2-D13 или A0-A3). Выберем D10 и D11. Подключение должно быть перекрестным: RXD контроллера подключается к TX Arduino, а TXD контроллера — к RX Arduino. Контакты RXD и RX читают данные, а TXD и TX их передают, — соответственно, в паре соединенных контактов один должен быть передающим, а другой — читающим. Это нужно будет учесть при создании последовательного порта в программе робота. Порядок соединения приведен на рис. 8.12. Подключение 34 ноги контроллера Bluetooth к порту A0 актуально, только если требуется сменить имя модуля HC-5, для HC-06 или SPP-C этого не требуется.

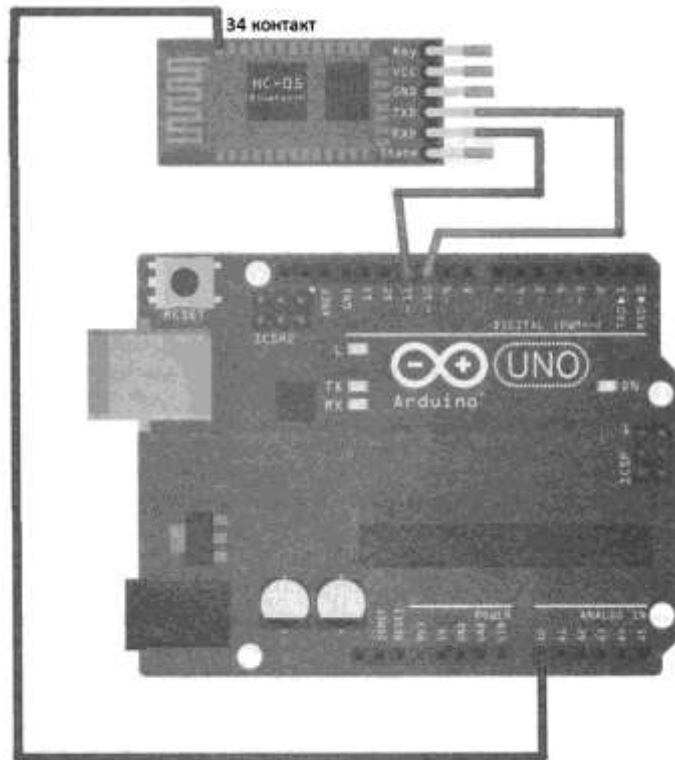


Рис. 8.12. Подключение контроллера Bluetooth HC-05 к Arduino UNO с контактом для модификации имени

Так как приемопередатчик Bluetooth потребляет относительно много энергии, электропитание для него следует получать от отдельного источника — в нашем случае от стабилизатора плата драйвера двигателя через Arduino Sensor Shield v5.0 (рис. 8.13). Теперь, если включить робота и выполнить на смартфоне поиск Bluetooth-устройств, в нем появится новое Bluetooth-устройство — оно будет называться так, как всю партию прошили на фабрике, — например, HC-05.

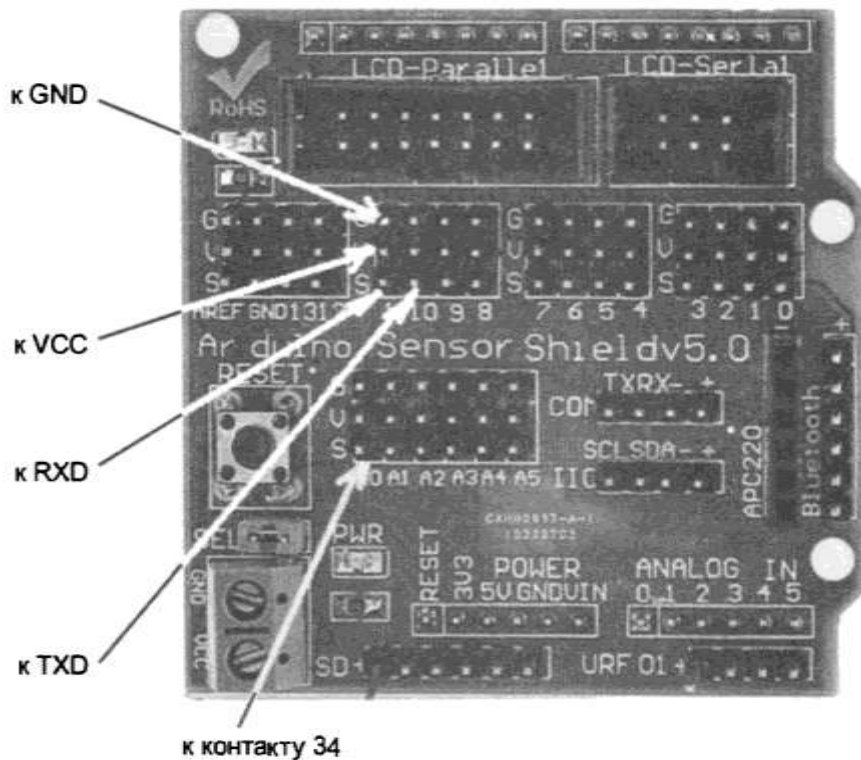


Рис. 8.13. Подключение контроллера Bluetooth HC-05 через Arduino Sensor Shield V5.0

Смена имени робота

HC-05 или HC-06 — не очень оригинальные названия, но они прошиты внутри модулей, и их можно изменить. В отличие от SPP-C и HC-6, перепрошивка HC-05 возможна только при наличии логической единицы на его 34-м контакте. Для подачи такой логической единицы потребуется присоединить к контакту 34 HC-05 (см. рис. 8.12) свободный порт контролера Arduino — пусть это будет порт АО. Согласно инструкции на Arduino ^NO, порты АО-А7 являются аналоговыми приемниками, но АО-А5 могут также работать и в двоичном режиме как для чтения, так и для вывода информации. Контакт 34 расположен довольно близко к 33-му контакту — паяйте осторожно! После изменения имени робота контакт 34 можно будет освободить.

Задание. Создайте скетч с программой переименования робота и тестирования работы Bluetooth обмена данными.

Программа переименования робота для описанных настроек (используемых портов) приведена в листинге 8.3. Новое имя присваивается роботу в строке BTSerial.println ("AT+NAME=MaxiBot"), здесь это имя MaxiBot. Придумайте и задайте имя своему роботу латинскими буквами без пробелов.

Проверьте результат прошивки, проведя поиск новых устройств Bluetooth на смартфоне.

Листинг 8.3. *Переименование робота и проверка работы Bluetooth*

// Подключаем библиотеку для работы с Serial через дискретные порты

#include <SoftwareSerial.h>

// Создаем последовательный порт на пинах 10 и 11

SoftwareSerial BTSerial(10, 11);

// RX, TX

// Переменная для приема данных по Bluetooth

char bt_input;

// Настройка void setup()

{

// Устанавливаем скорость передачи данных по Bluetooth

BTSerial.begin(9600);

// Переключаем АО в двоичный режим работы, на передачу

pinMode(14, OUTPUT);

//Только для HC-05

// Переводим HC-05 в режим AT-команд

digitalWrite(14, 1);

//Только для HC-05

// Присваиваем HC-05 новое имя - MaxiBot

BTSerial.println("AT+NAME=MaxiBot");

//Для HC-05

//Для модуля из набора HC-06 и SPP-C //

BTSerial.println ("AT+NAMEMaxiBot");

// Устанавливаем скорость передачи данных по кабелю

Serial.begin(9600);

// переводим HC-05 в нормальный режим работы

digitalWrite(14, 0);

//Только для HC-05

}

void loop()

{

// Если пришли данные по Bluetooth

if (BTSerial.available())

{

// Записываем, что пришло по Bluetooth, в переменную

bt_input bt_input = (char) BTSerial.read();

Serial.println(bt_input);

}

}

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №29.

Управление роботом через приложение

Цель работы: научиться управлять роботом через стандартное приложение.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Настройка смартфона

Для управления роботом по каналу Bluetooth вам потребуется смартфон на операционной системе Android и программа удаленного управления. Удачным выбором может стать программа Arduino Bluetooth RC Car. Она хорошо документирована и неплохо зарекомендовала себя в тестах. Установите ее с Google Play Маркет на смартфон. Интерфейс программы приведен на рис. 8.14.

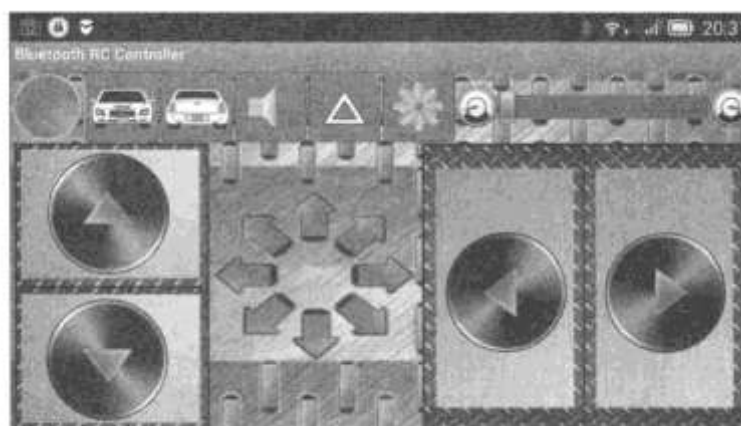


Рис. 8.14. Главное окно программы Arduino Bluetooth RC Car

Чтобы подключиться к роботу, следует нажать кнопку в виде шестеренки, активировать Bluetooth, после чего выбрать пункт Connect. Если робот уже прописан в устройствах смартфона, нужно будет выбрать его из меню (рис. 8.15), в противном случае произвести новое сканирование и подключить робота. Пароль по умолчанию у HC-05: 1234, это может потребоваться!



Рис. 8.15. Выбор Bluetooth-устройства

Когда робот соединится со смартфоном, интерфейс программы изменится: серая лампочка слева вверху станет зеленой, а в верхнем правом углу появится имя робота, к которому подсоединен смартфон (рис. 8.16).

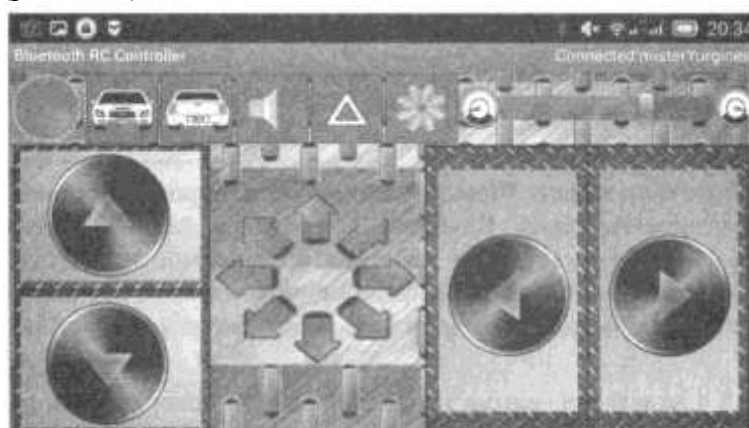


Рис. 8.16. Соединение установлено

Устранение радиопомех

Когда речь идет о работе по радиоканалу, возникают проблемы, связанные с электромагнитной совместимостью. Установка Bluetooth-модуля рядом с сервомотором головы, создающим радиопомехи, может приводить к частым обрывам связи. Лучше всего, если Bluetooth-модуль будет расположен на небольшой антенне и подальше от двигателей. При этом, если на все моторы будут установлены рекомендованные защитные конденсаторы, то радиус связи вашего робота значительно увеличится, и обрывы связи сократятся.

