Совместимо с Arduino

ПОП-БОТ

Набор для сборки мобильного робота

Руководство по Практическим Занятиям

версия 1.0 Стандартная

Авторские права

Arduino является проектом с открытыми исходными кодами, который поддерживается множеством людей во всем мире. Сообщество было основано и координируется Массимо Банзи (Massimo Banzi), Девидом Квартиелли (David Cuartielles), Томом Игое (Tom Igoe), Джиалука Мартино (Gianluca Martino) и Дэвидом Мэллисом (David Mellis). Николас Замбетти (Nicholas Zambetti) способствовал возникновению и развитию проекта. Янив Штейнер (Yaniv Steiner) и Джорджио Оливеро (Giorgio Olivero) поддерживали проект и работали на использованием его с платформой Instant Soup. Платформа Arduino включает в свой состав инструментарий avr-gcc, uisp и библиотеку Procyon AVR-LIB, написанную Паскалем Стэнгом (Pascal Stang). Синтаксис языка Arduino основывается на диаграммах Хернандо Баррагана (Hernando Barragan). Среда Arduino основана на работах Бена Фрая (Ben Fry) и Кэйси Риаса (Casey Reas). Огромное спасибо всем людям, участвующим в поддержке Arduino.

Оглавление

1 : Список комплектующих POP-BOT

1.1 Список комплектующих набора POP-BOT для сборки мобильного робота

1. Модуль микроконтроллера POP-168, совместимый с форматом Arduino-mini
2. Плата управления роботом RBX-168 с контейнером по 4 батареи типоразмера AA
3. Модуль переключателя с кабелем JST (2 комплекта)
4. Плата инфракрасного отражательного датчика с кабелем JST (2 комплекта)
5. Плата инфракрасного датчика расстояния GP2D120 с кабелем JST
6. Модуль ЖКИ (LCD) с организацией 16 символов в 2 строках со светодиодной подсветкой и кабелем
7. Электродвигатель постоянного тока на напряжение 4.5В с редуктором и кабелем IDC (2 комплекта)
8. Стандартный сервомотор (Рабочее напряжение от 4.8 до 7.2В постоянного тока)
9. Круглые колеса с резиновыми шинами и 2мм крепежными винтами (2 комплекта)
10. Пластиковые основания с отверстиями размером 80x60 см и 80x80 см (2 набора)
11. Круглое основание с шаровым колесом
12. Набор пластикового крепежа и прямых крепежных пластин (пластиковый крепеж 3-х типов и различного цвета в количестве 60 штук, 4 вида прямых крепежных пластин с количеством отверстий 3/5/12)
13. Набор прямоугольных металлических уголков (4 вида металлических уголков, с количеством отверстий на каждой из сторон 1x2, 2x2, 2x5)
14. Набор винтов и гаек
15. Бумажный лист, с примером траектории для движения робота
16. Преобразователь USB в последовательный интерфейс UCON-4 для загрузки прошивок и обмена данными
17. CD-ROM содержащий программный инструментарий, исходные коды примеров и документацию

1.2 Информация о компонентах микроконтроллера

1.2.1 Модуль микроконтроллера POP-168

Модуль Arduino POP-168 является платой с гибкими функциональными возможностями, с полным отсутствием скрытых компонентов, что обеспечивает полную реализацию их особенностей Стандартными инструментами разработки для AVR-микроконтроллеров, такими как IAR C/C++ , MikroElektronika Mikro BASIC/ MikroPascal for AVR, инструментарием с открытыми исходными кодами WINAVR: AVRGCC для Windows ... и т.п.

Arduino POP-168 использует микроконтроллер ATmega168 с архитектурой AVR компании Atmel ***(www.atmel.com). Назначение выводов Arduino POP-168 совпадает с назначением выводов модуля*** BASIC Stamp ***(www.parallax.com). Он содержит*** последовательный порт обмена данными RS-232 для загрузки прошивки и обмена данными с компьютером. Аппаратная часть модуля Arduino POP-168 совместима с модулем Arduino-mini в проекте Arduino ***(www.arduino.cc/en)***

Полная принципиальная схема модуля Arduino POP-168 показана на рисунке 1-1.

Модуль POP-168 имеет следующие функциональные особенности:

- Микроконтроллер ATmega168 с 10-разрядным АЦП (ADC) преобразователем, 16кБайт flash памяти программ, 512Байт ЭППЗУ (EEPROM), 1кБайт ОЗУ (RAM) и тактовой частотой 16МГц
- Встроенный интерфейс RS-232 для непосредственной загрузки кодов программ с помощью встроенного Начального Загрузчика (Bootloader)
- Кнопка Reset (Сброс) для обеспечения функции сброса
- Маленький размер, удобный для создания небольших конструкций
- Порт ISP для программирования с помощью устройства PX-400/PX-4000
- SMD светодиоды для индикации состояния
- Полностью совместим с Проектом Ardruino
- Назначение 16 линий Ввода/Вывода совместимы с модулем i-Stamp/i-Stamp2P24
- Диапазон напряжения питания от +3.3 до +5В при потребляемом токе 50мА

1.2.2 Плата управления роботом RBX-168 для Arduino POP-168

Плата управления RBX-168Robot является законченной, недорогой платформой, созданной для тех, кто интересуется изучением и использованием модулей Arduino POP-168 в приложениях робототехники. Ее небольшой размер, удобные функции и низкая стоимость делают ее идеальным инструментом для студентов и преподавателей. На Рисунке 1-2 показано расположение элементов на плате RBX-168, а ее полная принципиальная схема показана на Рисунке 1-3. Плата RBX-168 имеет следующие основные технические данные:

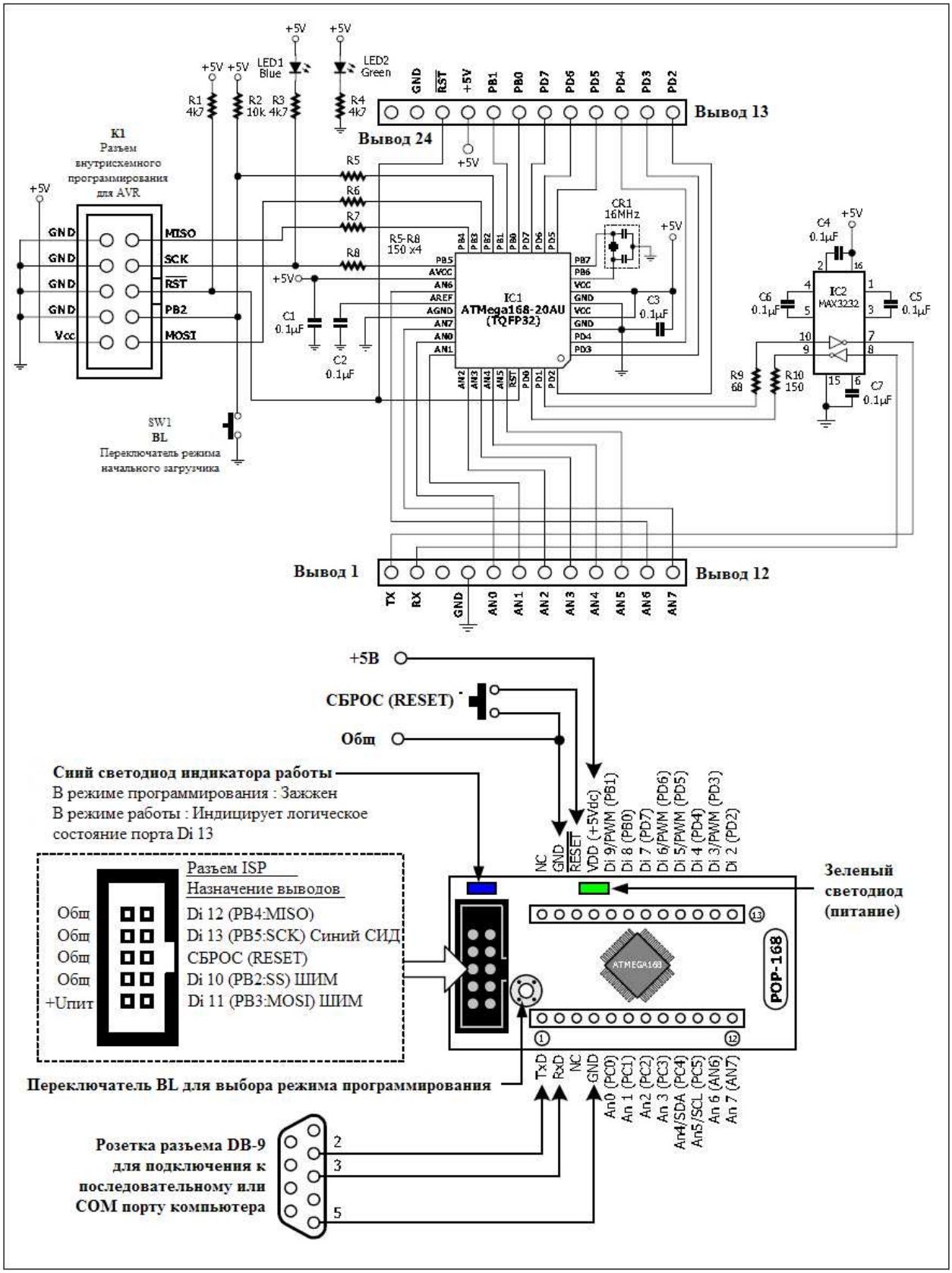


Рисунок 1-1. Схематическая диаграмма, назначение выводов и простейшая схема соединения модуля микроконтроллера POP-168

- Блок для подключения источника питания. Он поддерживает входное напряжение от +4.8 до +12 В постоянного тока и имеет встроенный выключатель питания.
- Схема импульсного стабилизатора напряжения +5В постоянного тока. Стабилизирует напряжение для модуля микроконтроллера POP-168 и всех портов сенсоров.
- Две кнопки, присоединенные к цифровым Портам 2 (Di2) и 4 (Di4). К выводам этих же портов подключены 2 светодиода, для индикации работы устройства.
- 5 универсальных портов поддерживают функции аналогового ввода и цифрового ввода/вывода; от An1 (Di15) до An5 (Di19)
- 2 порта аналогового ввода; An6 и An7. Оба вывода порта являются только аналоговыми входами.
- Порт шины I^{2}C; An4 (SDA) и An5 (SCL)
- Интерфейс последовательного порта RS-232.
- 2-канальный драйвер двигателя постоянного тока с индикаторами. Поддерживает моторы с рабочим напряжением от 2.5 до 13.5 В постоянного тока.
- 2 выхода для управления сервомоторами; присоединены к цифровым портам 7 (Di7) и 8 (Di8).
- Разъем для подключения пьезоизлучателя (не показан на рисунке 1-2; расположен на нижней стороне печатной платы модуля RBX-1689. Соединен с выводом An0/Di14 модуля POP-168.

