ВВЕДЕНИЕ

Дорогие робототехники, данное техническое руководство создано с целью «подружить» конструктор CLASSIC с ПЕРВЫМ изданием моей книги «Мобильные роботы на базе Arduino». В нем будут рассмотрены особенности конструктора CLASSIC при конструировании роботов по рассматриваемым в книге проектам.

Конструктор CLASSIC комплектуется оригинальными деталями корпуса, основан на использовании платы Arduino NANO (в книге делается упор на Arduino UNO), сервисной плате Arduino Nano Shield V3.0. Также оригинальную конструкцию крепления имеет ряд используемых датчиков.

Рассмотрим проекты из книги, которые относятся непосредственно к четырехколесному, четырехмоторному роботу.

Я уверен, что сборка роботов на основе конструктора CLASSIC по проектам из моей книги «Мобильные роботы на базе Arduino» позволит вам хорошо провести время, получить новые знания и умения, почувствовать себя настоящим Конструктором.

С уважением, Михаил Момот.

К ГЛАВЕ 1

Дополнительно к перечню электронных компонентов, рассмотренному в главе 1, стоит рассмотреть еще два — это контроллер Arduino Nano 3.0 и сервисная плата Arduino Nano Shield V3.0, при помощи которых осуществляется управление роботом CLASSIC.

Контроллер Arduino NANO 3.0

Плата Arduino NANO 3.0 отличается от Arduino UNO размерами, она немного меньше и, как правило, имеет два дополнительных аналоговых входа A6 и A7. Для программирования на плате имеется разъем Mini-USB, а контакты выполнены в виде штырьков. На плате Arduino Nano 3.0 отсутствует отдельный разъем внешнего питания. Других принципиальных различий нет.

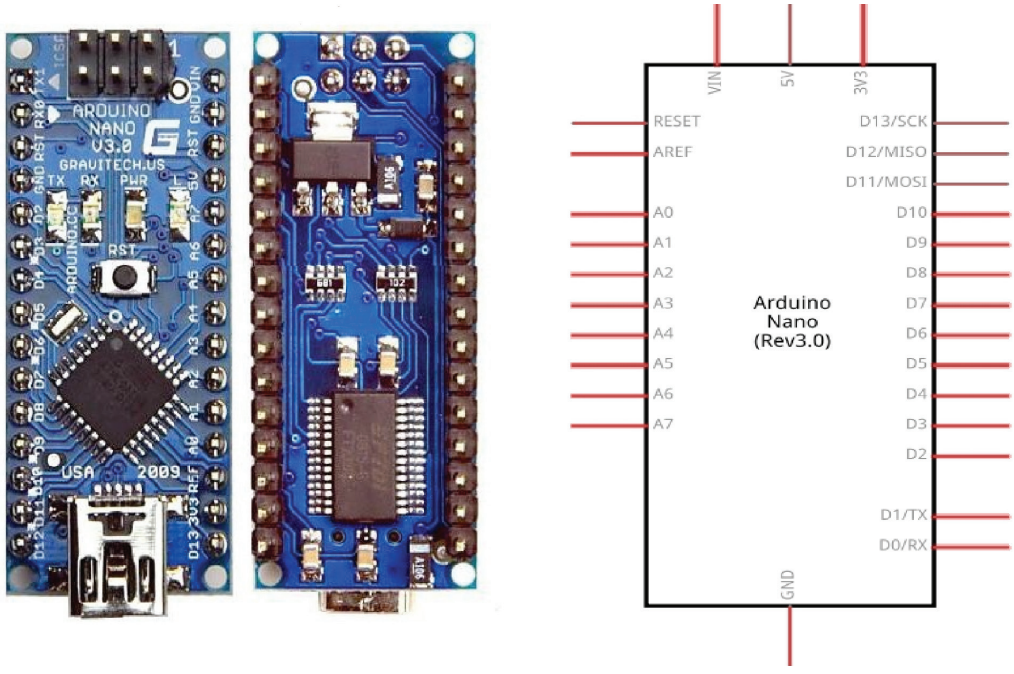


Рис. К1.1. Плата контроллера Arduino NANO

Сервисная плата Arduino Nano Shield V3.0

Плата Arduino Nano Shield V3.0 служит для более удобной работы с контроллером Arduino NANO, если не критично используемое платой пространство. На плате присутствует разъем для установки платы Arduino NANO, контакты которой разведены на сервисную плату в виде штырьковых разъемов.

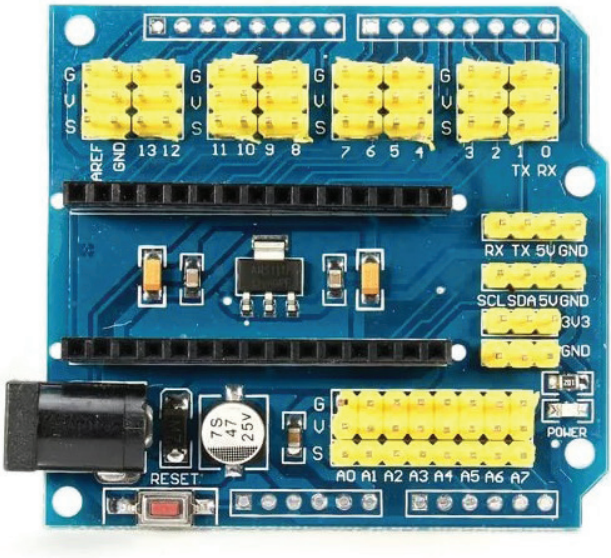


Рис. К1.2. Сервисная плата Arduino Nano Shield V3.0

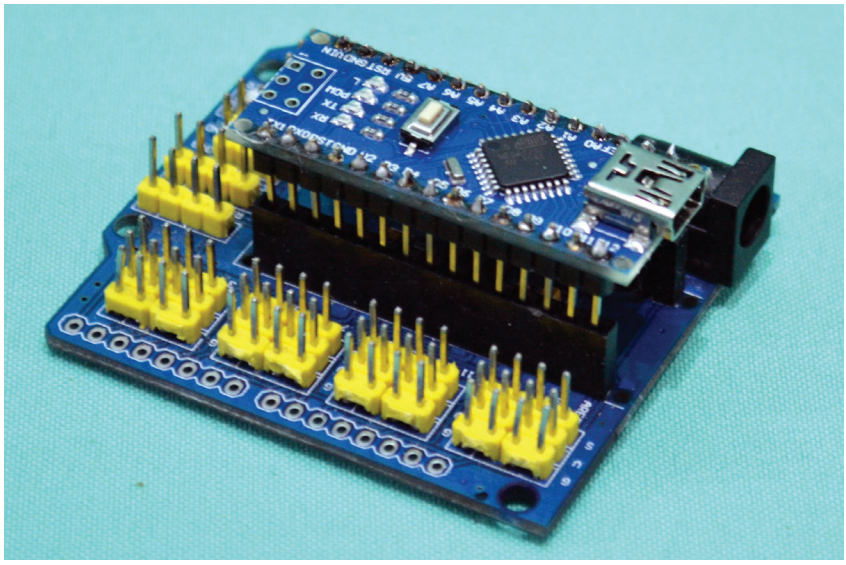


Рис. К1.3. Arduino NANO подготовлена к установке в разъем Arduino Nano Shield V3.0

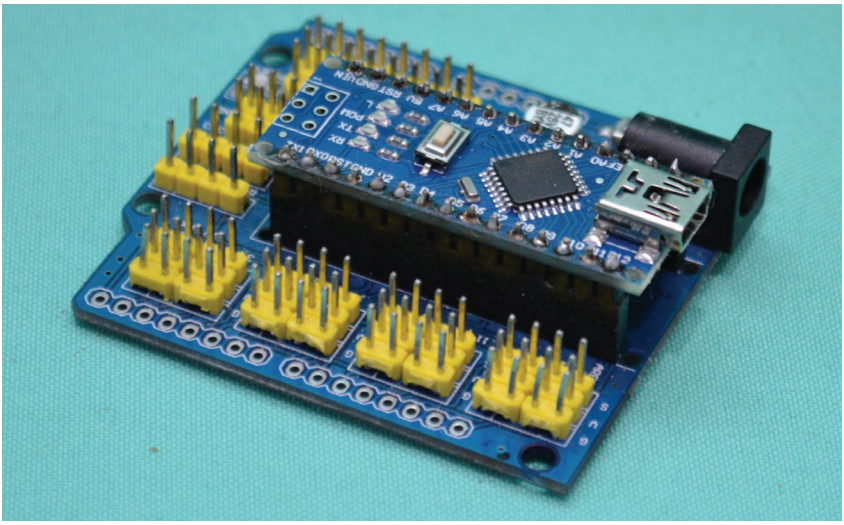


Рис. К1.4. Arduino NANO подключена к Arduino Nano Shield V3.0

Arduino Nano Shield V3.0 имеет собственный стабилизатор питания, который при подключении нестабилизированного питания не менее 6 вольт работает в паре со стабилизатором, расположенным на плате Arduino NANO, что увеличивает выходную мощность на линии 5 вольт (основное электропитание платы Arduino NANO, датчиков, маломощных сервомоторов, плат расширения).

Если установить с нижней стороны платы штырьки в контакты для пайки, то сервисную плату Arduino Nano Shield V3.0 можно подключать к Arduino UNO, эти контакты можно также использовать для подключения информационных входов устройств, для которых удобнее пайка.

Контактные площадки для подключения внешних устройств представлены тремя пинами (G, V, S), где G – это земля, или отрицательный контакт питания, V – 5В (стабилизированное напряжение +5 вольт), а S – сигнальный контакт, номер которого соответствует номеру на плате Arduino NANO.

Дополнительно на плате отдельно разведен Serial port (порт для последовательного обмена), он может быть использован для подключения устройств, работающих по данному порту, но при этом теряется возможность программировать Arduino NANO через USB-разъем.

I2C – это порт, к которому можно подключать несколько устройств параллельно. Стоит отметить, что контакт SCL – это пин A5, а контакт SDA – A4.

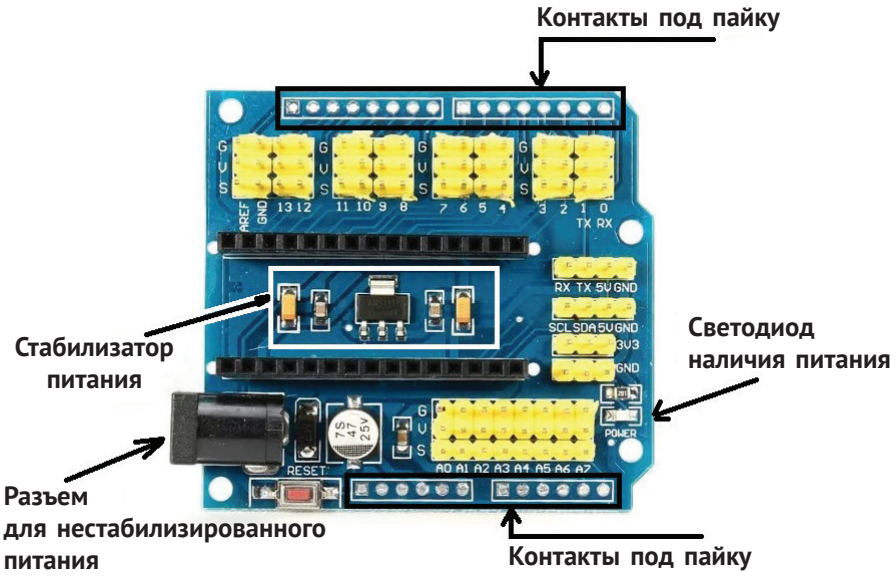


Рис. К1.5. Arduino Nano Shield V3.0 с пояснениями 1

Рис. К1.6. Arduino Nano Shield V3.0 с пояснениями 2

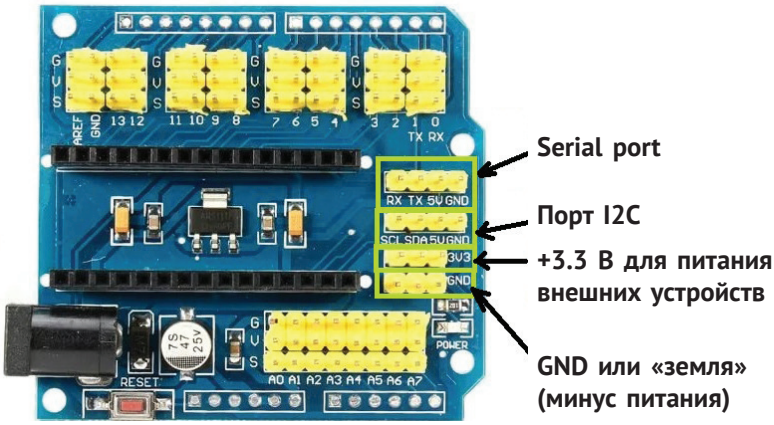


Рис. K1.7. Arduino Nano Shield V3.0 с пояснениями 3

Выводы

Выше приведены основные электронные компоненты конструктора CLASSIC, отличающие его от набора, использованного в книге. Конструкционные особенности конструктора CLASSIC будут рассмотрены далее.

К ГЛАВЕ 5

В пункте «Драйвер» пятой главы объясняется, как происходит управление моторами при помощи специального блока – драйвера двигателей. Для того чтобы проверить, как осуществляется управление работой моторов при помощи программы на Arduino, создадим ряд несложных макетов.

Сборка макета

Учитывая тот факт, что не все владеют навыками пайки или не имеют минимального паяльного оборудования, мы комплектуем конструктор предварительно спаянными узлами, чтобы вы могли собрать конструкцию без пайки, исключительно пользуясь отвертками, пассатижами, ключиком для гаек. Но в описании будет рассказано, как собрать макет и при условии неспаянных мотора, редукторов, отсека питания, тумблера и разъема.

Для сборки макета вам потребуются два мотора с редукторами с припаянными к ним конденсаторами и многожильными проводами (1). Система управления базируется на Arduino NANO (4), которая будет установлена в сервисную плату Arduino Nano Shield V3.0 (5). Драйвер моторов (3) будет подключен к питанию от бокса аккумуляторов с припаянным выключателем (6). Питание на плату Arduino Nano Shield V3.0 подается через разъем (2). Управление драйвером моторов от Arduino NANO осуществляется через сервисную плату Arduino Nano Shield V3.0 соединением ее контактов посредством проводов с разъемами Dupont «мама» с обеих сторон (8). Под номером 7 представлены литиевые аккумуляторы на 3.7 В формата 18650.

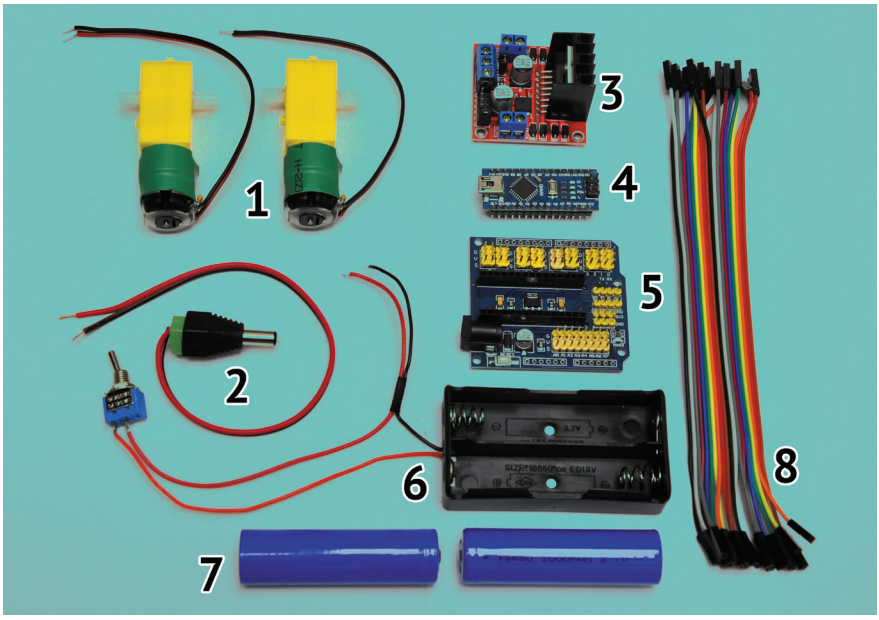


Рис. К5.1. Комплект для сборки макетов по 5-й главе

Моторы с редукторами согласно соединяем с проводами и керамическими конденсаторами (рис. К5.2). При этом капля припоя должна лежать только на контакте мотора и не касаться стального корпуса двигателя.

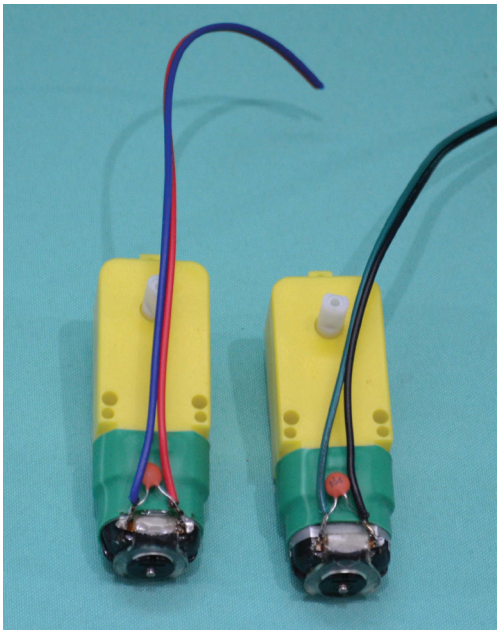


Рис. К5.2. Моторы постоянного тока с редукторами, проводами питания и гасящими помехи керамическими конденсаторами

Аккумуляторный бокс соединен пайкой с выключателем, как показано на рис. К5.3. Красный провод (от положительного контакта бокса) припаян к одному контакту выключателя, а к другому контакту припаян красный провод, который пойдет на драйвер моторов к плюсовому зажиму. Туда же, но к минусовому контакту клемника подключается черный (минусовой) провод от аккумуляторного бокса. В клемник драйвера моторовов вставляются провода, оканчивающиеся штекером, подключаемым к Arduino Nano Shield V3.0. Особое внимание уделяйте соблюдению полярности подключения. Монтажная схема полученных соединений представлена на рис. К5.4, после распайки проверьте по ней соединения.

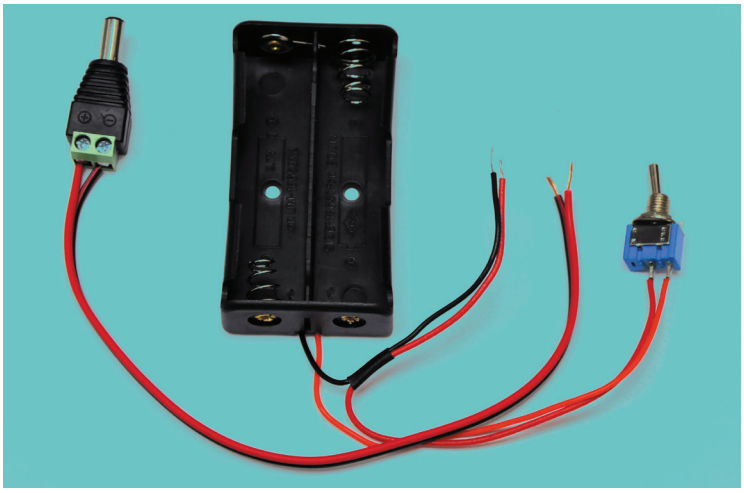


Рис. К5.3. Аккумуляторный бокс с распаянным выключателем и разъемом для питания платы Arduino Nano Shield V3.0. Одноцветные провода попарно вставляются в драйвер мотора и затягиваются винтами

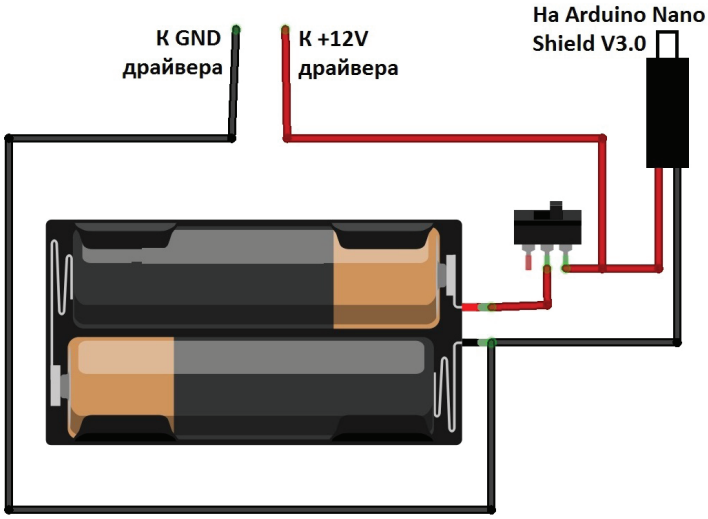


Рис. К5.4. Монтажная схема электропитания стенда

Теперь можно подключить провода от двигателей к драйверу, пайка не потребуется, выходы OUT1 и OUT2 имеют клеммы с винтовым зажимом (рис. К5.5).

Проводники разъема, питающего Arduino Nano Shield, с одной стороны закрепляются винтами в штекере, с другой стороны – винтами в драйвере моторов.

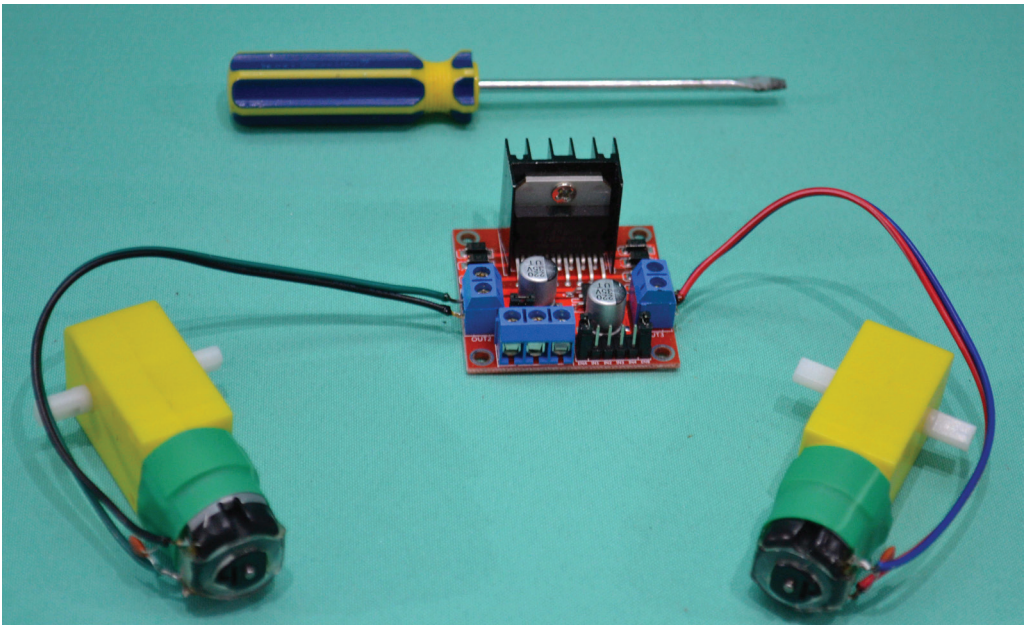


Рис. К5.5. Подключение моторов к драйверу

Управляем двигателем без Arduino

Подсоединяем питание к драйверу (рис. К5.6–К5.8) и поочередно подключаем второй контакт провода подключенного на клемму 5V драйвера к ножкам управления IN1–IN4. Наблюдаем, что при подключении провода к контактам IN1 и IN2 работает мотор, подключенный на клеммы OUT1, OUT2, а при подключении провода от 5V к IN3 и IN4 включается двигатель на клеммах OUT3, OUT4. При этом изменяется направленность вращения.

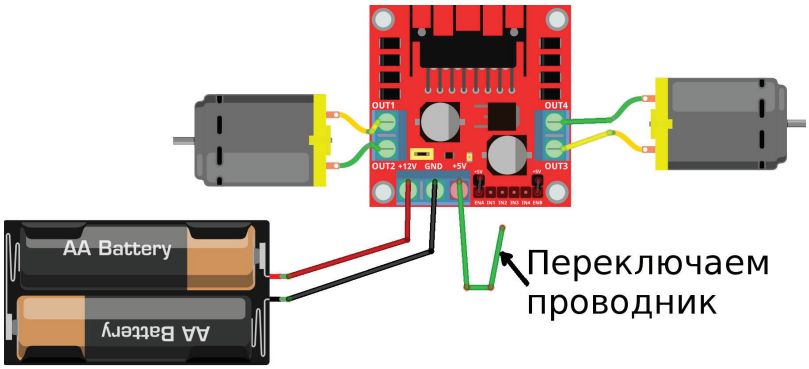


Рис. К5.6. Тестовый стенд для управления работой двигателей без контроллера Arduino

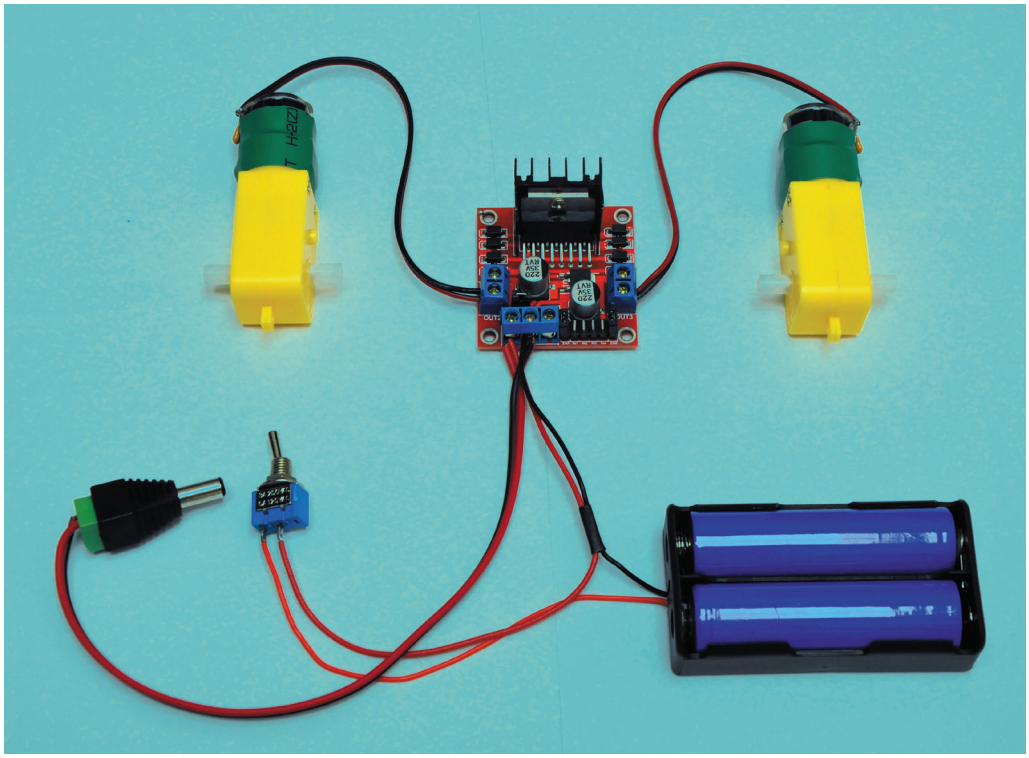


Рис. К5.7. Тестовый стенд для управления работой двигателей без контроллера Arduino (фото)

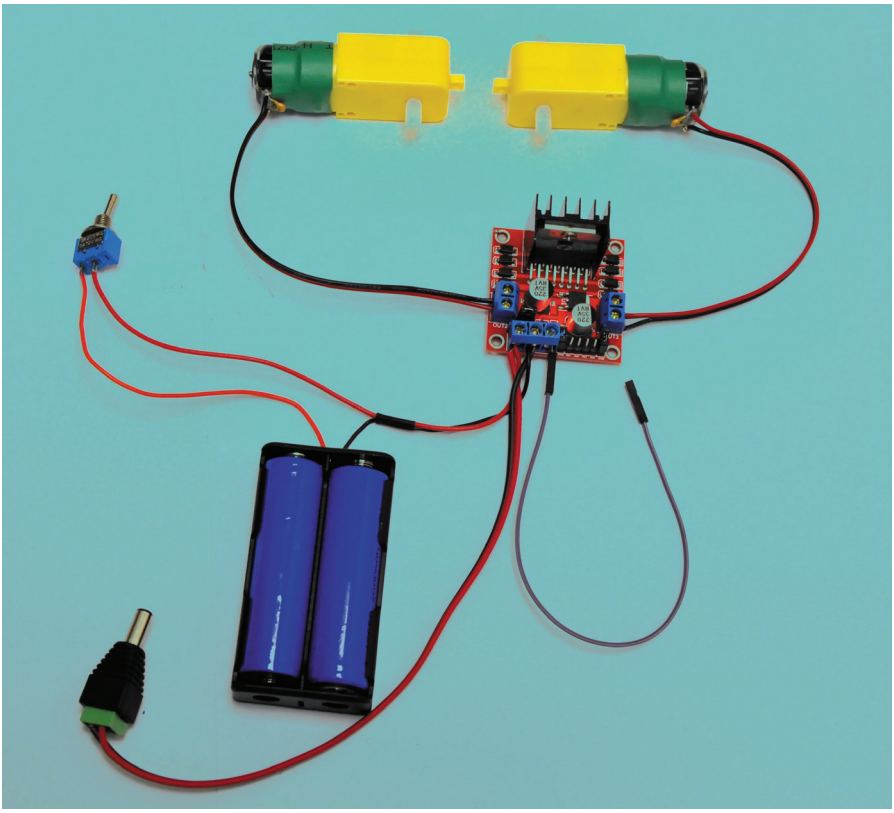


Рис. К5.8. К винтовому разъему (+5V) драйвера подключаем провод с разъемом «папа» и разъемом «мама» с другого конца

К ГЛАВЕ 6 СБОРКА БАЗОВОЙ МОДЕЛИ

Минимальный комплект

Начнем сборку робота с ходовой части, компоненты для сборки ходовой части робота из конструктора CLASSIC изображены на рис. К6.1.