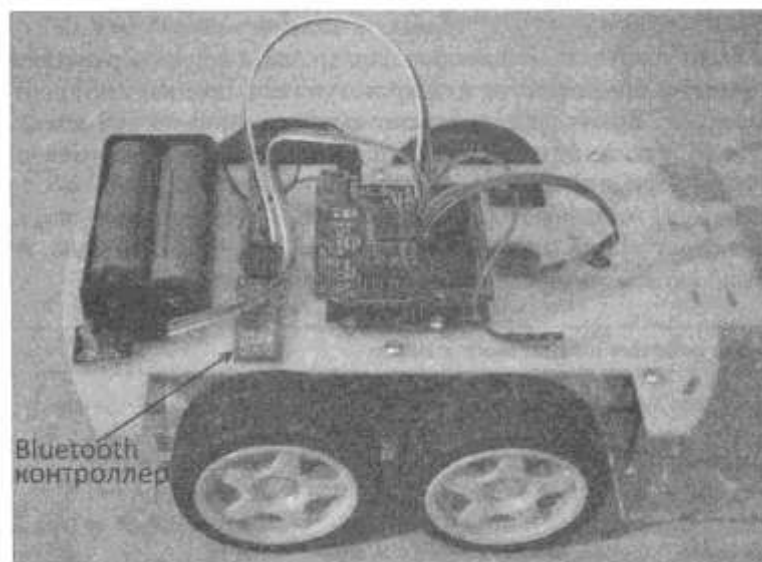


Рис. 8.17. Робот, оснащенный контроллером Bluetooth HC-05/HC-06

Общий вид робота в сборе, оснащенного контроллером Bluetooth, приведен на рис. 8.17.

Задание. Создайте скетч с программой управления поворотом колес и их блокировкой.

Программа управления роботом по каналу Bluetooth весьма проста: робот постоянно проверяет на последовательном порту Bluetooth - контроллера наличие байта команды. Если команда есть — изменяет свои действия в зависимости от того, какой байт пришел. Если никакие кнопки на смартфоне не нажимать, он все равно отправляет команду, а именно — символ `s`, которую наш робот интерпретирует как остановку.

Приведенная в листингах 8.4, а и б программа позволяет роботу поворачивать на полном ходу, используя принцип блокировки колес со стороны поворота. Если же робот стоит, то он поворачивает всеми колесами, — при этом правые и левые вращаются в разные стороны. Робот может поворачивать и при езде задним ходом. Кроме того, программа Arduino Bluetooth RC Car дает возможность управлять роботом посредством изменения ориентации смартфона в пространстве.

Кроме заявленных функций, программа Arduino Bluetooth RC Car позволяет, например, включать и отключать фары, если включение фар предусмотрено конструкцией робота.

Подробнее прочитать о том, какие символы-команды посылает смартфон роботу, можно на странице помощи в программе управления на смартфоне.

Собственно программа управления роботом по Bluetooth-каналу представлена в листинге 8.4, а. А для более удобной работы анализ нажатых кнопок вынесен в отдельную функцию `move_case` (`char bt_input`), помещенную в файл `move_case.h` (листинг 8.4, б), который подключается к основной программе инструкцией

```
#include "move_case.h".
```

Для того чтобы можно было управлять мощностью двигателей, введена переменная `_move_time`, в которой задается время работы в микросекундах в течение такта длительностью 500 микросекунд. Время работы можно изменять от 0 до 500 мксек. Сама организация тактов работы двигателей находится в конце программы. Сначала проверяется, не истек ли период времени текущего такта, выделенный на работу моторов, а если истек, моторы отключаются командой `_stop` (). Затем проверяется, не закончилось ли время самого такта (оно равно 500 мксек), а если оно истекло, то начинается новый такт, и двигатели запускаются вызовом функции `move_case`.

Листинг 8.4, а. *Управление роботом по Bluetooth-каналу*

```
// Подключаем библиотеку для создания дополнительных последовательных (Serial) портов.
```

```
#include <SoftwareSerial.h>
```

```
#include "motor.h"
```

```
#include "move_case.h"
```

```
// Создаем последовательный порт на пинах 13-чтение и 2-передача.
```

```
SoftwareSerial BTSerial(10, //); // RX, TX
```



```

// Переменная для приема данных по Bluetooth. char bt_input;
// Хранит время последнего нажатия кнопки. unsigned long _time;
// Функция инициализации void setup()
{
// Переменные - номера контактов (цинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2, 3, 4, 5);
_stop(); //Двигатели остановлены.
// Устанавливаем скорость передачи данных для HC-05 (Bluetooth-модуль).
  BTSerial.begin(9600);
// Переключаем A0 в двоичный режим работы, на передачу,
// если вы его еще не отключили
pinMode(14, OUTPUT);
// Устанавливаем скорость передачи данных по кабелю.
// Порт компьютера //Serial.begin(9600);
_time = micros ();
}
// Основная программа. void loop()
{
if (BTSerial.available())
{
// Читаем команду и заносим ее в переменную. char преобразует
// код символа команды в символ.
bt_input = (char)BTSerial.read();
// Отправляем команду в порт, чтобы можно было
// их проверить в "Мониторе порта".
//Serial.println(bt_input);
// Вызов функции выбора действия по команде
move_case(bt_input);
_time = micros ();
}
if ((micros() - _time) > _move_time)
{
  stop ();
}
if ( ( micros () - _time) >= 500)
{
_time = micros ();
move_case(bt_input);
}
}

```

Листинг 8.4, б. Содержимое файла *move_case.h*

```

// Переменная изменения скорости. unsigned long _move_time;
// == Выбор действий
void move_case(char bt_input)
{
switch (bt_input) {
// Вперед
case 'F' :
forward();
break;
// Назад case 'B':
backward();
}
}

```

```

break;
// Влево
case 'L':
    left ();
break;
// Вправо
case 'R':
    right();
break;
// Прямо и влево
case 'G':
    forward_left();
    break;
// Прямо и вправо
case 'Г':
    forward_right ();
    break;
// Назад и влево
case 'H':
    backward_left();
    break;
// Назад и вправо
case 'J':
    backward_right();
    break;
// Стоп case 'S':
    stop ();
break;
// Скорость 0% case '0':
    _move_time = 0;
break;
// Скорость 10% case '1':
    _move_time = 50;
break;
// Скорость 20% case '2':
    _move_time = 100;
break;
// Скорость 30% case '3':
    _move_time = 150;
break;
// Скорость 40% case '4':
    _move_time = 200;
break;
// Скорость 50% case '5':
    _move_time = 250;
break;
// Скорость 60% case '6':
    _move_time = 300;
break;
// Скорость 70% case '7':
    _move_time = 350;
break;
// Скорость 80% _move_time = 400;
break;

```

```

// Скорость 90% case '9':
_move_time = 450;
break;
// Скорость 100% case 'q':
_move_time = 500;
break;
case 'X':
//switch_rejim = 1;
break;
case 'x':
//switch_rejim = 0;
break;
case 'D':
_stop();
//switch_rejm = 0;
break;
}
}

```

Основные проблемы, которые могут возникнуть при подключении контроллера Bluetooth, связаны с электромагнитной несовместимостью и невнимательностью. Если контроллер Bluetooth не изменяет свое имя с первого раза, следует сделать повторную попытку.

Принцип управления роботом по каналу Bluetooth теперь должен быть вам понятен. Здесь не рассмотрен вопрос обратной передачи информации от робота и связи двух роботов между собой, но если вы успешно собрали и испытали рассмотренного до сих пор робота, то добьетесь остального самостоятельно.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие №30.

Движение робота по заданной траектории

Цель работы: научиться разрабатывать программу управления роботом, обеспечивающую движение робота по заданной траектории.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации-производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;

- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Классической задачей для мобильных роботов является движение по линии. В нашем случае фон будет белым, а линия — черного цвета, шириной 4 см. Наш робот должен двигаться по этой черной линии, не сбиваясь с курса.

Решение этой задачи предполагает:

- рассмотрение способов детектирования черной линии Arduino;
- обоснование выбора датчиков детектирования черной линии;
- подключение датчиков к роботу;
- составление алгоритма и практическую реализацию программы.

В качестве примера приведем типичную трассу для соревнований по «биатлону» для роботов (рис. 9.1), которые традиционно проводят на робофестивалях. В продолжение эстафеты робот должен начать движение от старта и, двигаясь только по черной линии, проехать, не задев фишки, установленные в позициях 5, 6, 7 и 8; проезжая мимо, сбить фишки, установленные в позициях 1 и 2; забрать с собой к финишу фишки, находящиеся в позициях 3 и 4.

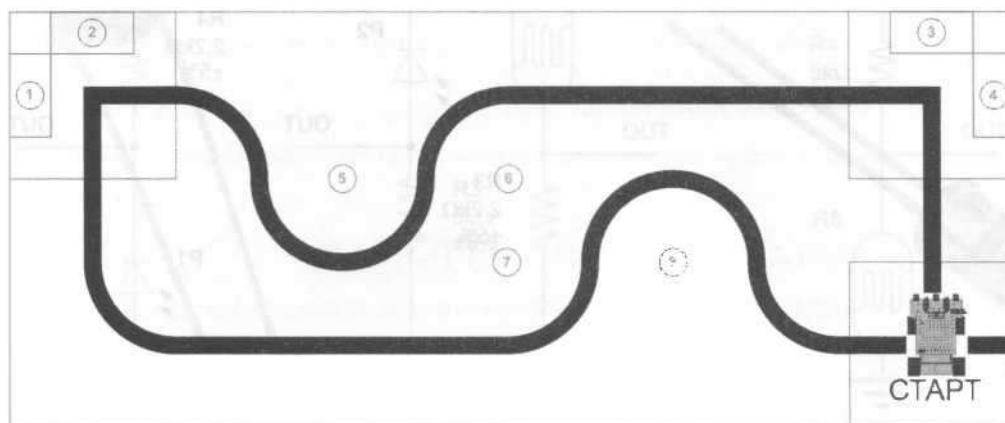


Рис. 9.1. Пример черной линии на соревнованиях роботов по «биатлону»

Такую сложную задачу мы пока ставить перед собой не будем, а ограничимся только движением по линии.

Конечно, можно попытаться запрограммировать робота таким образом, чтобы он четко знал, где ехать прямо, а где совершить поворот, но при этом малейшие непредвиденные обстоятельства: небольшая неровность, изменение шероховатости покрытия, удар о препятствие — собьют робота с курса. Намного эффективнее робот действует, если будет знать, где поверхность черная, а где светлая.

Обнаружение черной линии

Обнаружить черную линию нам помогут электронные приборы, которые реагируют на изменение освещенности. Рассмотрим четыре таких прибора, наиболее применимых для решения поставленной задачи.

1. Фотодиод (рис. 9.2, а) в темноте ведет себя как обычный диод — пропускает электрический ток только в одном направлении, а при освещении генерирует небольшой ток даже без источника питания. При обратном включении фотодиода он, при достижении освещенностью определенного уровня, начинает пропускать электрический ток в обратном направлении — это позволяет использовать фотодиоды в качестве датчиков наличия освещенности.