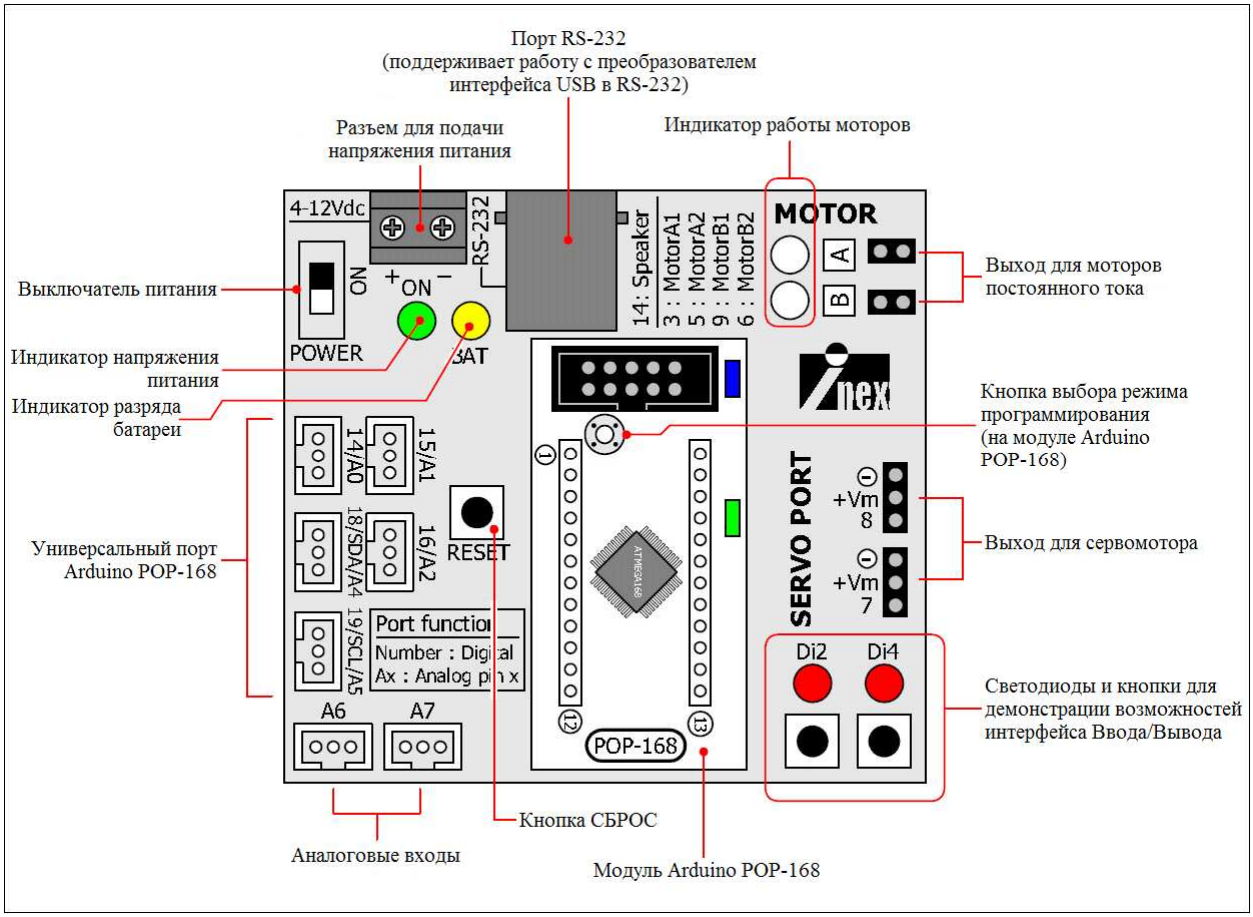


Рисунок 1-2. Схема расположения узлов на плате управления RBX-168

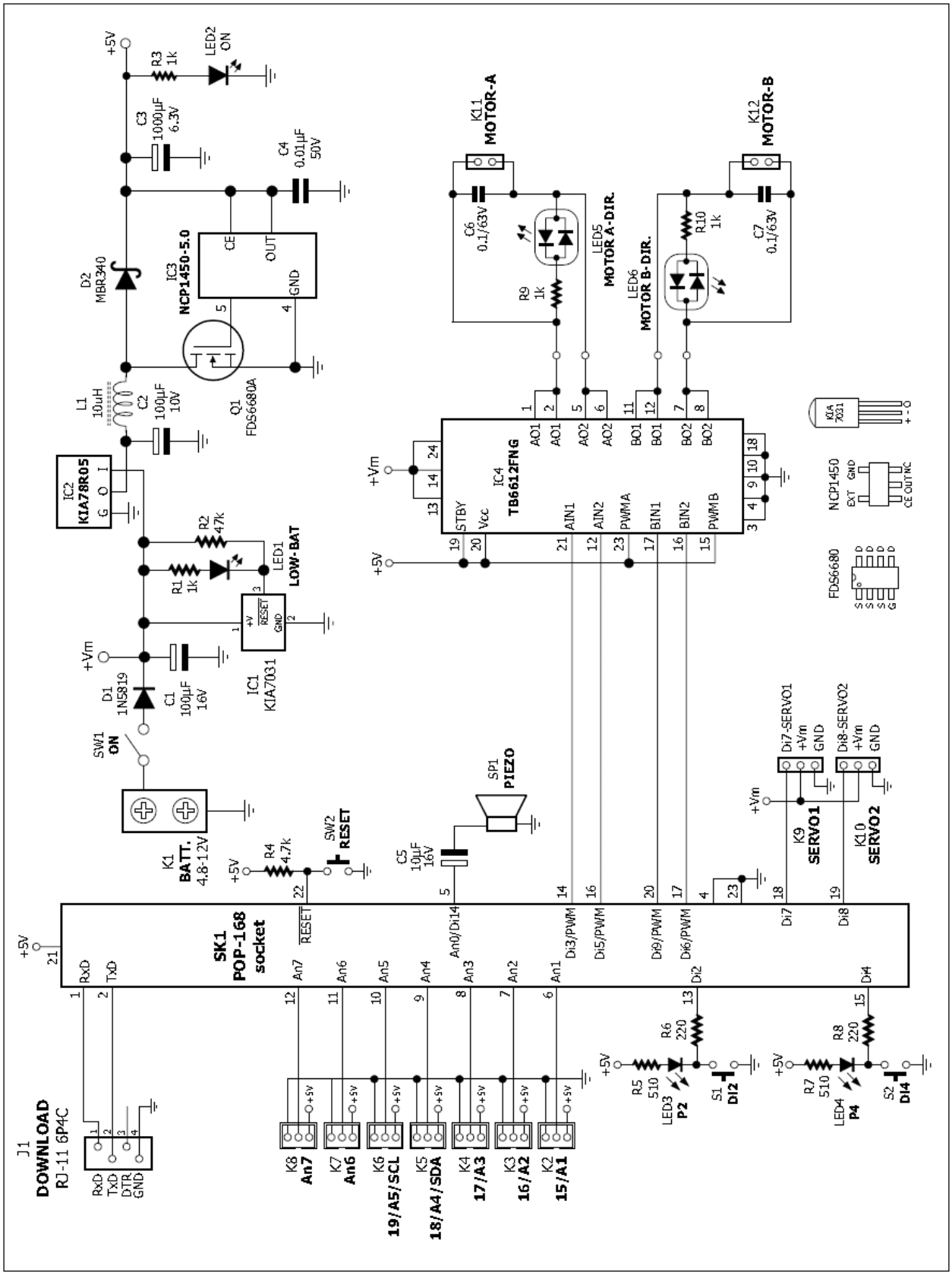
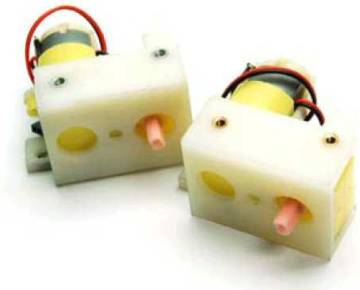


Рисунок 1-3. Полная принципиальная схема платы управления RBX-168

1.3 Особенности исполнительных устройств

1.3.1 Электродвигатели постоянного тока с редукторами

Данный набор для изготовления робота укомплектован электродвигателями постоянного тока с редукторами 48:1; тип BO-2 с соединительным кабелем IDC. Они имеют следующие особенности:



- Рабочее напряжение от +3 до +9В постоянного тока
- Потребление тока 130мА при напряжении +6В постоянного тока и отсутствии нагрузки
- Средняя частота вращения от 170 до 250 оборотов в минуту (RPM) при напряжении +6В и отсутствии нагрузки
- Вес 30 граммов
- Минимальный момент 0.5 кг*см.
- Присоединяется винтами с помощью 5 вмонтированных в пластик гаек
- Размер 42 x 45 x 22.7 мм (Ширина x Длина x Высота)

1.3.2 Стандартный RC-сервомотор

Стандартный сервомотор является идеальным для робототехнических применений и основных проектов перемещения. Эти сервомоторы позволяют осуществлять угловое перемещение в диапазоне от 0 до 180 градусов. Выходной механизм сервомотора обладает стандартной конфигурацией фирмы Futaba. Сервомотор имеет следующие технические характеристики:

- Максимальное рабочее напряжение: 6В постоянного тока.
- Средняя скорость выходного органа: перемещение в диапазоне от 0 до 180 градусов происходит за 1.5 секунды.
- Вес: 45.0 граммов/1.59 унции
- Момент: 3.40 кг*см/47унций*дюйм
- Размер, мм: (Д x Ш x В) 40.5x20.0x38.0

1.3.3 SLCD16x2 : Модуль ЖКИ с последовательным управлением на 2 строки по 16 символов

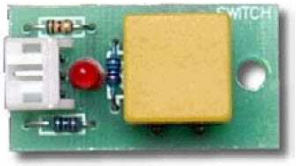
Модуль ЖКИ 16x2 с последовательным управлением Serial обеспечивает простейший путь отображения данных от микроконтроллера.
Для работы модуля требуется всего одна линия Ввода/Вывода микроконтроллера, напряжение питания +5 В и общая шина. Для обмена данными с модулем ЖКИ на скорости 2400 или 9600 бод (бит/сек) можно использовать простейшие команды с последовательного выхода данных модуля Arduino POP-168.

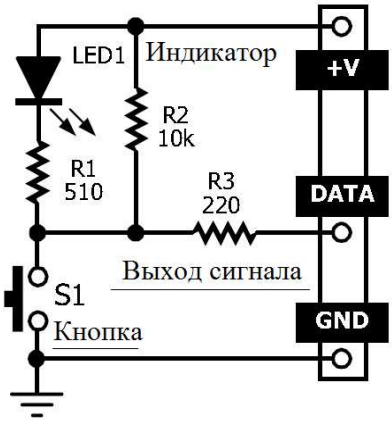
Особенности модуля ЖКИ 16x2 с последовательным управлением:

- Вход последовательных данных с Инвертированными/Не инвертированными логическими уровнями TTL
- Режим работы 1/8 или 1/16, выбираемый перемычками
- Совместимость с системой команд Scott Edwardshs LCD Serial Backpack™ , дополненной расширенными командами для упрощения управления ЖКИ
- Работа с одним напряжением питания +5 В постоянного тока
- SLCD16x2 обеспечивает регулировку яркости переменным резистором обозначенным на плате как **BRIGHTNESS**
- Разъем для подключения имеет 3 контакта : Напряжение питания +5В (+), Последовательный вход данных (**S**) и общая шина (**G**)

1.4 Особенности модулей датчиков

1.4.1 Модуль с кнопкой/Датчик нажатия (прикосновения)



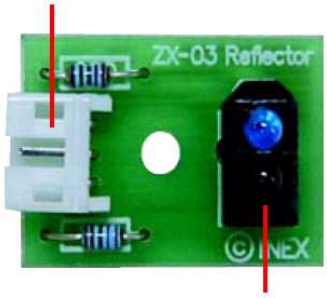


Вход кнопки используется для обнаружения столкновения, при котором на выходе появляется уровень логического “0”. В набор входят два комплекта датчиков, вместе с соединительными кабелями.

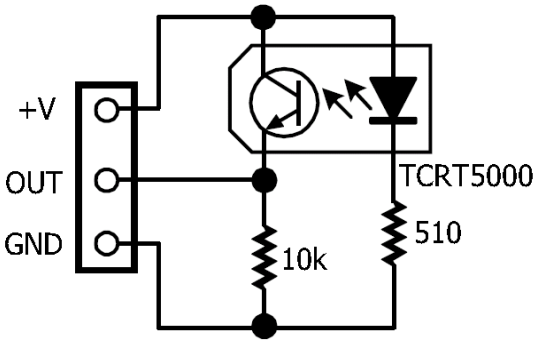
1.4.2 ZX-03 : Инфракрасный отражательный датчик

Сердцем этого датчика является инфракрасный (ИК) отражательный датчик объектов TCRT5000. Он сконструирован для обнаружения, с помощью ИК-излучения, объектов, расположенных в непосредственной близости от датчика. Позади прозрачного синего окошка расположен инфракрасный светодиод, позади черного окошка расположен инфракрасный фототранзистор. Когда инфракрасное излучение, испущенное светодиодом, отражается от поверхности и возвращается на черное окошко, оно попадает в базу инфракрасного транзистора, вызывая возникновение тока проводимости. Чем больше интенсивность инфракрасного излучения, попадающего в базу транзистора, тем больше интенсивность тока проводимости. При использовании в качестве аналогового датчика, ZX-03 может определять оттенки серого на бумаге и расстояние на коротких дистанциях, если освещенность в комнате остается постоянной.
Диапазон измеряемых расстояний от датчика до линии или пола лежит в пределах от 3 до 8 мм. При этом выходное напряжение будет меняться в диапазоне от 0.1 до 4.8 В, а значение оцифрованное 10-разрядным АЦП - от 20 до 1000. Таким образом, ZX-03 будет вполне подходящим для использования в качестве датчика, отслеживающего положение робота относительно линий.