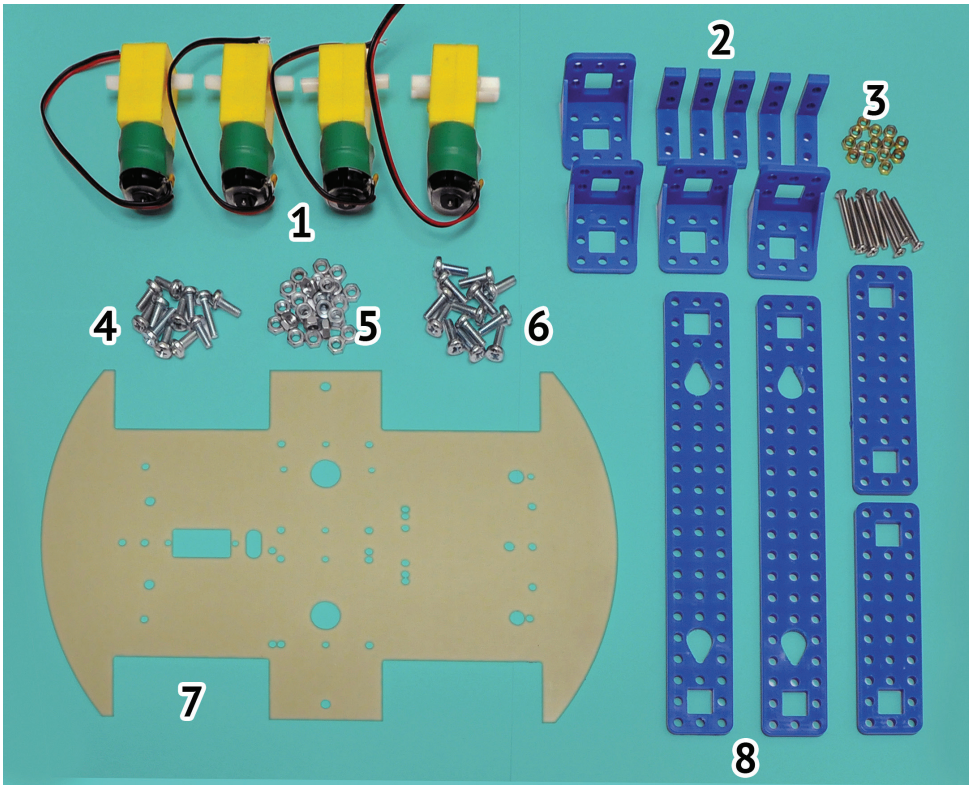


Рис. К6.1. Набор деталей и комплектующих нижнего уровня: двигатели с припаянными гибкими проводами и керамическими конденсаторами (1); уголки для сборки моторного отдела и крепления его к корпусу (2); 3-мм винты с потайной головкой для крепления двигателей с редукторами (3); 4-мм винты и гайки крепления моторного отдела к корпусу (4), (5), (6); несущая часть корпуса (7); конструкционные детали моторного отдела (8)

Начнем сборку с моторов. Как припаять провода[1], подробно рассмотрено в 6-й главе книги, но для сокращения количества проводов проведем небольшую модернизацию и к парным моторам, которые еще не распаяны при изготовлении макета к 5-й главе, припаяем конденсаторы и короткие провода (около 3 см).

Установим моторы, как показано на следующем рисунке (рис. К6.2), контактными клеммами друг к другу.

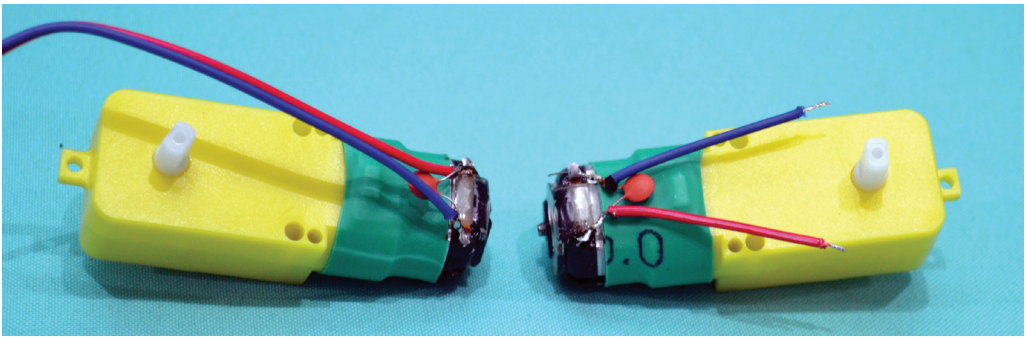


Рис. К6.2. Пара моторов: слева с длинными проводами (10–15 см), справа с короткими (3–4 см)

Теперь, как показано на следующем рисунке, перекрестно припаяем провода правого мотора к клеммам левого. Для сокращения числа проводов так и надо сделать. Однако если вы не владеете навыками пайки, то можно обойтись и без нее. Длины проводов достаточно для подключения моторов к драйверу. В винтовые клеммы будете вставлять не по одному, а по два провода.

Обращаю внимание!!! Если моторы распаять, не перекрещивая проводников, то их валы (колеса) будут вращаться в разные стороны.

Полученная пара моторов должна выглядеть как на рис. К6.3.

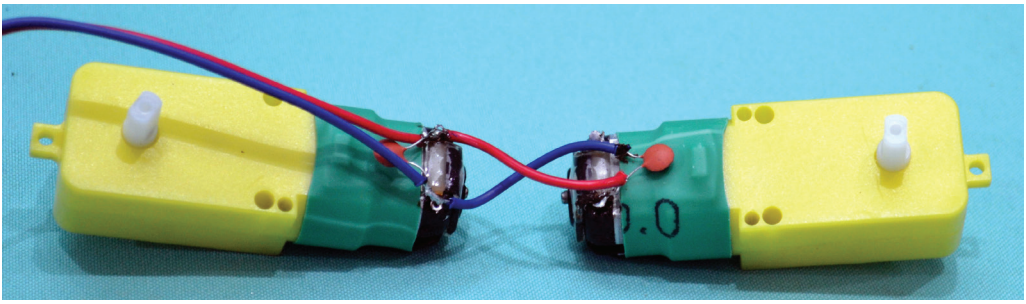


Рис. К6.3. Распаянная пара моторов

Теперь моторы подготовлены к установке в робота.

---

[1] В некоторых версиях конструкторов моторы идут с уже припаянными конденсаторами и проводами, что существенно облегчает и ускоряет сборку робота.

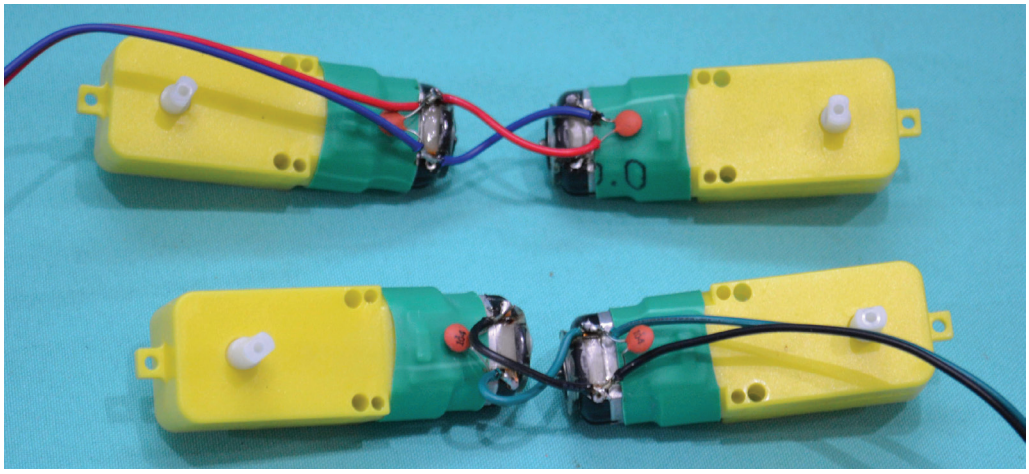


Рис. К6.4. Моторы подготовлены к установке в робота

Следом соберем рамку, основная роль которой – служить креплением двигателей к корпусу робота. Рамка состоит из четырех усиленных уголков, двух коротких конструкционных пластин и двух длинных. Рамка свинчивается при помощи 4-мм винтов длиной не менее 12 мм.

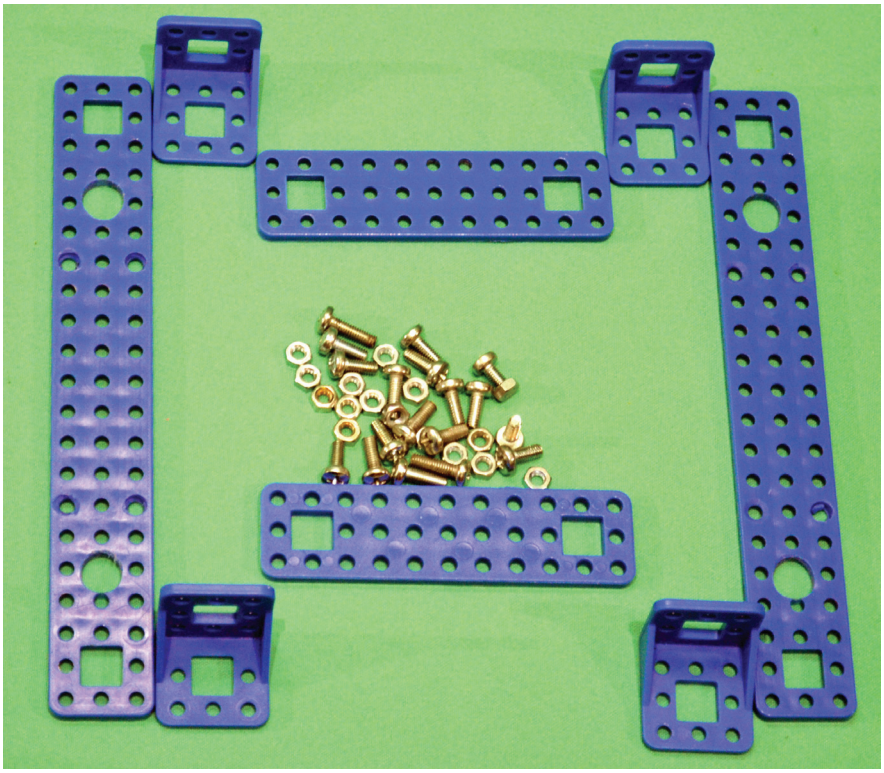


Рис. К6.5. Конструкционные детали крепления моторов с редукторами

Прикрутим уголки к коротким пластинам согласно рис. К6.6, сторона уголка с тремя отверстиями должна прилегать к короткой конструкционной пластине. Достаточно использовать по два винта на уголок.

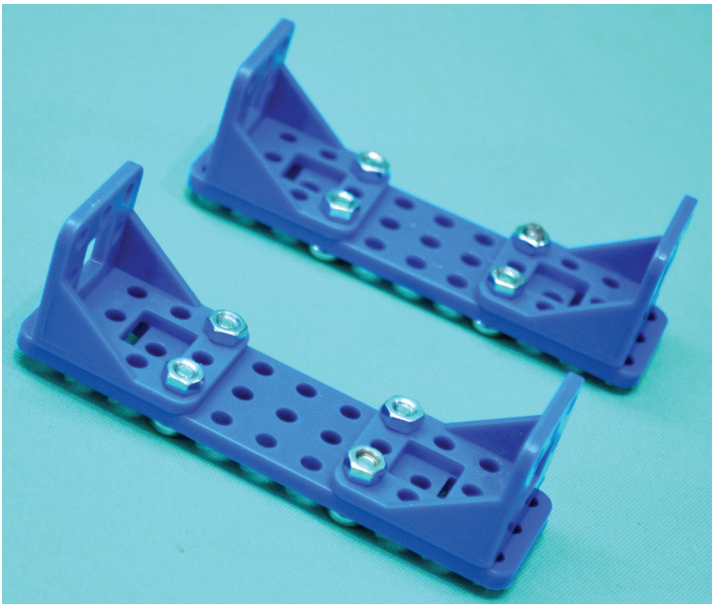


Рис. К6.6. Уголки прикручены к коротким соединительным пластинам

По рис. К6.7 соединим короткие и длинные пластины через уголки винтами. Полученная конструкция изображена на рис. К6.8.

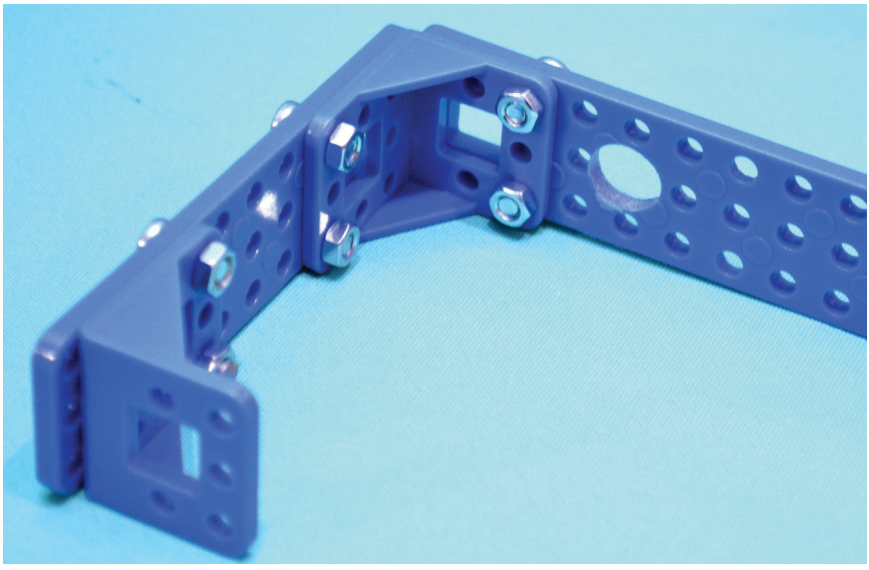


Рис. К6.7. Короткая и длинная пластины соединены уголком

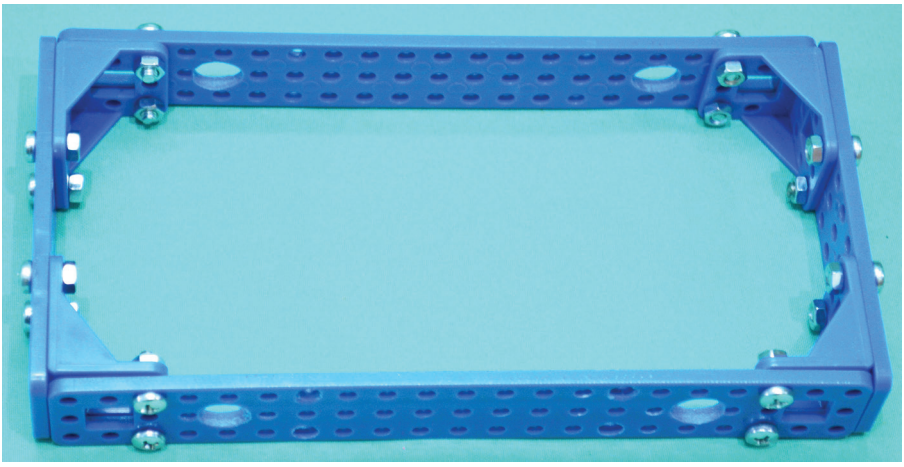


Рис. К6.8. Рамка для установки моторов собрана

Далее прикрепим к рамке уголки, которые в дальнейшем послужат креплением ее к корпусу робота (рис. К.6.9). В качестве подсказки используйте рис. К6.10-12.

Если данные уголки не закрепить, то после установки моторов это будет сделать затруднительно!

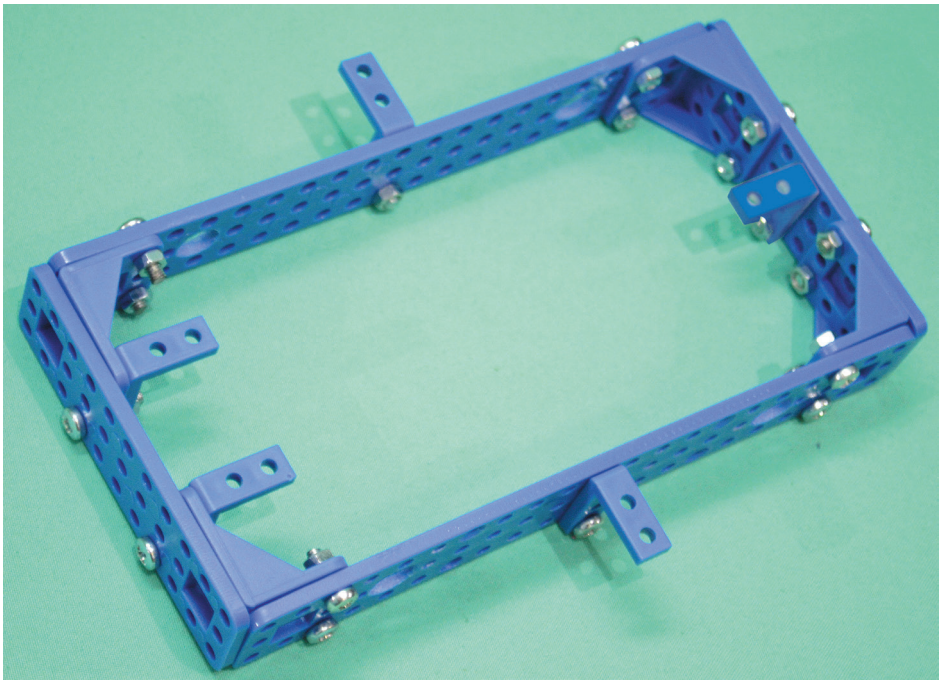


Рис. К6.9. Рамка с установленными уголками, фиксирующими ее на корпусе

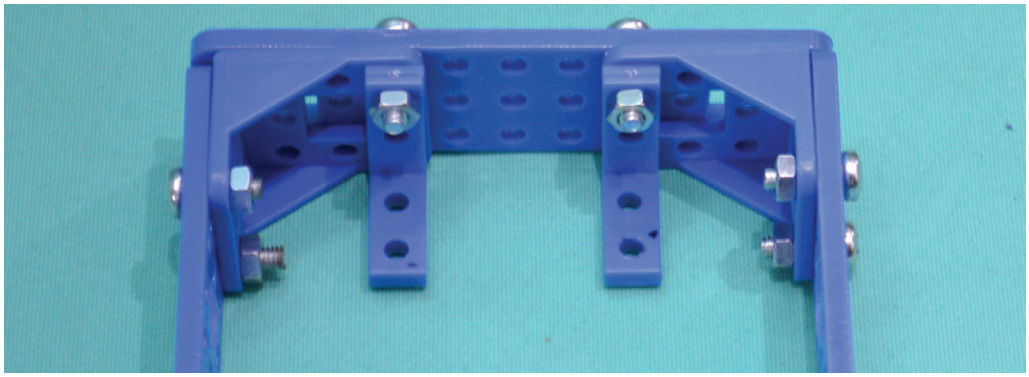


Рис. К6.10. Крепление уголков в передней части рамки

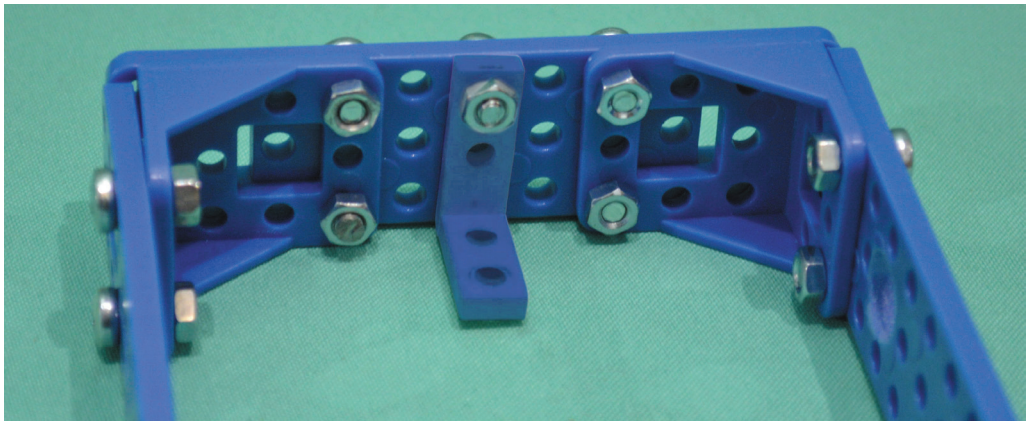


Рис. К6.11. Крепление уголка в задней части рамки

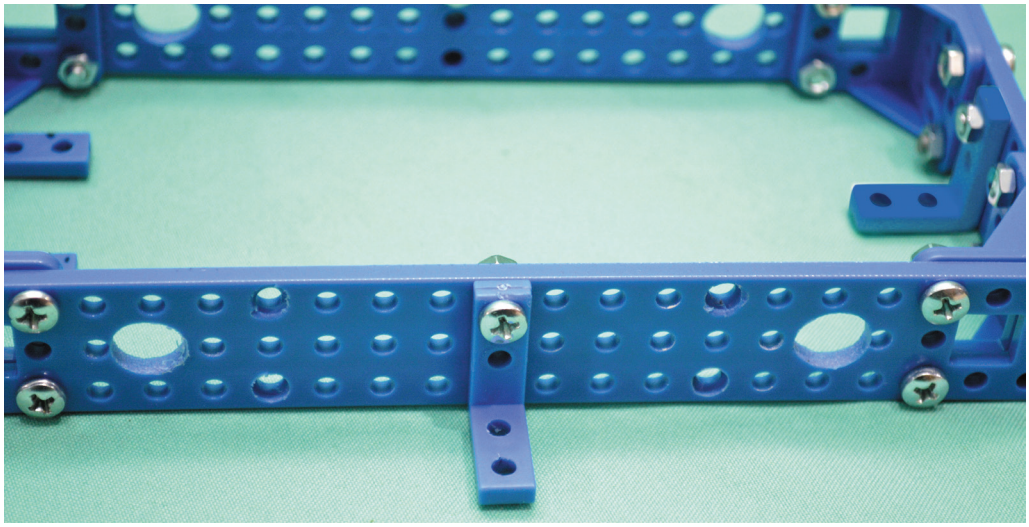


Рис. К6.12. Крепление боковых уголков к рамке

Прикрепим моторы с редукторами к моторной рамке. Моторы будут находиться внутри рамки, как показано на рис. К6.13. Крепить моторы следует 3-мм винтами длиной 35 мм. Для этого в комплекте имеются восемь винтов с потайной головкой.

Подробнее о креплении можно узнать из рис. К6.14, К6.15.

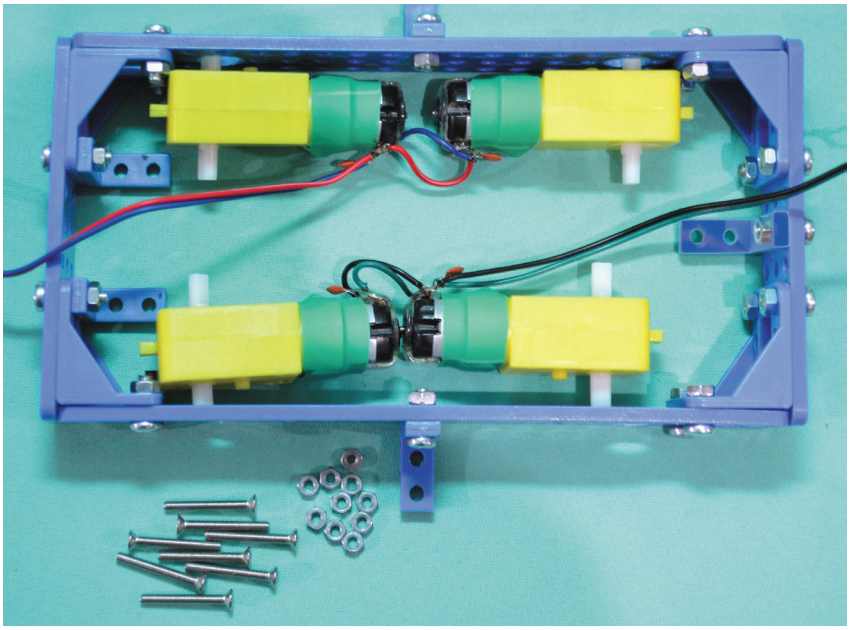


Рис. К6.13. Двигатели подготовлены для установки в рамку на винты М3 длиной 25 мм с потайной головкой

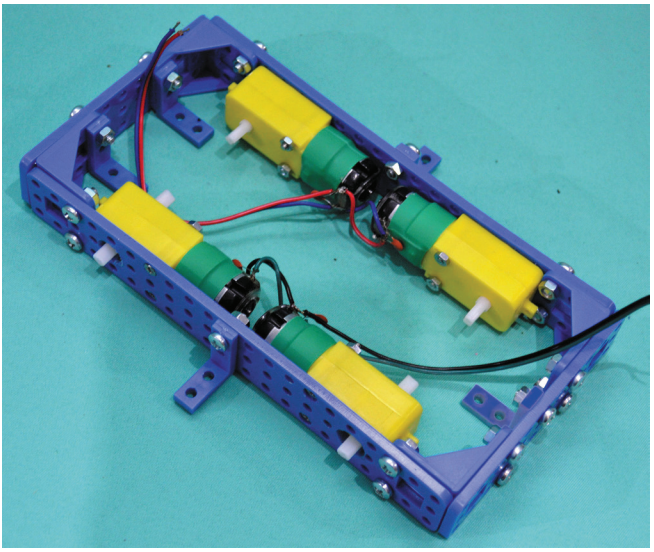


Рис. К6.14. Моторы закреплены на рамке (вид сверху)

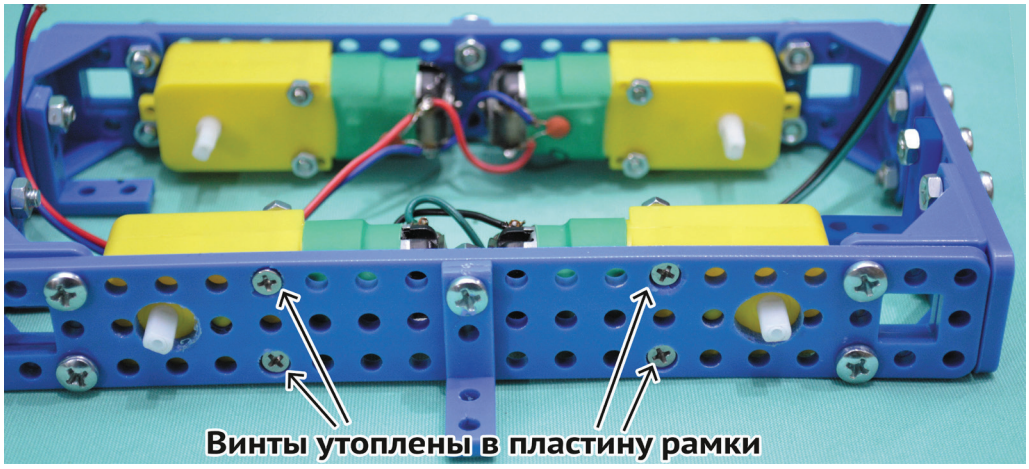


Рис. К6.15. Моторы закреплены на рамке (вид сбоку). Винты утоплены в отверстия со снятой фаской

Пластиковые насадки на вал редуктора с внутренней стороны робота придется снять, для этого следует использовать плоскогубцы. После того как пластиковая насадка снята, может возникнуть ситуация, когда стальной вал вышел вслед за пластиковой насадкой, в этом случае вал нужно утопить обратно, нажав на него плоскогубцами или другим твердым предметом. Вал не должен выступать из редуктора белее чем на 5 мм (рис. К6.17).

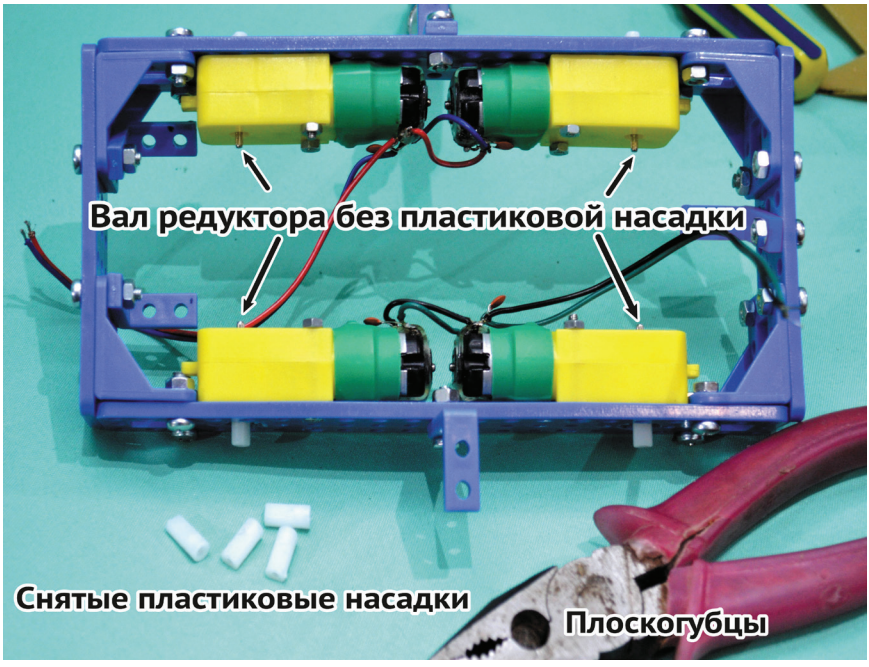


Рис. К6.16. Снятие пластиковых насадок с валов редукторов моторов с внутренней стороны рамки

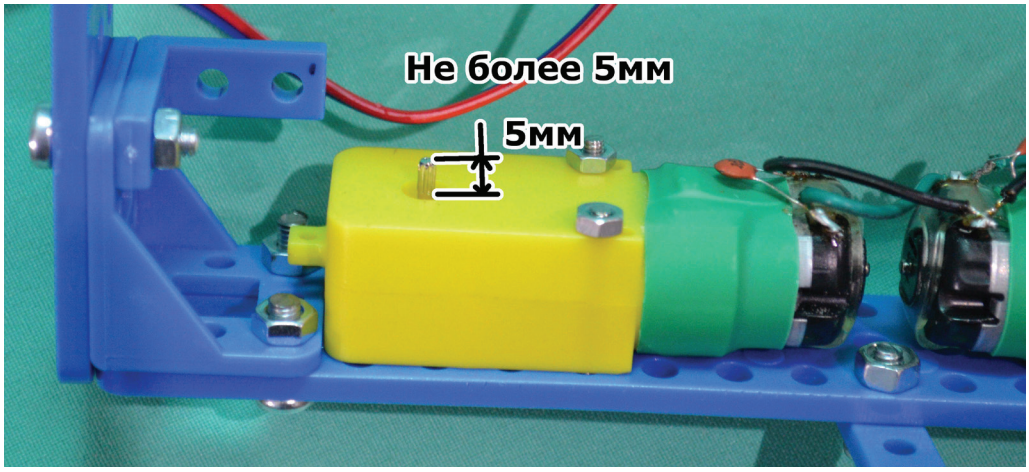


Рис. К6.17. Правильно утопленный в редуктор стальной вал

Теперь подключим электрическое питание.

Для этого потребуются провода, бокс для аккумуляторов, выключатель питания типа тумблер, круглый штекер. Для ускорения сборки робота необходимая пайка компонентов (тумблера, бокса, проводов) уже произведена. Красный провод традиционно подключается к положительному контакту питания, а черный – к отрицательному. Перечень других элементов приведен на рис. К6.18.