Схемы обратного включения фотодиодов приведены на рис. 9.2, б и в. Сигнал снимается с выхода OUT, при этом для схемы 9.2, б наличие освещенности характеризуется высоким выходным сигналом, а для схемы 9.2, в — низким. Подобные схемы не позволяют измерять уровень освещенности, а лишь фиксируют ее наличие, что говорит о необходимости подключения их к цифровым портам, которые воспринимают только два вида сигналов: высокий — 1 и низкий — 0.

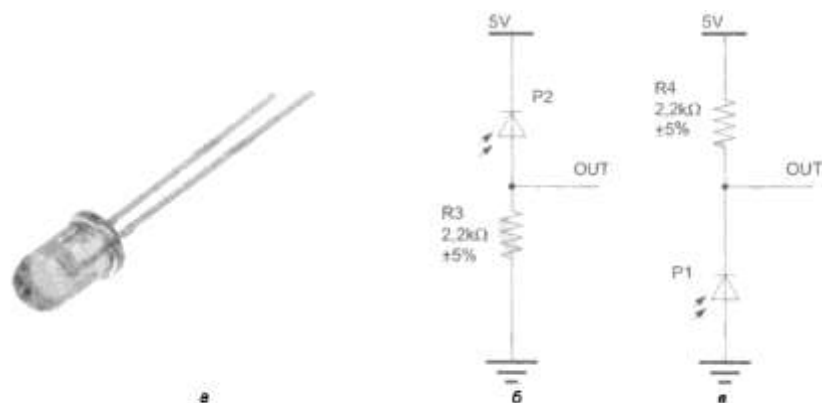


Рис. 9.2. Фотодиод (а) и схемы подключения фотодиода в качестве датчика наличия освещенности (б и в)

Чтобы фотодиод мог четко детектировать черную линию, нужно ее область подсветить, для чего, как правило, используется светодиод. Исходящий от него свет, отражаясь от светлой поверхности пола, попадает на фотодиод, вызывая в нем обратный ток. От черной же линии свет не отражается, что приводит к прекращению обратного тока фотодиода. Светодиод и фотодиод размещаются над исследуемой поверхностью так, чтобы отраженный от нее свет светодиода попадал на фотодиод. При этом расстояние до поверхности должно быть в пределах 1-2 см.

Фотодиоды и светодиоды производятся для разных спектров света, в том числе и невидимого, — например, инфракрасного диапазона.

2. Фоторезистор (рис. 9.3, а) под действием света изменяет свое сопротивление. При отсутствии освещения его сопротивление находится в пределах 1-200 МОм, а при освещении — уменьшается на несколько порядков. При этом фоторезистор имеет линейную зависимость сопротивления от освещенности, вследствие чего с его помощью удобно измерять освещенность в помещении.

Недостатками фоторезистора можно назвать достаточно высокое сопротивление и инертность — он не реагирует на изменение освещенности с частотой выше 1 кГц. Но при небольших скоростях движения робота использовать его все-таки можно. Схемы подключения фоторезистора приведены на рис. 9.3, б и в. Подсветка фоторезистора светодиодом и размещение их над исследуемой поверхностью организуются таким же образом, как и в случае применения фотодиода.

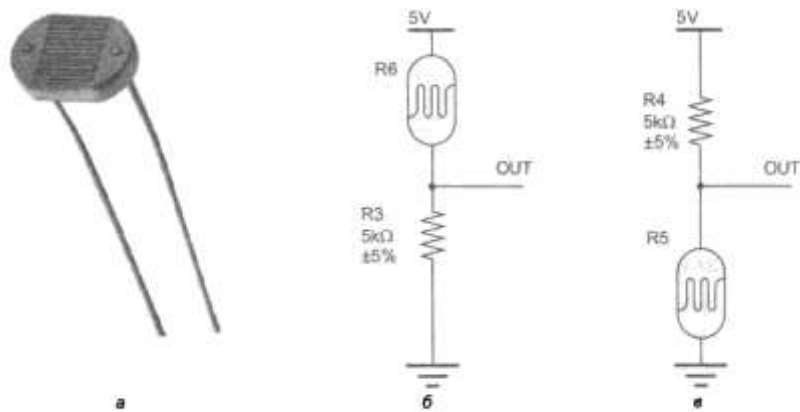


Рис. 9.3. Фоторезистор (а) и схемы подключения фоторезистора в качестве датчика освещенности (б и в)

3. Фототранзистор (рис. 9.4, **а**) не только преобразует световой поток в электрический ток, подобно фотодиоду или фоторезистору, но и многократно его усиливает. Визуально фототранзисторы могут не отличаться от фотодиодов, но подключаются и работают иначе. Схемы подключения фототранзисторов приведены на рис. 9.4, б и в. Основным преимуществом фототранзистора, как аналогового датчика, перед фотодиодом, представляющим собой пороговый, или цифровой, датчик, является возможность измерения уровня освещенности, что позволяет при определении цвета объекта фиксировать тона и полутона. Для этого следует подключать аналоговый выход OUT фототранзистора к аналоговым входам контроллера Arduino (АО-А5 или АО-А7 — в зависимости от модели контроллера). При этом робот должен будет экспериментально измерить уровень освещенности на белом фоне и уровень освещенности на черной линии, а затем проанализировать полученные значения в программе.

Подсветка фототранзистора светодиодом и размещение их над исследуемой поверхностью организуются таким же образом, как и в случае применения фотодиода.

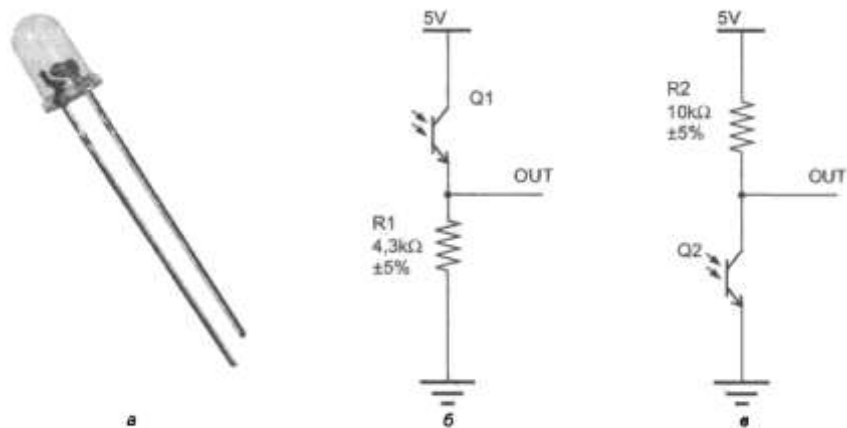


Рис. 9.4. Фототранзистор (а) и схемы подключения фототранзистора в качестве датчика освещенности (б и в)

4. В инфракрасном датчике TCRT 5000 (рис. 9.5) использована готовая связка инфракрасного светодиода и фототранзистора. На его выходах можно получать как аналоговый, так и цифровой сигналы:

- цифровой выход датчика можно использовать для определения наличия черной линии, при этом его чувствительность настраивается переменным резистором;
- на аналоговом выходе датчика интенсивность отраженного сигнала зависит от типа и цвета поверхности, что также можно использовать для выявления наличия черной линии. При этом надо будет экспериментально измерить уровень освещенности на белом фоне и уровень освещенности на черной линии, а затем проанализировать полученные значения в программе.



Рис. 9.5. Инфракрасный датчик отражения TCRT 5000: а — лицевая сторона; б — обратная

Цифровой выход датчика TCRT 5000 обозначен на плате D0, а аналоговый — A0. Цифровой выход подключается к цифровому порту Arduino D0-D13, а аналоговый — к аналоговому A0-A5/A7. Контакт VCC подключается к питанию 5 В, контакт GND — соответственно, к «земле».

На плате датчика расположены два светодиода: зеленый, информирующий о наличии линии, и красный, информирующий о наличии электропитания.

Аналоговые порты A0-A7 при программировании в среде Arduino IDE имеют номера 14-21. При этом порты A0-A5 могут работать как обычные цифровые порты. Порты A6 и A7 распаяны не на всех платах, и они могут работать только как порты аналогового ввода.

Вывод

Для определения наличия черной линии лучше всего использовать инфракрасный датчик отражения TCRT 5000 — как наиболее удобное и готовое решение. При этом воспользуемся цифровым выходом датчика, как наиболее простым с точки зрения обнаружения линии: для него в программе не потребуется анализировать уровень освещенности, а регулировка будет проходить на месте (трассе) подстройкой резистора.

Задание. Подключить к роботу и запрограммировать инфракрасный датчик черной линии.

Подготовка робота: установка датчиков

При движении по непрерывной линии двух датчиков — справа и слева относительно бампера робота — вполне достаточно, чтобы на этой линии удержаться. Датчики TCRT 5000, как правило, уже имеют припаянные контакты для разъемных соединений. Для подключения датчиков через сервисную плату потребуются провода с разъемами «мама-мама» (рис. 9.6).

Подсоединим провода к датчикам и прикрепим датчики на робота. Для настройки чувствительности доступ к подстроечным резисторам датчиков должен быть свободным. Если по конструктивным особенностям вашего робота доступ не возможен, следует настроить датчики заранее, для чего подключить их к питанию, установить на рабочее расстояние над черной линией/белым фоном и добиться четкого срабатывания, подкладывая под датчики черные и белые листы.

Итак, крепим датчики к нижней панели робота 3-мм винтом с гайкой (рис. 9.7, а), предварительно проложив провода до платы Arduino Sensor Shield (рис. 9.7, б).

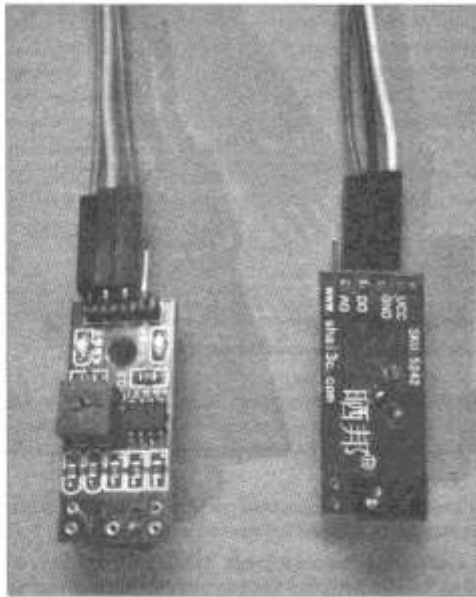


Рис. 9.6. Пара датчиков TCRT 5000, готовых к установке на робота

Датчики должны получить электропитание 5 В и соединение с двумя цифровыми выводами Arduino. Воспользуемся для этого двумя оставшимися свободными портами D8 и D9 — для правого и левого датчика соответственно (рис. 9.8).