Сигнальный разъем

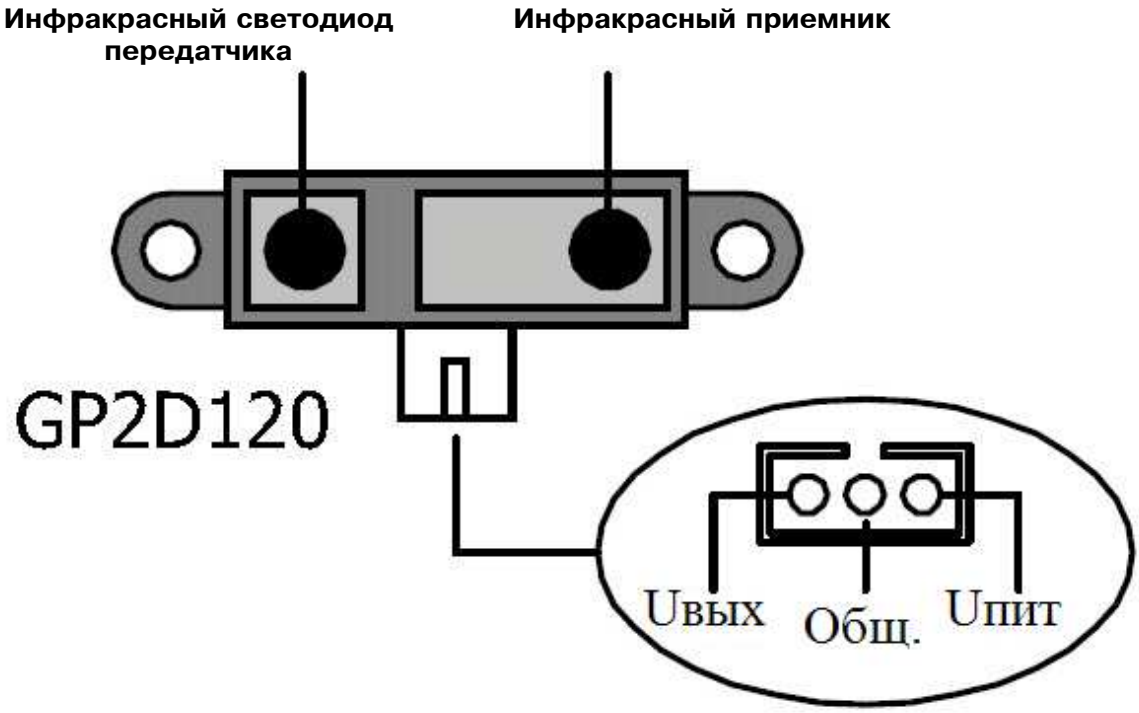


Инфракрасная отражательная оптопара



1.4.3 GP2D120 Инфракрасный датчик расстояния

Одним из специальных датчиков в робототехнике является GP2D120. Этот датчик выполняет функцию инфракрасного измерителя расстояния. Некоторые люди называют его ИК-дальномер, или ИК-локатор. Модуль GP2D120 добавляет нашему роботу функцию измерения расстояний и обнаружения препятствий, используя инфракрасное излучение. Робот MicroCamp может избегать препятствия без вхождения с ними в прямой физический контакт.



Особенности модуля GP2D120

- Для измерения расстояний используется инфракрасное излучение
- Диапазон измеряемых расстояний лежит от 4 до 30 см
- Напряжение питания от 4. 5 до 5 В, при потребляемом токе 33мА
- Диапазон выходного сигнала от 0.4 до 2.4 В, при напряжении питания +5В

Модуль инфракрасного локатора GP2D120 раъем с 3 выводами: вход напряжения питания (Vcc - Uпит), вывод Общей шины (GND - Общ.) и вывод выходного напряжения (Vout – Uвых). Чтобы прочитать выходное напряжение с модуля GP2D120, необходимо подождать окончания периода подтверждения, который колеблется от 32 до 52.9 мсек.

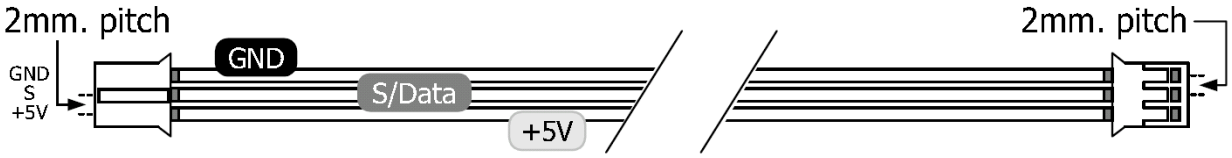
Выходное напряжение модуля GP2D120 при расстоянии до объекта 30 см и напряжении питания +5В находится в диапазоне от 0.25 до 0.55В, со средним значением 0.4В. При расстоянии до объекта равном 4 см, выходное напряжение будет равно 2.25В± 0.3В.

1.5 Информация о соединительных кабелях POP-BOT

Набор для сборки мобильного робота POP-BOT включает в свой состав несколько видов сигнальных кабелей для обмена данными между платой управления, модулями датчиков и компьютером. Набор состоит из кабелей JST3AA-8 для подключения модулей датчиков, кабеля UCON-4 преобразователя USB в RS-232 для обмена данными с компьютером.

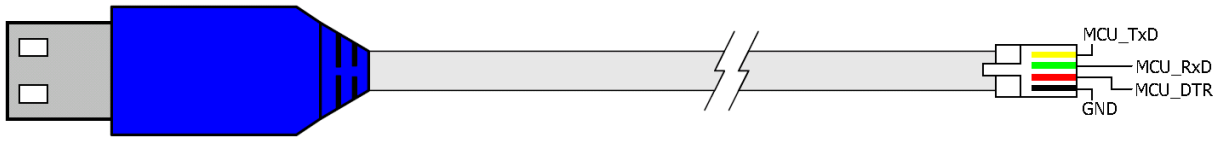
1.5.1 Кабель JST3AA-8

Это стандартный для конструкторов INEX-кабель, состоящий из трехпроводного плоского кабеля с шагом 2мм, длиной 8” (20см). На каждом конце кабеля установлены разъемы JST. Используется в наборе по изучению робототехники POP-BOT для подключения к плате микроконтроллера всех модулей датчиков. Назначение выводов кабеля показано на следующем рисунке.



1.5.2 Кабель UCON-4 преобразователя интерфейсов USB в Последовательный порт (RS-232)

Используется для обмена данными между USB-портом компьютера и платой управления RBX-168. На одном из концов кабеля установлен Модульный разъем RJ-11 6P4C (с 6 выводами к 4 из которых подведены сигналы). Его длина составляет приблизительно 1.5 метра. Назначение контактов кабеля показано на рисунке ниже.



Для нормальной работы кабеля требуется напряжение питания +5В от USB-порта, и он поддерживает работу по спецификации USB 1.0/2.0. Скорость обмена данных по кабелю может устанавливаться пользователем вплоть до 115 200 бит в секунду.

Перед первым подключением кабеля к компьютеру необходимо установить соответствующие драйвера с CD, входящего в комплект набора.

1.6 Описание механических частей набора

1.6.1 Набор круглых колес и шин

Состоит из 2 пар колес, пригодных для использования совместно с модулями электродвигателей постоянного тока с редукторами BO-2, и резиновых шин, одеваемых на колеса. На оси редуктора колесо крепится с помощью 2мм саморезов

1.6.2 Пластиковое основание с отверстиями

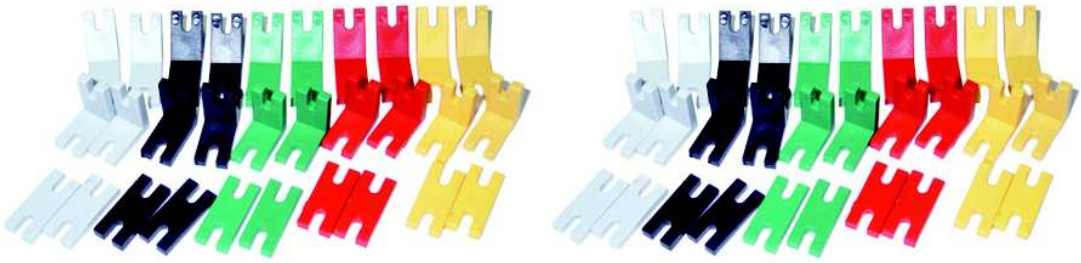
В каждый набор входят универсальные пластиковые основания двух типоразмеров; 80x60мм и 80x80мм. В каждом основании имеется сетка отверстий с шагом 5мм под 3мм винты.

1.6.3 Круглое основание

Основание отлито из высококачественного пластика ABS повышенной прочности. Диаметр основания составляет .....мм. С двух сторон основания свободно закреплены шаровые колеса. Основание имеет большое количество 3мм отверстий для крепления платы управления, датчиков и большого количества механических частей.

1.6.4 Пластиковый крепеж

В состав набора входят 60 крепежных элементов различного цвета, сделанных из PVC-пластика. Они могут быть соединены друг с другом или с другими деталями набора, с использованием винтов и гаек M3. Имеется 4 типа крепежных элементов: согнутые под прямым углом, согнутые под тупым углом, прямые планки без отверстий и прямые планки с отверстиями.



1.6.5 Крепежные планки с отверстиями

Крепежные планки с отверстиями сделаны из пластика. Каждая планка имеет отверстия диаметром 3мм, с шагом 5мм. Каждую планку можно присоединять к другой для увеличения общей длины. Имеется по 4 планки 3 типоразмеров: с 3, 5 и 12 отверстиями. Всего 12 штук.

1.6.6 Коробчатое основание

Представляет собой пластиковую коробку для крепления платы управления RBX-168. В коробке имеются отверстия диаметром 3мм для крепления ее на любом подходящем основании.

1.6.7 Металлические уголки

Уголки представляют собой изогнутые под прямым углом металлические полоски, шириной 7.5 мм. Каждый уголок имеет 3 мм отверстия для крепления к другим деталям набора с помощью 3 мм винтов и гаек. В набор входит по 4 металлических уголка с количеством отверстий 1x2, 2x2 и 2x5. Всего 12 штук.

1.6.8 Винты и гайки

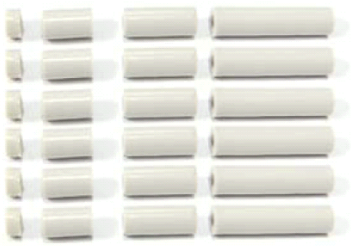
В набор входят: 2 самореза на 2 мм, 4 винта M3 x 8 мм, 30 винтов M3 x 10 мм, 4 винта M3 x 15 мм, 4 винта M3 x 40 мм, 10 винтов M3 x 8 с плоской потайной головкой и 30 гаек M3.

1.6.9 Металлические стойки

В наборе имеются металлические стойки для крепления оснований и модулей датчиков. Они сделаны из никелированного металла, для предотвращения коррозии и продления срока службы. В набор входят 6 шестигранных металлических стоек длиной 33 мм. Каждая стойка на обоих концах имеет отверстия с резьбой M3 для крепления винтами.

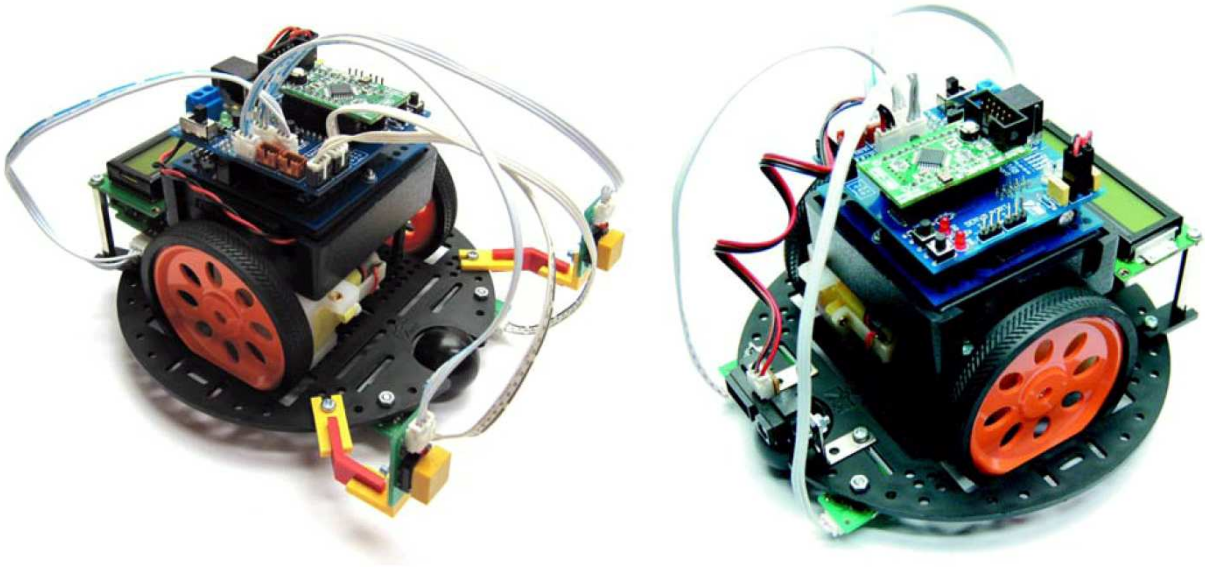
1.6.10 Пластиковые втулки

В наборе имеется комплект механических частей для закрепления основания и модулей датчиков. В этот комплект входят 4 типа пластиковых втулок (длиной 3 мм, 10 мм, 15 мм и 25 мм). Всего 24 штуки.