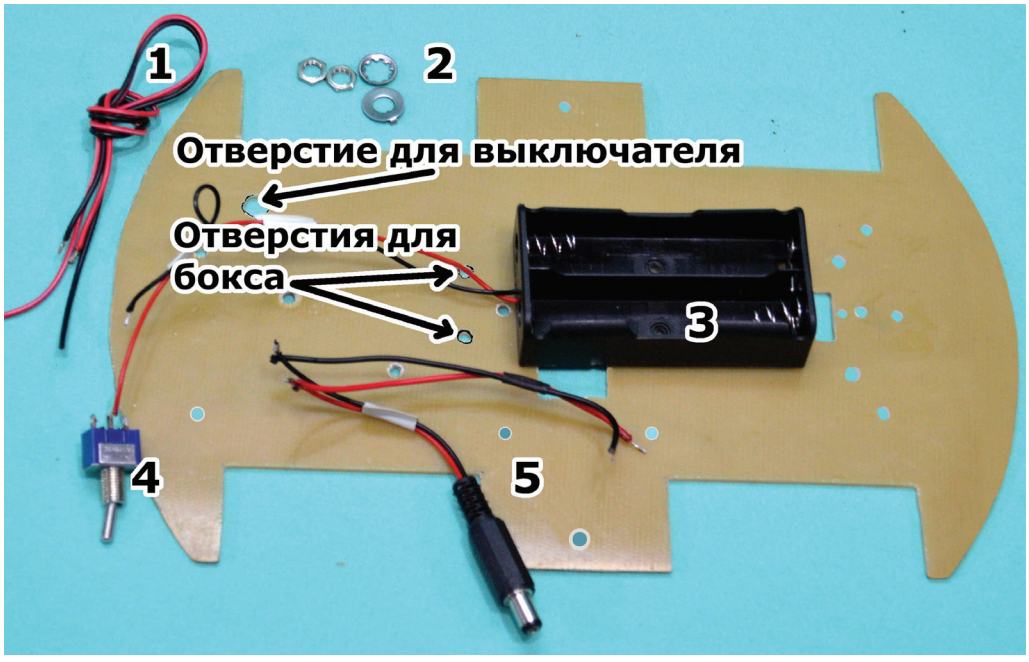


Рис. К6.18. Монтаж системы электропитания робота: дополнительные провода (1); крепеж выключателя/тумблера (2); аккумуляторный бокс (3); выключатель/тумблер (4); разъем для Arduino Nano Shield V3.0 (5)

Бокс для элементов питания прикрутим к корпусу при помощи двух 2.5-мм винтов впотай длиной 6 мм. Для этого в боксе и корпусе имеются соответствующие отверстия.

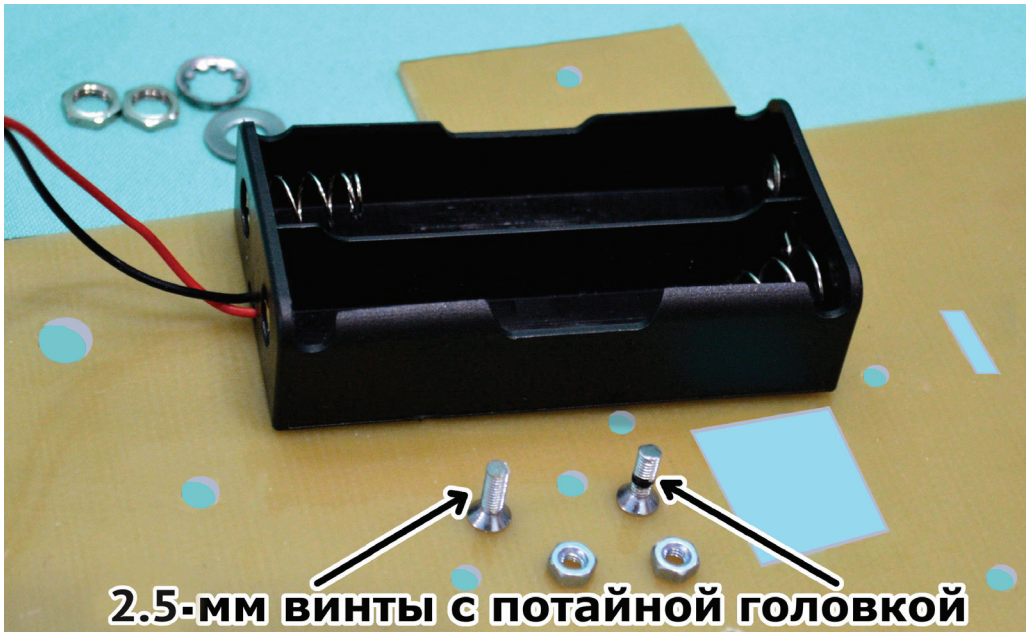


Рис. К6.19. Установка аккумуляторного бокса с использованием 2.5-мм винтов с потайной головкой (длина 6 мм) на корпус робота

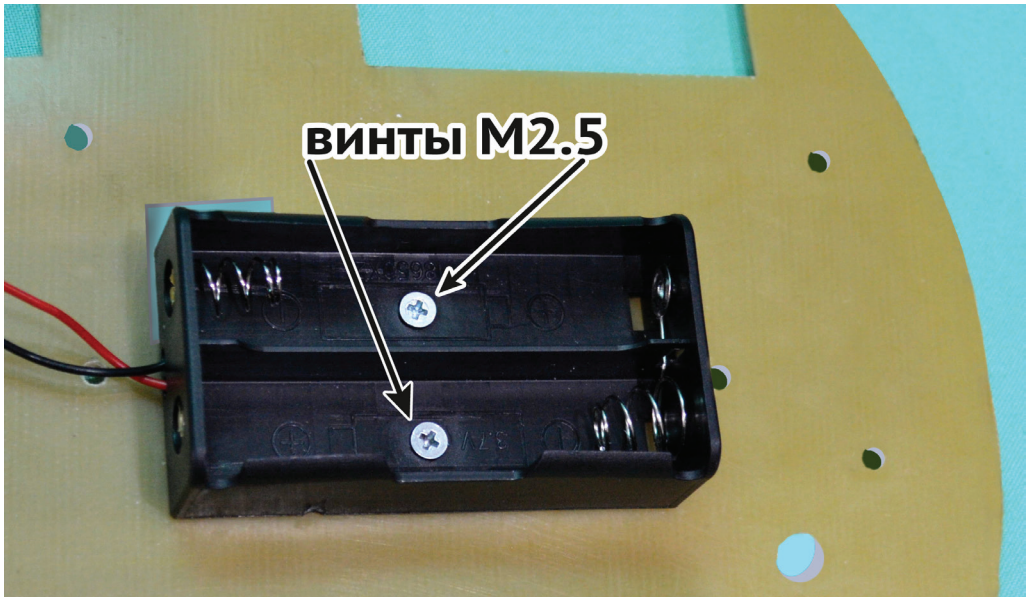


Рис. К6.20. Аккумуляторный бокс прикреплен винтами к корпусу (низ корпуса)

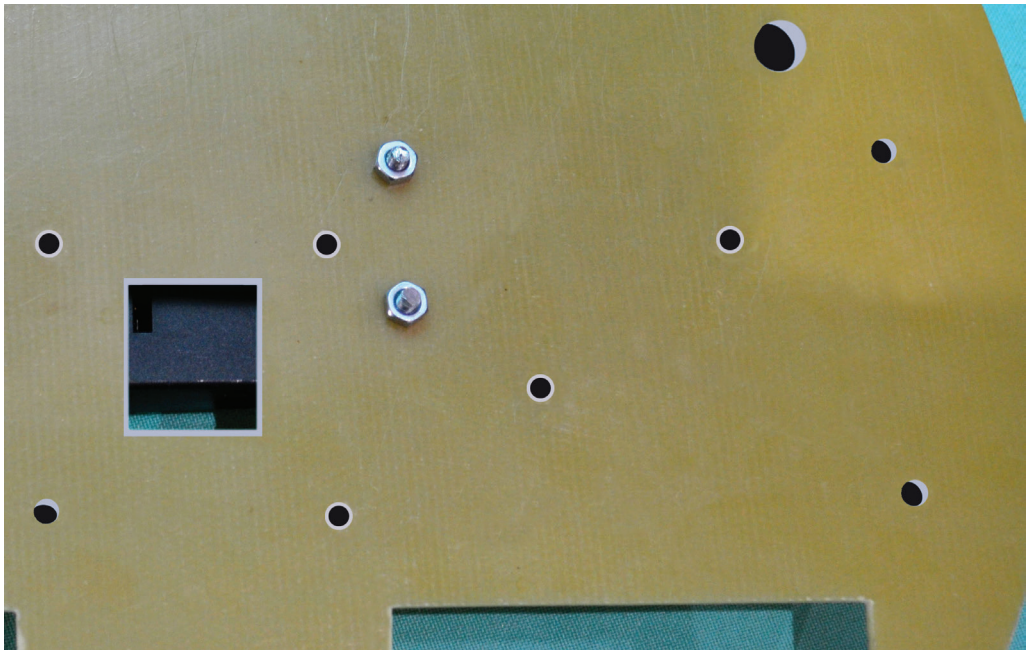


Рис. К6.21. Аккумуляторный бокс прикреплен винтами к корпусу (верх корпуса)

Выключатель также можно установить в корпус снизу, а сверху закрутить гайками со стопорным кольцом.

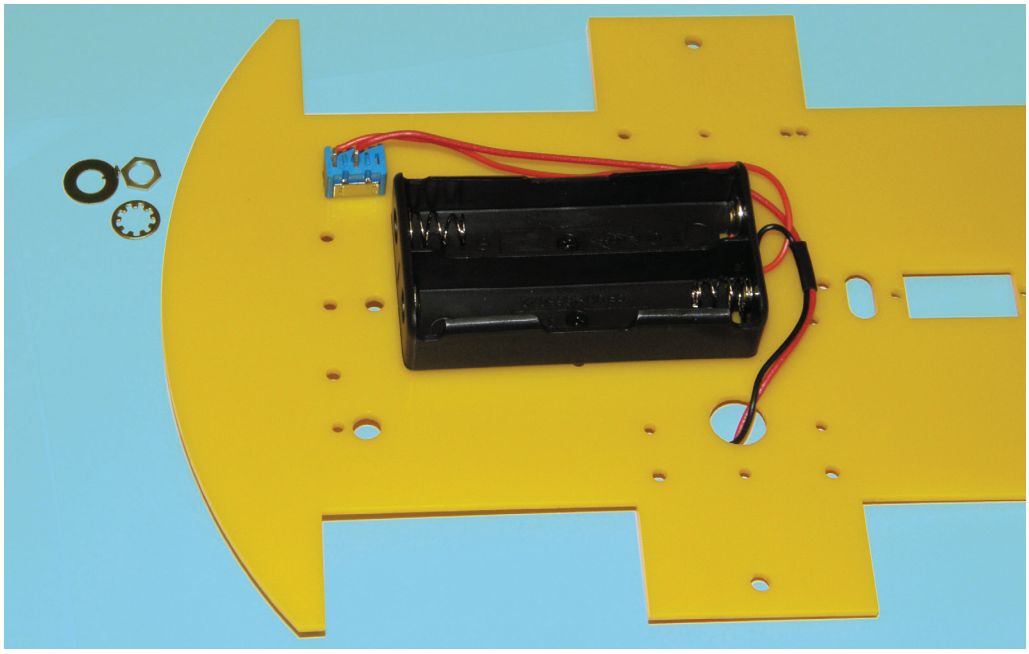


Рис. К6.22. Выключатель установлен в отверстие корпуса (вид снизу)

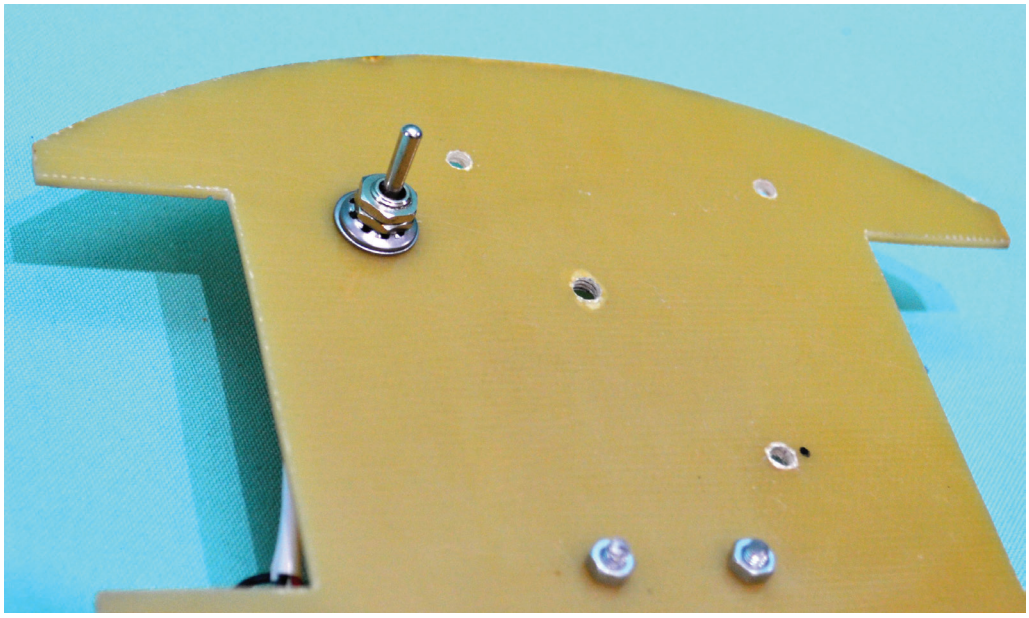


Рис. К6.23. Выключатель установлен в отверстие корпуса (вид сверху)

Следующим шагом будет разводка питающих проводов согласно монтажной схеме, представленной на рис. К6.24. Положительный контакт от батарейного бокса подключен через выключатель, а отрицательный – напрямую.

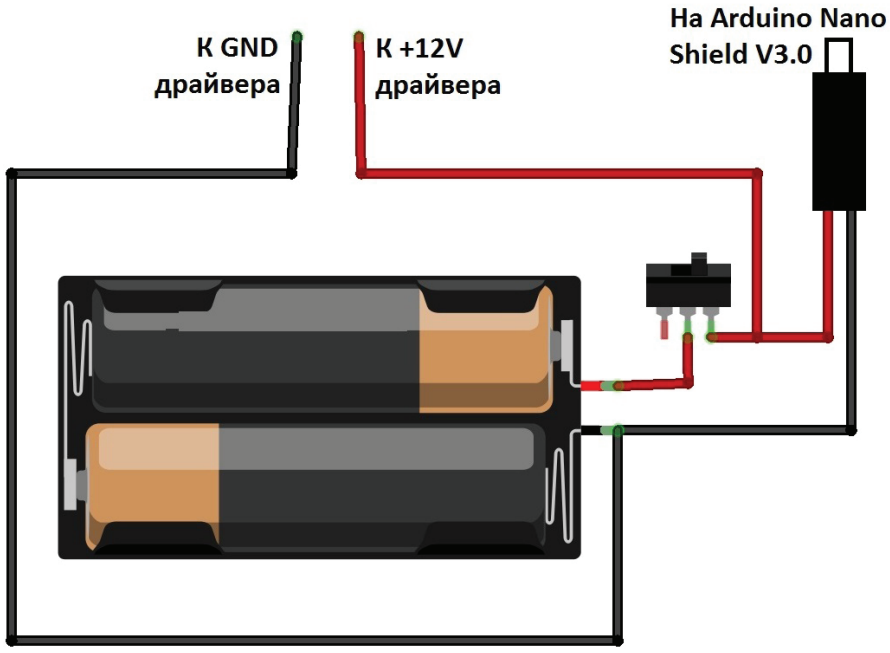


Рис. К6.24. Монтажная схема электропитания робота

Припаяем красный провод от бокса к среднему контакту выключателя, а проводник, который будет выведен наверх корпуса, – к одному из боковых контактов рис. К6.25. Черный провод от отрицательного контакта питания аккумуляторного бокса и красный с выключателя выведем напрямую через технологическое отверстие в корпусе, как показано на рис. К6.26.

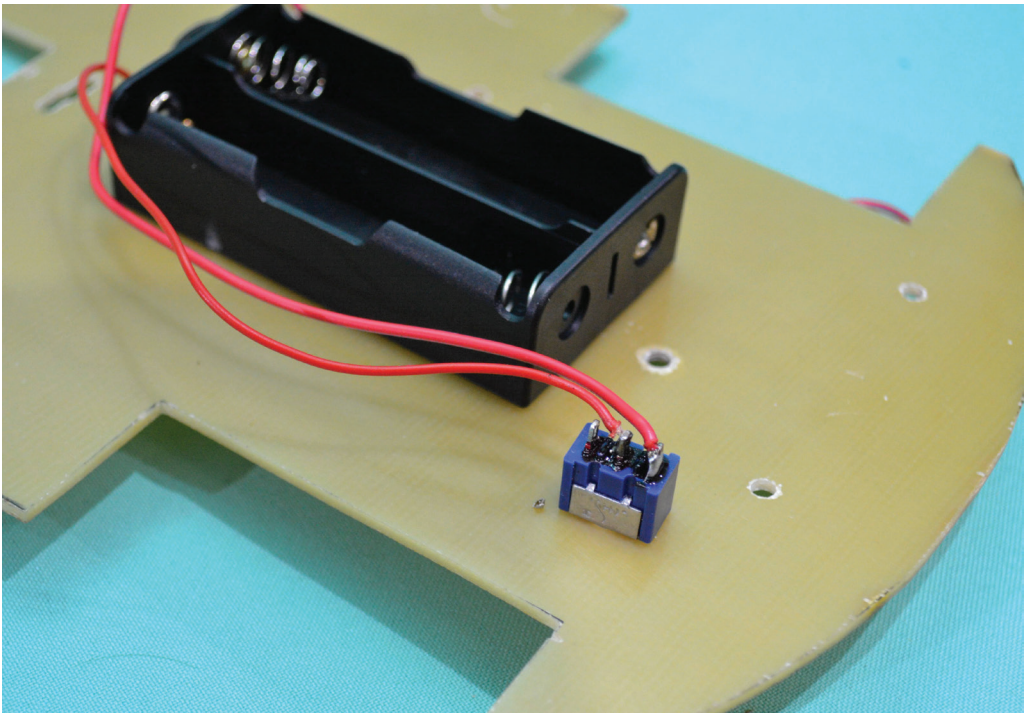


Рис. К6.25. Положительный контакт питания от аккумуляторного бокса с выключателем

Установим стойки, на которые в дальнейшем закрепим драйвер моторов и Arduino Nano Shield V3.0. Стойки снизу прикручены на 3-мм винты длиной 6 мм. Расположение стоек указано на рис. К6.26.

Теперь можно прикрутить рамку с моторами к корпусу. Для этого используются пять 4-мм винтов, как показано на рис. К6.27, К6.28.

Провода от моторов выведем через технологическое отверстие корпуса наверх рис. К6.30.

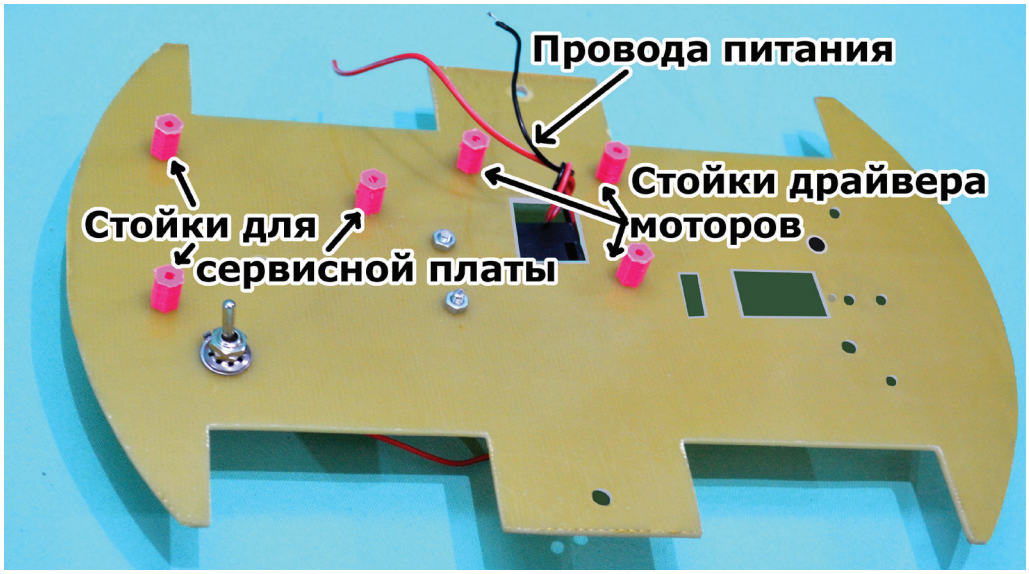


Рис. К6.26. Установлены стойки вверху корпуса для крепления драйвера моторов и сервисной платы Arduino Nano Shield V3.0

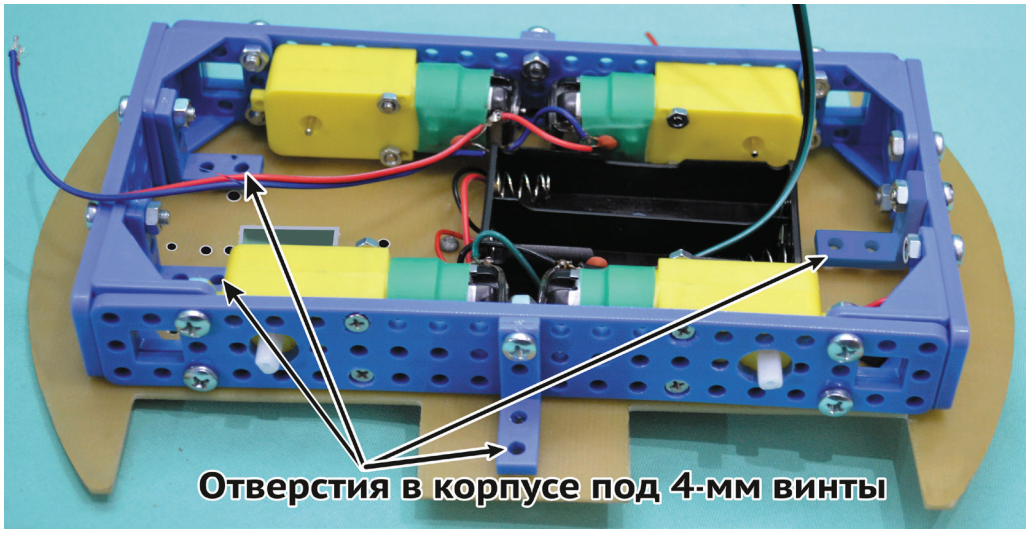


Рис. К6.27. Крепление рамки с моторами к корпусу робота

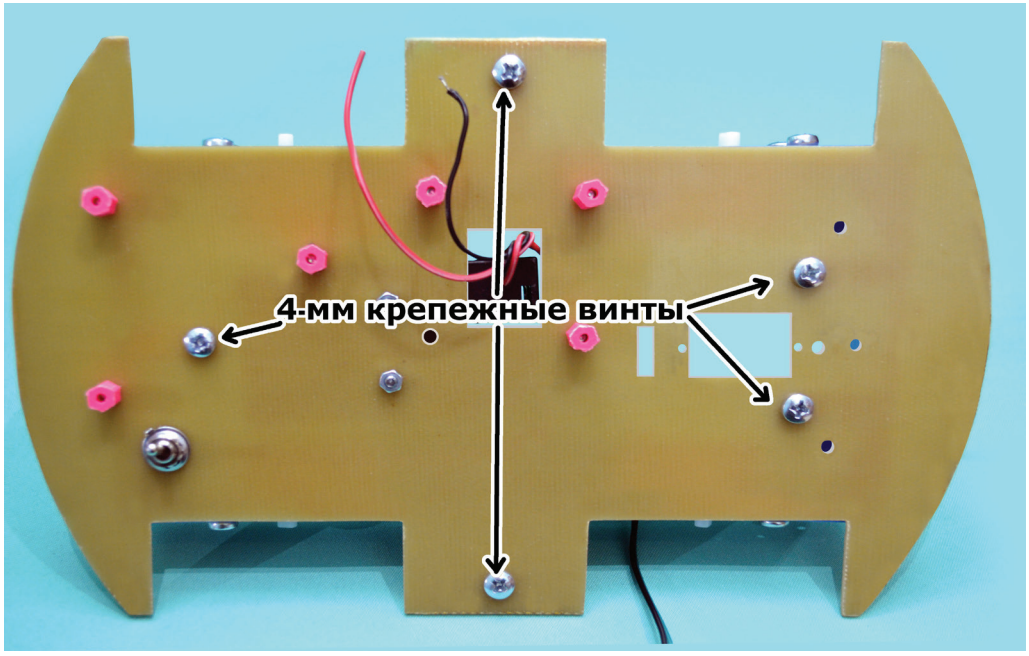


Рис. К6.28. Винты для крепления рамки с моторами вставлены в корпус

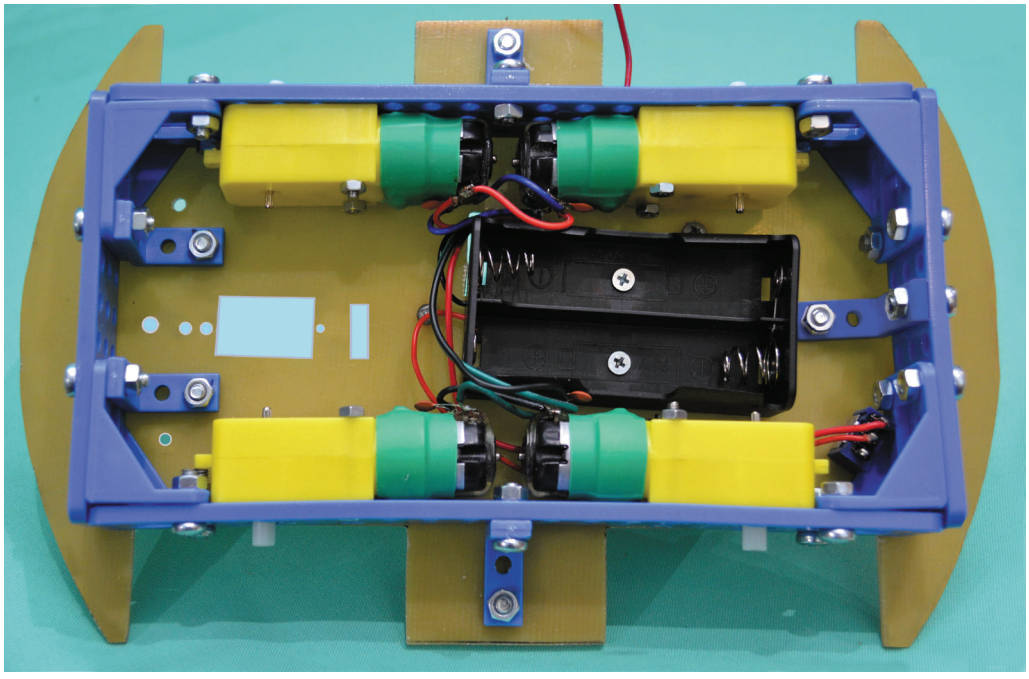


Рис. К6.29. Силовая часть робота в сборе (провода от моторов выведены наверх)

Установим драйвер на стойки, не прикручивая, замерим длину проводов. Если провода длинные, то их нужно подрезать (рис. К6.30).

Провода от моторов (предварительно оголенные и залуженные) закрепим в винтовых разъемах драйвера согласно схеме К6.31.

Отрицательный оголенный контакт от бокса скрутим с отрицательным контактом от разъема для Arduino Nano Shield V3.0, то же сделаем с положительным. После этого их желательно спаять попарно (рис. К6.32). Далее согласно монтажной схеме К6.31 закрепим контакты питания в винтовых разъемах драйвера +12V и GND.

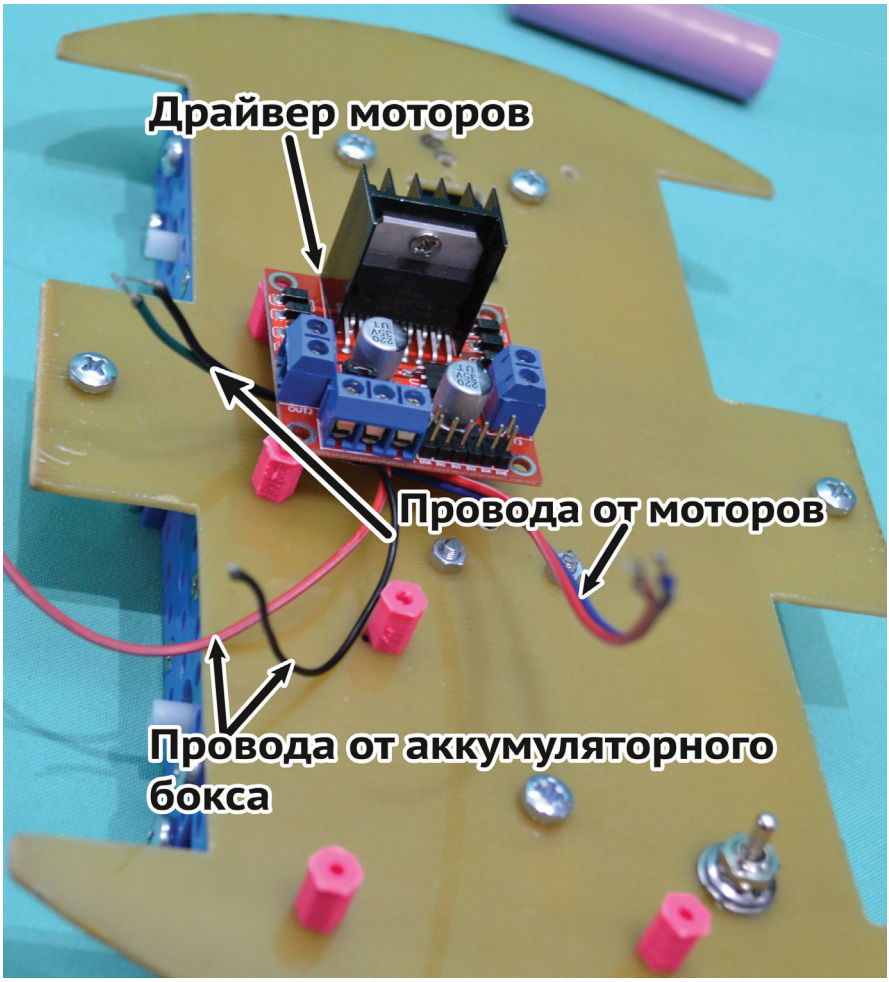


Рис. К6.30. Подсоединим провода от моторов к драйверу

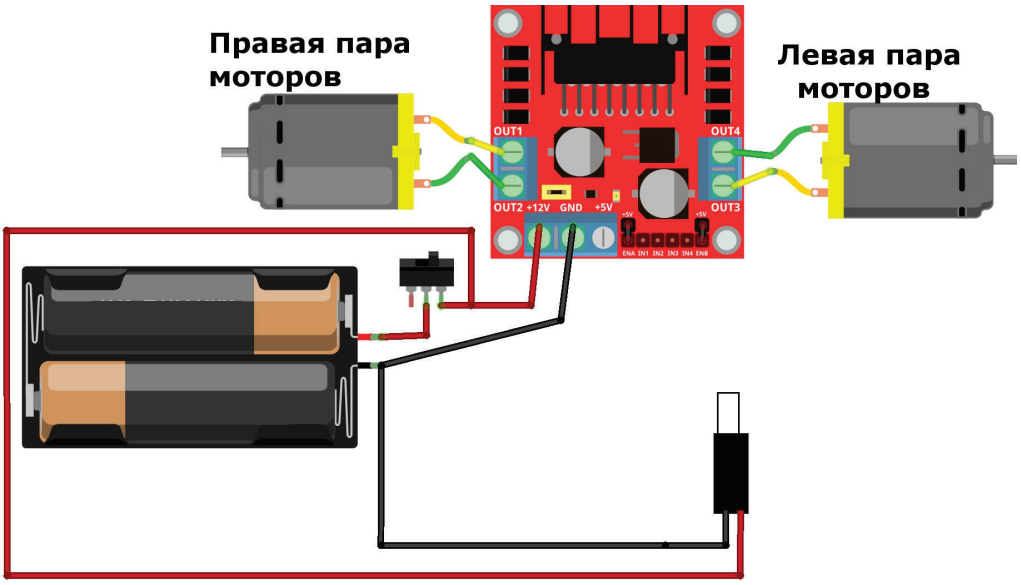


Рис. К6.31. Монтажная схема электропитания робота с драйвером

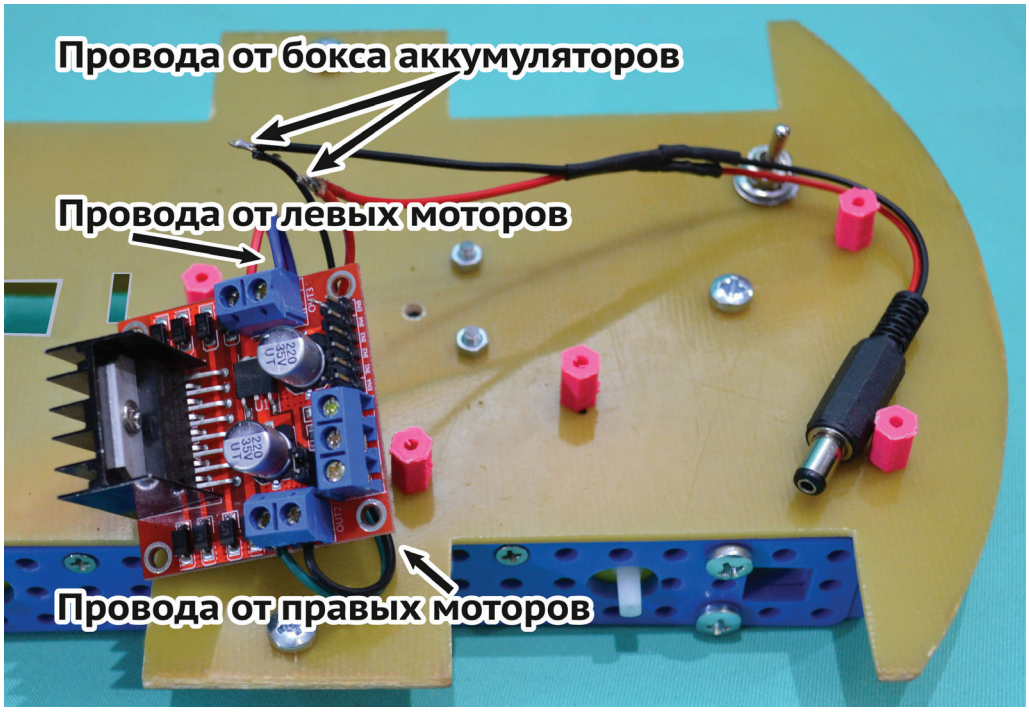


Рис. К6.32. Соединяем провода от разъема для Arduino Nano Shield V3.0 и аккумуляторного бокса (положительный контакт с положительным, отрицательный с отрицательным)

Далее драйвер следует закрепить короткими винтами диаметром 3 мм, как показано на рис. К6.33. Аналогично закрепим плату Arduino Nano Shield V3.0 и подключим к ней разъем питания (рис. К6.34). Если плата Arduino NANO еще не установлена на Arduino Nano Shield V3.0, следует ее установить, как показано на рис. К6.35.