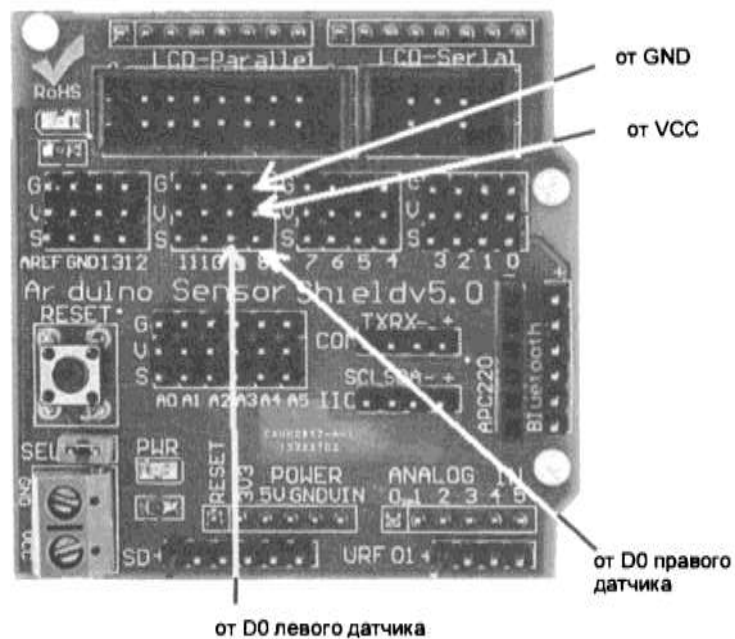


Рис. 9.8. Порядок подключения датчиков TCRT 5000 к Arduino Sensor Shield v5.0

После того как датчики подключены, желательно провести тестирование правильности их подключения к роботу. Для этого следует:

1. Загрузить в работа программа из листинга 9.1.

2. Включить питание.
3. Запустить монитор порта.
4. Подкладывая под датчики робота черные и белые листы, проследить изменение состояния датчиков на экране компьютера по монитору порта Arduino.

Если состояние портов изменяется адекватно с цветом листов, значит, датчики подключены верно.

Листинг 9.1. *Тестовая программа проверки подключения датчиков линии*

// Переменные с номерами пинов, к которым подключены датчики.

```
int left_sensor_line = 9;
```

```
int right_sensor_line = 8;
```

```
void setup ()
```

```
{
```

```
// Перевод пинов в состояние ввода данных.
```

```
pinMode(left_sensor_line, INPUT);
```

```
pinMode(right_sensor_line, INPUT);
```

```
// Устанавливаем скорость порта связи с ПК.
```

```
Serial.begin(9600);
```

```
}
```

```
// Основная программа.
```

```
void loop()
```

```
{
```

```
Serial.print("Left sensor = ");
```

```
Serial.println(digitalRead(left_sensor_line));
```

```
Serial.print("Right sensor = ");
```

```
Serial.println(digitalRead(right_sensor_line));
```

```
Serial.println("=====");
```

```
delay(500);
```

```
}
```

Теперь рассмотрим алгоритм, которого необходимо придерживаться при следовании по извилистой линии (рис. 9.9). Считаем, что в начале движения линия находится между правым и левым датчиком, т. е. робот установлен на линию. Пока оба датчика показывают состояние «светлый фон», робот будет ехать прямо. Если левый датчик показал «черный фон», робот должен повернуть налево. Если правый датчик показал «черный фон», робот должен повернуть направо.

Робот довольно тяжелый и может по инерции проскочить поворот и сбиться с линии. Поэтому следует отрегулировать скорость его движения, для чего в программу добавлены периодические остановки робота. Кроме регулировки скорости, подобные микроостановки позволяют более точно снимать показания с датчиков.

Программа движения робота с учетом указанных обстоятельств приведена в листинге 9.2.

Периоды работы двигателей и остановок следует подобрать экспериментально, поскольку они должны быть различными для разных моторов и массы робота.

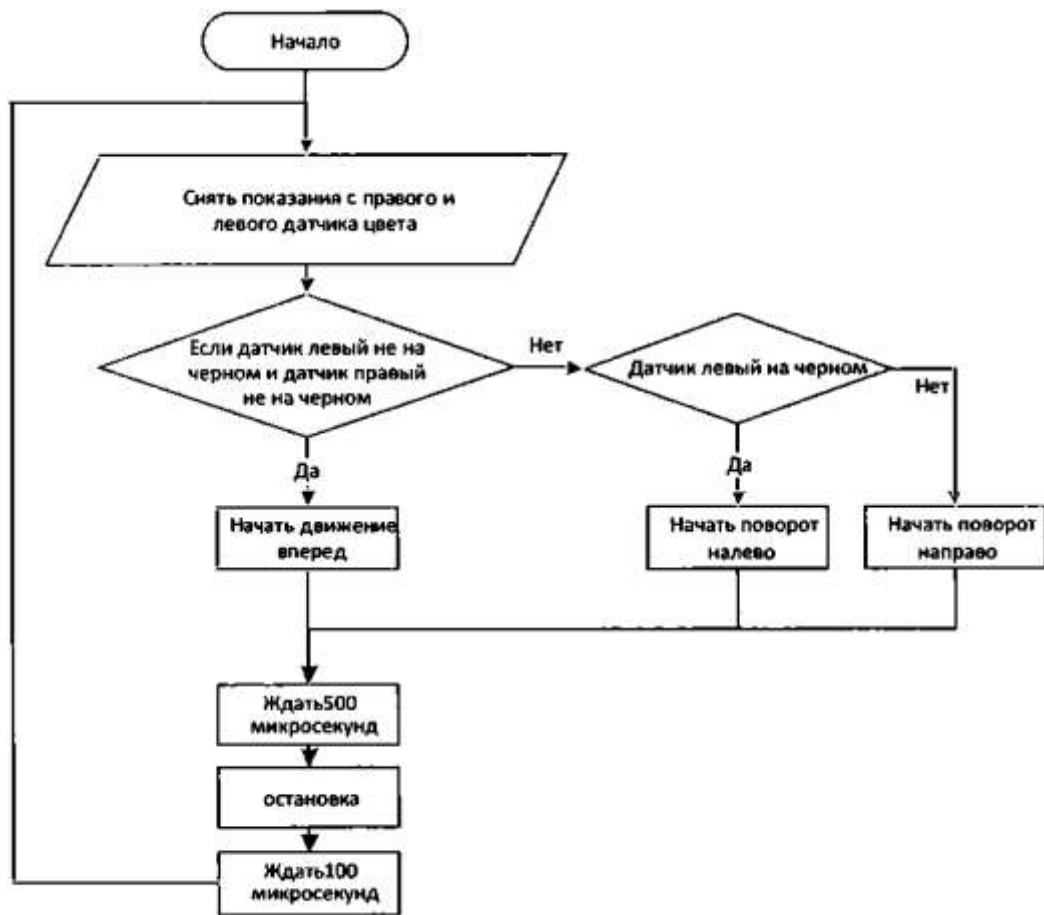


Рис. 9.9. Простейший алгоритм движения по черной линии

Листинг 9.2. Движение робота по черной линии

```

#include "rmotor.h" unsigned long _time;
//=====
int left_sensor_line = 9; int right_sensor_line = 8;
//=====
void setup()
{
// Переменные - номера контактов (пинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2, 3, 4, 5);
_stop();
// Двигатели остановлены.
// Сенсорные пины переводим в состояние ввода данных.
pinMode(left_sensor_line, INPUT);
pinMode(right_sensor_line, INPUT);
_time = millis();
// Устанавливаем скорость порта связи с ПК.
Serial.begin(9600);
}
// Основная программа. void loop()
{
while (true)
{
bool ls = digitalRead(left_sensor_line);
bool rs = digitalRead(right_sensor_line);
// Serial.println(ls);
// Serial.println(rs);
  
```

```

// Serial.println("=====");
// delay(500);
// Готов к приему.
if (!(ls) && !(rs))
{
forward();
}
else
{
if (ls)
{
left ();
}
else
{
right();
}
}
delayMicroseconds(500);
_stop ();
delayMicroseconds(100);
}
}

```

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие № 31.

Оснащение робота дальномером

Цель работы: научиться разрабатывать программу управления роботом, обеспечивающую определение расстояний до препятствий.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;

- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Чтобы робот мог определять расстояния до препятствий, причем в разных направлениях (а это может потребоваться при их объезде), мы оснастим его распространенным, простым и надежным ультразвуковым дальномером HC-SR04 (рис. 10.1), установив его на сервомотор с углом поворота 0-180 градусов.

В этой задаче мы задействуем на плате Arduino три свободных порта: один для управления сервоприводом и два для определения расстояния посредством датчика HC-SR04.

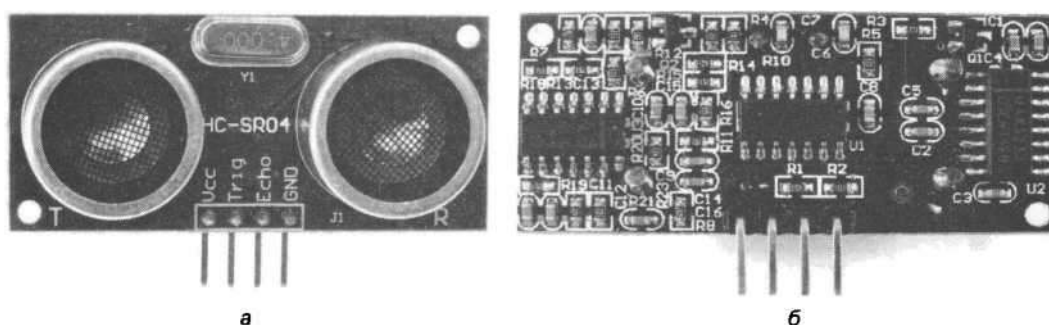


Рис. 10.1. Ультразвуковой дальномер HC-SR04: а — вид спереди; б — вид сзади

Схема подключения

Монтажная схема подключения дальномера и поворотного сервомотора к плате Arduino без использования коммутационной платы Arduino Sensor Shield v5.0 приведена на рис. 10.2.

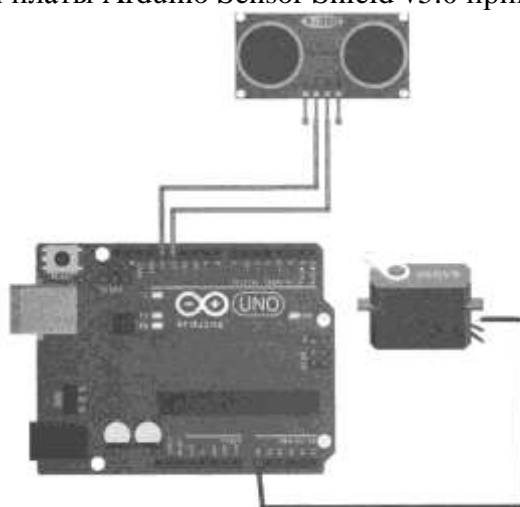


Рис. 10.2. Монтажная схема подключения сервомотора и дальномера

Питание 5 вольт для сервомотора и дальномера следует взять с контакта +5V платы драйвера двигателей (рис. 10.3).

Под датчиком HC-SR04 на плате Arduino будут заняты два порта: D 13 (Trig) и D 12 (Echo). Управление сервомотором, поворачивающим голову, будет производиться с порта A0 платы Arduino. Порт A0 хоть и является аналоговым, но позволяет работать и в цифровом (двоичном) режиме ввода/вывода, что и требуется в данном случае.

При использовании Arduino Sensor Shield v5.0 (рис. 10.4) потребуется четыре коммутационных провода с разъемами «мама-мама» для подключения датчика, сервомотор же не нуждается в дополнительных проводах, поскольку его стандартная колодка как раз подходит к контактам коммутационной платы. Но стоит обратить внимание на то, чтобы не перевернуть колодку в обратную сторону, перепутав G (GND) и S (Signal), — это может привести к порче либо мотора, либо контроллера Arduino.