2: Сборка мобильного робота POP-BOT

2.1 Особенности набора POP-BOT



- Движется при помощи электродвигателей с редукторами, на который установлены колеса с резиновыми шинами
- Управляется модулем контроллера POP-168, совместимого с Arduino, построенного на базе микроконтроллера ATmega168
- Программируется через последовательный порт и поддерживает использование преобразователя интерфейса USB в Последовательный порт
- Поддерживает работу с большим количеством разнообразных датчиков, таких как: инфракрасный отражательный датчик для отслеживания линий, датчик касания для предотвращения столкновения с препятствиями, Инфракрасный локатор или датчик расстояния для бесконтактного предотвращения столкновений с объектами
- Поддерживает проводное дистанционное управление, включая управление от PlayStation
- Поддерживает работу с модулями беспроводного обмена данными, такими как Xbee, XBee-Pro и Bluetooth
- В наборе имеется ЖКИ-модуль 16 x 2 с последовательным интерфейсом для контроля и отображения рабочего состояния робота
- 2 порта для подключения сервомоторов. Поддерживает малогабаритные RC-сервомоторы с рабочим напряжением от 4.8 до 6В
- Для работы требуется 4 батареи или аккумулятора типоразмера AA. Рекомендуется использовать батареи типа Alcaline, Evolta и Ni-MH аккумуляторы

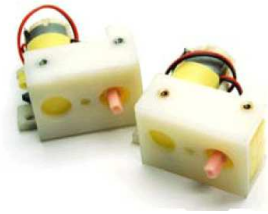
2.2 Список комплектующих

Круглое основание x 1

Плата RBX-168
с POP-168 x 1

Коробчатое основание x 1

Колеса с шинами x 2



Электродвигатель
с редуктором x 2

33-мм металлические стойки
x 4

ЖКИ с последовательным
интерфейсом x 1

Инфракрасные
отражательные
датчики
x 2

Инфракрасный датчик
расстояния x 1

Пластиковые крепежные
элементы

Металлические уголки
2 x 2 x 2

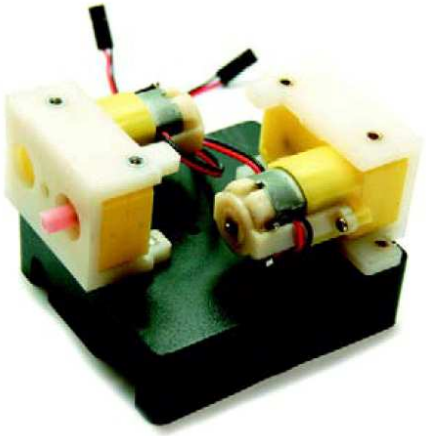
2-мм саморезы x 2

Набор пластиковых втулок

Набор винтов и гаек

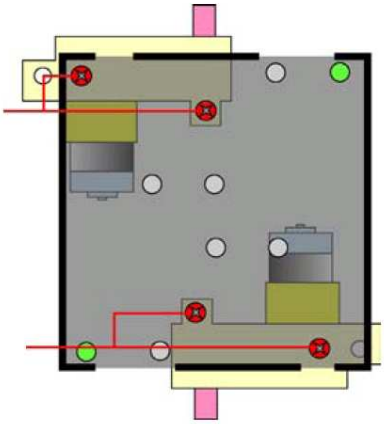
2.3 Последовательность сборки

(1) Присоедините оба электромотора с редуктором к коробчатому основанию с помощью винтов M3 x 8 мм с плоской потайной головкой.

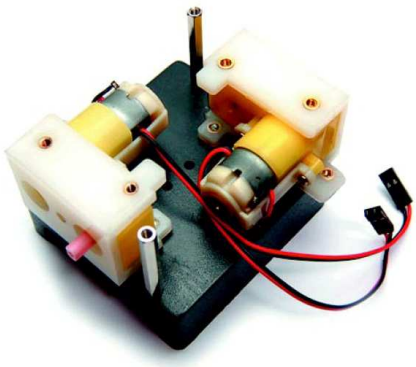


Винты M3 x 8 мм с потайной головкой

Винты M3 x 8 мм с потайной головкой

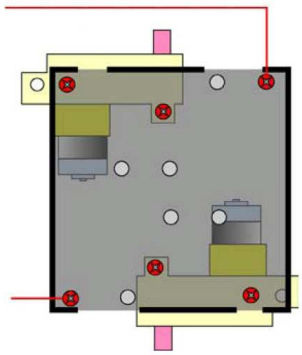


(2) Присоедините к коробчатому основанию 2 металлических стойки длиной 33 мм с помощью винтов M3 x 8 мм с плоской потайной головкой, как показано на следующих рисунках.

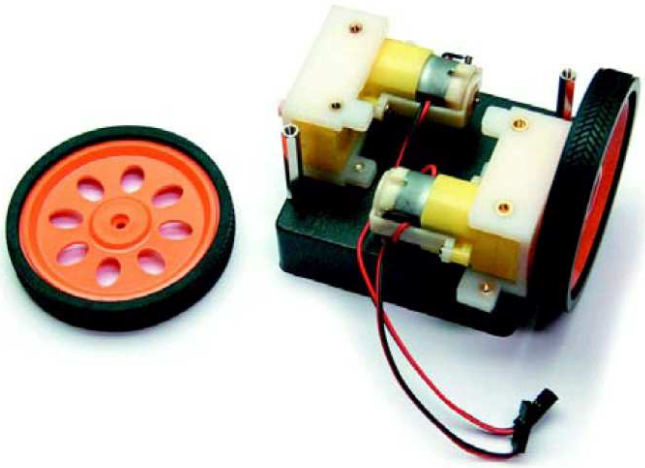


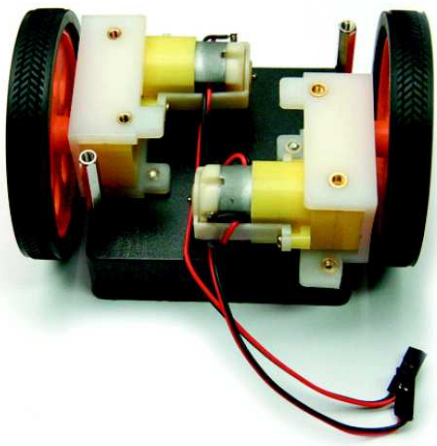
Винты M3 x 8 мм с потайной головкой

Винты M3 x 8 мм с потайной головкой



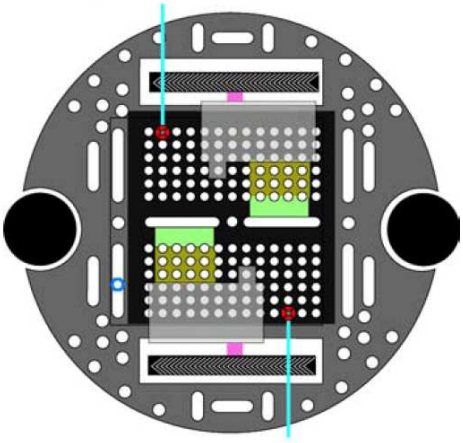
(3) Установите на валы редукторов электродвигателей постоянного тока круглые колеса с шинами и закрепите 2-мм саморезами.

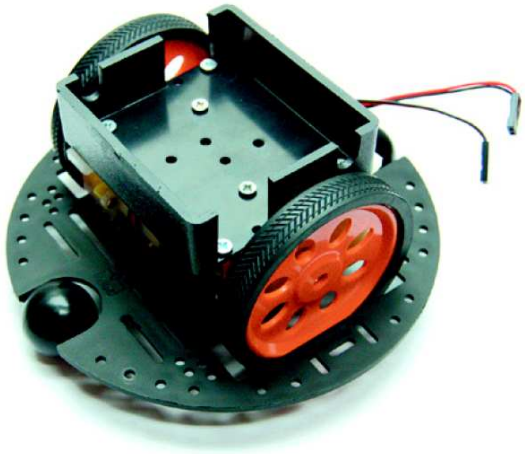




(4) Присоедините коробчатое основание с блоком электродвигателей, собранное на шаге (3), к круглому основанию, используя винты M3 x 6 мм, как показано на следующих рисунках. Старайтесь производить закрепление так, чтобы колеса располагались по центру прорезей под них в круглом основании.

Винт M3 x 8 мм с потайной головкой

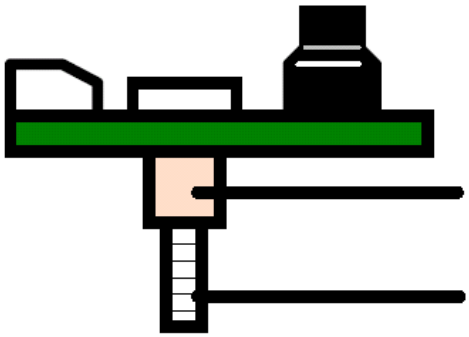




Винт M3 x 8 мм с потайной головкой

(5) Пропустите через отверстие в плате отражательного датчика винт M3 x 10 мм, и наденьте на него пластиковую втулку длиной 3 мм. Соберите 2 таких структуры.

Инфракрасный отражательный датчик



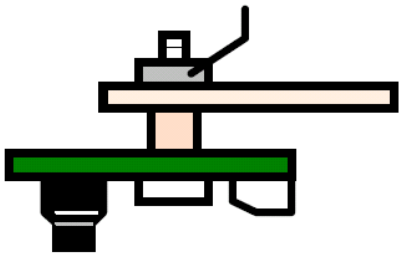
Пластиковая втулка длиной 3 мм

Винт M3 x 10 мм

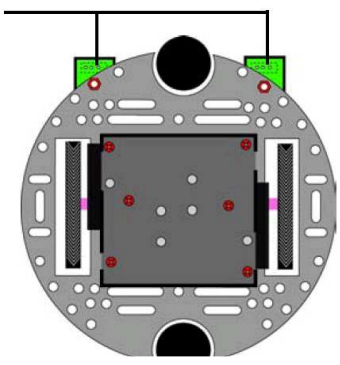
(6) Закрепите обе структуры инфракрасных отражательных датчиков, собранных на шаге (5), в передней части основания робота с двух сторон, используя гайки M3, как показано на следующем рисунке.

Инфракрасные отражательные датчики

Гайка M3

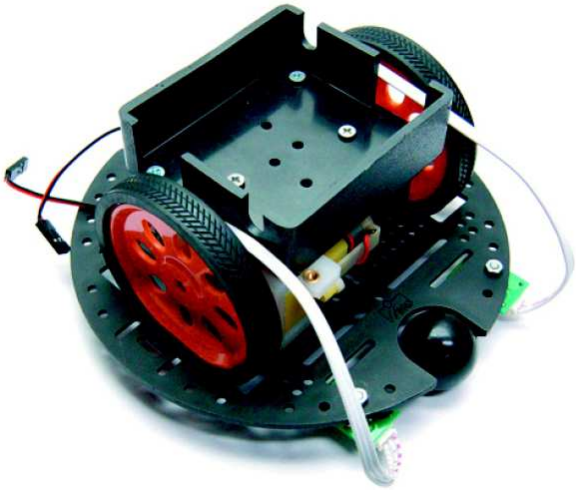


Основание

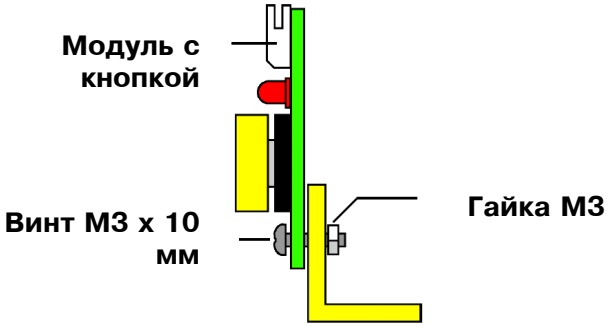


Инфракрасный отражательный датчик

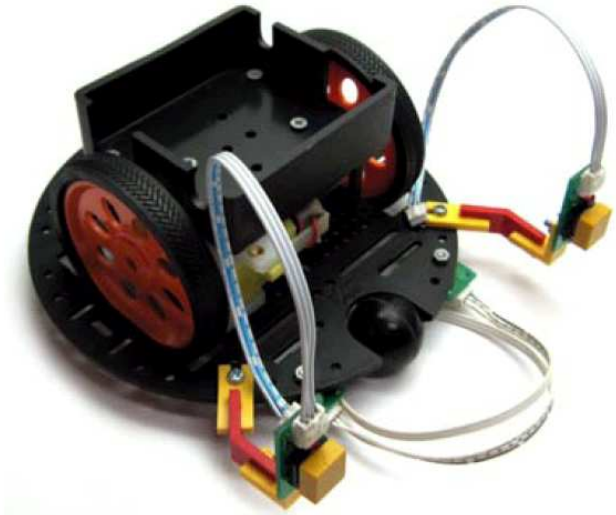
(7) Теперь корпус POP-BOT с инфракрасными отражательными датчиками готов.