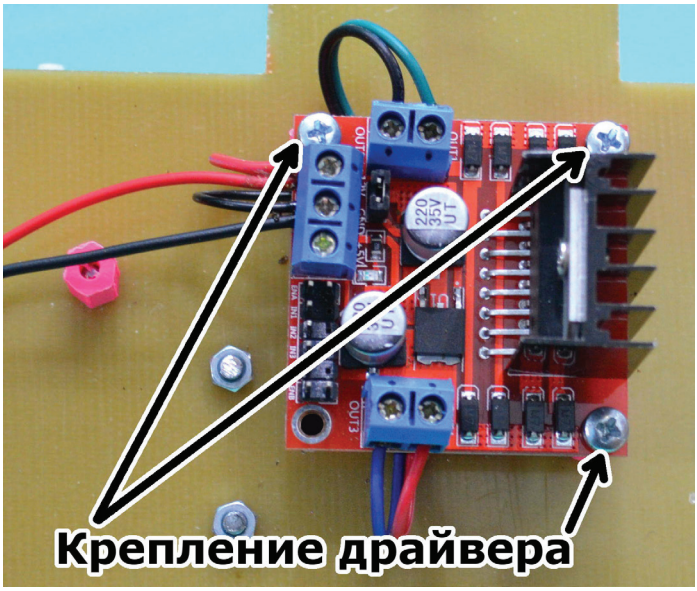


Рис. К6.33. Закрепляем драйвер моторов на корпус 2.5 мм винтами (длиной 6 мм)

Рис. К6.34. Сервисная плата Arduino Nano Shield V3.0 прикреплена 3-мм винтами на стойки

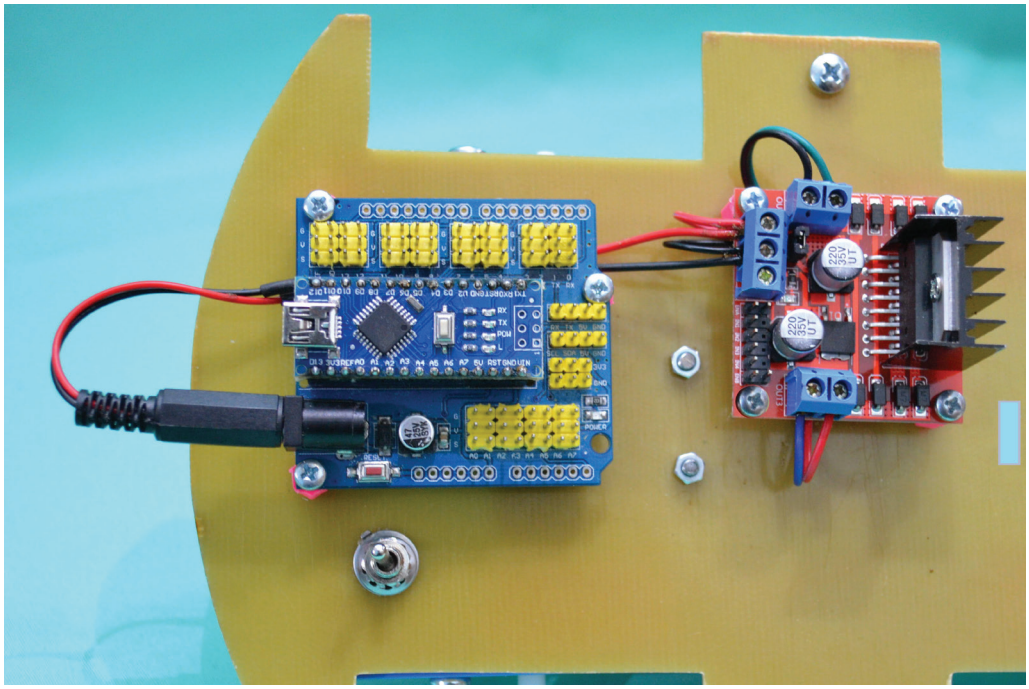


Рис. К6.35. На сервисной плате Arduino Nano Shield V3.0 установлена Arduino NANO и подключен разъем питания

Используя провода с разъемами типа Dupont «мама–мама» и руководствуясь схемой рис. К6.36, соединим сигнальные контакты Arduino NANO и контакты управления работой драйвера моторов. Нужные нам контакты представлены на сервисной плате Arduino Nano Shield V3.0, они выделены ярким на рис. К6.36. Контакты сервисной платы пронумерованы и имеют нумерацию, аналогичную номерам на плате Arduino NANO. Смотрим на схему (рис. К6.36) и соединяем последовательно пары контактов Arduino и драйвера. Для лучшей наглядности нужно использовать провода разного цвета.

В итоге должна получиться схема, представленная на рис. К6.38.

После проверки правильности соединений можно установить аккумуляторы. Особое внимание следует уделить проверке полярности питания!

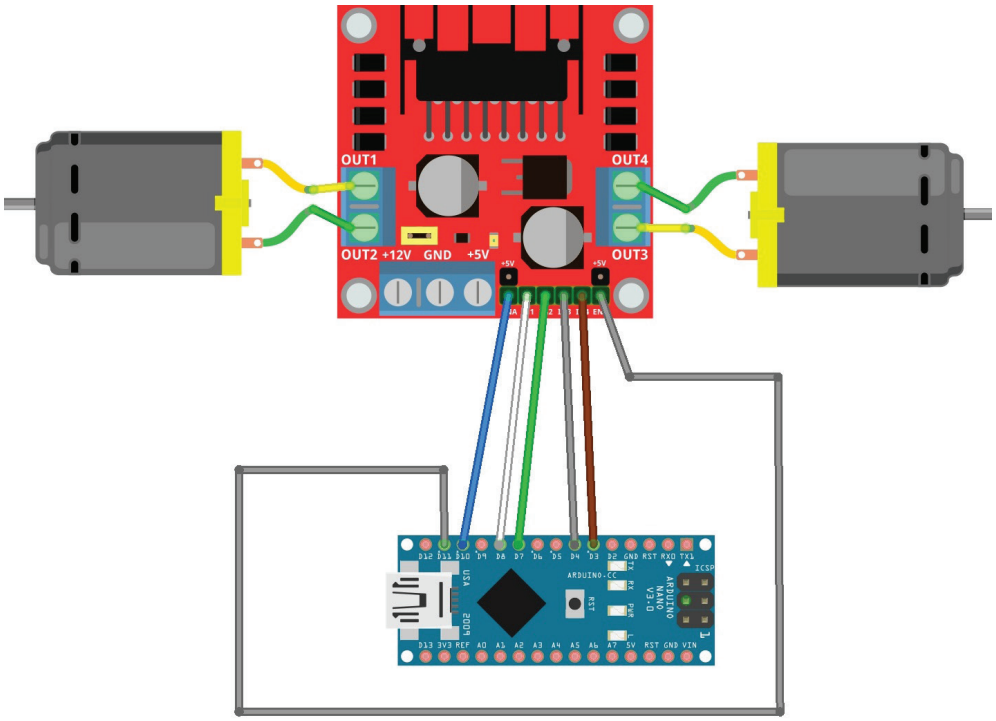


Рис. К6.36. Схема соединения информационных контактов Arduino NANO и драйвера

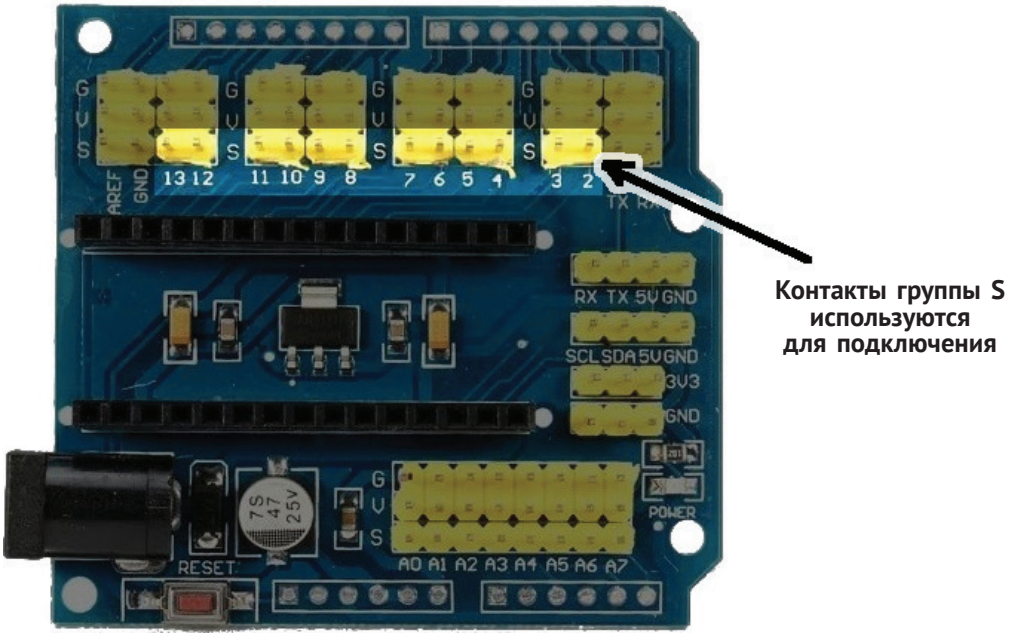


Рис. К6.37. Для управления драйвером используется полоска контактов, промаркированная буквой S (signal)

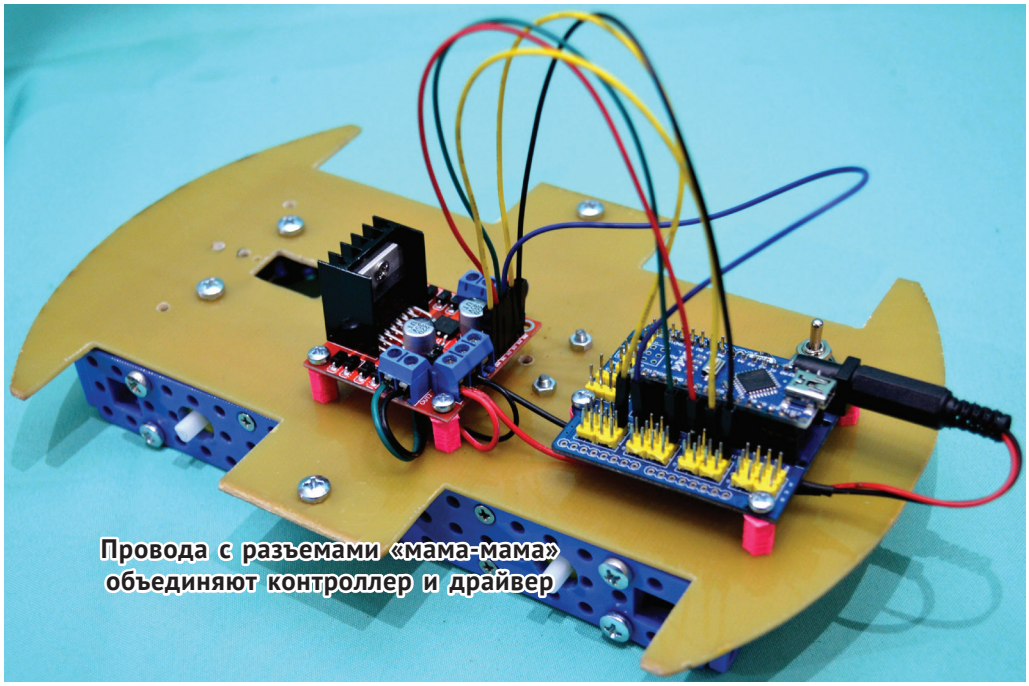


Рис. К6.38. Управление подключено к драйверу

Рис. К6.39. Аккумуляторы 18650 установлены в бокс робота

Теперь робот собран, устанавливаем на него колеса и приступаем к выполнению заданий из 7-й главы книги.

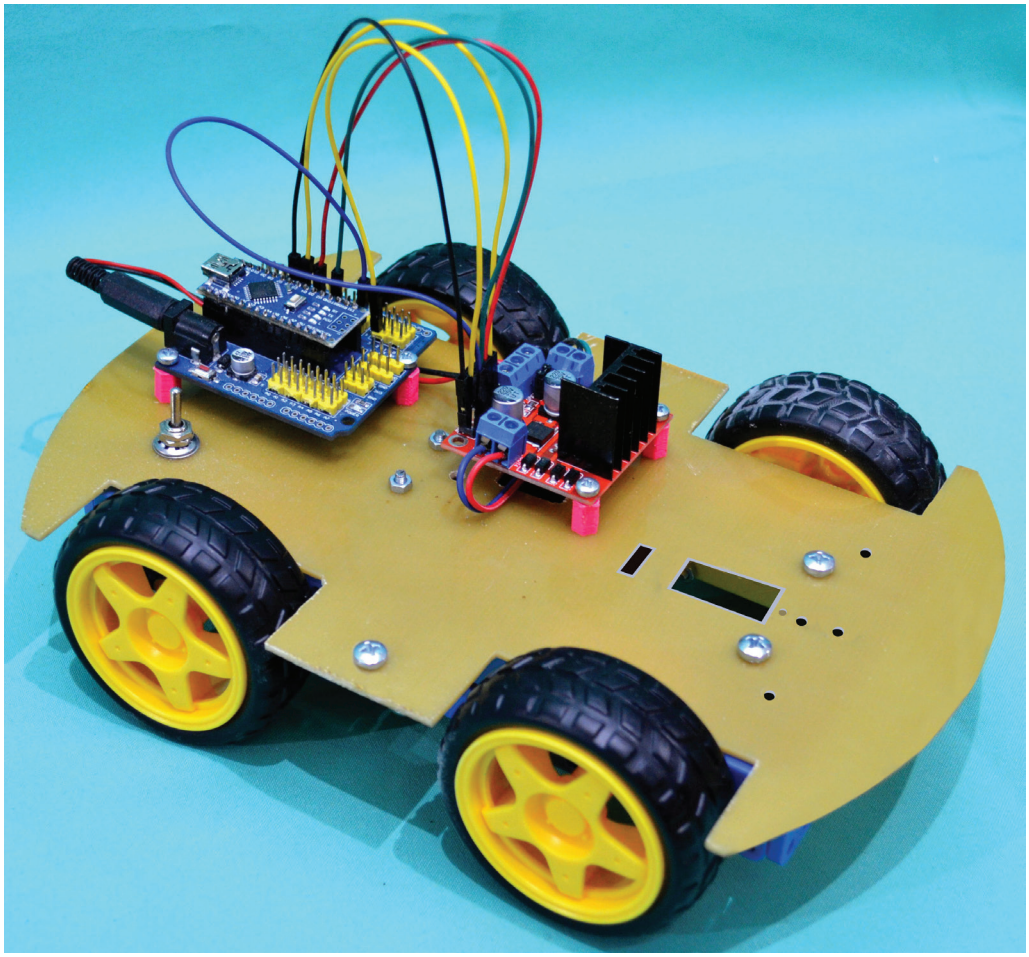


Рис. К6.40. Готовая базовая модель робота

К ГЛАВЕ 7 СХЕМА УПРАВЛЕНИЯ ДВИЖЕНИЕМ

Для программирования робота потребуется кабель с разъемом MINI-USB, именно через такой разъем подключается контроллер Arduino NANO к компьютеру.

Рис. К7.1. Кабель MINI USB

Подключите робота к компьютеру. Если на плате Arduino NANO зажегся красный светодиод, значит, кабель подключен верно.

Запустите программу Arduino IDE и перейдите в пункт **Инструменты** ⇒ **Порт**. Запомните, какие порты имеются в этом пункте. Теперь отключите кабель от компьютера или Arduino NANO. Снова перейдите в пункт меню Arduino IDE **Инструменты** ⇒ **Порт**. Определите, какой порт исчез. Теперь снова подключите робота к компьютеру и выберите появившийся порт в качестве порта Arduino.

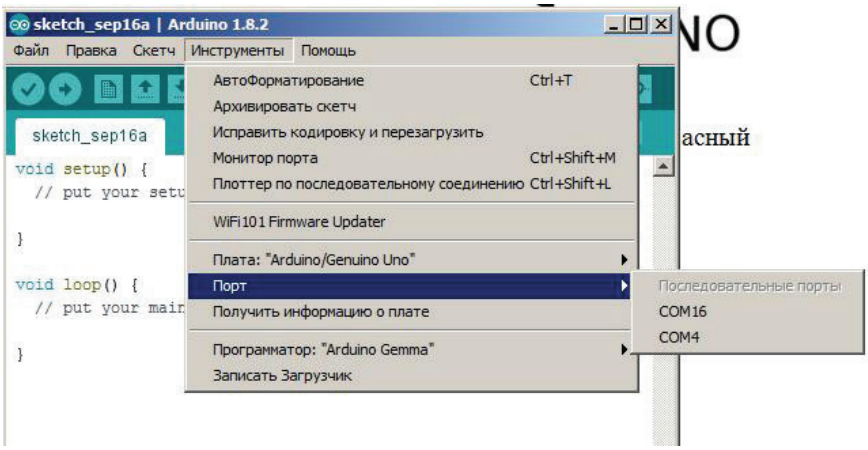


Рис. К7.2. Выбор порта программирования

В пункте **Инструменты** ⇒ **Плата** выберите плату Arduino Nano.

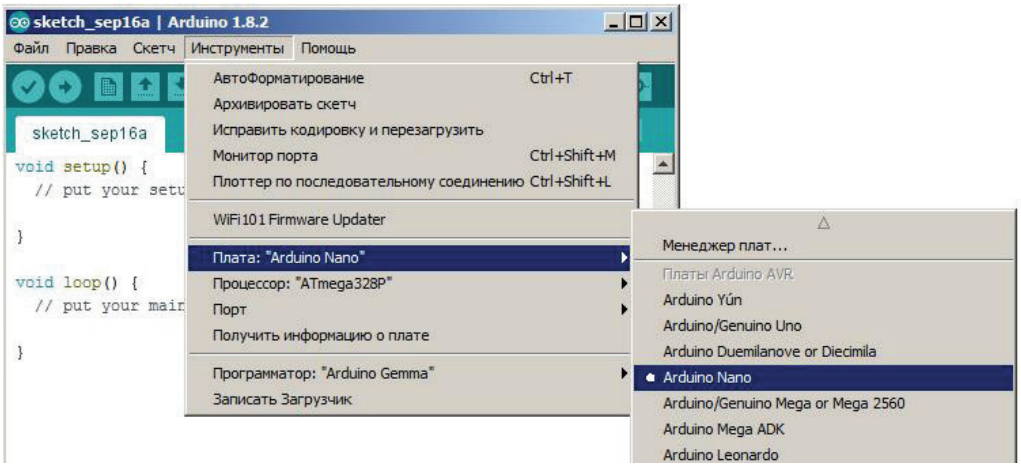


Рис. К7.3. Выбор платы контроллера

Теперь вы полностью готовы к выполнению заданий 7-й главы книги.

Но если у вас не появляется новый порт при подключении к компьютеру Arduino платы, возможно, требуется установить дополнительный драйвер порта, как это сделать – описано в главе 4.

Для лучшей визуализации движений и быстрого устранения неисправностей можно установить робота на подставку – таким образом, чтобы его колеса не касались пола.

Это позволит визуально наблюдать, как робот реагирует на команды, и не отключать его от компьютера. Можно также снять с робота колеса, что позволит работать с ним без отключения кабеля программирования.

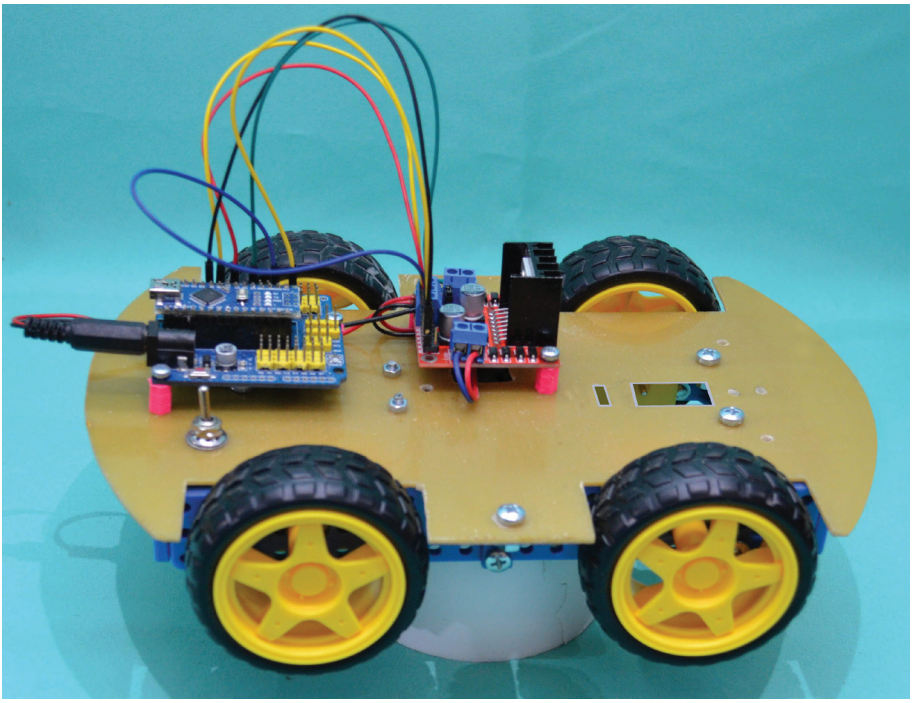


Рис. К7.4. Робот на подставке

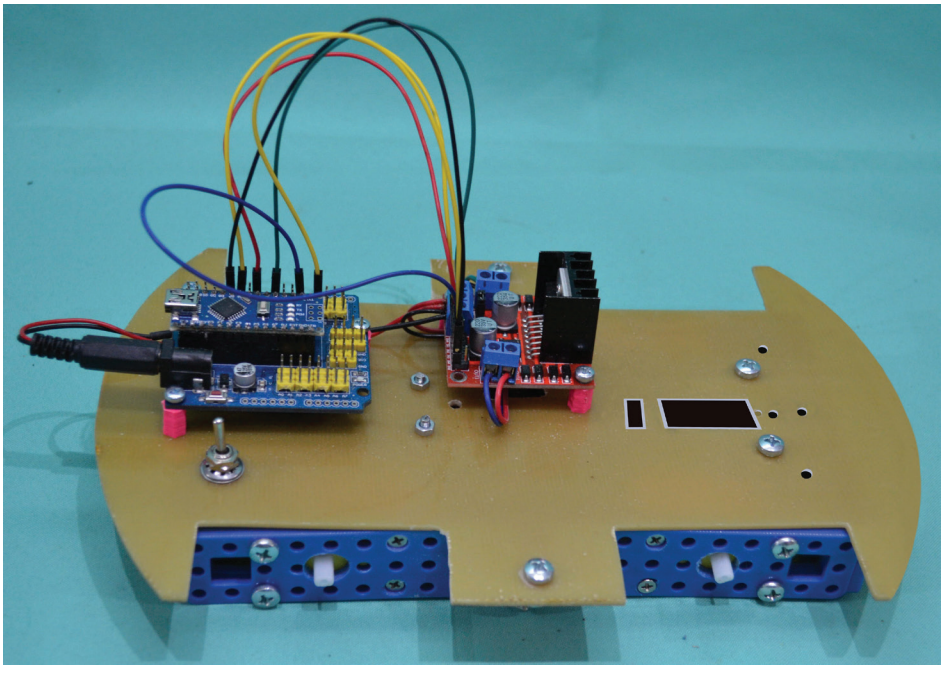


Рис. К7.5. Робот готов к тестированию
без отключения от кабеля программирования

К ГЛАВЕ 8 ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ

Использование инфракрасного канала

В этом случае в качестве пульта управления может выступать любой пульт, например от телевизора. Рассмотрим особенности доработки нашего робота.

Для управления роботом по инфракрасному каналу в конструкторе есть датчик TSOP31238 (или аналогичный) (рис. К8.1), который разработчики снабдили удобным разъемом, оптимизированным для установки на плату Arduino Nano Shield V3.0 (рис. К8.2).

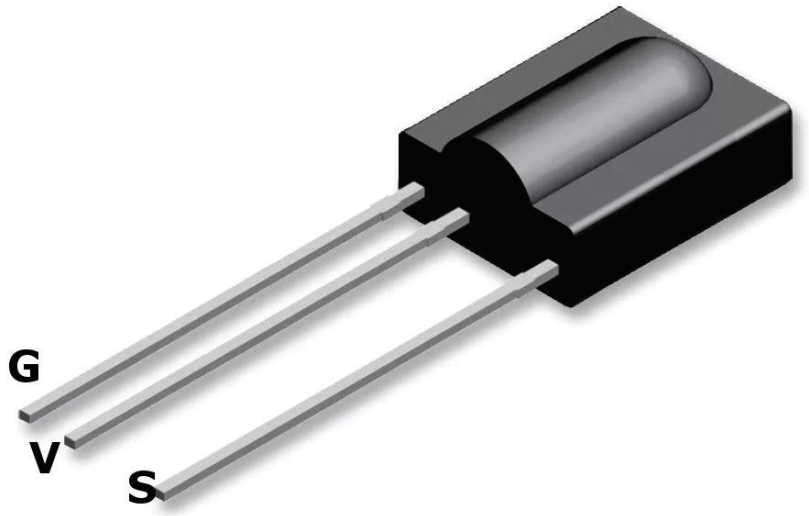


Рис. К8.1. Датчик TSOP31238 (G – земля, V – 5 вольт, S – сигнальный контакт)

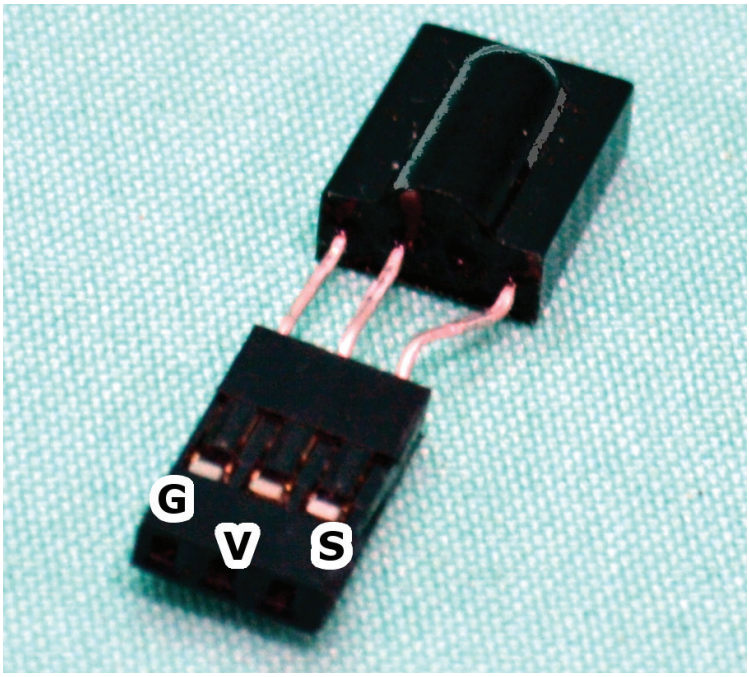


Рис. К8.2. Датчик TSOP31238 с разъемом для установки на плату Arduino Nano Shield V3.0

При помощи TSOP31238 можно только принимать сигналы. Стоит отметить, что сигналы от разных пультов отличаются, и следует вначале отследить и записать, какие «кодировки» посылает ваш пульт, как это сделать – описано в разделе «Получение кодов кнопок для используемого пульта» 8-й главы.

Используем 13-й разъем для подключения, согласно информации из рис. К8.2 устанавливаем датчик на плату Arduino Nano Shield V3.0 в 13-ю строку. G в G, V в V, S в S.

Если вставить неправильно, датчик может быть испорчен!!!