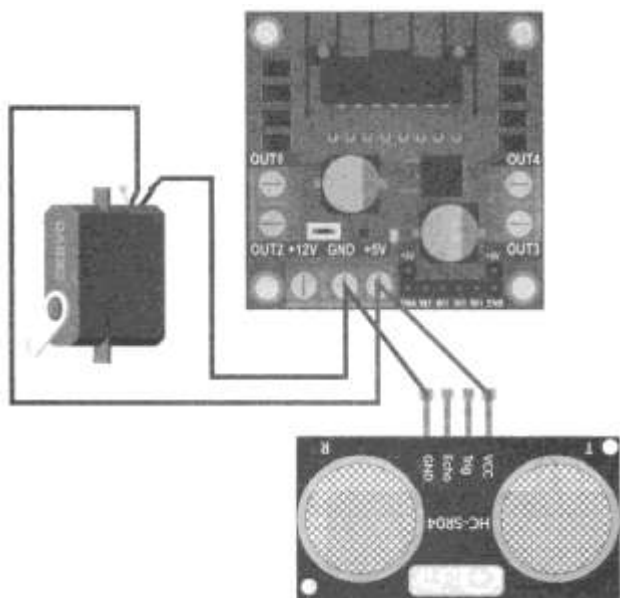
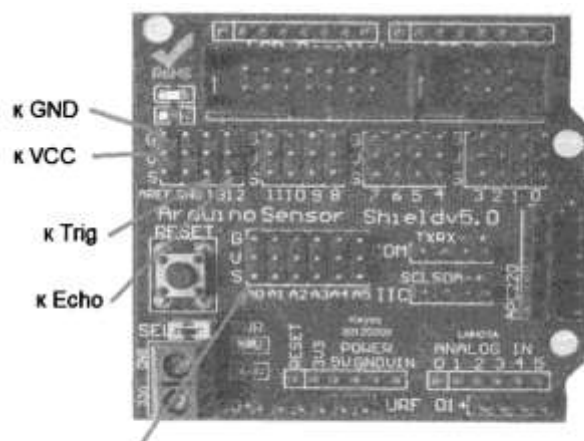


Рис. 10.3. Монтажная схема цепей питания сервомотора и датчика



Шлейф сервомотора:
S - оранжевый; G - коричневый

Рис. 10.4. Подключение датчика и сервомотора к Arduino Sensor Shield v5.0

Задание. Создайте скетч с программой измерения расстояния с помощью ультразвукового датчика и выводом результатов на монитор порта

Измерение расстояния

Ультразвуковой датчик посылает импульс, этот импульс отражается от препятствия и возвращается через некоторое время. Время между импульсом и его возвратом, умноженное на скорость звука и поделенное на два, даст расстояние.

Выделим для работы с ультразвуковыми функциями отдельный файл — это позволит не путаться в функциях и быстро находить нужное, дадим ему имя `sonar.h` (листинг 10.1). Объявим в этом файле две глобальные переменные: `Trig` и `Echo` — для хранения номеров портов подключения одноименных контактов датчика. Там же создадим функцию `sonar_init(int Tr, int Ec)`, которая будет запоминать значения портов для входов датчика и задавать им требуемые режимы работы. Измерением займется функция `int Sonar (unsigned long Limit)` — она будет обращаться к сонару и по его данным вычислять расстояние до препятствия. У этой функции всего один входной параметр, влияющий на время ожидания отражения от препятствия, — это максимальное измеряемое расстояние в сантиметрах, при превышении которого функция будет возвращать 0. Ограничение измеряемого расстояния позволит ускорить работу функции, иначе она может ждать возврата ультразвукового эха от препятствия целую секунду.

Файл `sonar.h` следует скопировать в рабочий каталог текущей программы и подключить инструкцией `#include "sonar.h"`.

Листинг 10.1. Содержимое файла `sonar.h`

```
// Номера пинов сонара. int Trig; int Echo;
//=Режим пинов/портов =====
Sonar_init(int Tr, int Ec)
{
  Trig=Tr;
  Echo=Ec;
}
```

```

// Задаем режим работы пинов.
pinMode (Trig, OUTPUT);
pinMode (Echo, INPUT) ;
)
//= Измерение расстояния =====
int Sonar(unsigned long Limit)
{
int Long_cm;
// Переводим лимит из сантиметров в относительные величины (микросек.) unsigned long
Lim=Limit*58;
// Генерируем импульс
digitalWrite(Trig, LOW);
delayMicroseconds(2) ;
digitalWrite(Trig, HIGH);
delayMicroseconds (10) ;
digitalWrite(Trig, LOW);
// Ждем отражения импульса.
// Пересчитываем время в расстояние (по скорости звука).
Long_cm = int(pulseIn(Echo, HIGH, Lim)/58);
// Если не дождалось отражения,
// присваиваем значение максимально измеряемому. if(Long_cm==0) return int(Limit); return
Long_cm;
}

```

Подключим сонар HC-SR04 к роботу, а робот — к компьютеру и протестируем возможности измерения расстояния. Для этого подойдет программа из листинга 10.2. В ней дальномеру назначаются порты 13 и 12, создается связь с ПК, а в основной программе с периодом 1,5 секунды измеряется расстояние до препятствия и полученное значение передается в порт ПК. В оболочке Arduino IDE следует открыть монитор порта, тогда в окне монитора будет отображаться расстояние (рис. 10.5). Максимальное измеряемое расстояние в примере — 300 см.

Листинг 10.2. Программа измерения расстояний

```

// Подключаем библиотеку, управляющую дальномером.
#include "sonar.h" void setup()
{
// Инициализируем дальномер Trig = 13, Echo = 12.
Sonar_init(13, 12);
// При инициализации задаем скорость порта для связи с ПК.
Serial.begin(9600); // start the serial port
}
void loop()
{
// В цикле loop отправляем значение,
// полученное с дальномера, в порт через 1,5 с.
// получаем дистанцию в сантиметрах с лимтом 300 см.
int prepyatstvie = Sonar(300);
Serial.print("Distance=");
// оформляем вывод.
Serial.print(prepyatstvie);
// выводим дистанцию.
Serial.println(" cm.");
// оформляем вывод.
delay(1500);
// приостанавливаем программу.
}

```



Рис. 10.5. Отображение измеренного расстояния в окне монитора порта

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие № 32.

Оснащение робота сервомотором

Цель работы: научиться разрабатывать программу управления роботом, обеспечивающую поворот верхней части робота с применением сервоприводов.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;

- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

Сервомотор требуется, чтобы робот поворачивал голову на шею. Для работы с сервомоторами в Arduino IDE имеется отдельная библиотека, при помощи которой порт контроллера может генерировать программный ШИМ на любом порту (кроме А6 и А7) с частотой 50 Гц. Принцип работы контроллера в режиме генерации ШИМ с частотой 50 Гц, но для генерации ШИМ не требуются порты с аппаратной широтно-импульсной модуляцией, — подойдет любой цифровой порт, так как генерация ШИМ будет производиться программно. От длины положительного фронта импульса при широтно-импульсной модуляции зависит угол поворота сервомотора. Простейшая программа, реализующая поворот вала сервомотора в разные стороны, представлена в листинге 10.3. Вал поворачивается рывками по 10 градусов сначала в одну, а затем в другую сторону. Задействуются значения углов поворота от 10 до 170 градусов (большие или меньшие значения на простых сервомоторах могут привести к заклиниванию).

Берегите шестерни сервомоторов!

Сервомоторы SG90 имеют довольно хрупкие шестерни. Не следует с усилием вращать вал сервомотора руками — это может привести к разрушению шестерней! Сервомоторы с металлическими редукторами таких недостатков лишены.

Монтаж головы

Если голова еще не сидит на роботе, то пора ее установить. Варианты крепления головы могут различаться, требуемые в нашем примере детали приведены на рис. 10.6: это сонар, сервомотор с крестообразным рычажком и крепежным винтом, пластина крепления сервомотора, четыре шестигранных стойки с гайками, конструкционные детали крепления сонара, четыре 3-мм винта и два 2-мм винта с гайками для крепления сервомотора.

Установку головы следует проводить в следующем порядке:

- 1) Установим четыре шестигранных стойки и закрепим их снизу гайками, как показано на рис. 10.7.
- 2) Вставим сервомотор в прямоугольное отверстие пластины крепления сервомотора и зафиксируем двумя 2-мм винтами с гайками.
- 3) Пластину с сервомотором закрепим на стойках четырьмя 3-мм винтами.
- 4) Сонар HC-SR04, установленный на серводвигатель, показан на рис. 10.8.
- 5) На нижнюю пластину крепления сонара наложим сверху две детали: пластину с крестообразным вырезом и уголок с круглыми отверстиями для сонара; зафиксируем каждую деталь двумя винтами М3 с гайками.