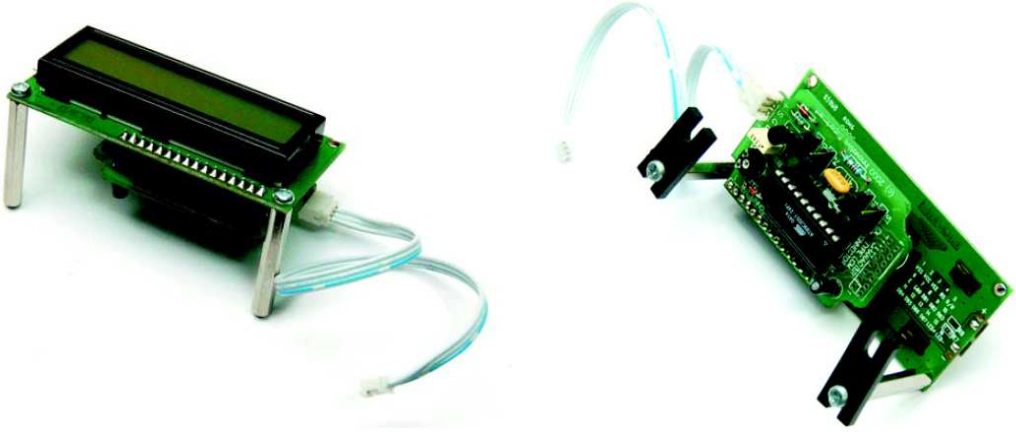
(8) Присоедините прямоугольный крепежный элемент к Модулю с кнопкой, используя винт M3 x 10 мм и гайку M3. Затем вставьте тупоугольный крепежный элемент и прямоугольный крепежный элемент датчика с кнопкой прорезями друг в друга. Сделайте 2 таких структуры.



(9) Закрепите 2 прямых крепежных элемента в передней части круглого основания робота, используя винты M3 x 6 мм и гайки M3. Далее, соедините структуры с Модулями с кнопкой, собранные на шаге (9), с прямыми крежными элементами, вставляя их в прорези друг друга.



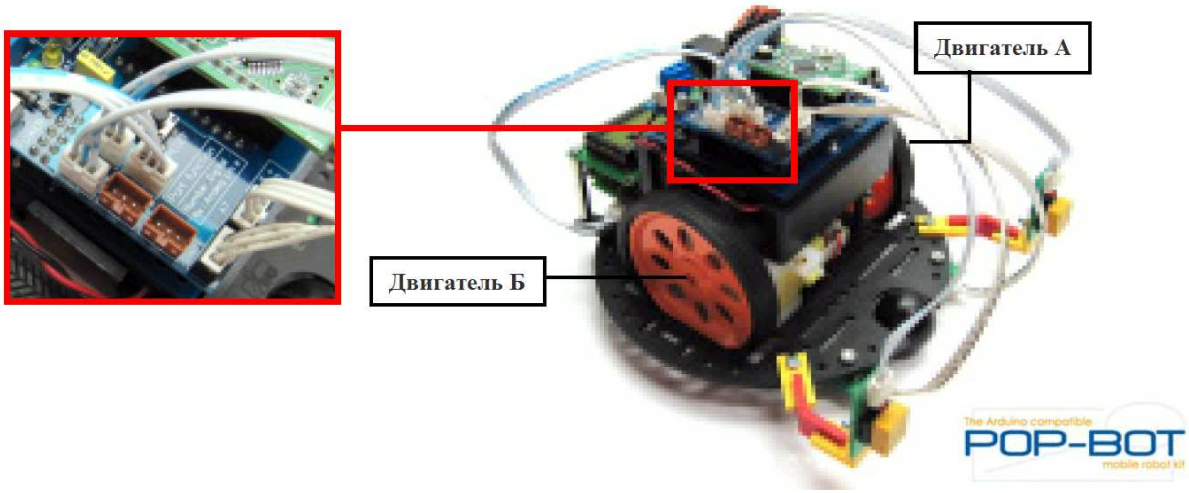
(10) Присоедините 2 металлических стойки длиной 33 мм к модулю SLCD16 x 2, используя винты M3 x 6 мм. Далее к свободным концам стоек прикрепите прямые крепежные элементы, используя винты M3 x 10 мм.



(11) Закрепите структуру SLCD16 x 2, собранную на шаге (10), в задней части круглого основания робота с помощью винтов M3 x 10 мм и гаек M3, как показано на следующем рисунке.

(12) Установите плату RBX-168 в коробчатое основание. Подключите все кабели. Начните с подключения кабеля левого электродвигателя к выходу MOTOR A, кабеля правого электродвигателя к выходу MOTOR B. Подключите кабель левого инфракрасного отражательного датчика к выводу **A7**, правого инфракрасного отражательного датчика к выводу **A6**. Далее подключите кабель левого модуля с кнопкой к выводу **15/A1**, кабель правого модуля с кнопкой к выводу **17/A3** и подключите SLCD16 x 2 к выводу **16/A2.**

Итак, только теперь POP-BOT готов к программированию.



3 : Введение в среду разработки IDE Arduino

Среда разработки Arduino является платформой макетирования электронных устройств с открытыми исходными кодами, основанной на гибком, простом в использовании аппаратном и программном обеспечении. Она предназначена для художников, проектировщиков, людей, увлечённых своим хобби, и всех интересующихся созданием интерактивных объектов или сред. [1]

В данной главе описано краткое введение в Arduino, начиная с процесса установки, объяснения функций компонент Arduino IDE и детального описания панели Меню.

3.1 Установка программного обеспечения

(1) Вставьте POP-BOT CD-ROM в CD-драйвер вашего компьютера.
(2) Войдите в каталог **Software _ Arduino**. Найдите там файл **ArduinoSetup.exe**, запустите его на исполнение двойным щелчком мыши. После этого запустится процедура установки.

POP-BOT CD-ROM содержит программное обеспечение Arduino версии V15, все исходные коды примеров для выполнения Заданий для изучения POPBOT и необходимые файлы библиотек. Последнюю версию Arduino можно загрузить с www.arduino.cc. Однако, после установки новой версии Arduino IDE, необходимо будет убедиться в правильности указания к библиотеке POP-BOT.

---

[1] Параграф введения взят с веб-сайта Arduino (www.arduino.cc)

3.2 Среда разработки Arduino

После запуска IDE Arduino, появится главное окно приложения, похожее по внешнему виду на изображенное на рисунке 3-1. В состав среды разработки Arduino входят следующие компоненты.

- **Меню (Menu):** Выбор рабочих команд
- **Панель инструментов (Toolbar):** Содержит кнопки вызова большинства команд
- **Закладки (Tabs)**: Позволяют одновременно работать более чем с одним файлом скетча (каждый из которых доступен на собственной закладке)
- **Текстовый редактор (Text editor)**: Область текстового редактора для создания скетчей
- **Область сообщений (Message area):** отображает состояния работы программы, такие как результат компиляции
- **Текстовая область (Text area):** Пространство, в котором отображается информация компилятора и окно Терминала последовательного обмена данными (если доступно)

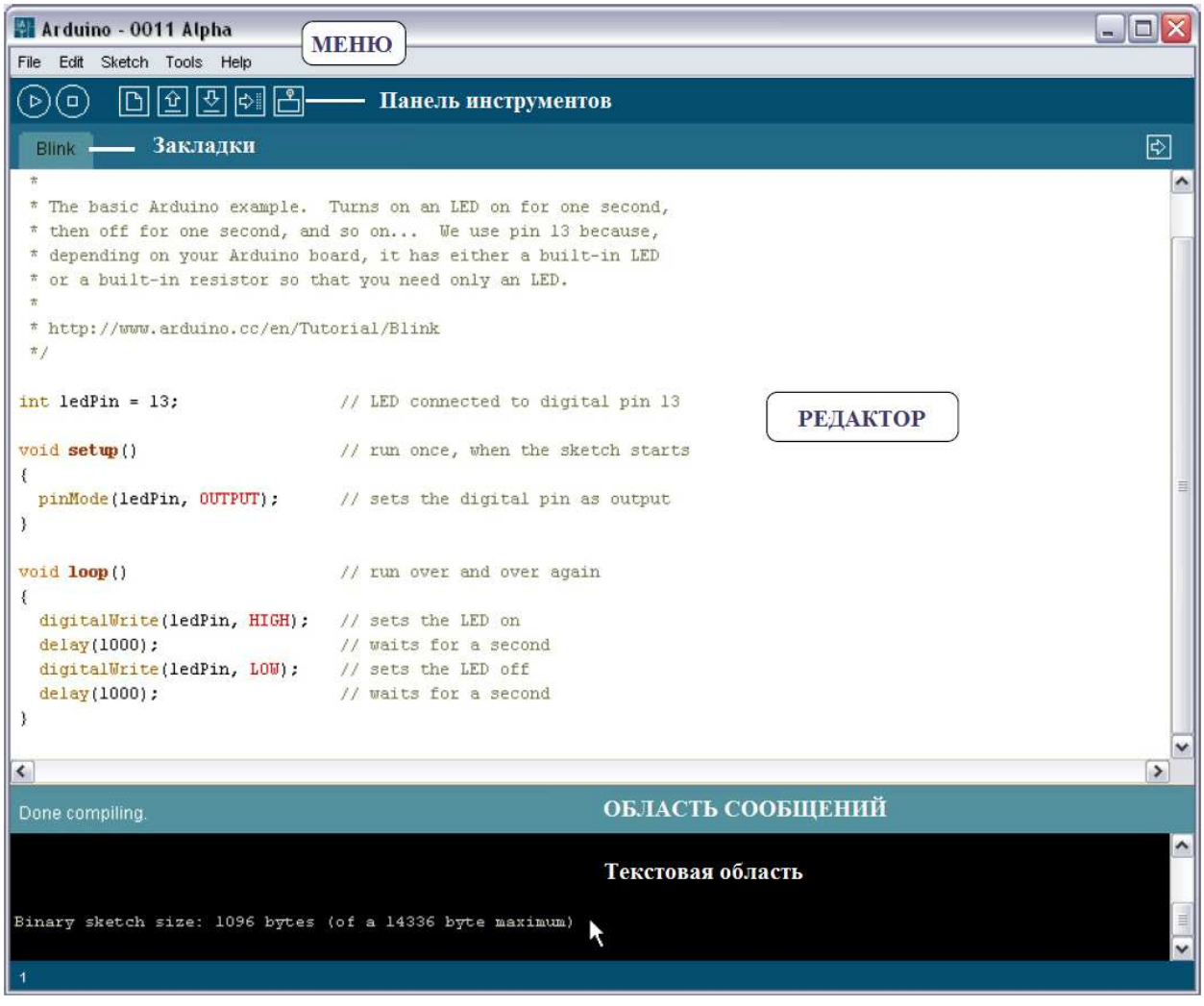


Рисунок 3-1. Интегрированная среда разработки Arduino

3.3 Строка меню

3.3.1 Файл (File)

Исходные коды программы в среде Arduino называются **Скетч (Sketch**). Этот пункт меню содержит множество следующих команд: открыть (open), сохранить (save) и закрыть (close) скетч:

- **Новый (New)**: Создает новый скетч, в наименовании которого присутствует текущая дата в формате "sketch_ГГММДДа"
- **Подшивка скетчей (Sketchbook)**
  - **Открыть (Open)**: Открывает существующий скетч
  - **Пример (Example)**: Открывает скетч примера
- **Сохранить (Save)**: Сохраняет текущий скетч
- **Сохранить как (Save as)**: Сохраняет текущий скетч с новым именем
- **Загрузить на плату Ввода/Вывода (Upload to I/O board)**: Загружает скомпилированный код текущего скетча на плату Ввода/Вывода Arduino (модуль POP-168). Перед загрузкой убедитесь, что Ваш скетч был сохранен и проверен на наличие ошибок
- **Предпочтения (Preference)**: Некоторые индивидуальные настройки среды Arduino
- **Выйти (Quit)**: Выход из Arduino IDE

3.3.2 Редактировать (Edit)

Пункт меню Редактировать (Edit) обеспечивает вызов набора команд для редактирования файлов Arduino.

- **Отменить (Undo)**: Отменяет последнюю команду или стирает последний введенный символ
- **Отменить последнюю операцию Undo (Redo)**: Отменяет результаты выполнения последней команды Undo. Этот пункт доступен, только если уже выполнялись операции отмены предыдущих действий (Undo)
- **Вырезать (Cut)**: Удаляет и копирует выделенный текст в буфер обмена
- **Копировать (Copy)**: Копирует выделенный текст в буфер обмена
- **Вставить (Paste)**: Вставляет содержимое буфера обмена в место расположения курсора и заменяет весь текущий выделенный текст
- **Выделить все (Select All)**: Выделяет весь текст в текущем файле, открытом в текстовом редакторе
- **Найти (Find)**: Находит местоположение текстовой строки в открытом в редакторе файле и предлагает возможность замены его другим текстом
- **Найти далее (Find Next)**: Находит следующее местоположение текстовой строки в открытом в редакторе файле

3.3.3 Скетч (Sketch)

Этот пункт меню предназначен для вызова ряда команд компиляции кода и управления библиотекой.

- **Проверить/ Компилировать (Verify/Compile)**: Проверить и компилировать код, если не были обнаружены ошибки
- **Остановить (Stop)**: Остановить выполнение текущей операции
- **Добавить файл (Add file)**: Открыть файловый навигатор. Выбрать файл с исходным кодом и добавить его в каталог "data" скетчей
- **Импортировать библиотеку (Import Library)**: Импортирует дополнительную библиотеку
- **Показать каталог скетча (Show Sketch folder)**: Открывает каталог с текущим скетчем

3.3.4 Инструментарий (Tools)

В этот пункт меню входят команды вызова инструментов для разработки скетч-файлов Arduino и настройки параметров аппаратного обеспечения Arduino.