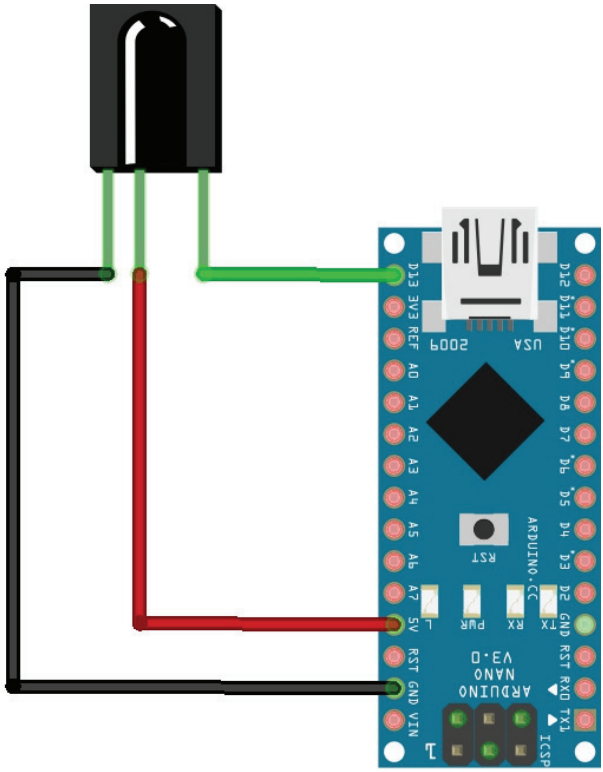


Рис. К8.3. Схема соединения датчика TSOP31238 с Arduino Nano

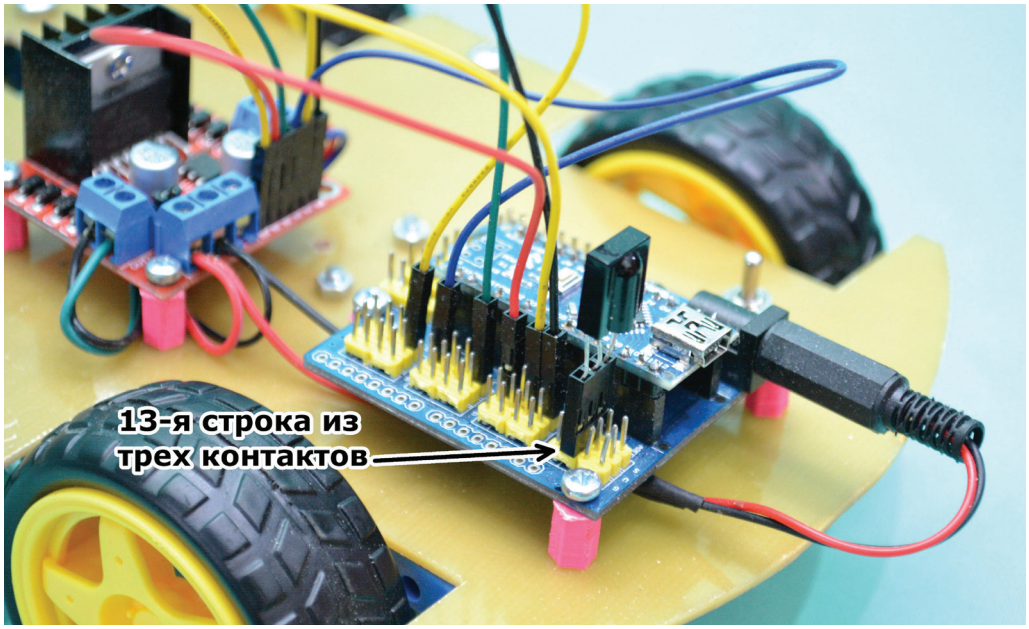


Рис. К8.4. Датчик TSOP31238 установлен на плату Arduino Nano Shield V3.0 в 13-й пин Arduino

После того как датчик установлен, можно приступать к программированию, пользуясь разделом «Управление роботом по каналу инфракрасной связи» 8-й главы книги.

В связи с особенностью инфракрасного управления для лучшей чувствительности робота к командам следует направлять луч пульта на приемник, робот будет лучше управляем, когда пульт находится сзади от робота (приемник направлен назад).

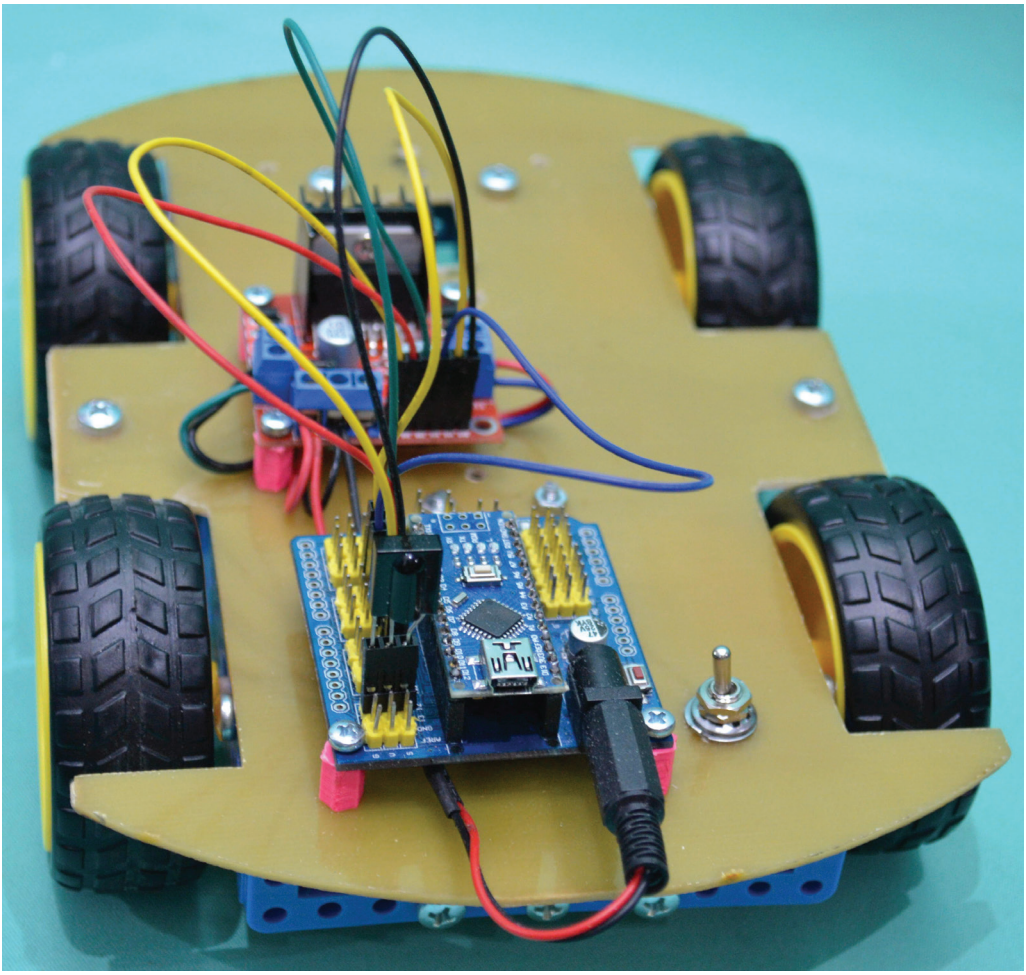


Рис. К8.5. Вид робота сзади с установленным датчиком TSOP1738

Использование канала Bluetooth

Для наглядности на рис. К8.6 и К8.7 приведены схемы вариантов подключения модуля Bluetooth. Схема на рис. К8.7 предполагает пайку, но позволяет изменить скорость обмена или имя, под которым робот будет виден по каналу Bluetooth окружающим устройствам, см. книгу.

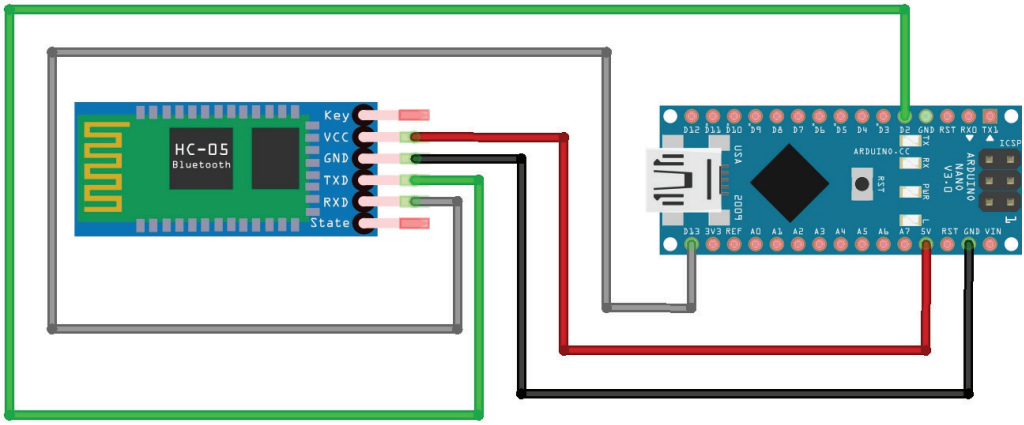


Рис. К8.6. Подключение HC-05 к Arduino Nano (без возможности менять настройки HC-05)

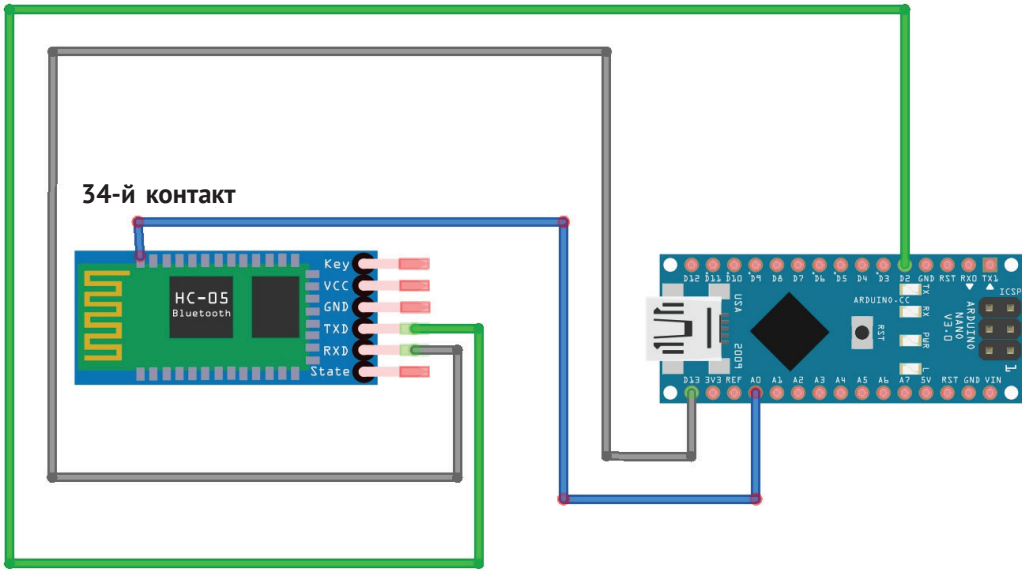


Рис. К8.7. Подключение HC-05 к Arduino Nano (меняем настройки HC-05, управляя 34-м контактом через пин A0 Arduino), питание не показано

Подключить модуль Bluetooth можно по-разному, например используя четыре соединительных провода с разъемами «мама–мама» (рис. К8.8), но более удобно сделать это при помощи макетной платы (рис. К8.9). Данная макетная плата служит для соединения нескольких контактов через объединенные проводником строки по четыре разъема.

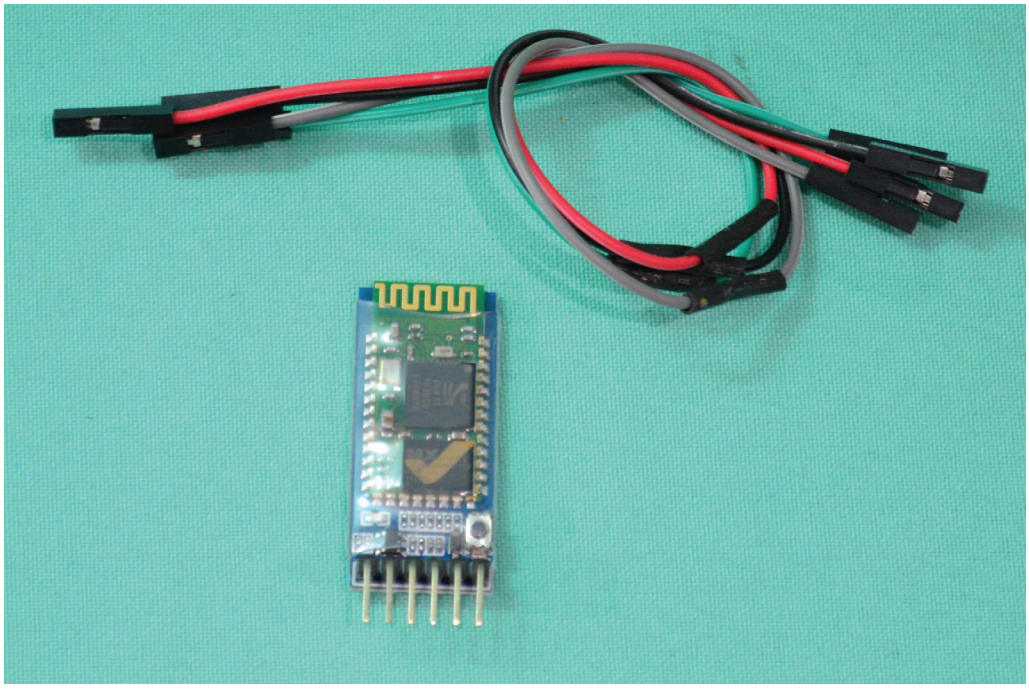


Рис. К8.8. Bluetooth-контроллер HC-05 и провода с разъемами «мама–мама»

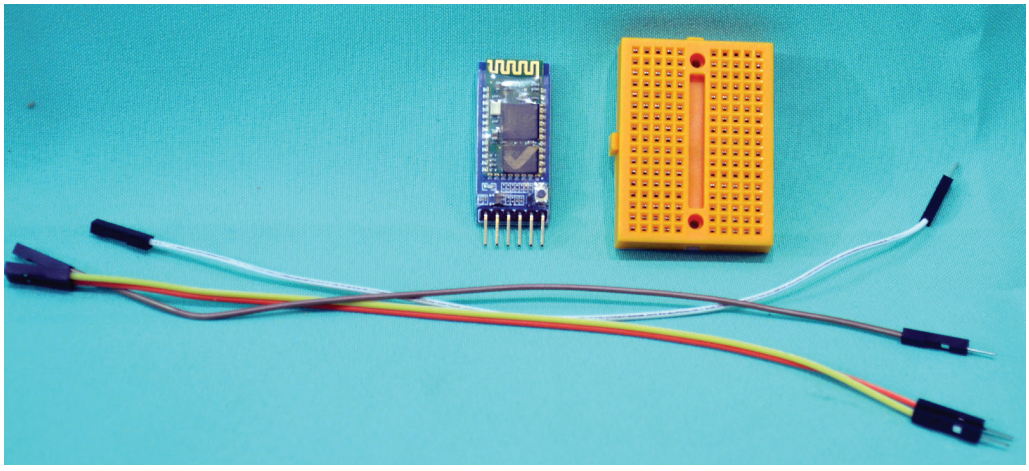


Рис. К8.9. Bluetooth-контроллер HC-05, макетная плата и провода с разъемами «мама–мама»

Удобная установка контроллера Bluetooth показана на рис. К8.10 и К8.11; через макетную плату провода пойдут на Arduino Nano Shield V3.0.

Рис. К8.10. Bluetooth-контроллер HC-05 установлен в макетную плату, видны названия разъемов контроллера

Рис. К8.11. Bluetooth-контроллер HC-05 установлен в макетную плату, требуемые контакты (RX, TX, GND +5V) соединены проводами

Стоит отметить, что снизу макетная плата имеет клейкую основу. Следует снять защитную пленку и приклеить ее к роботу, как показано на следующем рисунке.

Для удобства модификации робота можно, не снимая защитной пленки, прикрутить макетную плату входящими в комплект винтами M2.5.

Контакты соединяем согласно схеме (рис. К8.6).

Теперь можно приступать к программированию, возвращаемся к книге, используем раздел «Управление роботом по каналу Bluetooth», программируем и управляем роботом.

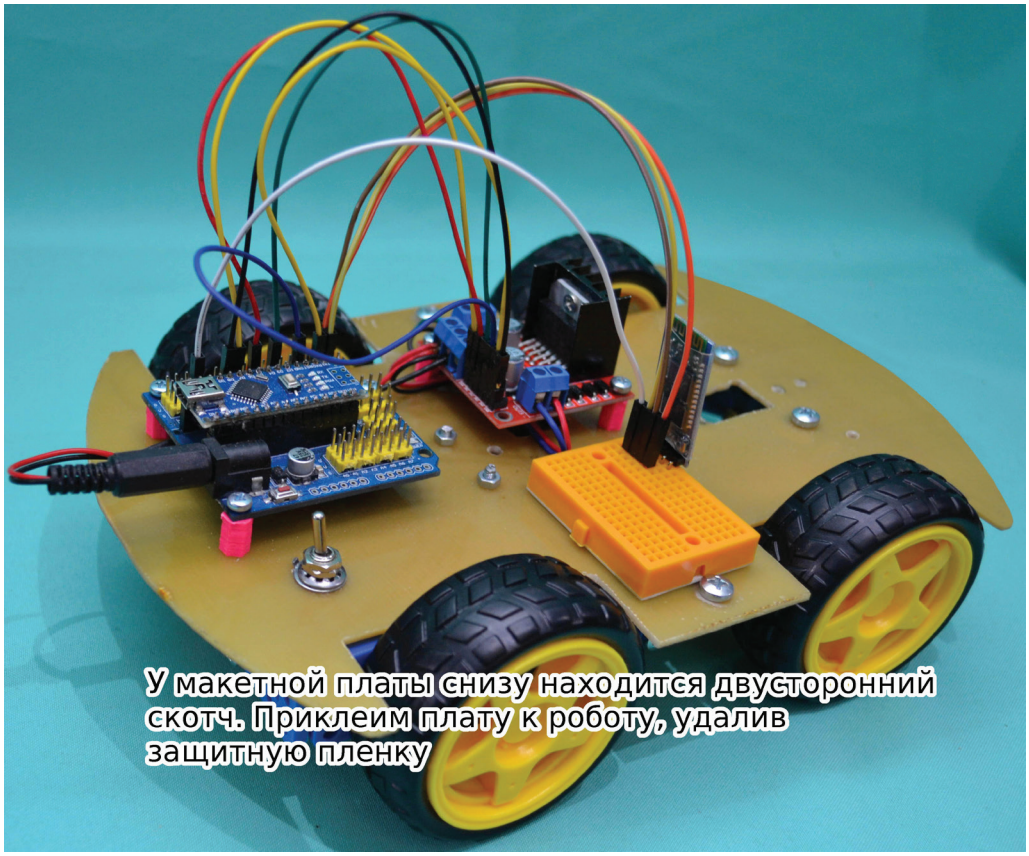


Рис. К8.12. Робот – вид справа

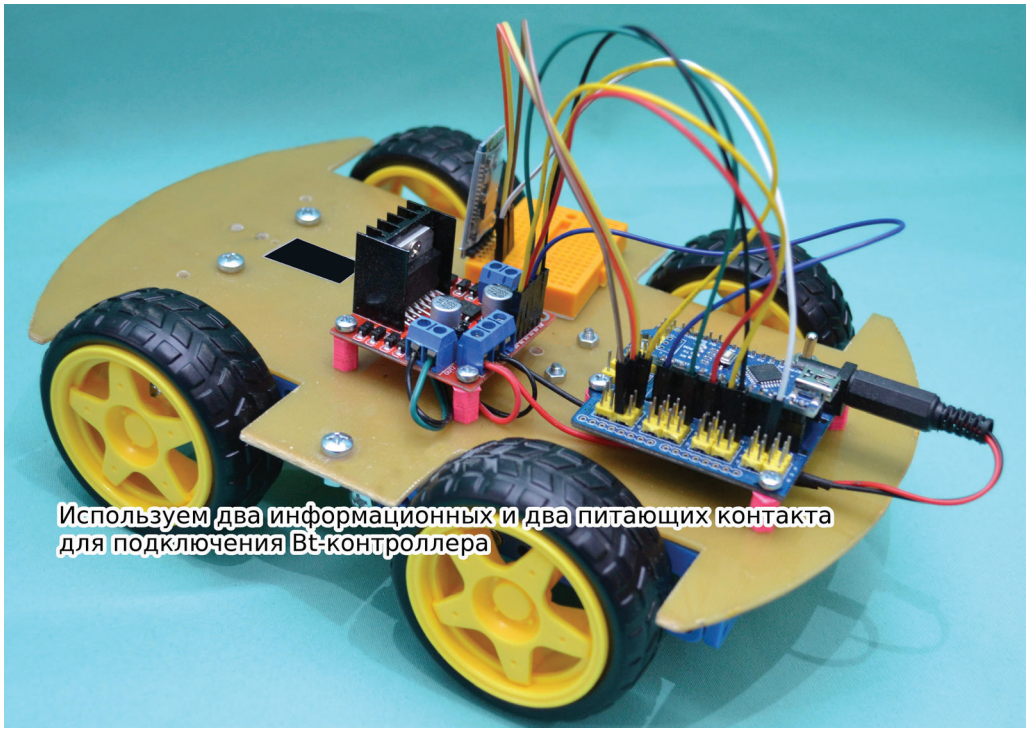


Рис. К8.13. Робот – вид слева, видны контактные площадки Arduino Nano Shield V3.0, к которым подключен Bt-контроллер

К ГЛАВЕ 9

Инструкция по установке головы

Для добавления вращающейся головы с ультразвуковым датчиком расстояния потребуются: 1) крепление датчика расстояния – основа головы; 2) сервомотор; 3) крепежный рычажок сервомотора к основе головы; 4) мелкие шурупы; 5) винт крепления рычажка к валу сервомотора; 6) ультразвуковой сонар HC-SR04; 7) четыре провода «мама–мама».

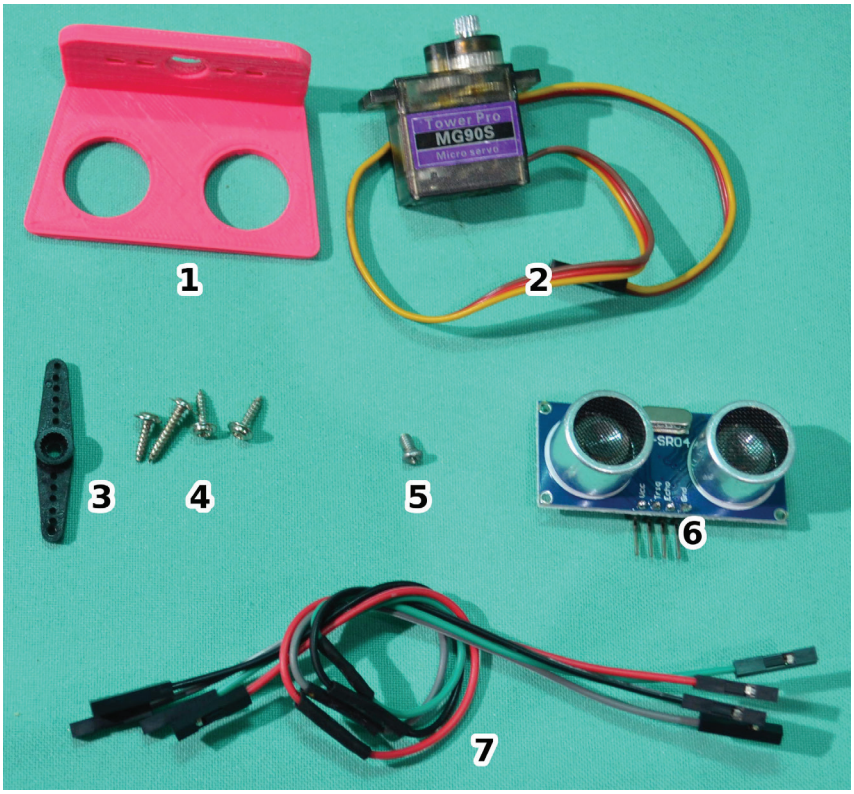


Рис. К9.1. Элементы для конструирования головы робота

Сервомотор установим в специально предназначенное отверстие в передней части корпуса робота согласно рис. К9.2–К9.5.

Рис. К9.2. Установка сервомотора

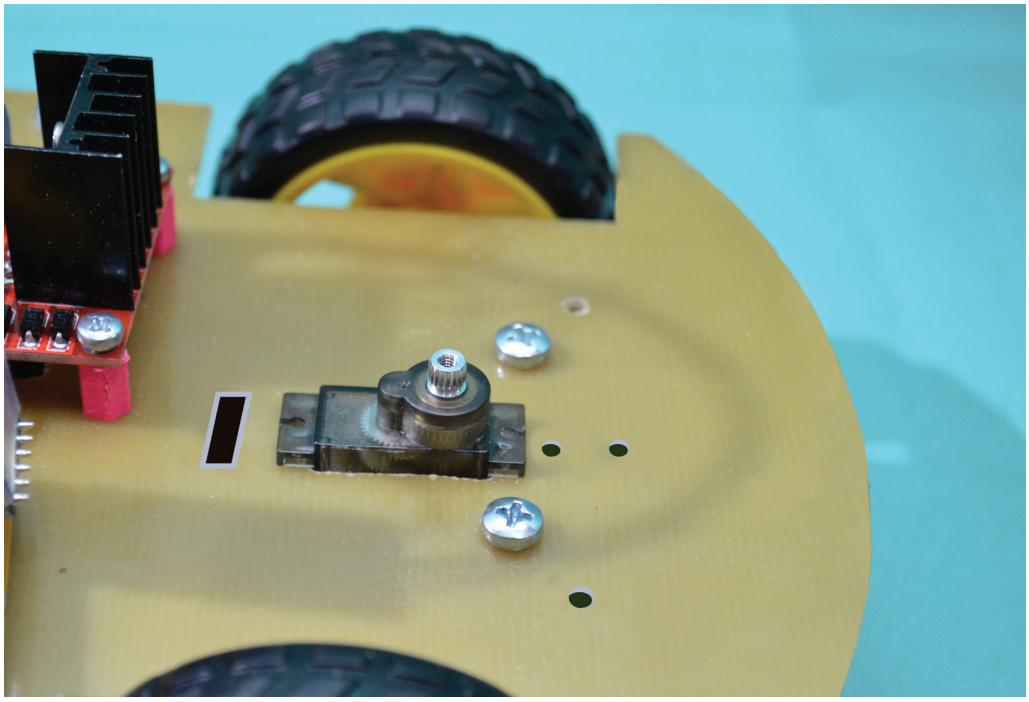


Рис. К9.3. Установка сервомотора (сервомотор не закреплен)

Рис. К9.4. Крепим сервомотор

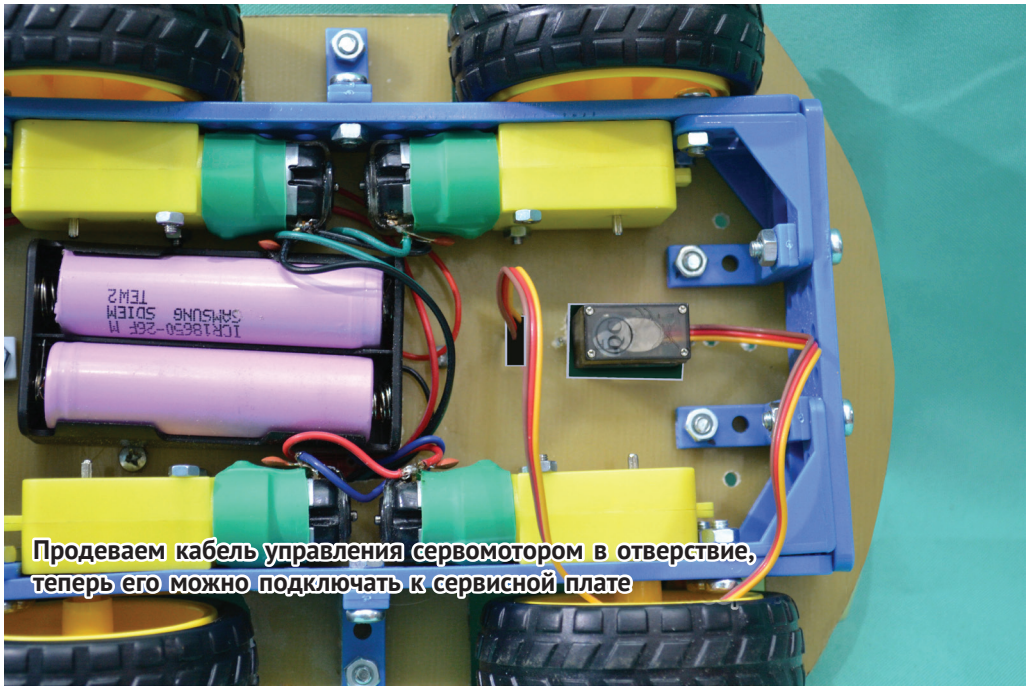


Рис. К9.5. Установка сервомотора (вид снизу)

Используя номера контактов по схеме К9.6, коммутируем шлейф сервомотора на сервисную плату в строку из трех контактов под номером 9. Желтый провод в пин S (сигнальный), красный в пин V (+5 вольт), коричневый в пин G (земля или минус).

Неправильное подключение может испортить сервомотор или плату Arduino!