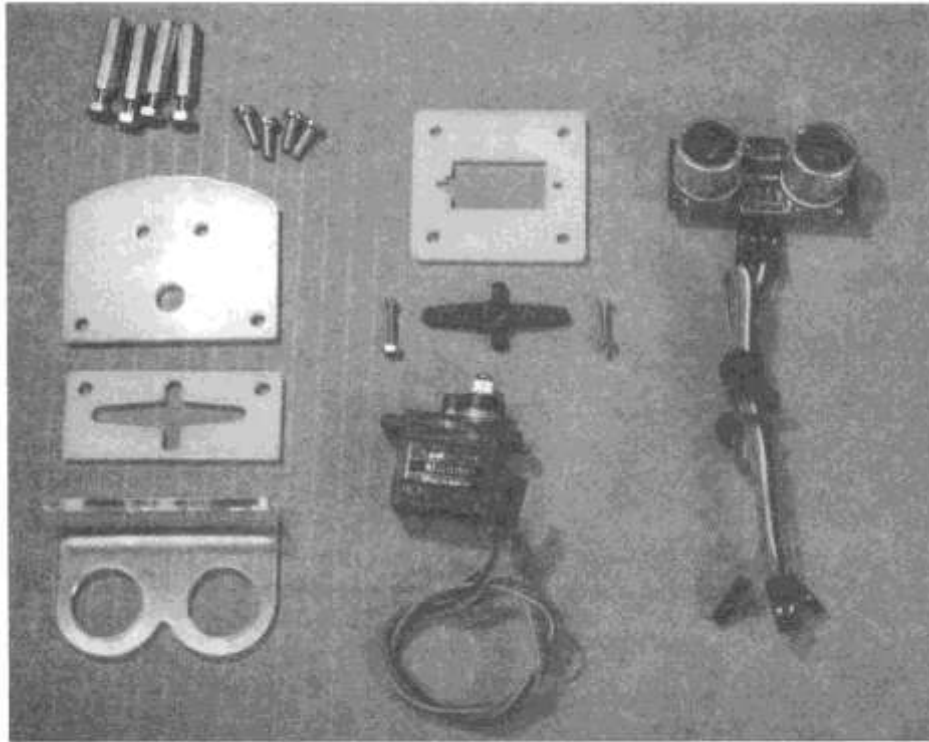


Рис. 10.6. Детали для сборки головы

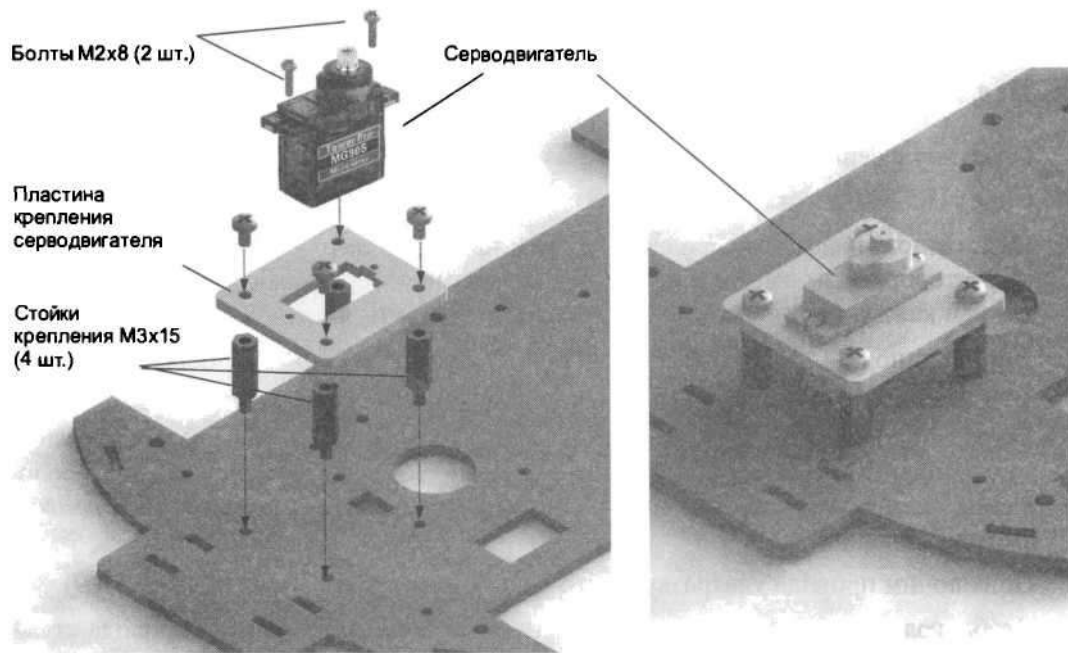


Рис. 10.7. Установка сервомотора

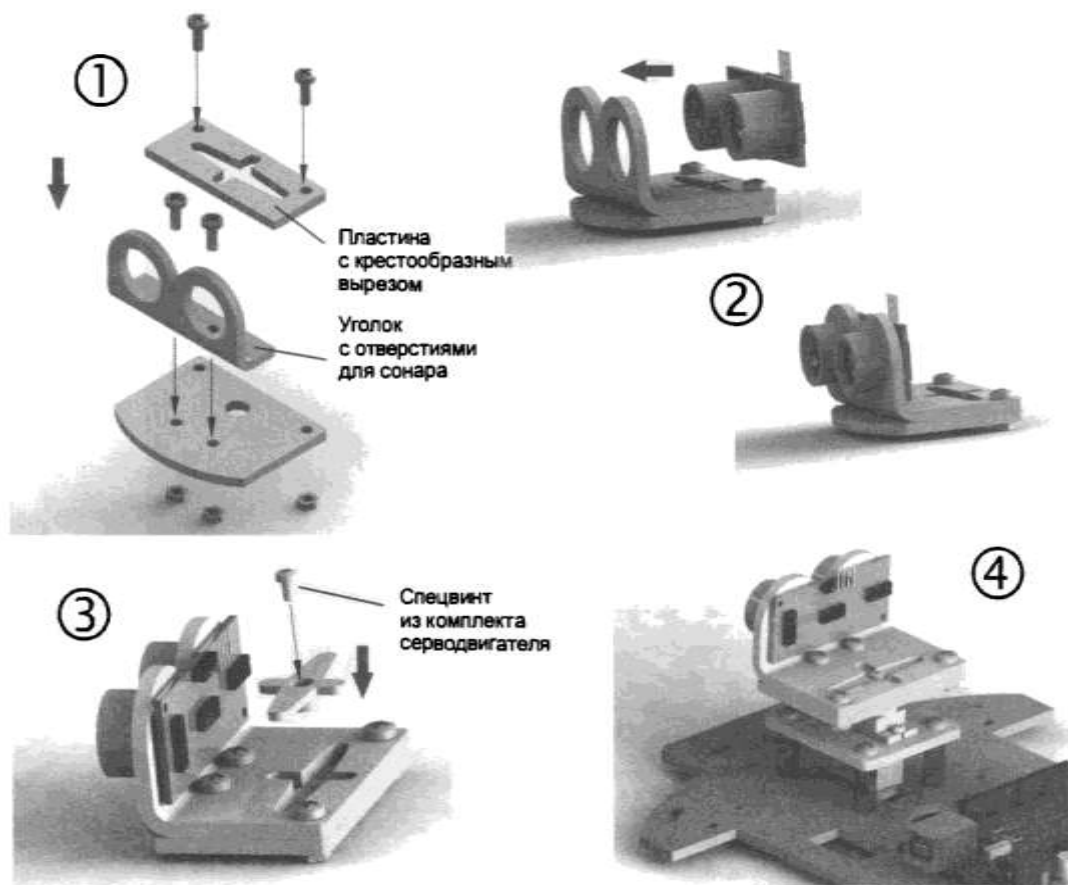


Рис. 10.8. Установка головы на работа

Программно (`neck.write (90)`;) установим сервомотор в среднее положение (листинг 10.4).

Закрепим собранную конструкцию на оси сервомотора специвинтом из комплекта серводвигателя. После программной установки сервомотора в среднее положение голову следует установить ровно в центральное положение (робот должен смотреть вперед).

Подключим провода ультразвукового дальномера к плате Arduino (рис. 10.9).

Чтобы голове ничего не мешало вращаться, стянем болтающиеся провода стяжками. Стянутые в пучки провода реже подвергаются изгибам и реже переламываются. Теперь робот готов (рис. 10.10).

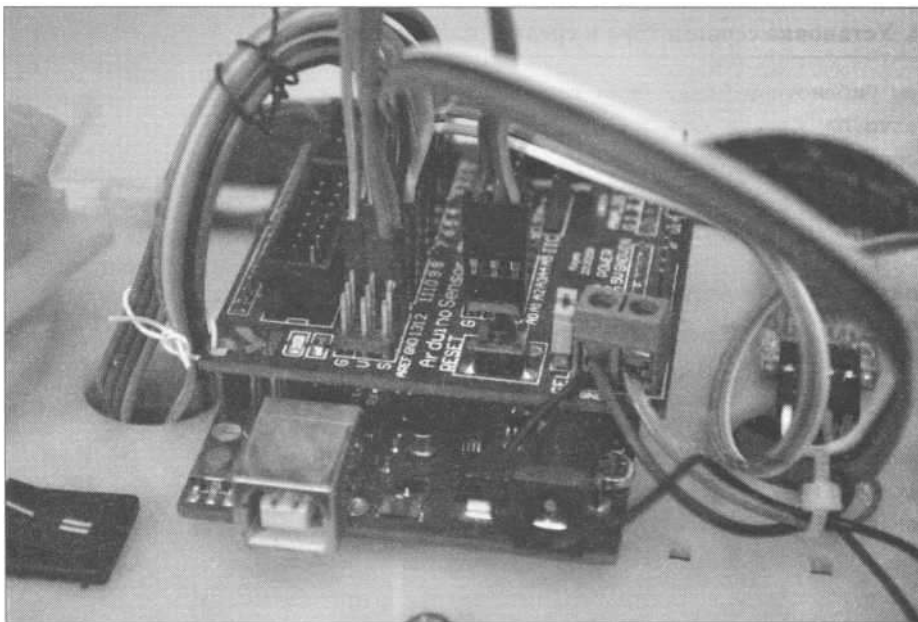


Рис. 10.9. Подключение проводов дальномера HC-SR04 к плате Arduino Sensor Shield v5.0

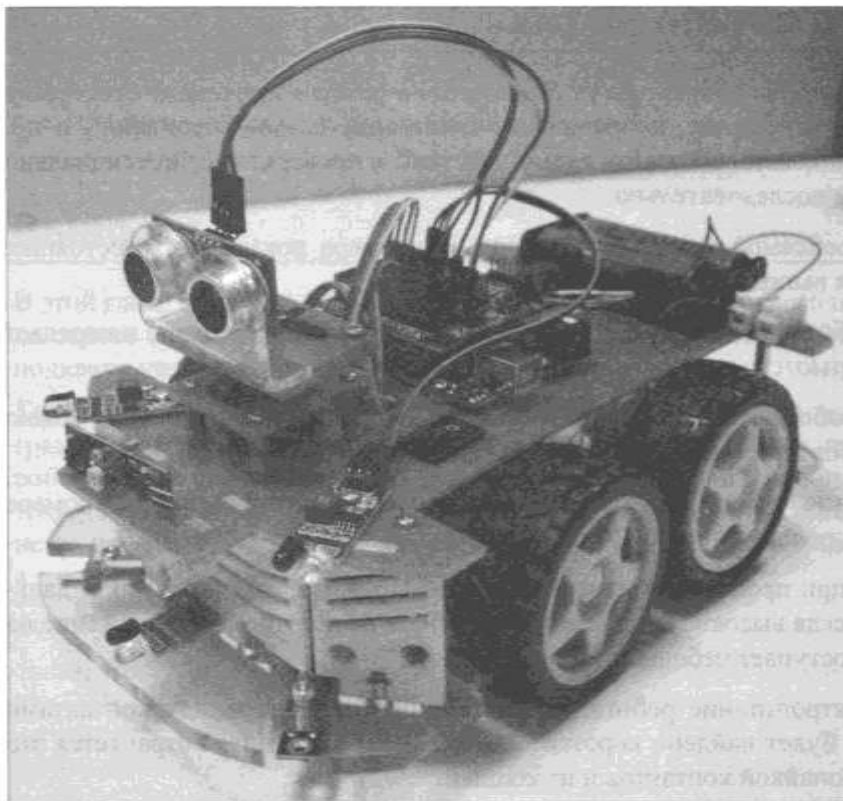


Рис. 10.10. Робот с установленной головой

Задание 1. Создайте скетч с программой управления сервоприводом для верхней части робота.

Листинг 10.3. Программа поворота головы робота в разные стороны

// Подключаем библиотеку управления сервомоторами.

```
#include <Servo.h>
```

// Создаем сервомотор поворота головы.

```

Servo neck; void setup()
{
// Инициализируем сервомотор, управление 14-м портом. neck.attach(14);
}
void loop()
{
int i;
// Создаем цикл для равномерного поворота сервомотора. for(i=10; i<=170; i=i+10)
{
// Поворачиваем вал на угол i. neck.write(i) ;
// приостанавливаем программу. delay(100) ;
}
// Возвращаем голову назад. for(i=170; i>=10;i=i-10)
{
// Поворачиваем вал на угол i.
neck.write(i) ;
// приостанавливаем программу.
delay(100);
}
}

```

Листинг 10.4. Установка сервомотора в среднее положение

```

// Подключаем библиотеку управления сервомоторами.
#include <Servo.h>
// Создаем сервомотор поворота головы.
Servo neck;
void setup()
{
//Инициализируем сервомотор, управление 9-м портом.
neck.attach(9) ;
}
void loop()
{
// Поворачиваем вал на угол 90. neck.write(90) ;
// приостанавливаем программу.
delay(100);
}

```

Возможные проблемы

Неисправности во время сборки могут возникать по разным причинам: отсутствие внимания, плохое освещение, неправильное понимание схемы соединений и пр. Главное — не паниковать, составить схему действий и проверять все потенциально неисправные узлы последовательно.