- **Авто-формат (Auto Format)**: Делается попытка форматирования кода в более удобочитаемом для человека виде. Перед выполнением команды Авто-формат (Auto Format) вызывается команда Сделать код красивым (Beautify).
- **Архивировать скетч (Archive Sketch)**: Сжатие текущего скетча в zip-файл
- **Каталог экспорта (Export Folder)**: Открывает каталог, в котором содержится текущий скетч
- **Плата (Board)**: Выбор конкретной аппаратной платформы, совместимой с Arduino. Для POP-BOT следует выбрать ***POP-168*** или ***Arduino Mini***
- **Последовательный порт (Serial Port)**: Позволяет определить, какой из последовательных портов будет использоваться по умолчанию для загрузки кода в плату Ввода/Вывода Arduino или для анализа данных, поступающих от платы. Данные, поступающие от платы Ввода/Вывода Arduino I/O, распечатываются в символьном формате в текстовой области консоли.

3.3.5 Помощь (Help)

Это меню содержит многочисленную информацию в формате HTML для поддержки пользователей Arduino.

- **Начало работы (Getting Start)**: Открывает раздел справочной системы Как начать работать с Arduino (How to start Arduino).
- **Диагностика (Troubleshooting)**: Предлагаются возможные варианты решения проблем, возникающих в Arduino.
- **Среда разработки (Environment)**: Дается описание среды разработки Arduino
- **Ссылки (Reference)**: В интернет обозревателе, используемом по умолчанию, открываются полезные ссылки, включая ссылки на описание языка программирования, среды разработки, библиотек и сравнение разных языков программирования.
- **Часто задаваемые вопросы (Frequently Asked Question)**: Содержит наиболее часто задаваемые вопросы и ответы на них об Arduino.
- **Посетить www.arduino.cc (Visit www.arduino.cc)**: По умолчанию открывает в интернет-обозревателе домашнюю страницу Arduino.
- **Об Arduino (About Arduino)**: Открывает краткую информационную панель о программном обеспечении.

3.4 Панель инструментов

 **Verify/Compile**	Проверка кода на наличие ошибок.
 **Stop**	Останавливает монитор обмена данными по последовательному порту или делает недоступными другие кнопки.
 **New**	Создает новый скетч.
 **Open**	Вызывает меню выбора скетча из подшивки скетчей.
 **Save**	Сохраняет скетч.
 **Upload to I/O Board**	Загружает код в плату Ввода/Вывода Arduino (модуль POP-168). Перед загрузкой скетча убедитесь, что он был сохранен, проверен на наличие ошибок и откомпилирован.
 **Serial Monitor**	Показывает последовательные данные, поступающие от платы Arduino (через интерфейс USB или последовательный порт). Чтобы послать данные плате, введите текст и нажмите кнопку “послать” ("send") или нажмите клавишу Ввод (Enter) на клавиатуре. Выберите в поле со списком значение скорости обмена, соответствующее скорости обмена, заданной в параметре функции **Serial.begin** в скетче. Следует отметить, что в операционных системах Mac или Linux, плата Arduino будет сбрасываться (возвращать скетч к началу его выполнения) всякий раз, когда Вы будете подключать ее к монитору последовательного обмена данными.

3.5 Замечания относительно ссылок по программированию Arduino

В этом сборнике практических работ не будет описываться программирование Arduino. Вы можете прочитать и изучить синтаксис Arduino и получить информацию о программировании из справочной системы, пункт меню Помощь (Help), или изучить на веб-сайте Arduino www.arduino.cc.

Кроме того, вы можете изучить документ **Arduino Programming Notebook**, объемом 40 страниц. А также загрузить rom Arduino со страницы *Playground* веб-сайта.

4 : Разработка программ для POP-BOT с использованием Arduino

Разработку программ для POP-BOT в общих чертах можно представить как диаграмму, изображенную на рисунке 4-1.

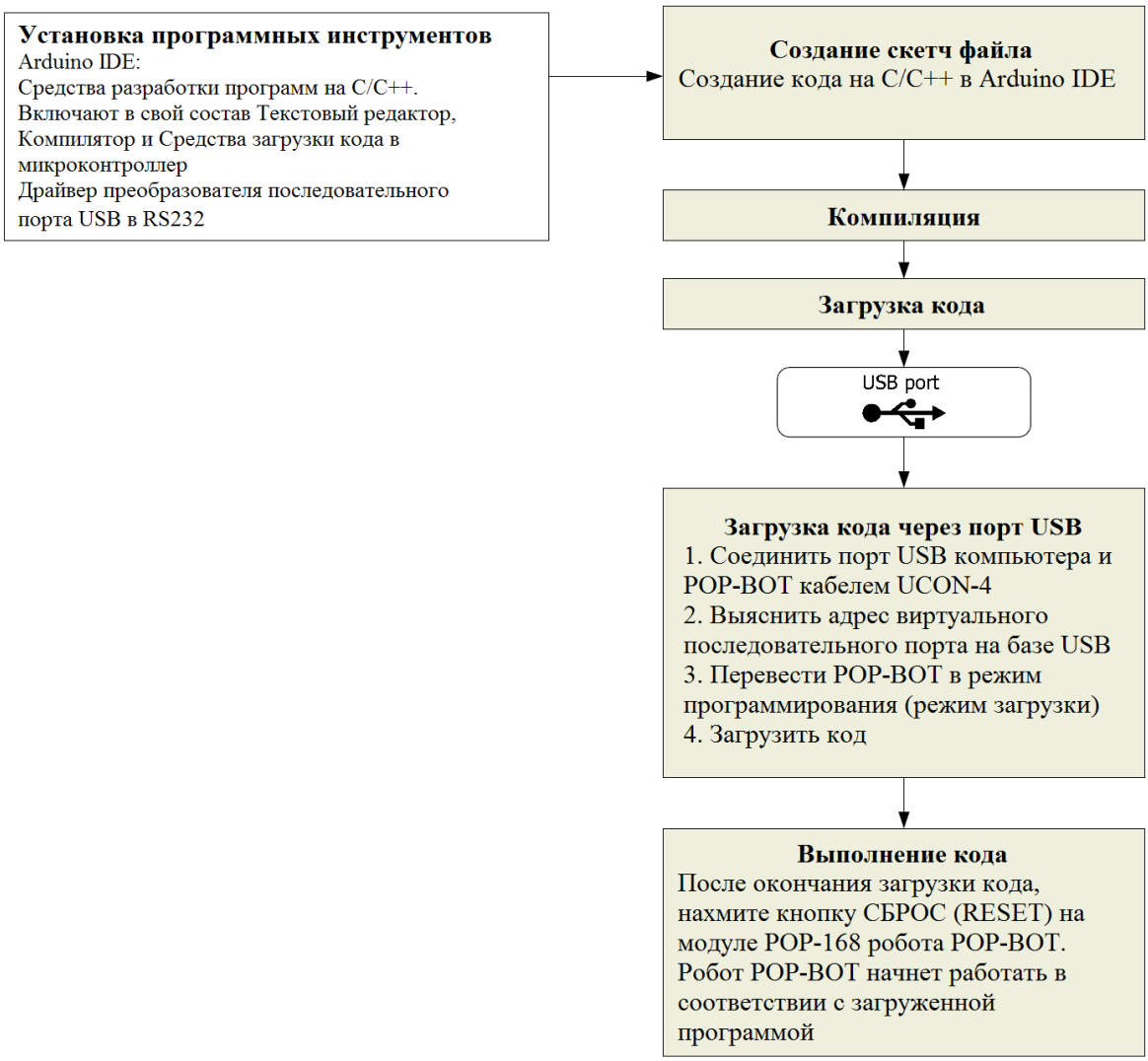


Рисунок 4-1. Диаграмма разработки программ для POP-BOT с использованием Arduino IDE

4.1 Подготовка кабеля UCON-4, преобразователя USB в последовательный порт RS-232

Для загрузки кода в POP-BOT требуется компьютерный интерфейс и среда Arduino. Обычно используется последовательный порт RS-232 или COM-порт. Для современных компьютеров основным интерфейсом обмена данными является порт USB. Поэтому необходимо использовать преобразователь интерфейса USB в последовательный порт RS-232. В наборе POP-BOT для этих целей подготовлен кабель UCON-4.

Перед первым использованием кабеля UCON-4, необходимо установить соответствующие драйвера и проверить получившуюся после установки драйверов конфигурацию оборудования.

4.1.1 Установка драйвера

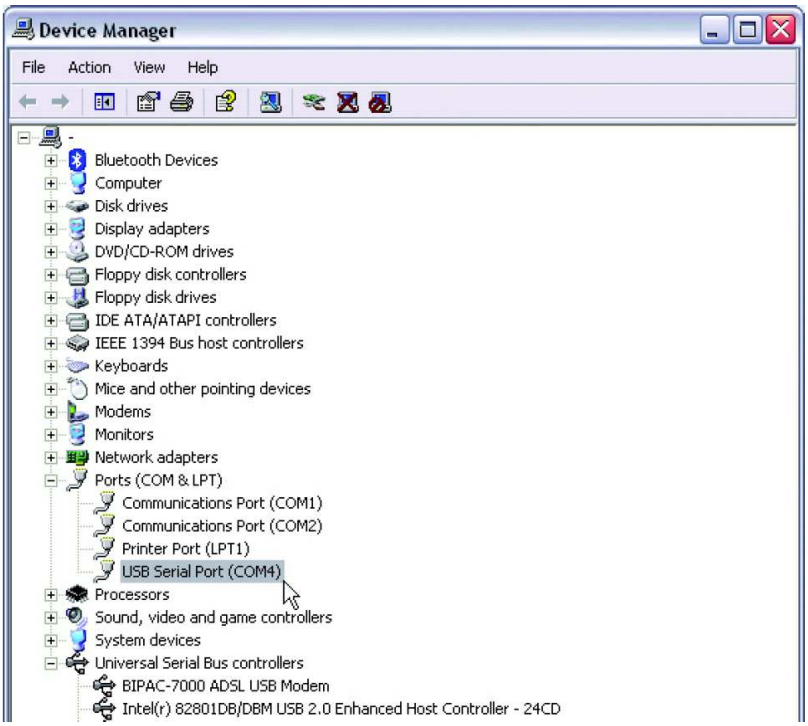
Для начала установки драйверов необходимо сделать двойной клик по файлу USBDriverInstallerV2.0.0.exe с CD-ROM, включенного в комплект робота POP-BOT. После этого появится диалоговое окно установки драйверов, похожее на изображенное ниже.

4.1.2 Проверка адреса последовательного порта USB

(1) Подключите кабель USB к порту USB и плате управления POP-BOT. Подождите некоторое время.

(2) Проверьте адрес Виртуального COM порта (Virtual COM) или последовательного порта USB (USB Serial port) Последовательно нажимая мышкой **Пуск → Панель управления → Система → Оборудование → Диспетчер устройств** (**Start → Control Panel → System → Hardware → Device Manager**)

(3) Просмотрите список последовательных портов USB и запомните адрес COM-порта для работы с кабелем UCON-4. Обычно создается последовательный порт COM3 или с более высоким номером. В рассматриваемом примере последовательный порт USB имеет адрес **COM4**



4.1.3 Замечания по использованию кабеля UCON-4 совместно с Arduino

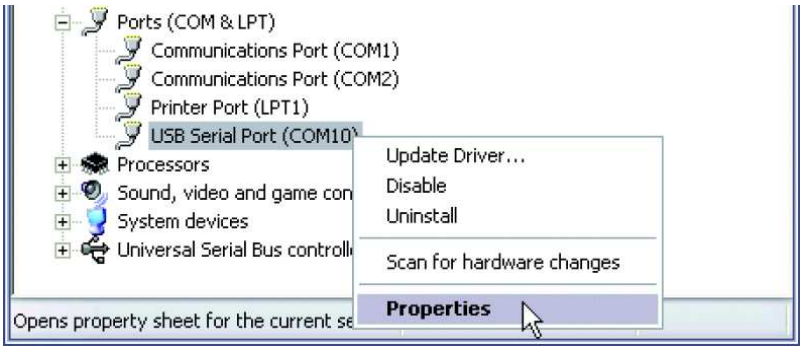
Обычно программное обеспечение Arduino может обслуживать COM-порты с адресами не выше COM9. Поэтому, пользователь должен убедиться, что адрес последовательного порта USB не более, чем COM9. Если он оказался выше, проделайте, пожалуйста, следующую процедуру.

(1) Присоедините кабель UCON-4 к компьютерному USB порту.