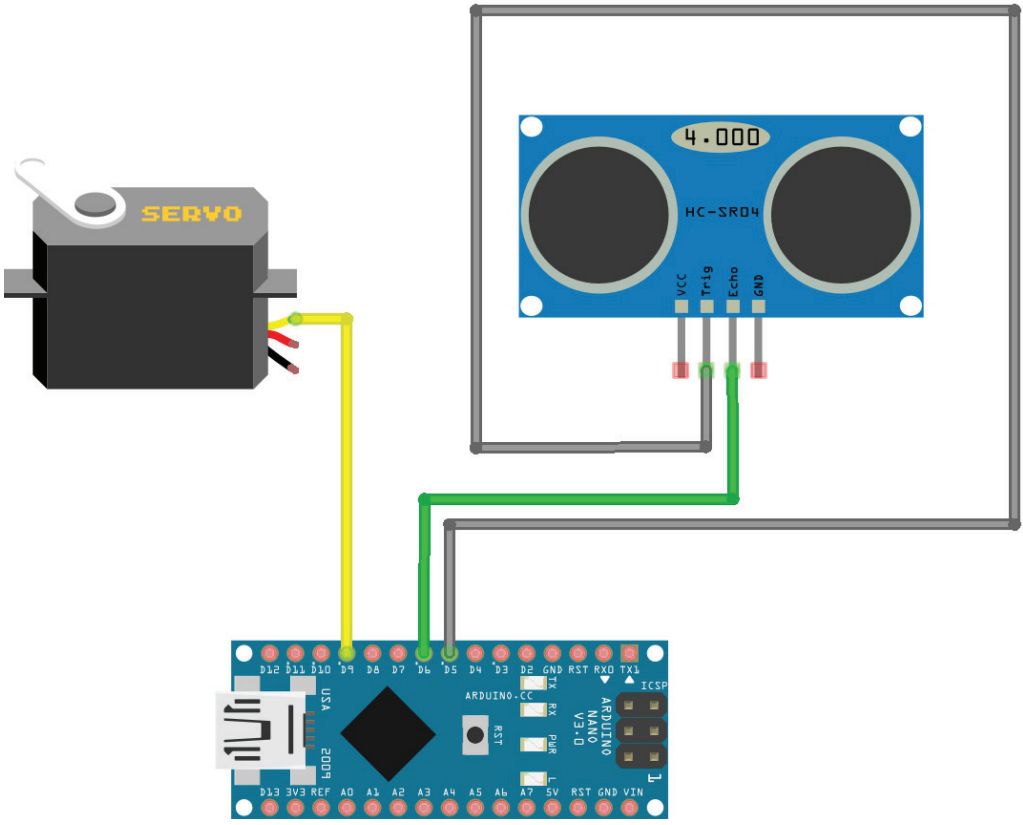


Рис. К9.6. Схема логических соединений сервомотора и сонара (HC-SR04)

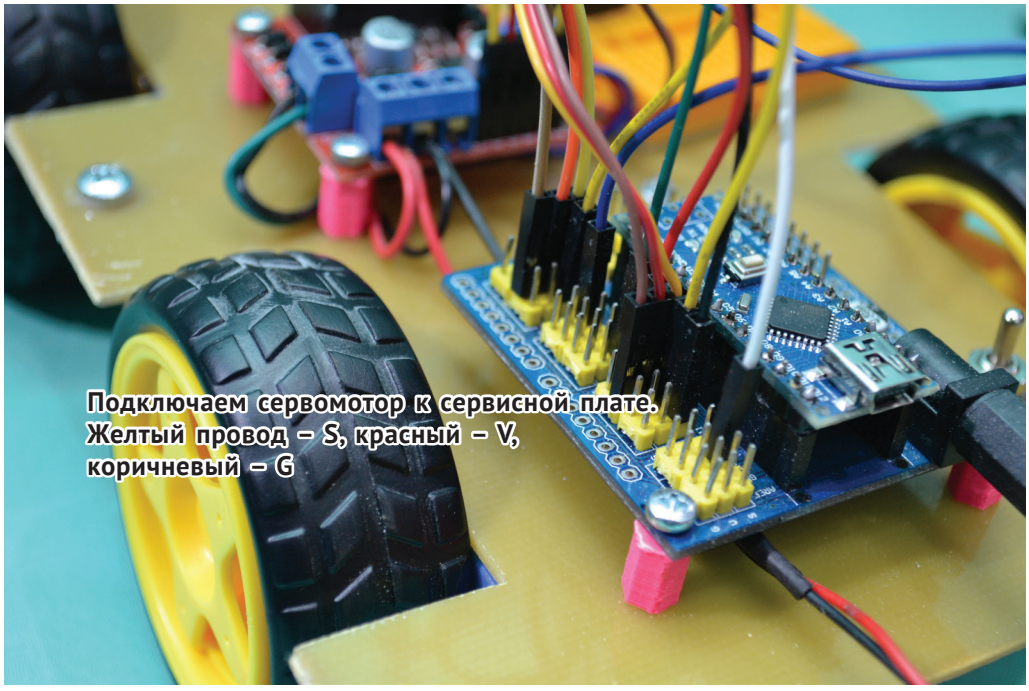


Рис. К9.7. Контакты сервомотора соединены с контроллером Arduino

Далее шурупами прикрепляем рычажок сервомотора к основе головы, как показано на рис. К9.8 и К9.9.

Рис. К9.8. Рычажок с основой головы (вид сверху)

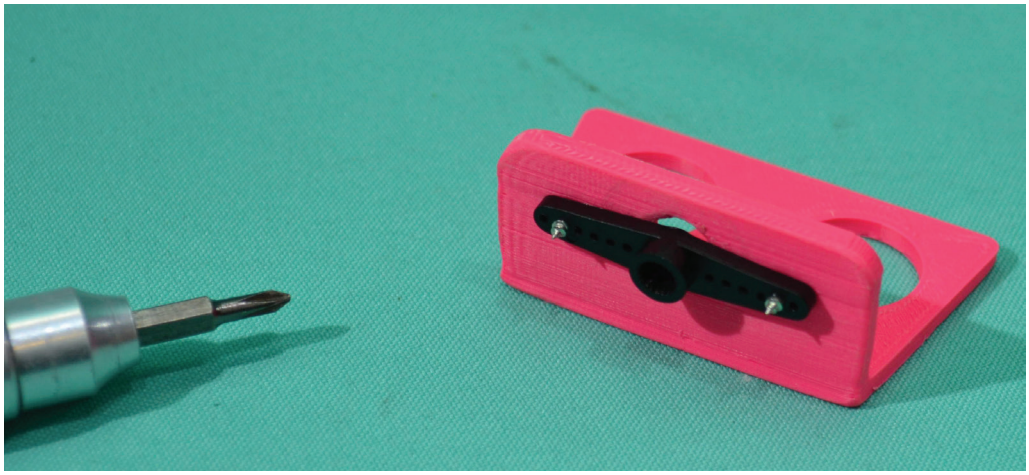


Рис. К9.9. Рычажок с основой головы (вид снизу)

Устанавливаем сонар в соответствующие отверстия основы головы так, чтобы контакты оказались сверху.

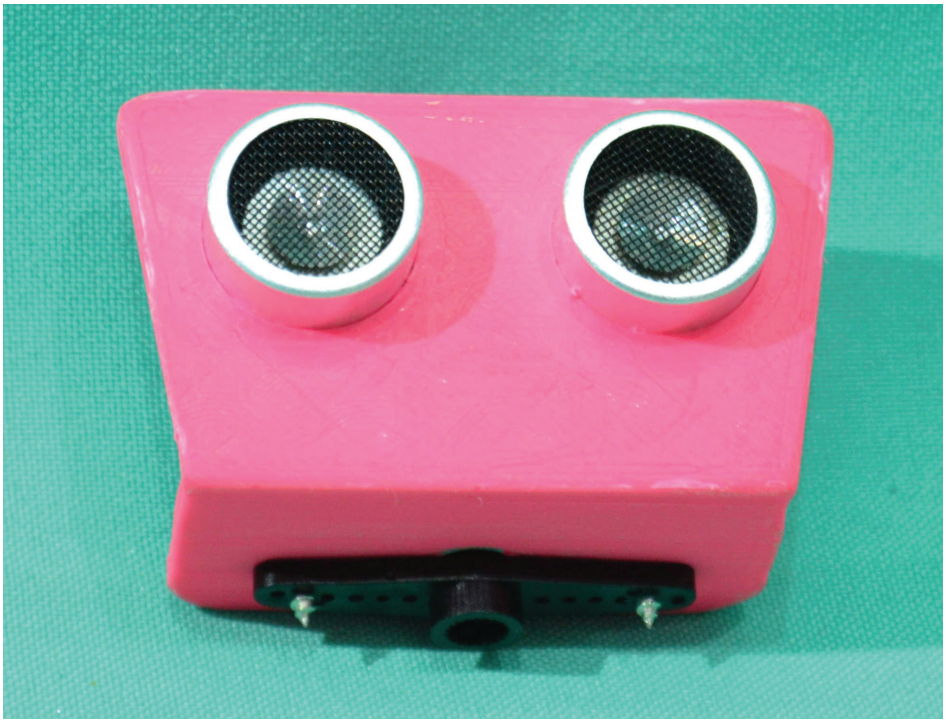


Рис. К9.10. Голова в сборе (вид спереди)

Теперь к ультразвуковому сонару можно прикрепить провода.

Рис. К9.11. Голова в сборе с прикрепленными проводами (вид сзади)

Инструкция по правильной регулировке положения головы есть в пункте «Монтаж головы» книги. Непосредственно сам монтаж показан на рис. К9.12–К9.14.

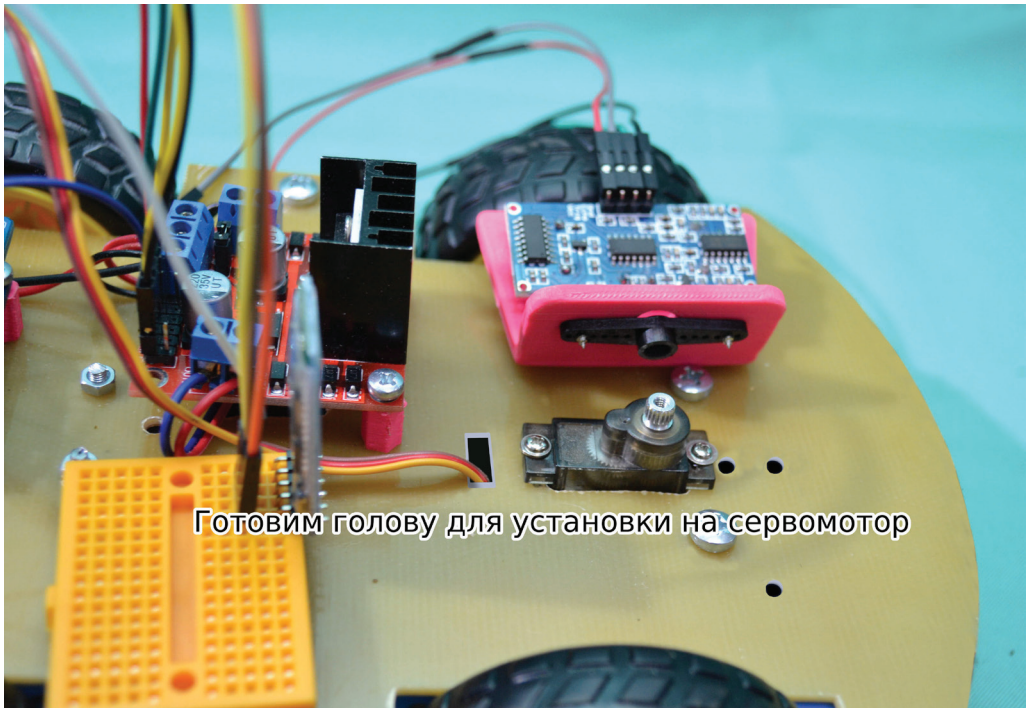


Рис. К9.12. Голова готова к установке

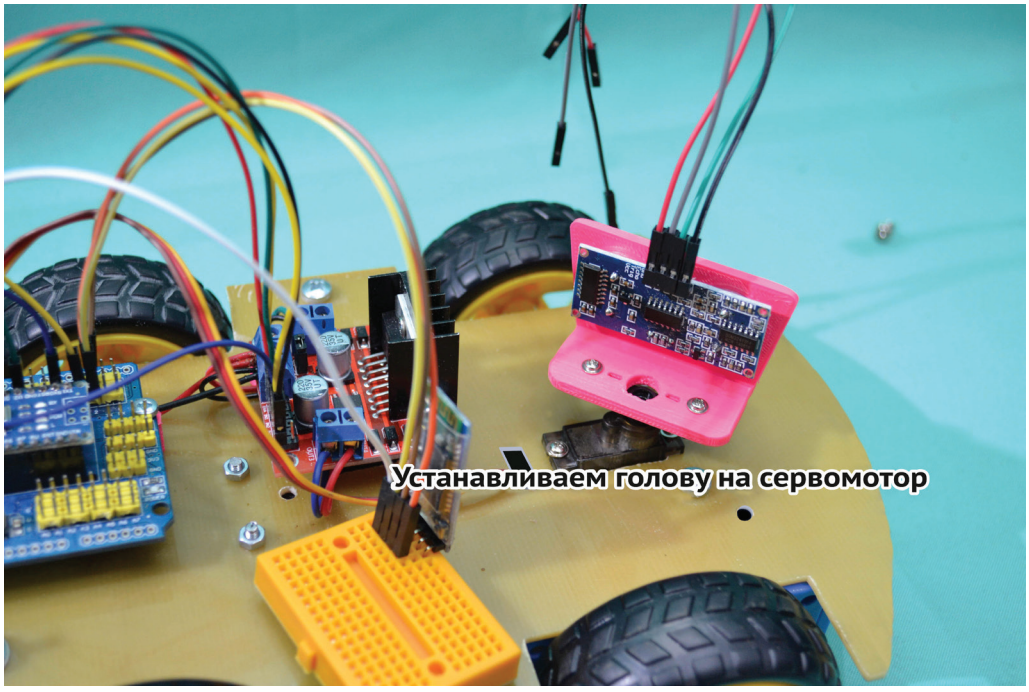


Рис. К9.13. Голова насажена на вал сервомотора

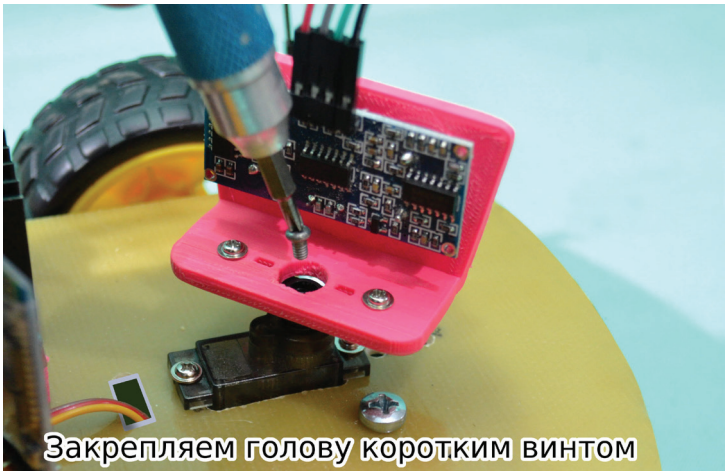


Рис. К9.14. Голова крепится винтом

Согласно схеме рис. К9.6, проводник от Trig присоединяем 5-му контакту (5 S) сервисной платы, Echo – к 6-му контакту (6 S) сервисной платы, Vcc – к любому свободному контакту в столбце V, GND – к любому свободному контакту в столбце G.

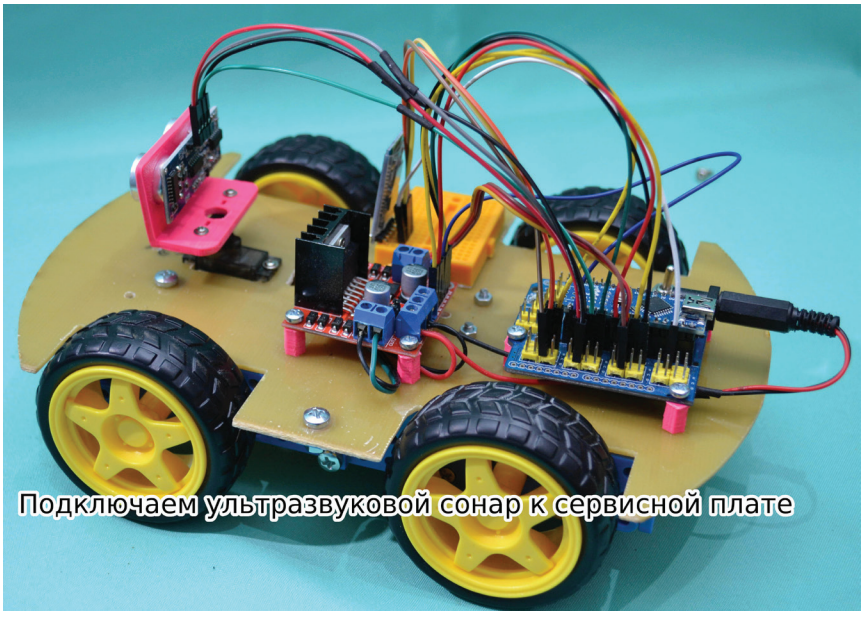


Рис. К9.15. Робот с головой в сборе

Робот собран, теперь можно приступать к программированию по 9-й главе книги, проверить работоспособность сонара и сервомотора, правильность установки головы. Все это потребуется для выполнения заданий 10-й главы книги. 10-я глава выполняется полностью по книге.

К ГЛАВЕ 11 РОБОТ, НАХОДЯЩИЙ ВЫХОД ИЗ ЛАБИРИНТА

Робот, находящий выход из лабиринта, использующий только ультразвуковой сонар, уже создан. Теперь можно загружать программу «Листинг 11.1», корректировать значения констант, оттачивать программу под своего робота.

Если все получилось, можно приступить к модернизации робота, установив на него инфракрасные детекторы препятствия.

В состав конструктора CLASSIC входят три платы IR-детекторов препятствия, соединительные и крепежные элементы.

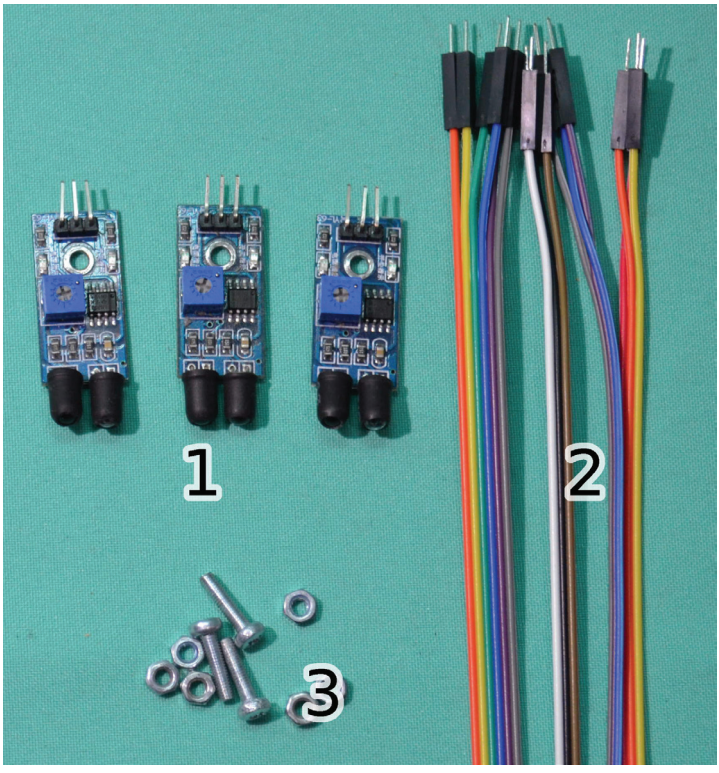


Рис. К11.1. IR-датчики (1), провода с разъемами «папа–мама» (2), крепежные элементы (3)

К датчикам следует подсоединить провода, как показано на рисунке.

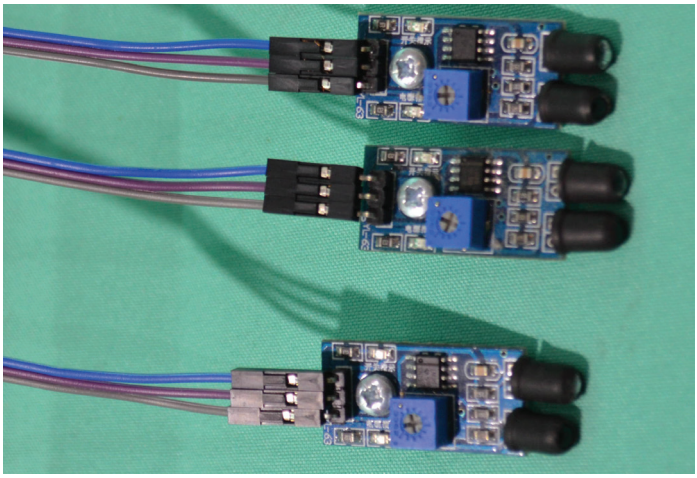


Рис. К11.2. IR-датчики с провода

Чтобы не тянуть провода через всего робота, воспользуемся возможностями макетной платы. Рисунок К11.3 демонстрирует подключение питания к датчикам, а рис. К11.4 – подключение информационных контактов.

Уточню, что мы действуем строго по 11-й главе, логическая схема робота не меняется!

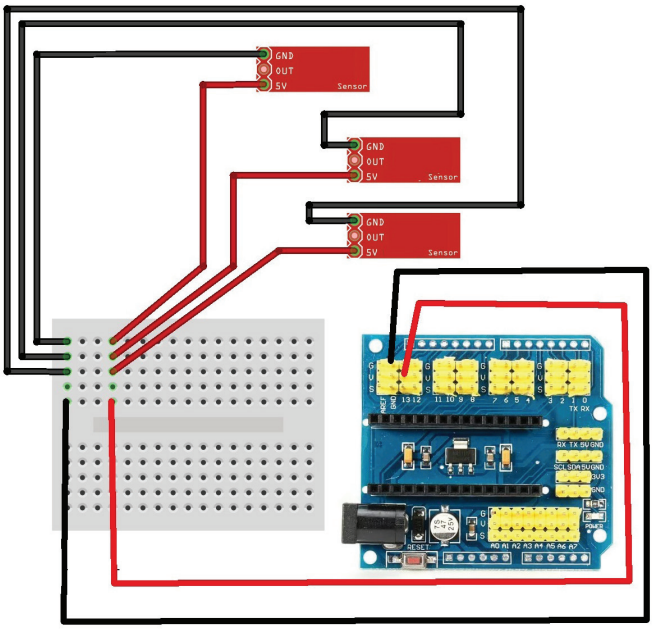


Рис. К11.3. IR-датчики – подключение питания

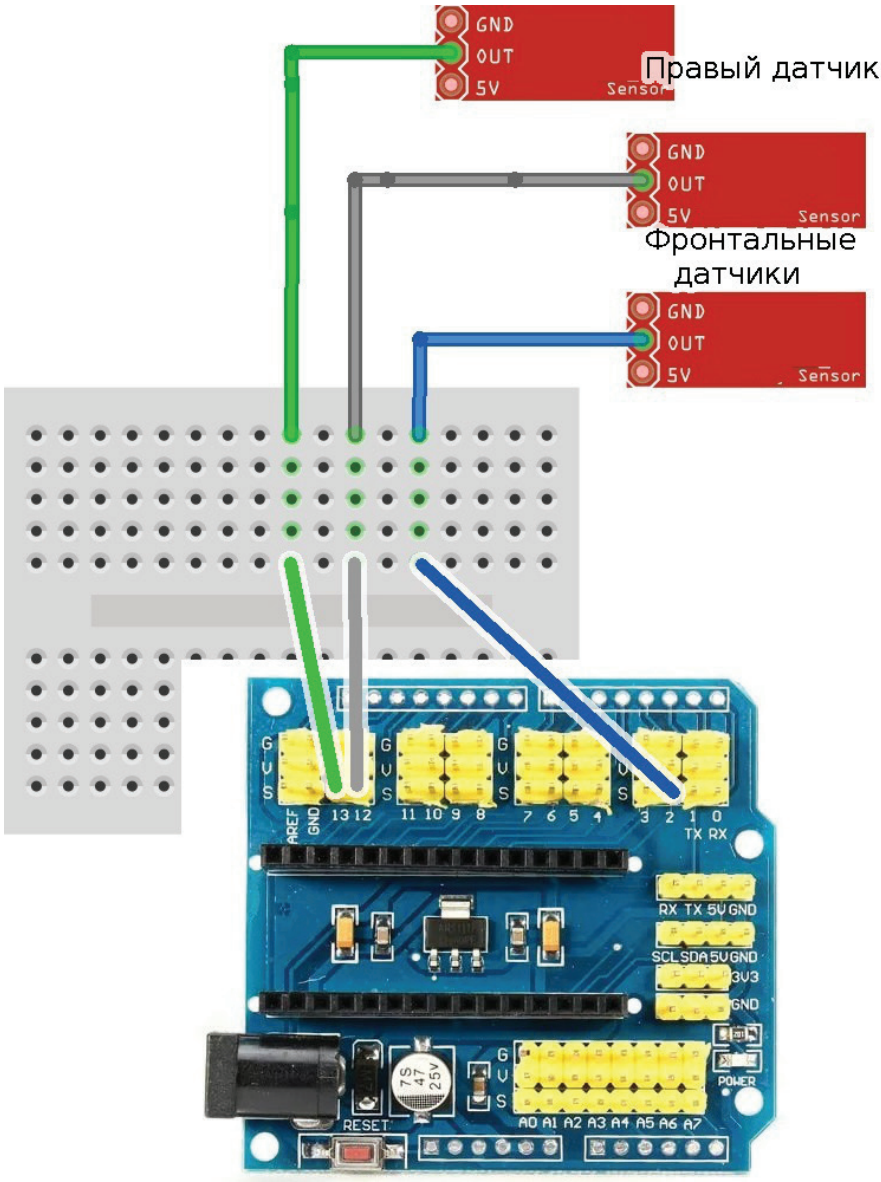


Рис. K11.4. IR-датчики – подключение информационных контактов

Снизу корпуса робота установим и закрепим гайками три винта, как показано на следующем рисунке. Они будут служить для крепления датчиков.

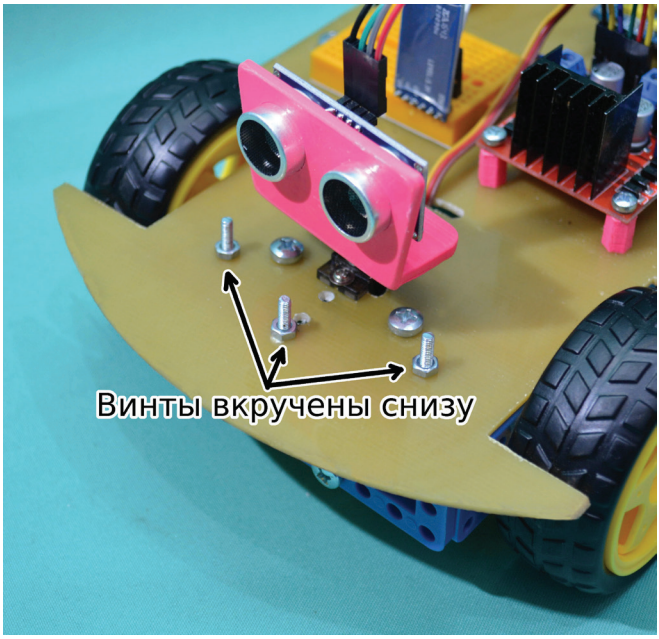


Рис. К11.5. Винты в передней части робота для крепления IR-детекторов препятствия

У датчиков есть крепежное отверстие. Установим датчики на винты.

Рис. К11.6. Датчики установлены на винты

Закрепим установленные датчики гайками и направим их согласно следующему рисунку.

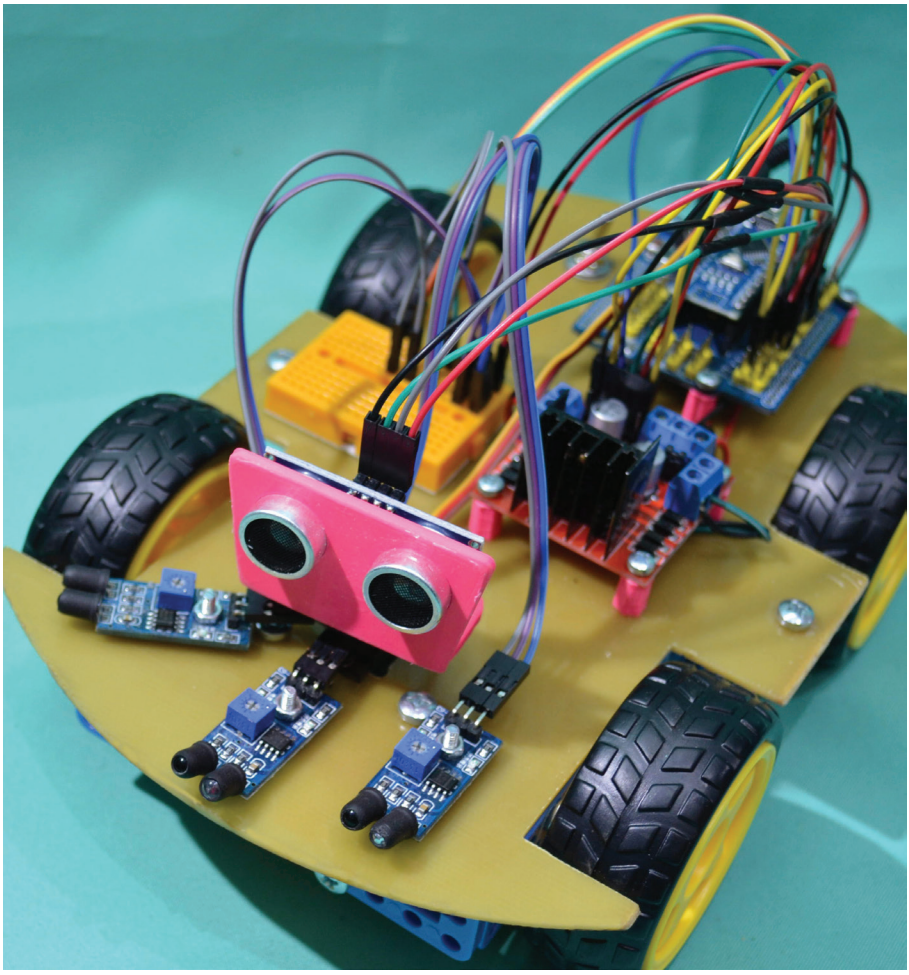


Рис. К11.7. Датчики установлены на винты и закреплены гайками

Теперь вернемся к рис. К11.3 и К11.4, произведем коммутацию строго по схемам. Полученный робот изображен на рис. К11.8. Он полностью готов к выполнению программ 11-й главы.

Обращаю внимание, что данные датчики не защищены от влияния внешнего инфракрасного излучения, например солнечного света!

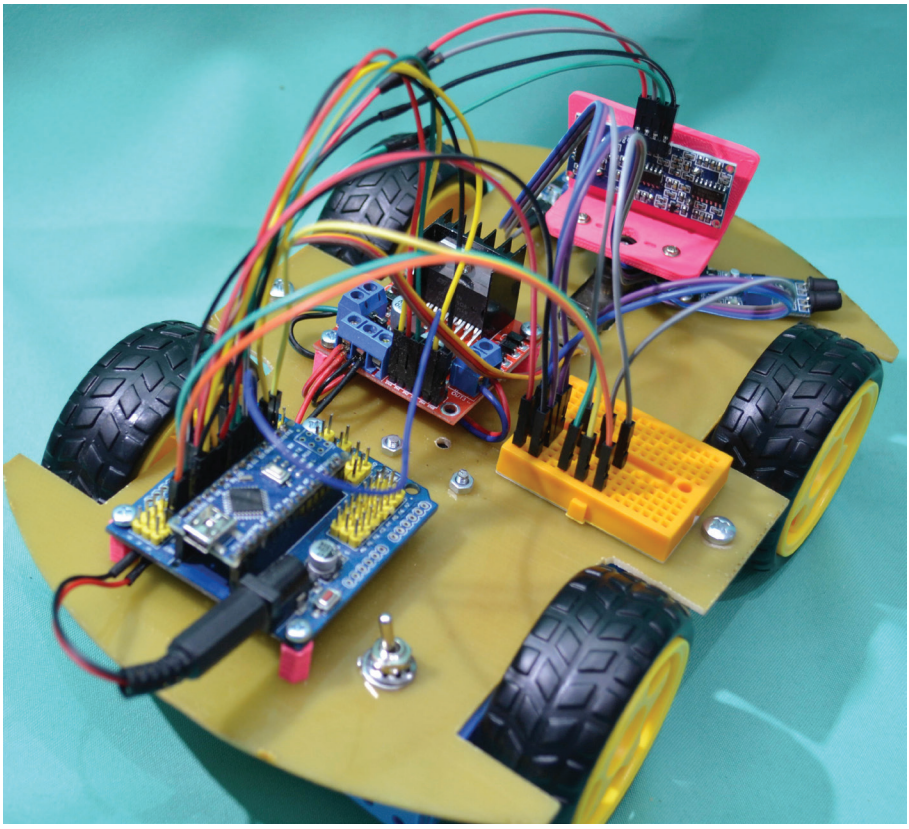


Рис. К11.8. Готовый робот для обхода лабиринта по трем IR-датчикам

Расстояние срабатывания датчика можно изменять, вращая подстроечный резистор, процесс срабатывания можно проследить визуально по соответствующему светодиоду. Расстояние срабатывания датчиков зависит от цвета поверхности, оно дальше для белого цвета и ближе для черного.