Так, основной проблемой может стать отказ датчика показывать расстояние. Для ее устранения выполните следующие шаги:

- 1) Загрузите в робота программу, которая только определяет расстояние и передает его в порт компьютера (см. листинг 10.2).
- 2) Подключите робота к компьютеру, в Arduino IDE откройте порт. Как можно видеть, через порт поступают нулевые значения.
- 3) При включенном роботе проверьте наличие электропитания на датчике HC-SR04 (между контактами Vcc и GND).

4) Скорее всего, при проведении измерений вы обнаружите, что на контакте даль-номера Trig всегда высокое напряжение, чего быть не должно, — напряжение на этот контакт поступает небольшими импульсами.

5) Отключив электропитание робота, измерьте сопротивление между контактами VCC и Trig— будет найдено короткое замыкание (рис. 10.11). Устраняется это тщательной пропайкой контактов и изоляцией.

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Лабораторное занятие № 33.

Движение робота с объездом препятствий

Цель работы: научиться разрабатывать программу управления роботом, обеспечивающую заданное движение робота с возможностью объезда препятствий.

Выполнив работу, Вы будете:

уметь:

- применять выбранные языки программирования для написания программного кода;
- использовать выбранную среду программирования;
- использовать возможности имеющейся технической и/или программной архитектуры;
- применять методы и приемы отладки программного кода;
- выполнять действия, соответствующие установленному регламенту используемой системы контроля версий;
- выявлять ошибки в программном коде;
- соблюдать процедуру установки прикладного программного обеспечения в соответствии с требованиями организации- производителя;
- документировать произведенные действия, выявленные проблемы и способы их устранения;
- создавать резервные копии программ и данных, выполнять восстановление, обеспечивать целостность программного продукта и данных;
- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;
- анализировать задачу и/или проблему и выделять её составные части;
- применять средства информационных технологий для решения профессиональных задач;
- использовать современное программное обеспечение;
- применять современную научную профессиональную терминологию;
- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке;
- соблюдать нормы экологической безопасности;
- пользоваться средствами профилактики перенапряжения, характерными для данной специальности;
- понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы;

Материальное обеспечение:

1. Компьютер с лицензионным программным обеспечением для программирования МК.
2. Мультимедиа проектор.
3. Робототехнический набор

Теоретические сведения.

После устранения явных неисправностей следует перейти к проверке правильности работы и тонкой настройке основных функций робота. Для этого подойдет программа, которая позволяет роботу объезжать препятствия.

Придумаем простой и понятный алгоритм объезда препятствий:

- 1) Вращаем головой и измеряем расстояние до препятствий слева, справа и впереди.
- 2) Сравниваем расстояния, выбираем наибольшее.
- 3) Туда и едем.
- 4) Если ехать нельзя, тупик — разворачиваемся.
- 5) Подробная схема алгоритма приведена на рис. 11.1.

Следует заметить, что наша программа не является идеальной программой объезда препятствий — в частности, ее эффективность зависит от размеров робота. В программу заложено условие, разрешающее движение, если расстояние до препятствия превышает 10 см. Попробуйте это расстояние увеличить до 20-30 см. Подойдите к процессу тестирования робота и программирования его настроек творчески!

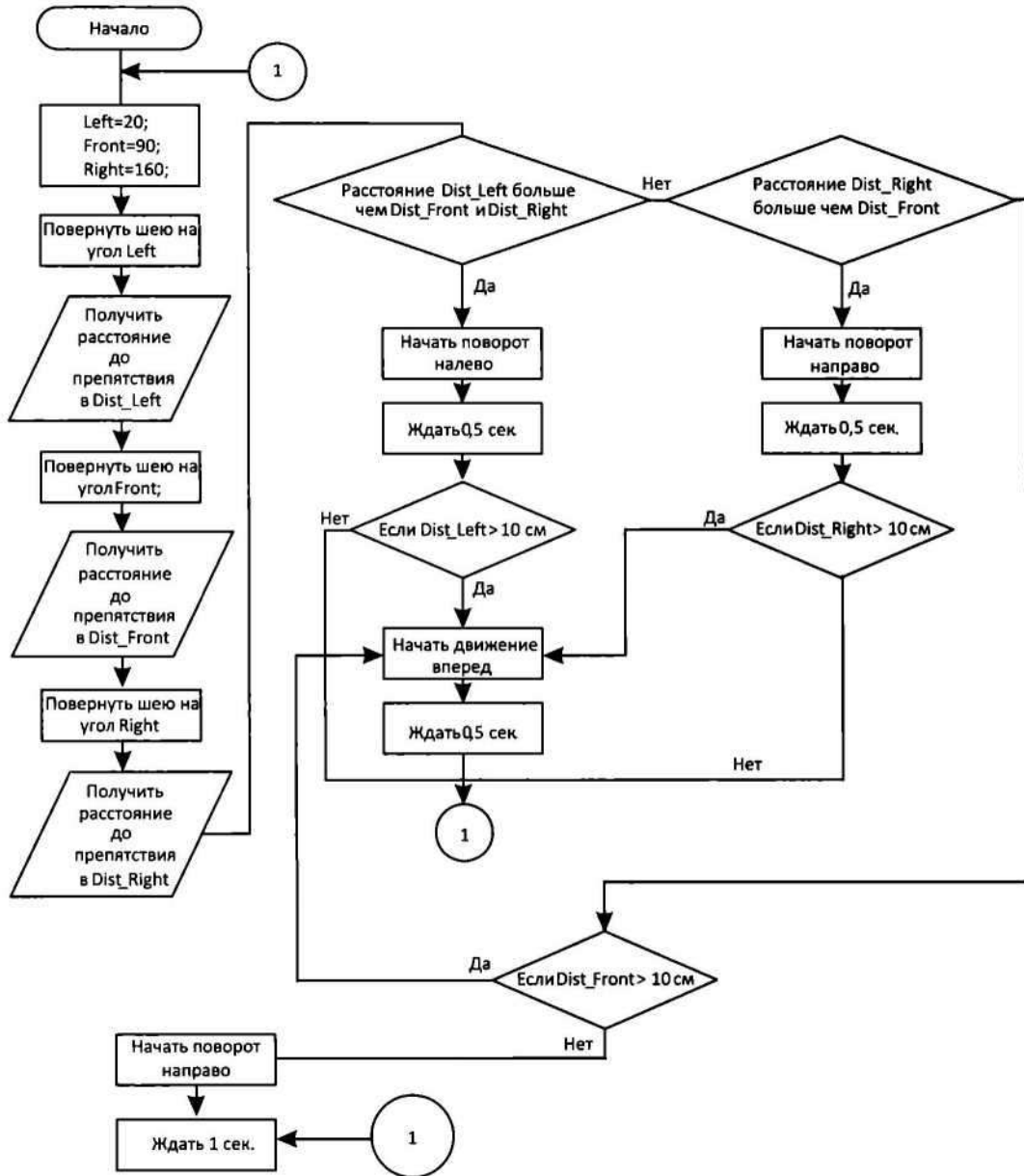


Рис. 11.1. Алгоритм программы обхода препятствий роботом

Задание. Реализуйте программно алгоритм движения, приведенный на рис. 11.1

Листинг 11.1. Программа обхода препятствий роботом

// Подключаем библиотеку управления сервомоторами.

```
#include <Servo.h>
```

// Подключаем библиотеку, управляющую моторками.

```
#include "motor.h"
```

// Подключаем библиотеку, управляющую датчиком расстояния.

```
#include "sonar.h"
```

// Создаем сервомотор поворота головы.

```
Servo neck;
```

// Константы - постоянные значения для уточнения углов.

```
const int left_ang = 168;
```

```
const int front_ang = 98;
```

```
const int right_ang = 28;
```

// Временные константы служат для точного задания времени на поворот,

// разворот, движение вперед

// в миллисекундах.

```
const int time_90 = 390;
```

```

const int time_180 = 750;
const int time_10cm = 220; void setup()
{
// Инициализируем дальномер Trig = 13, Echo = 12.
Sonar_init(13, 12);
// Инициализируем сервомотор, управление 9-м портом.
neck.attach(14);
// Переменные - номера контактов (пинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2, 3, 4, 5);
_stop(); //Двигатели остановлены.
}
// Основная программа.
void loop()
{
stop();
// Создаем переменные для хранения трех дистанций - слева, впереди, справа.
int Dist_left, Dist_front, Dist_right;
// Поворачиваем голову налево.
neck.write(left_ang);
// Ждем, т. к. поворот занимает небольшое время.
delay(150);
// Записываем расстояние до препятствия слева.
Dist_left = Sonar(400);
// Поворачиваем голову прямо вперед.
neck.write(front_ang);
// Ждем, т. к. поворот займет небольшое время.
delay(150);
// Записываем расстояние до препятствия впереди.
Dist_front = Sonar(400);
// Поворачиваем голову направо.
neck.write(right_ang);
// Ждем, т. к. поворот займет небольшое время.
delay(150);
// Записываем расстояние до препятствия впереди.
Dist_right = Sonar(400);
neck.write(left_ang);
// Если расстояние до препятствия слева наибольшее
if ((Dist_left > Dist_front) && (Dist_left > Dist_right))
{
left(); // поворачиваем налево 0,5 секунд.
delay(time_90);
if (Dist_left > 10)
{
forward(); // едем вперед 0,5 секунды.
delay(time_10cm);
}
}
else
{
if (Dist_right > Dist_front)
{
right(); // поворачиваем направо 0,5 секунд. delay(time_90);
}
}
}
}

```



```

if (Dist_right > 10)
{
forward(); // едем вперед 0,5 секунды.
delay(time_10cm);
}
else
{
if (Dist_front > 10)
{
forward(); // едем вперед 0,5 секунды. delay(time_10cm);
}
else
{
right(); // разворачиваемся. delay(time_180);
}
}
}
}
}

```

Отладка программы

Для тонкой настройки программы потребуется экспериментально определить ряд значений:

Значения углов поворота сервомотора для прямого, правого и левого взглядов.

Довольно сложно установить голову на вал точно прямо, но можно экспериментально подобрать и задать в программе нужный угол. Правый и левый повороты — это поворота головы направо и налево на 60-75 градусов, чего достаточно, чтобы робот смог определить расстояние до препятствия справа и слева соответственно. Высокая точность здесь не нужна, поскольку луч сонара довольно широкий, — около 30 градусов. Поверните сервомотор программно на 90 градусов. Если при этом робот смотрит прямо, примите значение угла для прямого взгляда `front_ang` равным 90 градусов. В противном случае уточните значение `front_ang` экспериментально.

Значения времени, за которые робот поворачивает на 90 градусов и на 180 градусов.

Здесь экспериментально определяется время, за которое робот выполняет эти движения. На разных типах поверхности это время может быть различным.

Время, за которое с места робот проезжает 10 сантиметров.

Полученные значения заносятся в программу как константы с интуитивно понятными названиями (табл. 11.1).

Значения функции `loop ()`, содержащей основные действия программы из листинга 11.1, можно для своего робота уточнить, если удалить из этой функции все строки, а затем поочередно вставлять в нее строки, представление в 4-м столбце табл. 11.1. Изменяя в программе значения указанных параметров, нужно добиться удовлетворительного результата.