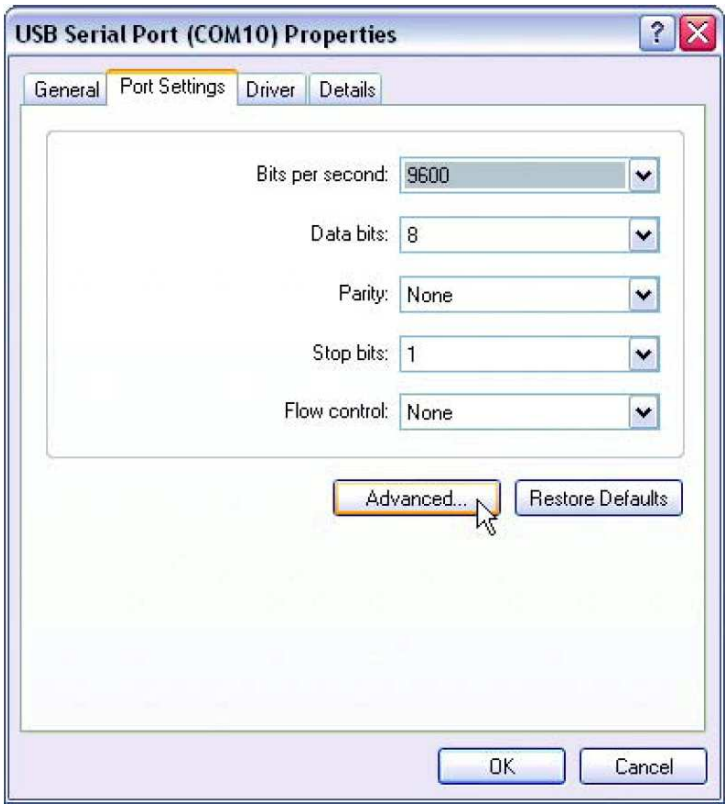
(2) Проверьте адрес COM-порта, нажимая мышкой последовательно **Пуск→Панель управления→Система** (**Start → Control Panel →System)**

(3) Выберите закладку **Оборудование** (**Hardware)** и нажмите кнопку **Диспетчер устройств** (**Device Manager**).

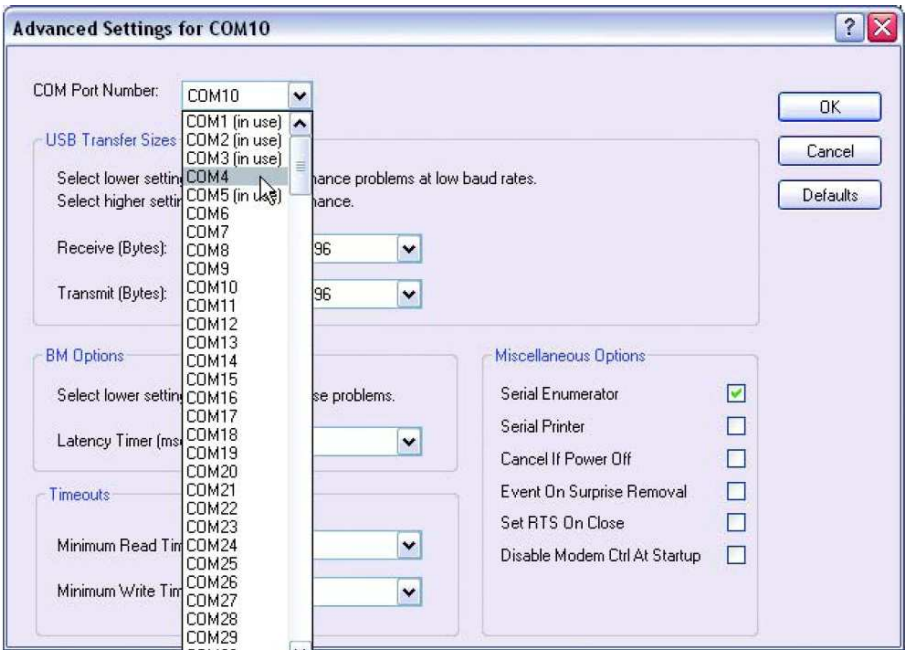
(4) Просмотрите список оборудования. В списке портов должна присутствовать строка **Последовательный порт USB (COMx)** (**USB Serial port (COMx))**. Если адрес COM-порта оказался выше, чем COM9 (в данном примере это COM10), пожалуйста, кликните на строке с названием порта мышкой и в появившемся списке выберите строку **Свойства (Properties)**



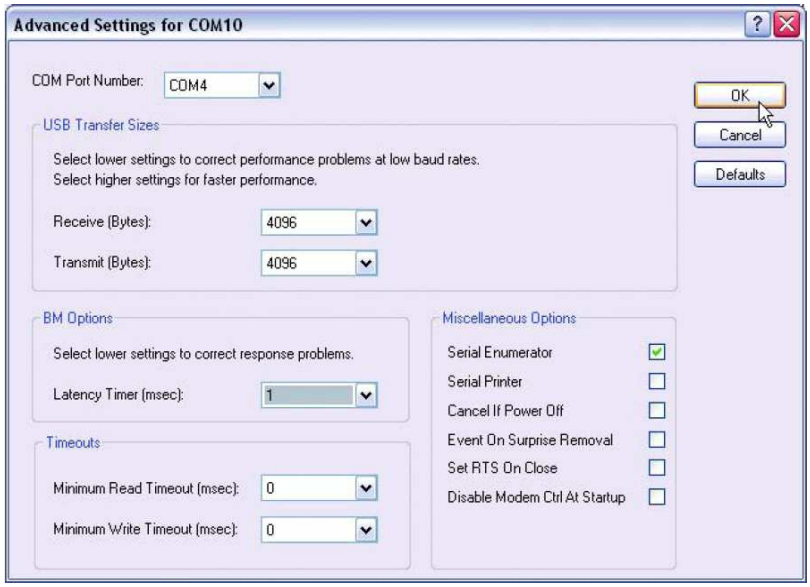
(5) Появится окно **Свойства последовательного порта USB (COM10)** (**USB Serial Port (COM10) Properties)**. Выберите закладку **Настройка порта** (**Port Setting)** и установите значения настроек как показано на рисунке ниже, после этого нажмите кнопку **Дополнительно Advance**



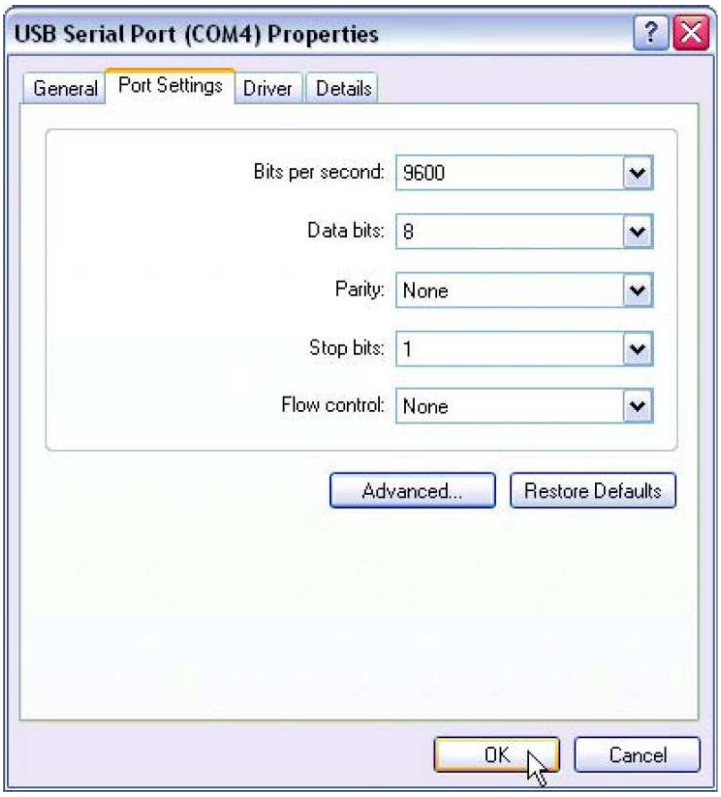
(6) Появится окно с дополнительными настройками для COM10. Кликните на поле со списком **Номер COM-порта** (**COM Port Number),** чтобы изменить номер порта на ***COM4*** или любой другой свободный в диапазоне от ***COM1*** до ***COM9***.



(7) Установите значения других настроек как показано на рисунке ниже. Особое внимание следует обратить на то, чтобы для настройки **Таймер времени ожидания (мсек)** (**Latency Timer (msec)) было установлено значение, равное 1,** и был отмечен флажок **Последовательная нумерация** (**Serial Enumerator).** Нажмите кнопку **OK**.



(8) Вернитесь к свойствам Последовательного порта USB (USB Serial Port Properties). *Теперь номер COM-порта в заголовке окна должен измениться на* ***COM4****. Нажмите кнопку* **OK**.

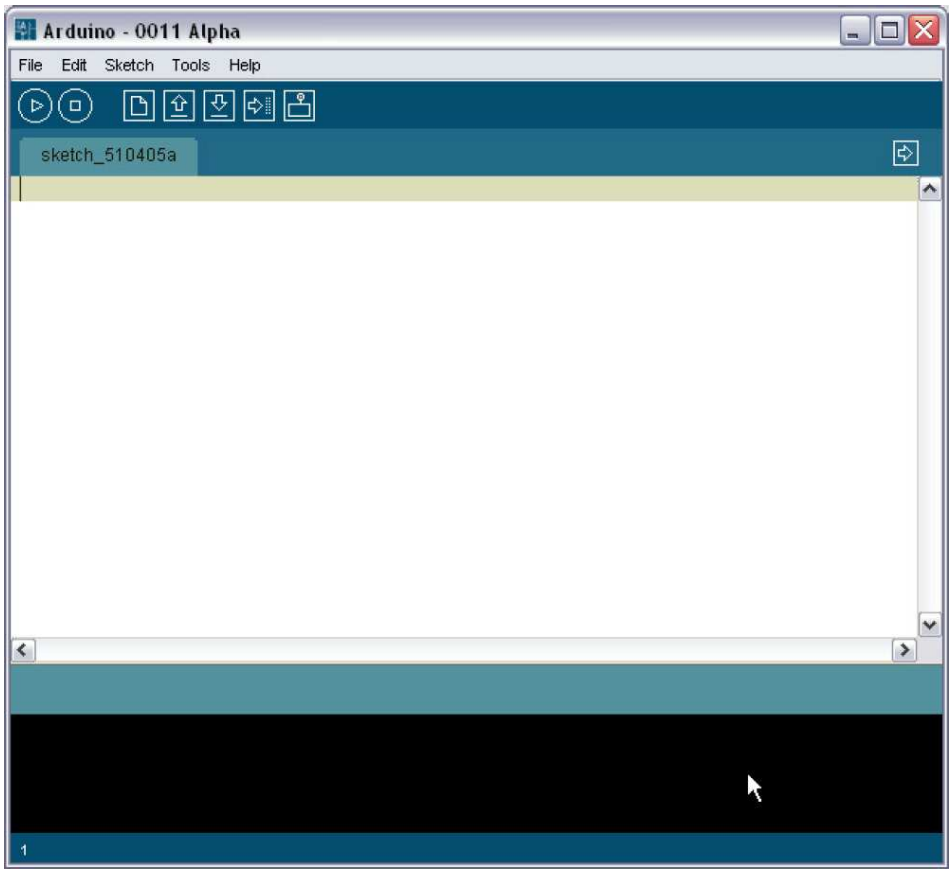


(9) Отключите кабель UCON-4 от USB-порта и подключите его снова. Еще раз проверьте адрес последовательного порта USB. Новый адрес должен быть **COM4**. Теперь кабель UCON-4 готов к использованию со средой разработки Arduino IDE.

4.2 Начало работы POP-BOT с программным обеспечением Arduino

Запустите Arduino IDE, последовательно нажав **Пуск → Все программы→ POP-168 Software Package → Arduino (Start → All Programs → POP-168 Software Package → Arduino)**

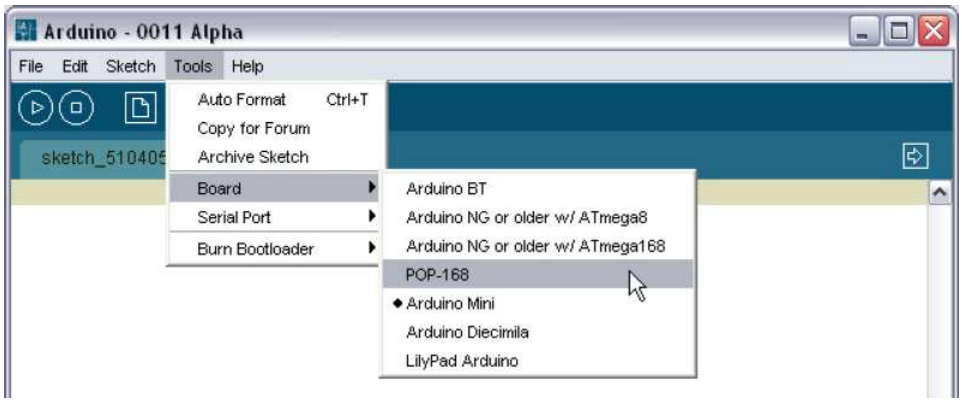
При первом запуске главное окно Arduino выглядит как показано на рисунке ниже.