К ГЛАВЕ 12 СБОРКА РОБОТА ДЛЯ ВЫБИВАНИЯ ЦВЕТНЫХ КЕГЛЕЙ ИЗ КРУГА

Подробнее о кегельринге можно узнать из 12-й главы моей книги, но в этой сборке мы отойдем от примера, приведенного в книге, и создадим альтернативную программу, а также будем использовать специализированный датчик для определения цвета кегли.

Сборку данного робота нужно производить после того, как вы научили робота ориентироваться по электронному компасу HC5883L и разобрались, как подключать и использовать электронный гироскоп MPU-6050 (главы 13 и 15). Я буду использовать в программе электронный гироскоп MPU-6050 для осуществления поворотов и контроля прямолинейности движения.

В качестве датчика цвета будем использовать многофункциональный датчик APDS-9960. APDS-9960 применяется, как правило, в смартфонах, он может определять направление жеста, расстояние до объекта (до 20 см), величину освещенности и то, что нам как раз требуется, – цвет находящегося перед сенсором объекта. Датчик на плате показан ниже.

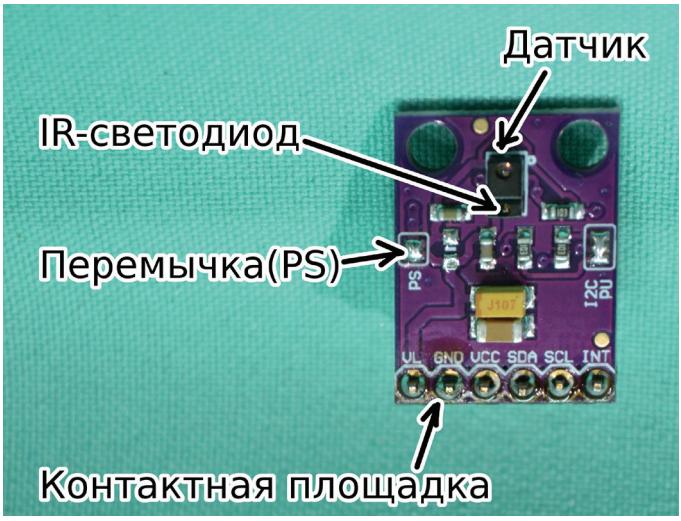


Рис. К12.1. Датчик APDS-9960 на плате

Датчик APDS-9960 подключается по интерфейсу I2C, а значит, он будет подключен к тем же проводам, что электронный компас и гироскоп. Питание датчика только 3.3 вольта, ни в коем случае его нельзя подключать к напряжению 5 вольт.

Сам датчик следует установить так, чтобы его сенсор смотрел прямо перед роботом. Для этого используем акриловую панель с вырезом под выводы датчика APDS-9960, два узких уголка, пару винтов M4-10 с гайками и пару винтов M2.5-6 впотай с соответствующими гаечками.

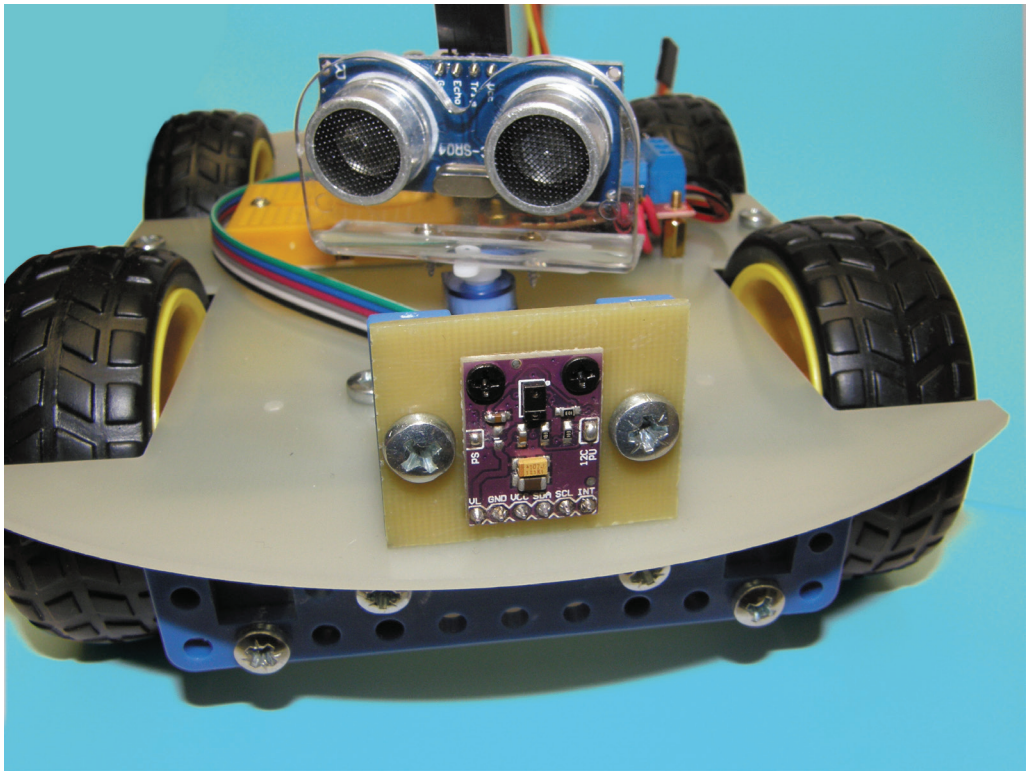


Рис. K12.2. Датчик APDS-9960 установлен на робота

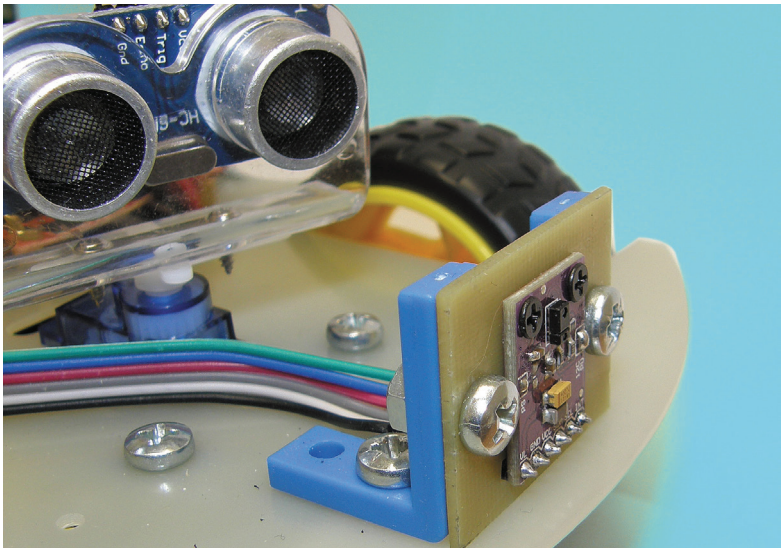


Рис. K12.3. Датчик APDS-9960 установлен на робота (вид сбоку)

Чтобы не осталось вопросов по правильному подключению датчика, подключим его через макетную плату, как показано на следующем рисунке. Фактически датчик APDS-9960 станет третьим датчиком, подключенным по интерфейсу I2C.

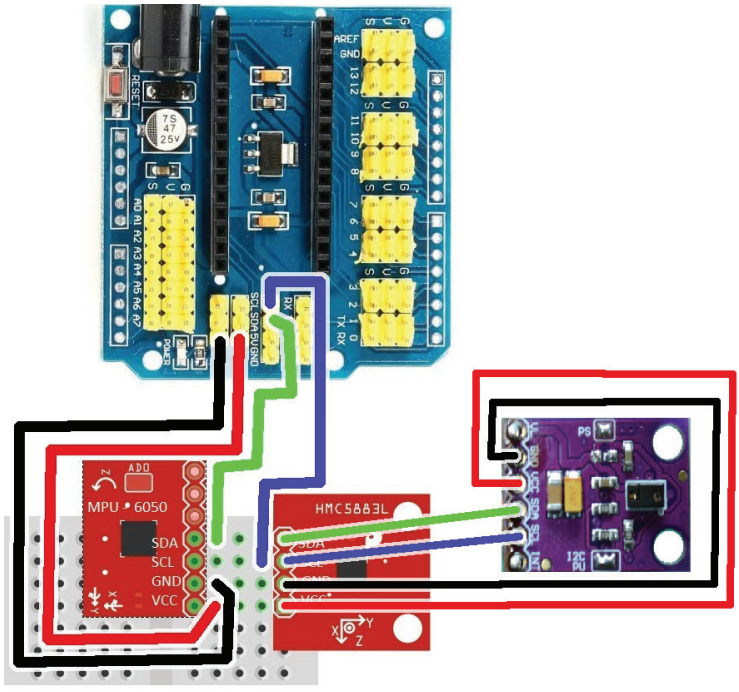


Рис. K12.4. Монтажная схема подключения датчика APDS-9960 через макетную плату, совместно с компасом и гироскопом

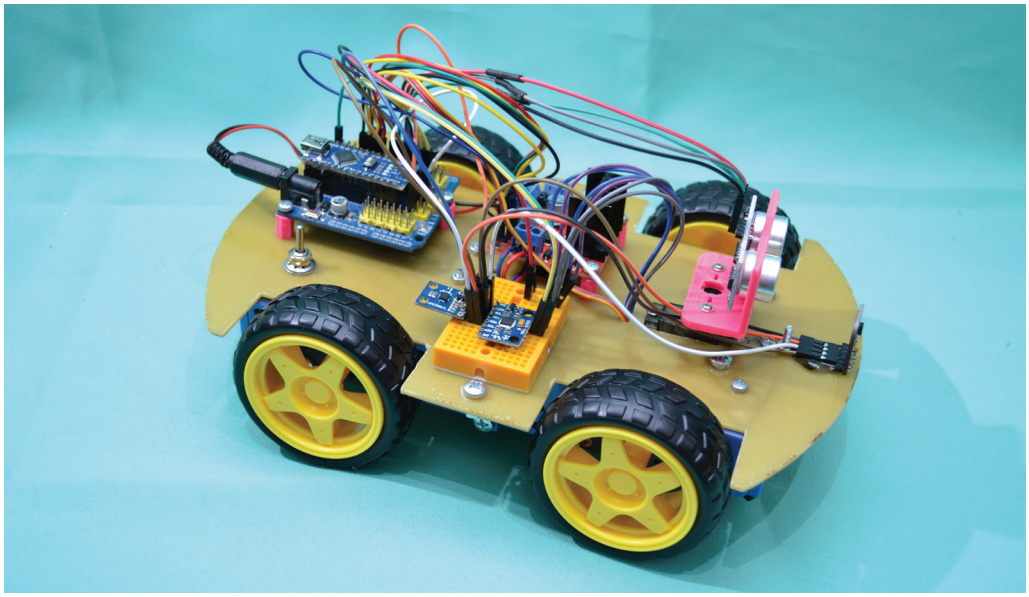


Рис. К12.5. Робот с установленным и подключенным датчиком APDS-9960

Ниже приведена программа по выбиванию разноцветных кеглей. В программе они обозначены как красные и желтые, но два цвета (по выбору) для сбиваемых кеглей запоминаются (записываются) роботом предварительно и могут быть другими по факту.

Еще следует установить инфракрасный приемник в порт A1 Arduino Nano Shield V3.0. Используем A1 разъем для подключения, согласно информации из рис. К8.2 устанавливаем датчик на плату Arduino Nano Shield V3.0 в A1 строку. G в G, V в V, S в S.

Если подключить неправильно, датчик может быть испорчен!!!

Программа состоит из нескольких модулей и приведена здесь не полностью. Полный комплект программного обеспечения можно скачать с сайта http://www.robotbin.ru/CLASSIC/.

Порядок работы программы следующий. Сначала следует дать команды роботу запомнить два цвета:

1. Следует поднести кеглю выбранного цвета на расстояние 2–3 см от датчика APDS-9960. По нажатии цифры «3» на миниатюрном IR-пульте (как в книге) значения RGB-составляющих цвета кегли будут занесены в память робота.
2. Следует поднести кеглю второго выбранного цвета на расстояние 2–3 см от датчика APDS-9960. По нажатии цифры «6» на миниатюрном IR-пульте (как в книге) значения RGB-составляющих цвета второй кегли будут занесены в память робота.
3. Далее следует на IR-пульте нажать цифру «0», по этой команде полученные RGB-составляющие будут занесены в энергонезависимую память робота. Теперь робот будет различать указанные цвета даже после перезагрузки.

4. Теперь нужно расставить кегли внутри ринга на одинаковом угловом расстоянии 45 градусов. Если робота поставить в центр окружности и направить на кеглю (не важно, какого из двух цветов), то следует использовать команды «1» и «2» пульта.

5. Порядок работы робота следующий: робот подъезжает к кегле и останавливается. Если при этом робот не доезжает до кегли, следует немного увеличить в программе значение константы TIME_TO_KEG (она в начале программы). Если же робот, наоборот, сбивает кеглю, а потом останавливается, то следует уменьшить данную (TIME_TO_KEG) константу.

6. Если робот «узнал» кеглю, он должен ее выпихнуть за пределы окружности. За это отвечает константа TIME_TO_LINE. Если робот выезжает слишком далеко – уменьшите ее, а если недоталкивает кеглю – увеличьте ее значение.

7. В случае установки робота точно между кеглями следует нажимать на кнопки «4» и «5», в этом случае робот перед началом работы повернет на 22.5 градуса.

Если у вас другой пульт, замените коды кнопок на свои!!! Они указаны внутри блока switch, после ключевого слова case.

Все нестандартные библиотеки для робота находятся в рабочем каталоге.

Если на робота установить яркий светодиодный фонарь, который будет светить на объекты (кегли) перед роботом, качество распознавания им цветов значительно возрастет.

Листинг К12.1. Программа обработки цветных кеглей

```
#define TIME_TO_KEG 500000 // в микросекундах
#define TIME_TO_LINE 500000 // в микросекундах

//Подключение библиотеки I2C шины.
#include <Wire.h>
//Подключение библиотеки удаленного управления.
#include «IRremote.h»
//Побключение библиотеки управления моторами.
#include «motor.h»
//Подключение библиотеки управления гироскопом.
#include «gyro_acsel.h»
//Подключение библиотеки обработки цветов.
#include «memcolor.h»
void yellow_8(float aaa);
void red_8(float aaa);
// Порт для IR-приемника.
int RECV_PIN = A1;
// Создание IR-приемника.
IRrecv irrecv(RECV_PIN);
//Переменная для результатов IR-приемника.
decode_results results;
//========================================
void setup()
{
  Serial.begin(9600);
  //запуск гироскопа.
  giroscop_setup();
  apds9960_setup();
  // Переменные - номера контактов (пинов) Arduino.
  // Для левых и правых моторов машинки.
  setup_motor_system(3, 4, 11, 7, 8, 10);
  _stop(); //Двигатели остановлены.
```

```
  // Запуск IR-приемника.
  //Расчет компенсации утечки нуля гироскопа.
  delay(1000);
  Calc_CompensatorZ(3000);
  irrecv.enableIRIn();
  Serial.print(" Start ");
  Ang_ = 0;
  setspeed(255, 255);
}
//=====================================
// Основная программа.
void loop()
{
  // float Ang2 = Ang_;
  while (1)
  {
    // delay(30000);
    if (irrecv.decode(&results))
    {
      //  Serial.println(results.value, HEX);
      switch (results.value) {
        // 1 Желтые
        case 0xC101E57B:
          yellow_8(0);
          break;
        // 2 Красные
        case 0x97483BFB:
          red_8(0);
          break;
        // 3 фото желтые
        case 0xF0C41643:
          foto_yellow();
          break;
        // 4 желтые со смещением
        case 0x9716BE3F:
          yellow_8(22.5);
          break;
        // 5 красные со смещением
        case 0x3D9AE3F7:
          red_8(22.5);
          break;
        // 6 фото красные
        case 0x6182021B:
          foto_red();
          break;
        // Вперед 10 сек
        case 0x511DBB:

          break;
        // Назад 10 сек
        case 0xA3C8EDDB:
          break;
        case 0x32C6FDF7: // "*"
          number_test_yellow = 0;
          number_test_red = 0;
          Serial.println("number_test_yellow = 0; number_test_red = 0;");
          break;
        case 0x1BC0157B: // "0"
          write_color_data_eeprom();
          break;
      }
```

```
      irrecv.resume();
    }
    // delay(40);
    time_gyro(0.5);
    apds9960_read();
  }
}
//====================================
void yellow_8(float aaa)
{
  float x;
  x = aaa;
  while (x > 2)
  {
    Angle(x / 2);
    x = aaa - Ang_;
  }
  time_gyro(100);
  Ang_ = 0;
  for (int i = 0; i < 8; i++)
  {
    forward_t(TIME_TO_KEG);
    time_gyro(500);
    if (test_yellow())
    {
      //Serial.println(" Yellow ");
      forward_t(TIME_TO_LINE);
      time_gyro(500);
      backward_t(TIME_TO_LINE);
    }
    backward_t(TIME_TO_KEG);
    time_gyro(500);
    x = float((i + 1) * 45) - Ang_;
    while (x > 2)
    {
      Angle(x / 2);
      x = float((i + 1) * 45) - Ang_;
    }
    time_gyro(500);
  }
}
//====================================
void red_8(float aaa)
{
  float x;
  x = aaa;
  while (x > 2)
  {
    Angle(x / 2);
    x = aaa - Ang_;
  }
  time_gyro(100);

  Ang_ = 0;
  for (int i = 0; i < 8; i++)
  {
    forward_t(TIME_TO_KEG);
    time_gyro(500);
    if (test_red())
    {
      // Serial.println(" red ");
```

```
      forward_t(TIME_TO_LINE);
      time_gyro(500);
      backward_t(TIME_TO_LINE);
    }
    backward_t(TIME_TO_KEG);
    time_gyro(500);
    x = float((i + 1) * 45) - Ang_;
    while (x > 2)
    {
      Angle(x / 2);
      x = float((i + 1) * 45) - Ang_;
    }
    time_gyro(500);
  }
}
```

Листинг К12.2. Содержимое файла memcolor.h

```
#include <EEPROM.h>
// Датчик цвета
#include "SparkFun_APDS9960.h"

// Global Variables
SparkFun_APDS9960 apds = SparkFun_APDS9960();
uint16_t ambient_light = 0;
uint16_t red_light = 0;
uint16_t green_light = 0;
uint16_t blue_light = 0;
uint8_t proximity_data = 0;

byte Y_max_mass_red_light;
byte Y_min_mass_red_light;
byte Y_max_mass_green_light;
byte Y_min_mass_green_light;
byte Y_max_mass_blue_light;
byte Y_min_mass_blue_light;

byte R_max_mass_red_light;
byte R_min_mass_red_light;
byte R_max_mass_green_light;
byte R_min_mass_green_light;
byte R_max_mass_blue_light;
byte R_min_mass_blue_light;

byte mass_red_light;
byte mass_green_light;
byte mass_blue_light;

byte number_test_yellow = 0;
byte number_test_red = 0;
byte correct_data = 0;
//============================================================
void write_int_to_eeprom(int ny, int data) //Младший адрес и переменная
{
  byte temp = data;// младший байт;
  EEPROM.write(ny, temp);
  temp = data >> 8;
  EEPROM.write(ny + 1, temp);
}
```

```
//============================================================
int read_int_from_eeprom(int ny) //Младший адрес и переменная
{
  int  temp = EEPROM.read(ny + 1) << 8; //старший байт
  temp |= EEPROM.read(ny); //младший байт

  return temp ;
}
//============================================================
bool read_color_data_eeprom()
{
  correct_data = EEPROM.read(0);
  if (correct_data == 55)
  {
    Y_max_mass_red_light = EEPROM.read(1); //
    Y_min_mass_red_light = EEPROM.read(2); //
    Y_max_mass_green_light = EEPROM.read(3); //
    Y_min_mass_green_light = EEPROM.read(4); //
    Y_max_mass_blue_light = EEPROM.read(5); //
    Y_min_mass_blue_light = EEPROM.read(6); //

    R_max_mass_red_light = EEPROM.read(7); //
    R_min_mass_red_light = EEPROM.read(8); //
    R_max_mass_green_light = EEPROM.read(9); //
    R_min_mass_green_light = EEPROM.read(10); //
    R_max_mass_blue_light = EEPROM.read(11); //
    R_min_mass_blue_light = EEPROM.read(12); //
    number_test_yellow = 1;
    number_test_red = 1;
    Serial.println(" Data read ");
  }
  else   return false;
  return true;
}
//============================================
void apds9960_setup()
{
  apds.init();
  apds.enableLightSensor(false);
  apds.setProximityGain(PGAIN_4X);
  apds.enableProximitySensor(false);
  read_color_data_eeprom();
}
//============================================================
void apds9960_read()
{
  apds.readAmbientLight(ambient_light);
  apds.readRedLight(red_light);
  apds.readGreenLight(green_light);
  apds.readBlueLight(blue_light) ;
  apds.readProximity(proximity_data);
  long mass = red_light + green_light + blue_light;
  mass_red_light = byte((long(red_light) * 100) / mass);
  mass_green_light = byte((long(green_light) * 100) / mass);
  mass_blue_light = byte((long(blue_light) * 100) / mass);
//  Serial.println(mass_red_light);
//  Serial.println(mass_green_light);
//  Serial.println(mass_blue_light);
//  Serial.println("//=======================================");
}
//============================================================
```

```
bool write_color_data_eeprom()
{
  if (number_test_yellow > 0)
  {
    EEPROM.write(1, Y_max_mass_red_light); //
    delay(1);
    EEPROM.write(2, Y_min_mass_red_light); //
    delay(1);
    EEPROM.write(3, Y_max_mass_green_light); //
    delay(1);
    EEPROM.write(4, Y_min_mass_green_light); //
    delay(1);
    EEPROM.write(5, Y_max_mass_blue_light); //
    delay(1);
    EEPROM.write(6, Y_min_mass_blue_light); //
    delay(1);
    EEPROM.write(0, 55);
    delay(1);
    Serial.println(" Write Yellow ");
  }
  if (number_test_red > 0)
  {
    EEPROM.write(7, R_max_mass_red_light); //
    delay(1);
    EEPROM.write(8, R_min_mass_red_light); //
    delay(1);
    EEPROM.write(9, R_max_mass_green_light); //
    delay(1);
    EEPROM.write(10,  R_min_mass_green_light); //
    delay(1);
    EEPROM.write(11,  R_max_mass_blue_light); //
    delay(1);
    EEPROM.write(12,  R_min_mass_blue_light); //
    delay(1);
    EEPROM.write(0, 55);
    delay(1);
    Serial.println(" Write Red ");
  }
  EEPROM.write(13,  99); //
}
//================================================
//Запоминает текущий цвет как желтый
bool foto_yellow()
{
  apds9960_read();
  //проверить расстояние
 // Serial.print("proximity_data = "); Serial.println(proximity_data);
  if(proximity_data<37) return false;
  if (number_test_yellow == 0)
  {
    Y_max_mass_red_light = Y_min_mass_red_light = mass_red_light;
    Y_max_mass_green_light = Y_min_mass_green_light = mass_green_light;
    Y_max_mass_blue_light = Y_min_mass_blue_light = mass_blue_light;
    number_test_yellow++;
  }
  else
  {
    if (Y_max_mass_red_light <  mass_red_light)
Y_max_mass_red_light =  mass_red_light;
    if (Y_min_mass_red_light > mass_red_light )
```

```
Y_min_mass_red_light = mass_red_light;

    if (Y_max_mass_green_light <  mass_green_light)  Y_max_mass_green_light =  mass_green_light;
    if (Y_min_mass_green_light > mass_green_light ) Y_min_mass_green_light = mass_green_light;

    if (Y_max_mass_blue_light <  mass_blue_light)
Y_max_mass_blue_light =  mass_blue_light;
    if (Y_min_mass_blue_light > mass_blue_light )
Y_min_mass_blue_light = mass_blue_light;
    number_test_yellow++;
  }
  Serial.println(" photo yellow ");
  return true;
}
//================================================
//Запоминает текущий цвет как красный
bool foto_red()
{
  apds9960_read();
  //проверить расстояние
 // Serial.print("proximity_data = "); Serial.println(proximity_data);
 //  if(proximity_data<37) return false;
  if (number_test_red == 0)
  {
    R_max_mass_red_light = R_min_mass_red_light = mass_red_light;
    R_max_mass_green_light = R_min_mass_green_light = mass_green_light;
    R_max_mass_blue_light = R_min_mass_blue_light = mass_blue_light;
    number_test_red++;
  }
  else
  {
    if (R_max_mass_red_light <  mass_red_light)
R_max_mass_red_light =  mass_red_light;
    if (R_min_mass_red_light > mass_red_light )
R_min_mass_red_light = mass_red_light;

    if (R_max_mass_green_light <  mass_green_light)  R_max_mass_green_light =  mass_green_light;
    if (R_min_mass_green_light > mass_green_light ) R_min_mass_green_light = mass_green_light;

    if (R_max_mass_blue_light <  mass_blue_light)
R_max_mass_blue_light =  mass_blue_light;
    if (R_min_mass_blue_light > mass_blue_light )
R_min_mass_blue_light = mass_blue_light;
    number_test_red++;
  }
  Serial.println(" photo red ");
  return true;
}
//================================================
bool test_red()
{
  apds9960_read();
  //вставить проверку расстояния
//  Serial.print("proximity_data = "); Serial.println(proximity_data);
  if(proximity_data<37) return false;
    if (R_min_mass_red_light <= mass_red_light )
      if (R_max_mass_green_light >=  mass_green_light)
          if (R_max_mass_blue_light >=  mass_blue_light)
    return true;
  return false;
}
```

```
//================================================
bool test_yellow()
{
  apds9960_read();
  if(proximity_data<37) return false;
    if (R_min_mass_red_light <= mass_red_light )
      if (R_max_mass_green_light >=  mass_green_light)
          if (R_max_mass_blue_light >=  mass_blue_light)
    return false;
  return true;
}
```

Научим робота двигаться по черной линии

Для этого используем два датчика черной линии. Датчики черной линии, входящие в комплект робота CLASSIC, не имеют аналогового выхода и, как следствие, не смогут передавать роботу информацию о полутонах. Но при помощи подстроечного резистора можно настраивать уровень отраженного света, при котором срабатывает датчик. Для начала установим и подключим датчики.

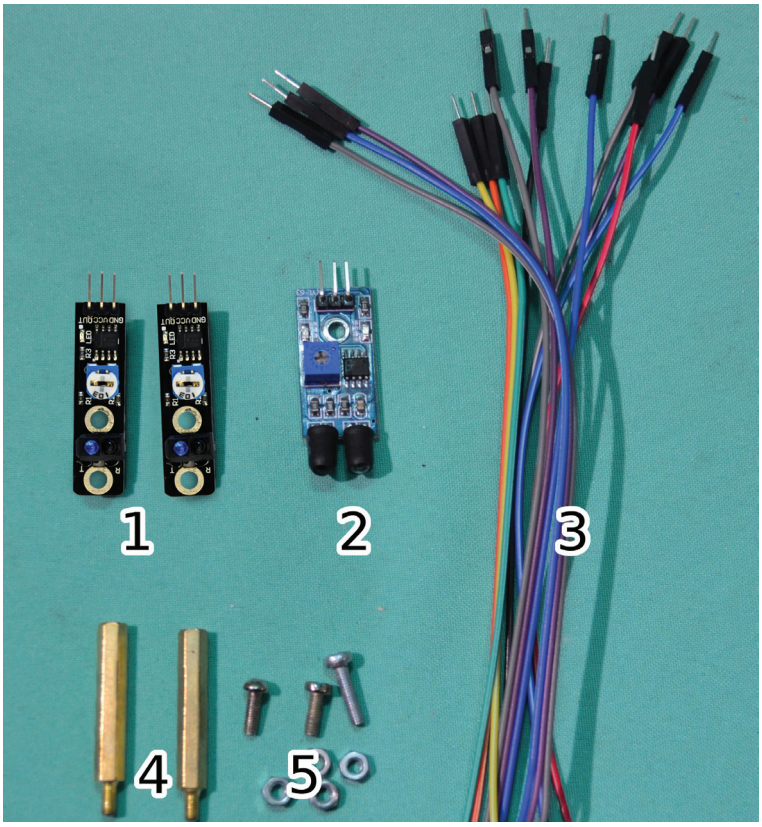


Рис. К12.6. Датчики линии (1), датчики препятствия (2), провода с коннекторами (3), крепежные стойки (4), винты и гайки (5)

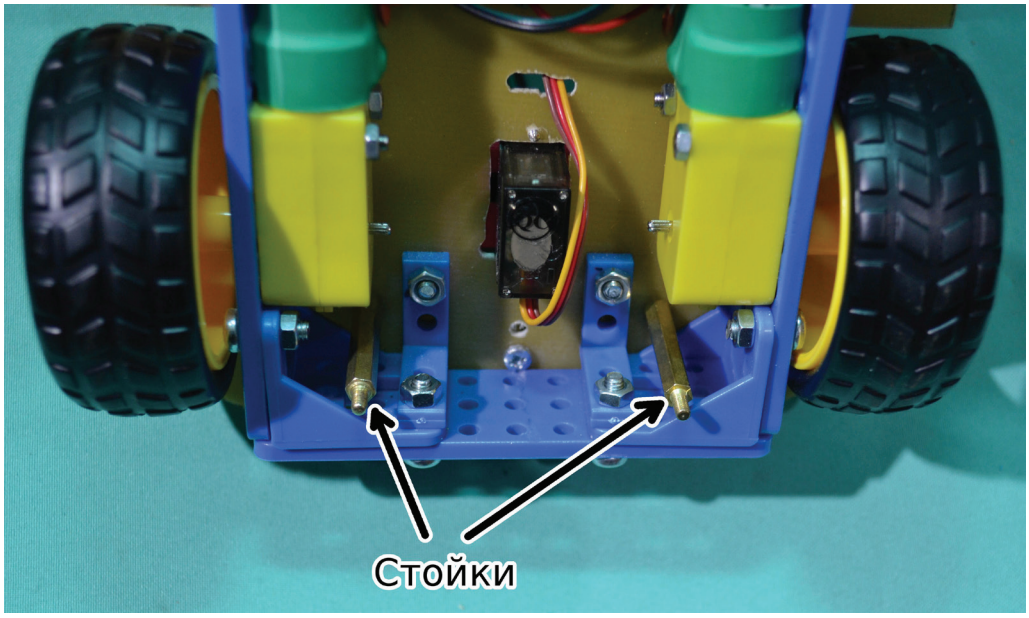


Рис. К12.7. Стойки для датчиков линии установлены в передней части робота

Рис. К12.8. Датчик линии (вид, обращенный на определяемую линию)

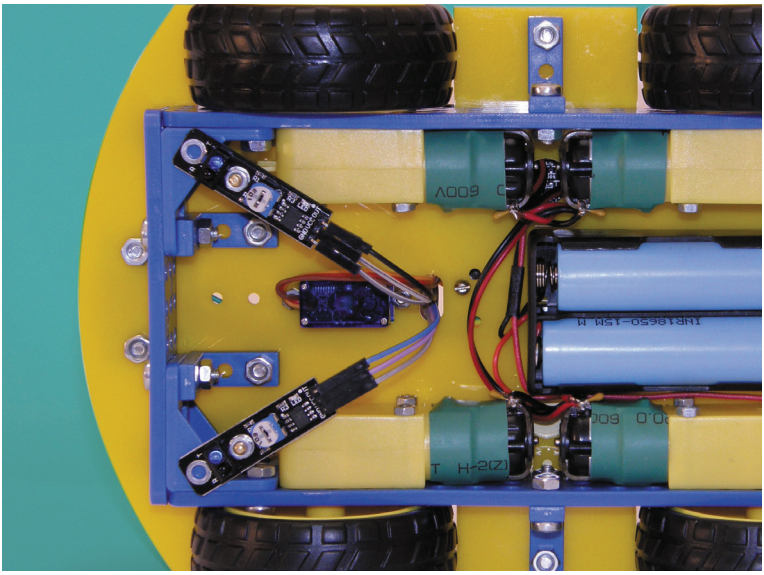


Рис. К12.9. Датчики линии закреплены на стойках, провода проложены через технологическое отверстие

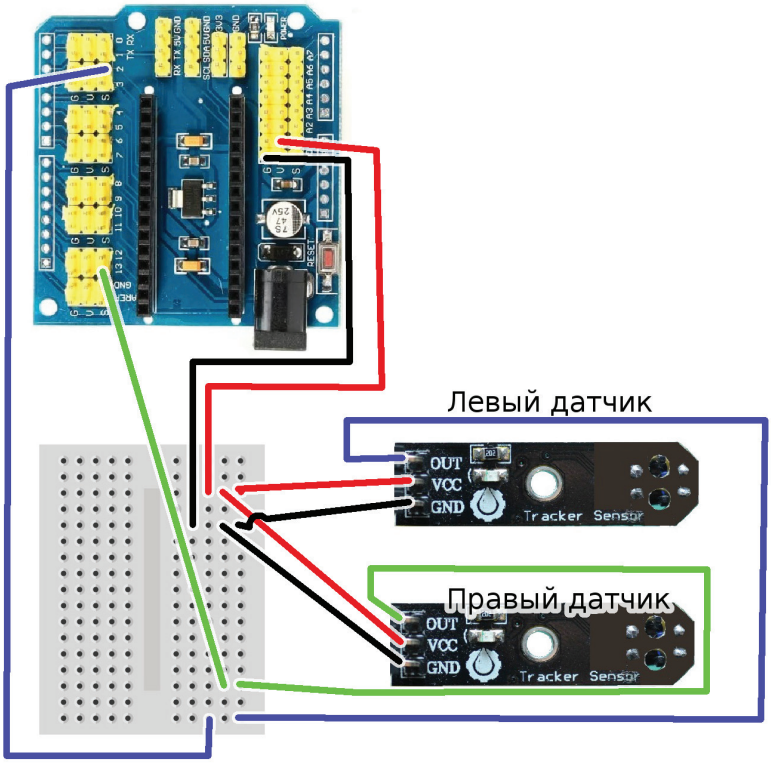


Рис. К12.10. Монтажная схема подключения датчиков линии к роботу (информационные контакты D2 и D13)

Датчик линии из комплекта CLASSIC имеет на выходе «1», если под сенсором черный цвет, и переходит в «0», если под сенсором светлая поверхность.