Например, если команда `neck.write (front_ang)` приводит к тому, что робот смотрит немного вбок, то следует, понемногу увеличивая или уменьшая значение `const int front_ang=90`, добиться того, чтобы робот смотрел прямо. Полученное значение может отличаться от 90, и его следует запомнить и применить в программе. Аналогично надо поступить для поворота головы влево и вправо.

Для поворотов робота на 90 градусов нужно определить время, за которое робот совершает поворот на указанный угол. Стоит заметить, что это время будет отличаться для разного вида поверхностей, по которым движется робот. Если робот поворачивается на угол больший, чем 90°, следует это время уменьшать, а если угол поворота мал, то увеличивать. В программе используется поворот на 90 градусов, но можно применять и меньшие значения углов.

Время движения вперед/назад на заданное расстояние определяется аналогично. В программе это расстояние задано величиной 10 см, но вы можете его увеличить.

Таблица 11.1. Переменные и константы для программы обхода препятствий

№ п.п.	Переменная или константа	Назначение	Содержимое функции loop ()
1.	left_ang	Угол поворота сервопривода головы робота для поиска препятствий слева — 70° от прямого положения	neck.write(left_ang); delay(1000);
2.	front_ang	Угол поворота сервопривода головы робота для поиска препятствий прямо по ходу робота	neck.write(front_ang); delay(1000);
3.	right_ang	Угол поворота сервопривода головы робота для поиска препятствий справа — 70° от прямого положения	neck.write(right_ang); delay(1000);
4.	time_90	Время, за которое робот поворачивает на 90° из состояния покоя	left(); delay(time_90); _stop(); delay(10000);
5.	time_180	Время, за которое робот поворачивает на 180° из состояния покоя	left(); delay(time_180); _stop(); delay(10000);
№ п.п.	Переменная или константа	Назначение	Содержимое функции loop ()
6.	time_10cm	Время, за которое робот из состояния покоя преодолевает расстояние 10 см и полностью останавливается	forward(); delay(time_10cm); _stop(); delay(10000);

Проверку правильности функционирования робота рекомендуется производить в следующем порядке:

- 1) Снять с робота колеса.
- 2) Загрузить программу из листинга 11.1 и дополнить ее следующими тестовыми командами согласно листингу 11.2 (требуемые изменения в листинге 11.2 выделены полужирным шрифтом):
- 3) в разделе setup () организовать вывод в порт;
- 4) в основной программе loop () вставить в интересующих нас местах вывод служебной информации.
- 5) Запрограммировать робота и оставить его соединенным кабелем с ПК.
- 6) В Arduino IDE открыть окно Монитор порта.
- 7) Создавая перед роботом различные варианты препятствий, соответствующие определенной ситуации в программе, проверить правильность реакции робота по выводу сообщений в окно Монитор порта и по вращению колес, — робот при этом будет выдавать в порт расстояния до препятствий и этапы работы программы (какой блок выполняется).
- 8) Если робот действует не верно, проанализировать его работу и внести в программу необходимые правки.

Можно также скорректировать время, которое требуется роботу для поворота головы (в листинге 11.2 это время сильно увеличено до 1,5 сек.), — этого времени должно быть достаточно для поворота головы, но не приводить к простоя робота. В то же время, если голова еще вращается, а робот уже начал измерять расстояние до препятствия, результат измерения окажется неверным.

Листинг 11.2. Отладочная (неполная) программа обхода препятствий с выводом в порт

```
// Отладочная программа
void setup()
{
// Инициализируем дальномер Trig = 13, Echo = 12.
Sonar_init(13, 12);
// Инициализируем сервомотор, управление 9-м портом.
neck.attach(14);
```

```

// Переменные - номера контактов (пинов) Arduino.
// Для левых и правых моторов машинки.
setup_motor_system(2, 3, 4, 5);
_stop(); // Двигатели остановлены.
Serial .begin(9600);
}
// Основная программа.
void loop()
{
  stop();
  // Создаем переменные для хранения трех дистанций - слева, впереди, справа.
  int Dist_left, Dist_front, Dist_right;
  // Поворачиваем голову налево.
  neck.write(left_ang);
  // Ждем, т. к. поворот занимает небольшое время.
  delay(1500);
  // Записываем расстояние до препятствия слева.
  Dist_left = Sonar(400);
  Serial.print("Dist_left="); // оформляем вывод.
  // выводим дистанцию.
  Serial.print("Dist_left=") ;
  Serial.println(" см") ; // оформляем вывод.
  // приостанавливаем программу.
  delay(500);
  // Поворачиваем голову прямо вперед.
  neck.write(front_ang);
  // Ждем, т. к. поворот занимает небольшое время.
  delay(1500) ;
  // Записываем расстояние до препятствия впереди.
  Dist_front = Sonar(400);
  Serial .print ("Dist_front=") ; // оформляем вывод.
  // выводим дистанцию.
  Serial.print(Dist_front);
  Serial.println(" см"); // вывод.
  // приостанавливаем программу.
  delay(500);
  // Поворачиваем голову направо.
  neck.write(right_ang);
  // Ждем, т. к. поворот занимает небольшое время.
  delay(1500);
  // Записываем расстояние до препятствия впереди.
  Dist_right = Sonar(400);
  Serial.print("Dist_right=") ; // оформляем вывод
  // выводим дистанцию.
  Serial .print (Dist_right) ;
  Serial.println(" см"); // оформляем вывод.
  // приостанавливаем программу.
  delay(500);
  neck.write(front_ang);
  // оформляем вывод.
  Serial.println(" _____ " ) ;
  // Если расстояние до препятствия слева наибольшее.
  if ((Dist_left > Dist_front) && (Dist_left > Dist_right))

```

```

{
// оформляем вывод.
Serial.println (" (Dist_left>Dist_front) && (Dist_left>Dist_right) POVOROT LEFT" );
// приостанавливаем программу.
delay(500);
left(); // поворачиваем налево 0,5 секунд.
delay(time_90);
if (Dist_left > 10)
{
// оформляем вывод.
Serial.println(" Dist_left>10  VPERED 0.5 SEC");
// приостанавливаем программу.
delay(500);
forward(); // едем вперед 0,5 секунды. delay(time_10cm);
}
}
else
{
if (Dist_right > Dist_front)
{
// оформляем вывод.
Serial.println(" Dist_right>Dist_front POVOTOROT RIGHT" );
// приостанавливаем программу.
delay(500);
right(); // поворачиваем направо 0,5 секунд.
delay(time_90); if (Dist_right > 10)
{
// оформляем вывод.
Serial.println (" Dist_right>10  VPERED 0.5 SEC" );
// приостанавливаем программу.
delay(500);
forward(); // едем вперед 0,5 секунды.
delay(time_10cm);
}
}
}
else
{
if (Dist_front > 10)
{
// оформляем вывод.
Serial.println(" Dist_front>10 VPERED 0.5 SEC");
// приостанавливаем программу.
delay(500);
forward(); // едем вперед 0,5 секунды. delay(time_10cm) ;
}
}
else
{
// оформляем вывод.
Serial.println(" right *****RAZVOROT*****" );
// приостанавливаем программу.
delay(500);
right(); // разворачиваемся. delay(time_180);
}
}

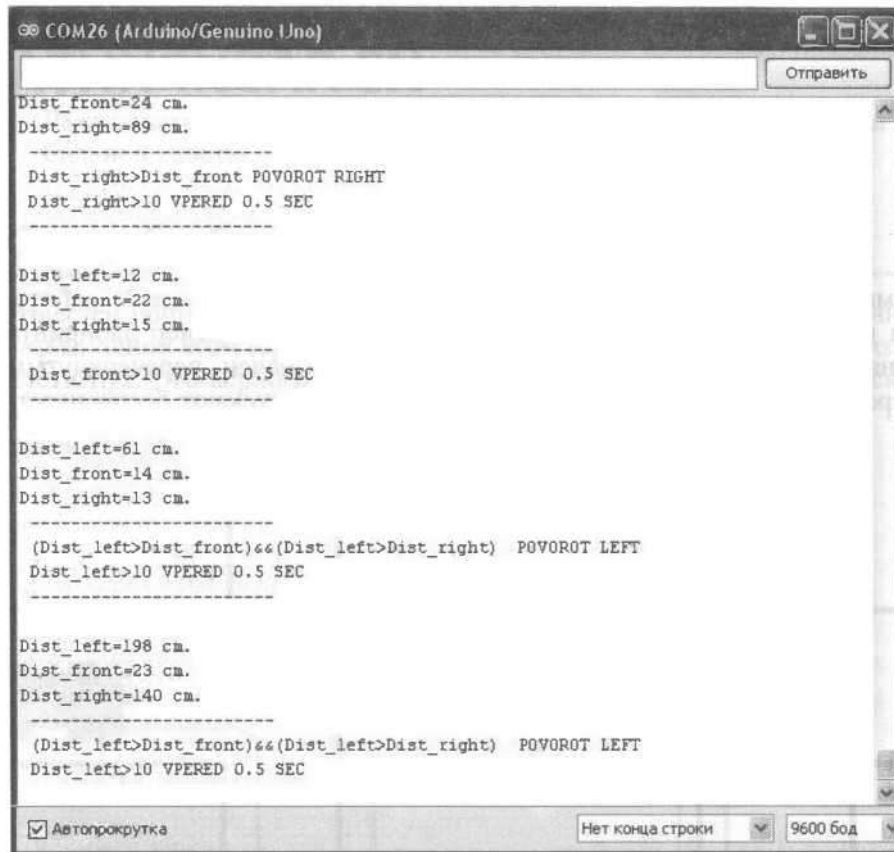
```

```
)  
}  
}
```

Пример вывода сообщений в окно монитора приведен на рис. 11.2.

Здесь наглядно видно, что когда слева 12 см, прямо 22 см, а справа 15 см, то самое большое расстояние впереди (и оно больше 10 см), и, соответственно, выполняется команда «Ехать вперед». Удостоверьтесь, что при этом моторы действительно вращаются вперед.

Когда же слева 61 см, прямо 14 см, а справа 13 см, то самое большое расстояние слева (и оно больше 10 см), и, соответственно, выполняются команды «Повернуть влево» и «Ехать вперед». Удостоверьтесь, что при этом моторы вращаются согласно командам.



```
COM26 (Arduino/Genuino Uno)
Отправить
Dist_front=24 cm.
Dist_right=89 cm.
-----
Dist_right>Dist_front POVOROT RIGHT
Dist_right>10 VPERED 0.5 SEC
-----
Dist_left=12 cm.
Dist_front=22 cm.
Dist_right=15 cm.
-----
Dist_front>10 VPERED 0.5 SEC
-----
Dist_left=61 cm.
Dist_front=14 cm.
Dist_right=13 cm.
-----
(Dist_left>Dist_front)&&(Dist_left>Dist_right) POVOROT LEFT
Dist_left>10 VPERED 0.5 SEC
-----
Dist_left=198 cm.
Dist_front=23 cm.
Dist_right=140 cm.
-----
(Dist_left>Dist_front)&&(Dist_left>Dist_right) POVOROT LEFT
Dist_left>10 VPERED 0.5 SEC
-----
 Автопрокрутка
Нет конца строки
9600 бод
```

Рис. 11.2. Пример вывода сообщений в окно монитора

Форма представления результата:

Отчет по работе должен содержать:

1. наименование работы и цель работы;
2. результаты работы;
3. выводы по работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно и полностью.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.