Создадим функцию fast_time_move(long ttime), которая осуществляет движение робота по черной линии в течение времени, указанного в параметрах «ttime».

Алгоритм движения робота следующий.

Если правый и левый датчики показывают «0» (оба на светлом фоне), считаем, что черная линия между датчиками и двигаемся вперед; иначе, если оба датчика на черном «1», поворачиваем направо; иначе, если на черном «1» только левый сенсор, поворачиваем налево; иначе, если на черном «1» только правый сенсор, поворачиваем направо.

Листинг К12.3. Программа движения робота по черной линии

```
//Подключение библиотеки управления моторами.
#include «motor.h»
//Номера пинов датчиков линии
const int left_line = 2;
const int right_line = 13;
//=======================================
void fast_time_move(long ttime);
void setup()
{
  Serial.begin(9600);
  //1-черная, 0-белая
  pinMode(left_line, INPUT);
  pinMode(right_line, INPUT);

  // Переменные - номера контактов (пинов) Arduino.
  // Для левых и правых моторов машинки.
  setup_motor_system(3, 4, 11, 7, 8, 10);
  _stop(); //Двигатели остановлены.
  setspeed(255, 255);
  delay(1000);
  Serial.print(" Start ");
}
//=====================================
// Основная программа.
void loop()
{
  fast_time_move(10000);
  delay(10000);
}
//=====================================
void fast_time_move(long ttime)
{
  bool  left_li, right_li;
  long timerTec;
  long timerStart;
  timerTec =  micros();
  timerStart = timerTec + ttime * 1000;
  while ( (timerTec < timerStart) )
  {
    left_li = digitalRead(left_line);
    right_li = digitalRead(right_line);
    if ((!left_li) && (!right_li))
    {
      forward();
```

```
    }
    else
    {
      if (left_li && right_li)
      {
        right();
      }
      else
      {
        if (left_li)
        {
          left();
        }
        else
        {
          right();
        }
      }
    }
    delayMicroseconds(150);
    _stop();
    delayMicroseconds(100);
    timerTec =  micros();
  }
}
```

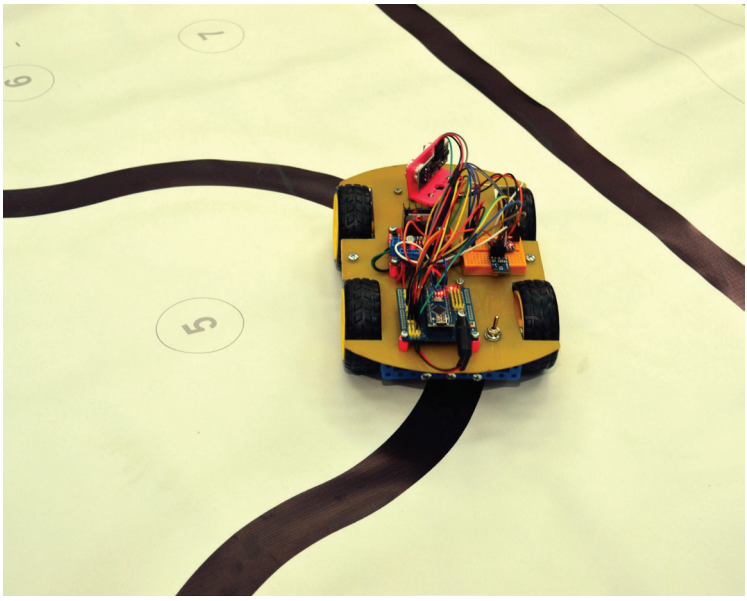


Рис. K12.11. Робот на черной линии

К ГЛАВЕ 13

Сборка робота, использующего для ориентации электронный компас

Используя монтажную схему, изображенную на следующем рисунке, подключим электронный компас к Arduino NANO.

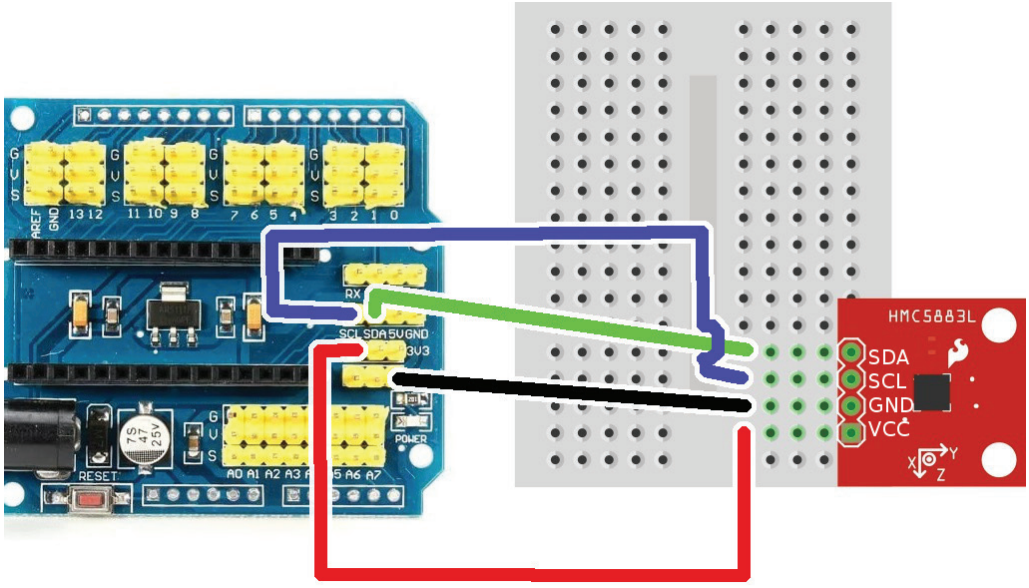
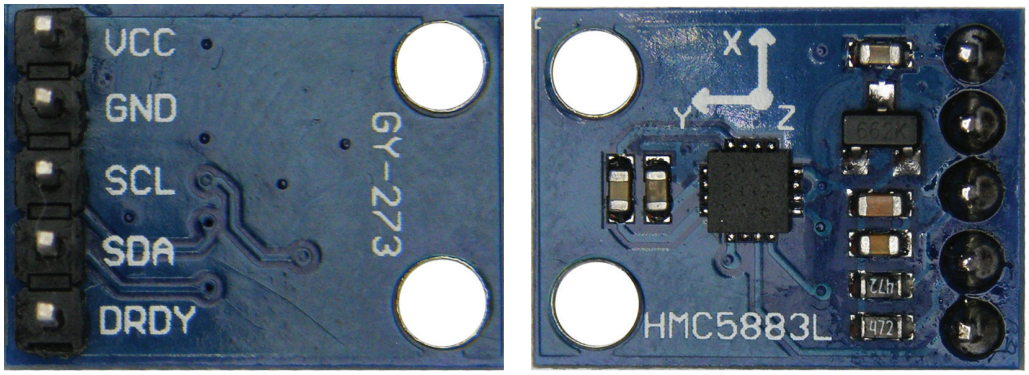


Рис. К13.1. Монтажная схема подключения электронного компаса HMC5883L (GY-273)

Обращаю внимание, что расположение контактов, приведенное на схеме, относится к датчику на плате, маркируемому как GY-273.



а) *б)*

Рис. K13.2 Электронный компас HMC5883L (GY-273) (вид снизу – а, вид сверху – б)

Непосредственно на роботе датчик должен быть расположен, как указано на следующей фотографии.

Компас постарайтесь расположить горизонтально, от этого зависит точность прибора.

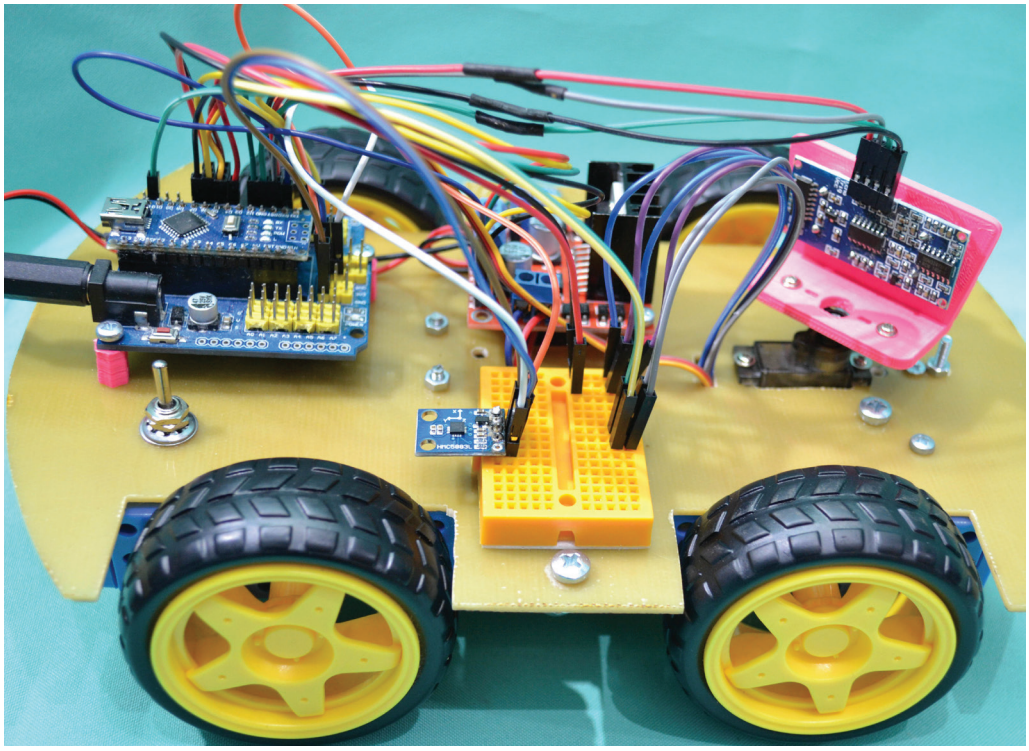


Рис. K13.3. Робот Classic с установленным электронным компасом HMC5883L (GY-273)

Теперь можно приступать к написанию и выполнению программ 13-й главы. Этого будет достаточно, чтобы познакомиться с прибором, но для использования

компаса на практике следует ознакомиться с учебным роликом «Магнитометр HMC-5883L, анализ и настройка», который я специально записал и выложил на YouTube: https://youtu.be/rwXrT2hQ5W8.

Рис. К13.4. Учебный видеоролик «Магнитометр HMC-5883L, анализ и настройка»

К ролику приложены листинги и библиотека, позволяющая значительно улучшить – откорректировать работу прибора: https://yadi.sk/d/ITeD9ZVB3Kjtbq.

К ГЛАВЕ 15 УСТАНОВКА ЭЛЕКТРОННОГО ГИРОСКОПА – АКСЕЛЕРОМЕТРА MPU-6050

Для реализации возможности осуществления точных поворотов и прямолинейного движения робота по программе можно использовать программу, приведенную в листинге 15.1 книги, но для наглядности опишем измененную программу, которая использует гироскоп для анализа углов поворота робота. Данная программа без подключенных библиотек приведена в листинге данного описания К15.1, подключенные библиотеки и саму программу можно скачать с сайта http://www.robotbin.ru/CLASSIC/. Программа называется «`test_gyro.ino`».

В программе реализована поддержка приема команд по инфракрасному порту, IR-приемник подключен на порт A0. Коды кнопок пульта заранее считаны и записаны в виде макроподстановок в начале программы (`#define STR_FORWARD 0x511DBB`). Действия робота по нажатии определенных кнопок пульта приведены в следующей таблице.

Таблица К1. Список команд, выполняемых роботом

Нажатая кнопка	Выполняемая команда
Стрелка вверх	Ехать строго вперед 5 секунд
Стрелка вниз	Ехать строго назад 5 секунд
Стрелка влево	Повернуть влево на 5 градусов по показаниям гироскопа
Стрелка вправо	Повернуть вправо на 5 градусов по показаниям гироскопа
Кнопка «OK»	Удерживать текущее положение в течение 10 секунд
Кнопка «1»	Повернуть влево на 10 градусов по показаниям гироскопа
Кнопка «2»	Повернуть вправо на 90 градусов по показаниям гироскопа
Кнопка «3»	Повернуть вправо на 10 градусов по показаниям гироскопа
Кнопка «4»	Повернуть влево на 20 градусов по показаниям гироскопа
Кнопка «5»	Повернуть влево на 180 градусов по показаниям гироскопа (разворот)
Кнопка «6»	Повернуть вправо на 20 градусов по показаниям гироскопа
Кнопка «7»	Повернуть влево на 60 градусов по показаниям гироскопа
Кнопка «8»	Повернуть влево на 90 градусов по показаниям гироскопа
Кнопка «9»	Повернуть вправо на 60 градусов по показаниям гироскопа
Кнопка «0»	Ехать назад 2 секунды
Кнопка «*»	Удерживать текущее положение в течение 10 секунд
Кнопка «#»	Удерживать текущее положение в течение 5 секунд

Функции управления движениями робота по гироскопу вынесены в отдельный файл (библиотеку) `gyro_acsel.h`, который находится в каталоге с основной про-

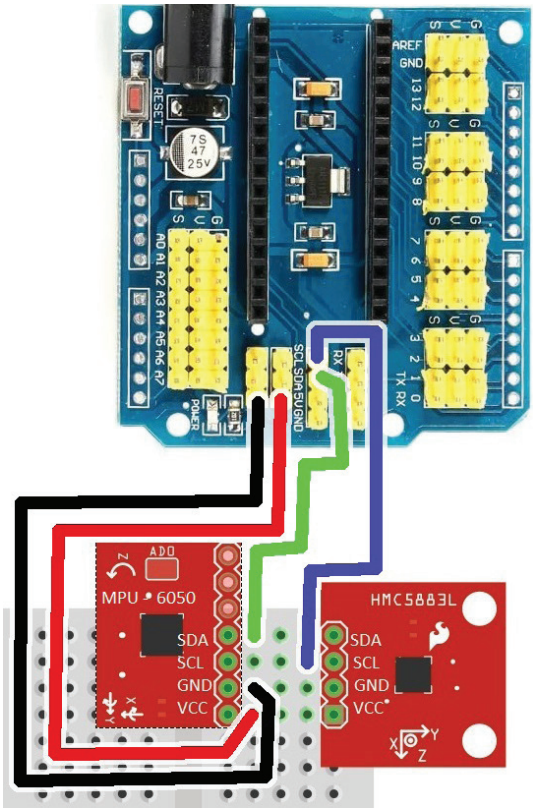


Рис. К15.1. Подключение MPU-6050 к роботу (монтажная схема)

граммой. Также в отдельном файле находятся функции управления моторами, это файл `motor.h`, остальные файлы библиотек, начинающиеся с «IR», относятся к работе с приемником сигналов инфракрасного пульта. Отдельные файлы можно редактировать совместно с основной программой, они присутствуют в виде дополнительных вкладок, между которыми можно переключаться (рис. К15.3).

Подключим к роботу гироскоп-акселерометр MPU-6050 согласно рис. К15.1. Фактически он будет подключен параллельно электронному компасу, более того, нередко при выполнении реальных задач эти приборы можно использовать вместе.

Обращаю внимание, что такая схема подключения возможна при использовании соответствующих плат с установленными приборами. Если контакты VCC, GND, SCL, CDA нельзя соединить через макетную плату попарно, следует использовать для этого провода.

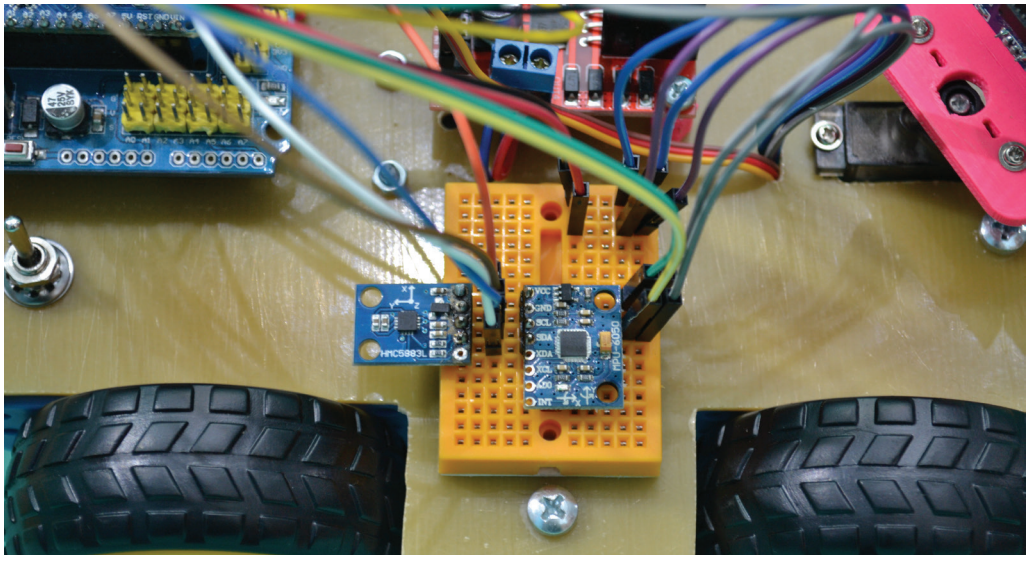


Рис. К15.2. Подключение MPU-6050 к роботу (вид сверху)

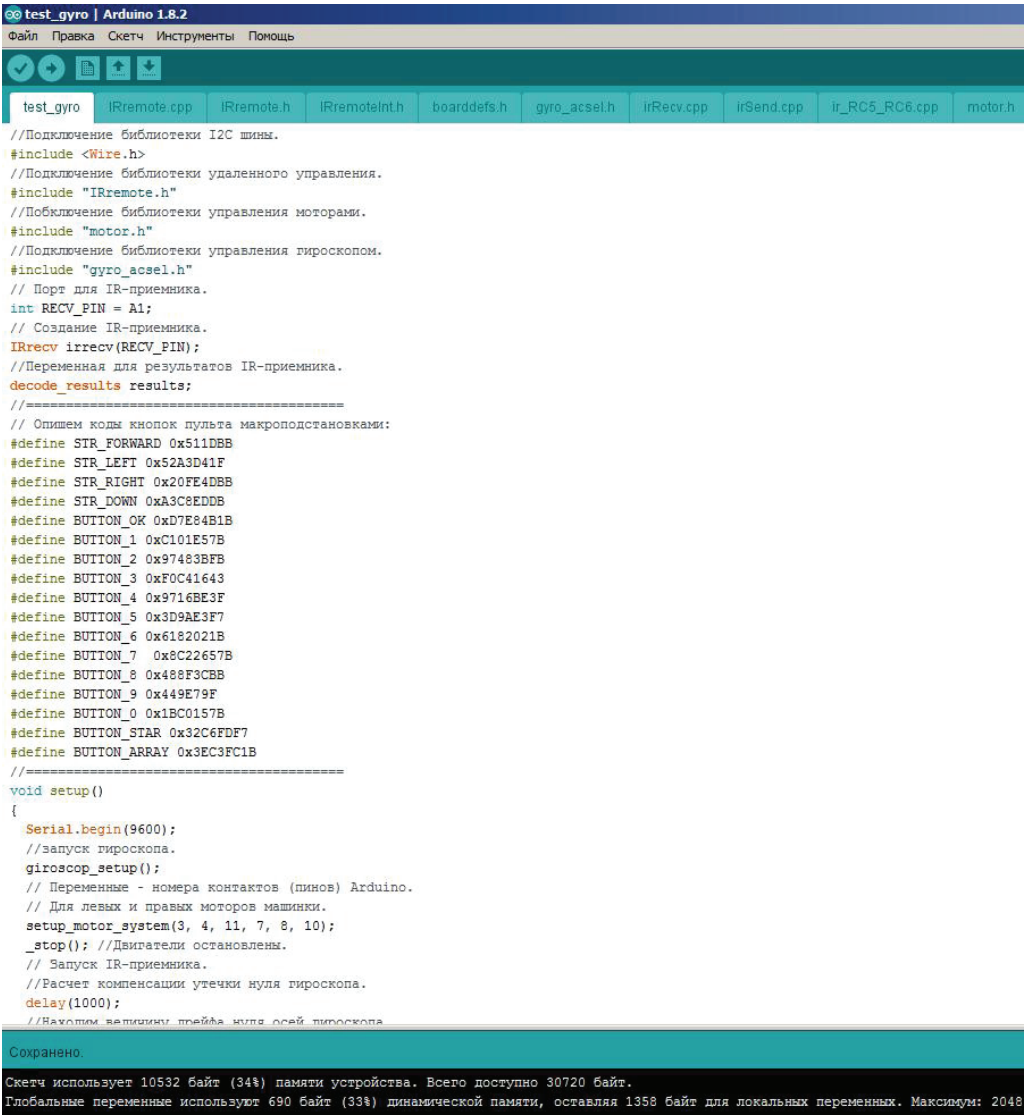


Рис. К15.3. Демонстрация вкладок в среде Arduino IDE (управление мобильным роботом с использованием гироскопа)

Листинг К15.1. Управление мобильным роботом с использованием гироскопа

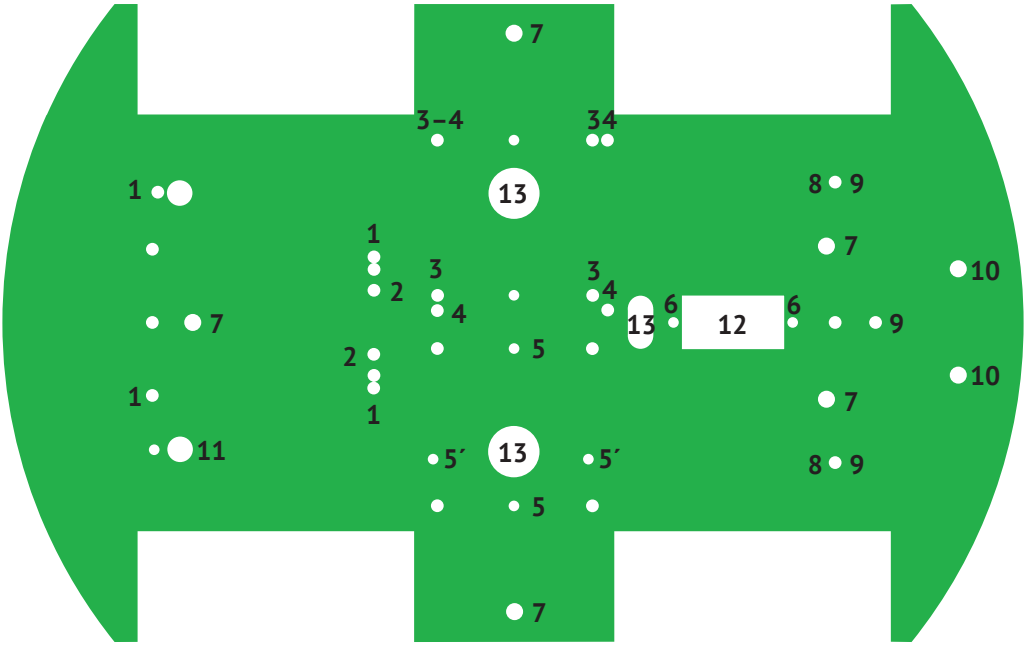
```
//Подключение библиотеки I2C шины.
#include <Wire.h>
//Подключение библиотеки удаленного управления.
#include «IRremote.h»
//Подключение библиотеки управления моторами.
#include «motor.h»
//Подключение библиотеки управления гироскопом.
#include «gyro_acsel.h»
// Порт для IR-приемника.
int RECV_PIN = A1;
// Создание IR-приемника.
```

```
IRrecv irrecv(RECV_PIN);
//Переменная для результатов IR-приемника.
decode_results results;
//=======================================
// Опишем коды кнопок пульта макроподстановками:
#define STR_FORWARD 0x511DBB
#define STR_LEFT 0x52A3D41F
#define STR_RIGHT 0x20FE4DBB
#define STR_DOWN 0xA3C8EDDB
#define BUTTON_OK 0xD7E84B1B
#define BUTTON_1 0xC101E57B
#define BUTTON_2 0x97483BFB
#define BUTTON_3 0xF0C41643
#define BUTTON_4 0x9716BE3F
#define BUTTON_5 0x3D9AE3F7
#define BUTTON_6 0x6182021B
#define BUTTON_7 0x8C22657B
#define BUTTON_8 0x488F3CBB
#define BUTTON_9 0x449E79F
#define BUTTON_0 0x1BC0157B
#define BUTTON_STAR 0x32C6FDF7
#define BUTTON_ARRAY 0x3EC3FC1B
//=======================================
void setup()
{
  Serial.begin(9600);
  //запуск гироскопа.
  giroscop_setup();
  // Переменные - номера контактов (пинов) Arduino.
  // Для левых и правых моторов машинки.
  setup_motor_system(3, 4, 11, 7, 8, 10);
  _stop(); //Двигатели остановлены.
  // Запуск IR-приемника.
  //Расчет компенсации утечки нуля гироскопа.
  delay(1000);
  //Находим величину дрейфа нуля осей гироскопа.
  Calc_CompensatorZ(3000);
  //Включаем прием IR-команд.
  irrecv.enableIRIn();
  Serial.print(" Start ");
  Ang_ = 0;
  _stop(); //Двигатели остановлены.
  setspeed(255, 255);
}
//====================================
// Основная программа.
void loop()
{

  while (true)
  {
    // delay(30000);
    if (irrecv.decode(&results))
    {
      switch (results.value) {
        case STR_FORWARD:
          // Ехать вперед 5 секунд (5 000 000 микросекунд)
          forward_t(5000000);
          break;
        case STR_LEFT:
          // Повернуть влево на 5 градусов (разворот)
          Angle(5);
          break;
        case STR_RIGHT:
          // Повернуть вправо на 5 градусов (разворот)
```

```
          Angle(-5);
          break;
        case STR_DOWN:
          // Ехать назад 5 секунд (5 000 000 микросекунд)
          backward_t(5000000);
          break;
        case BUTTON_OK:
          // Держать угол 10 секунд
          Angle_t(10000000);
          break;
        case BUTTON_1:
          // Повернуть влево на 10 градусов
          Angle(10);
          break;
        case BUTTON_2:
          // Повернуть вправо на 90 градусов
          Angle(-90);
          break;
        case BUTTON_3:
          // Повернуть вправо на 10 градусов
          Angle(-10);
          break;
        case BUTTON_4:
          // Повернуть влево на 20 градусов
          Angle(20);
          break;
        case BUTTON_5:
          // Повернуть влево на 180 градусов
          Angle(180);
          break;
        case BUTTON_6:
          // Повернуть вправо на 20 градусов
          Angle(-20);
          break;
        case BUTTON_7:
          // Повернуть влево на 60 градусов
          Angle(60);
          break;
        case BUTTON_8:
          // Повернуть влево на 90 градусов
          Angle(90);
          break;
        case BUTTON_9:
          // Повернуть вправо на 60 градусов
          Angle(-60);
          break;
        case BUTTON_0:
          // Ехать назад 2 секунды
          backward_t(2000000);
          break;
        case BUTTON_STAR:
          //Держать угол 10 секунд
          Angle_t(10000000);
          break;
        case BUTTON_ARRAY:
          //Держать угол 5 секунд
          Angle_t(5000000);
          break;
      }
      irrecv.resume();
    }
    time_gyro(10);
  }
}
```

Карта отверстий и их назначение



1 – стойки для Shield Nano v3;
2 – винты для аккумуляторного бокса;
3 – стойки для драйвера моторов;
4 – стойки для драйвера моторов RobotDYN;
5 – винты для макетной платы;
5´ – отверстия для крепления макетной платы при установке драйвера моторов RobotDYN;
6 – крепление сервомотора;
7 – крепление платформы к моторному отсеку;
8 – стойки датчиков линии;
9 – крепление датчика препятствия;
10 – крепление датчика цвета;
11 – тумблер питания;
12 – место для сервомотора;
13 – отверстия для проводов