4.2.1 Конфигурация аппаратной части POP-168

4.2.1.1 Выбор типа микроконтроллера

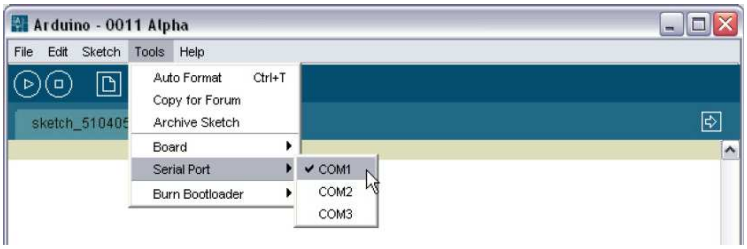
Выберите последовательно пункты меню **Инструменты → Плата→ POP-168** или **Arduino Mini** (**Tools → Board → POP-168** или **Arduino Mini)** (можно использовать обе версии)



4.2.1.2 Выбор COM-порта

Для загрузки скетча из IDE Arduino в модуль POP-168 необходим обмен данными по последовательному порту. Среда разработки IDE Arduino может работать с виртуальным COM-портом, который создается для преобразователя интерфейса USB в Последовательный порт.

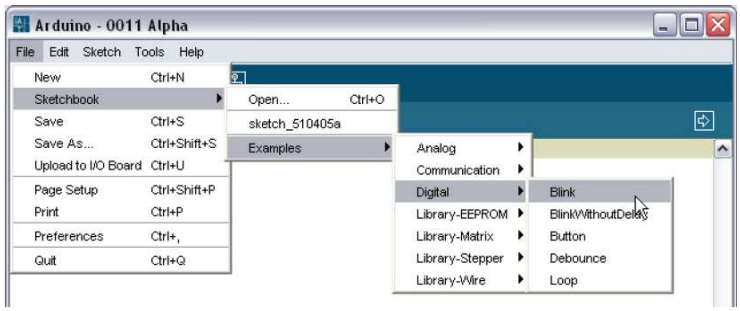
Выберите в меню **Инструменты → Последовательный порт (Tools → Serial Port)**. В открывшемся списке можно выбрать COM-порт, который будет использоваться для обмена данными с роботом.



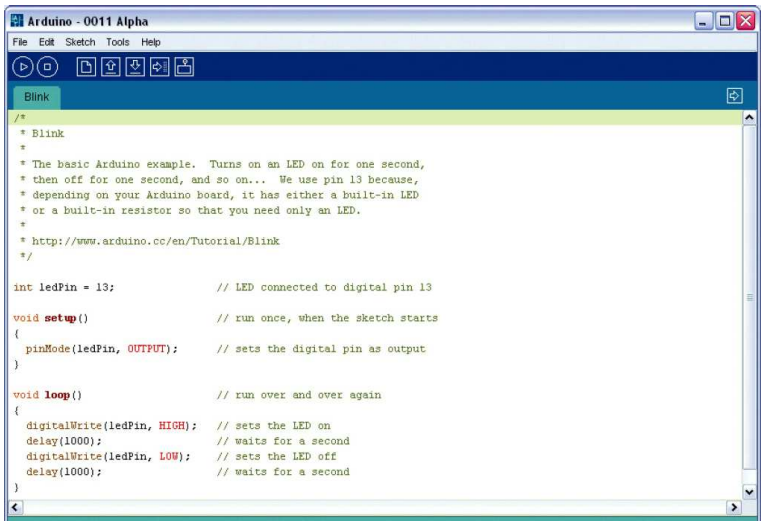
Обычные встроенные последовательные порты имеют адреса COM1 и/или COM2. Для Последовательного порта USB адрес будет COM3 или выше. Но Arduino может поддерживать COM-порты не выше COM9.

4.2.2 Загрузка скетч-файла примера

Последовательно выберите в меню **Файл → Подшивка скетчей Примеры → Цифровые → Мигание** (**File → Sletchbook → Examples → Digital → Blink)**

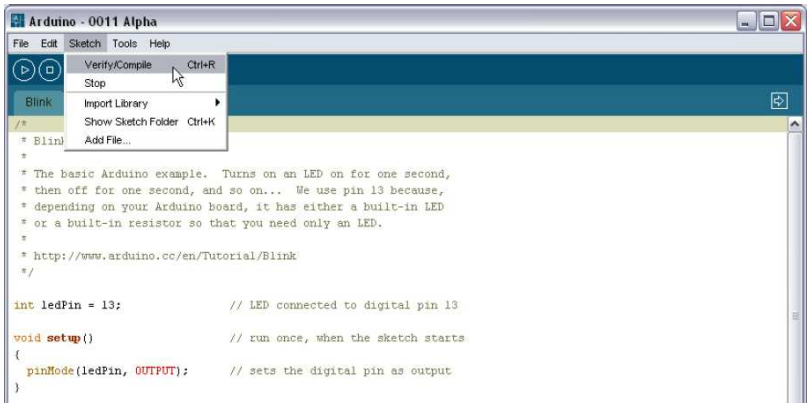


Код примера, Blink.pde, появится в области редактирования текстового редактора.



4.2.3 Компиляция скетча

После открытия скетч-файла в редакторе, его можно откомпилировать, выбрав последовательно пункты меню **Скетч → Проверить/Откомпилировать** (**Sketch → Verify/Compile)** или нажав кнопку  на панели инструментов.

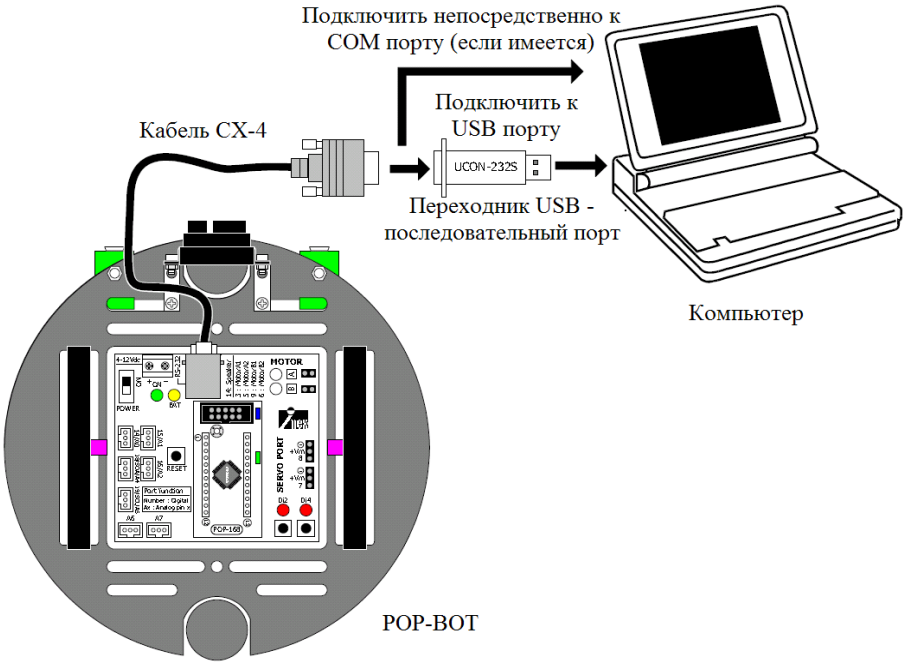


В строке состояния в нижней части основного окна будет отображаться состояние компиляции. Если компиляция прошла без ошибок, то отобразится сообщение **Компиляция завершена** (**Done compiling),** и в Текстовой области появится сообщение с указанием размера двоичного файла скетча.

4.2.4 Загрузка скетча в модуль POP-168

Передача скомпилированного машинного кода в аппаратную часть Arduino называется Загрузка (Uploading). Перед загрузкой кода аппаратную часть Arduino необходимо подготовить, переведя POP-168 в режим Начального загрузчика (Bootloader mode). Для этого необходимо выполнить следующую последовательность действий:

(1) Соединить POP-BOT с COM-портом компьютера посредством кабеля CX-4 или кабеля UCON-4 преобразователя интерфейса USB в Последовательный порт.



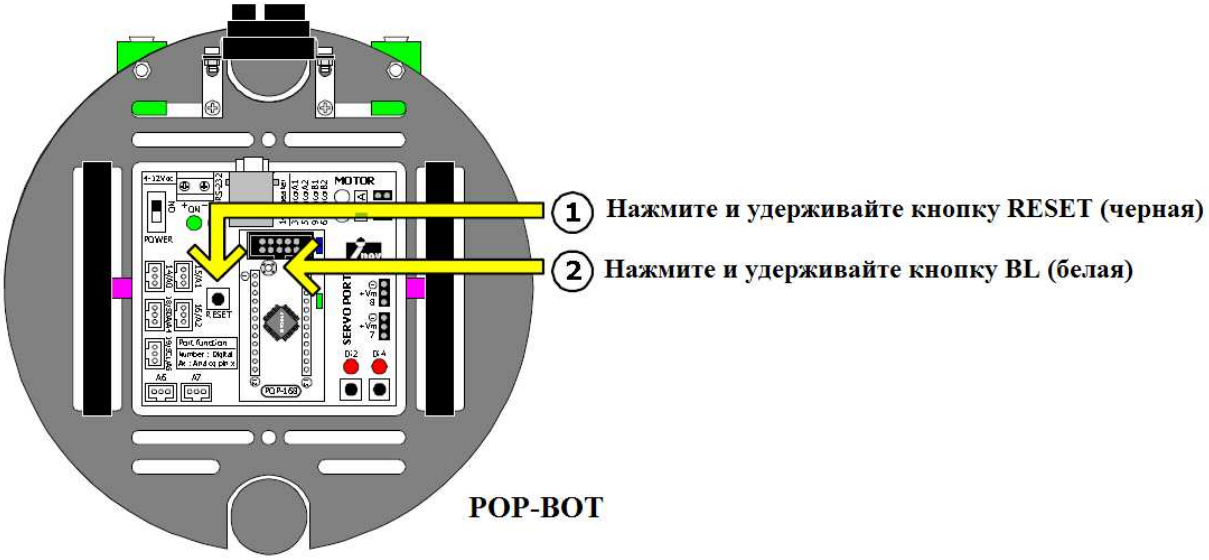
(2) Перевести модуль POP-168 в режим программирования. Для этого необходимо сделать 2 шага.

(2.1) Используя кнопку СБРОС (RESET) на плате RBX-168 и кнопку BL в модуле POP-168

(2.1.1) Включите POP-BOT

(2.1.2) Нажмите и удерживайте кнопку СБРОС (RESET) на плате управления RBX-168.

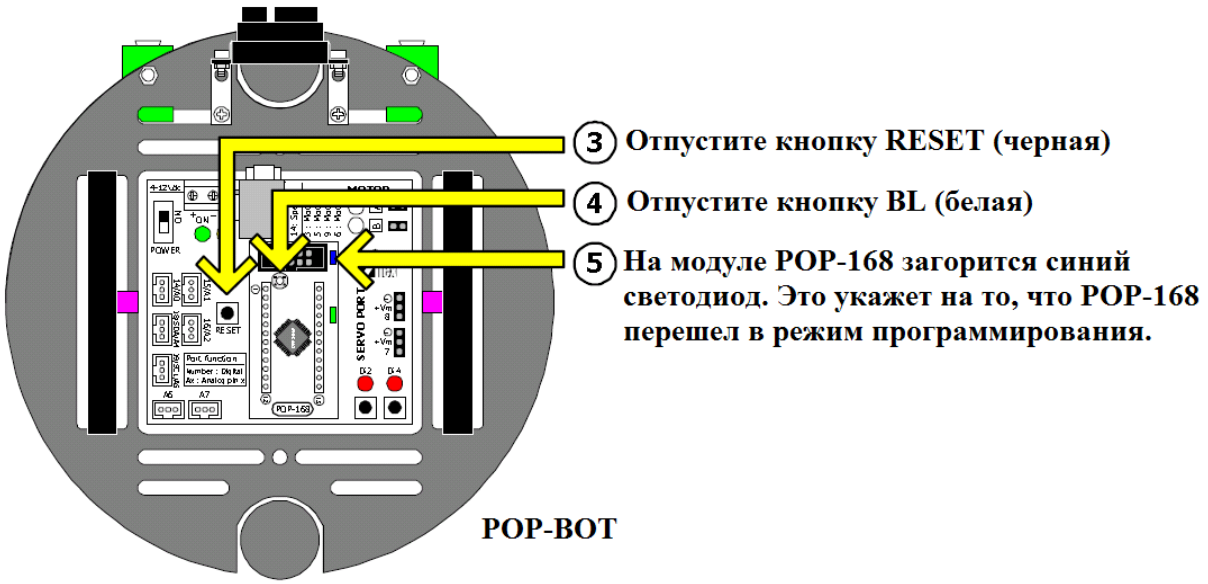
(2.1.3) Нажмите и удерживайте кнопку BL в модуле POP-168.



(2.1.4) Отпустите кнопку СБРОС (RESET).

(2.1.5) Затем отпустите кнопку BL.

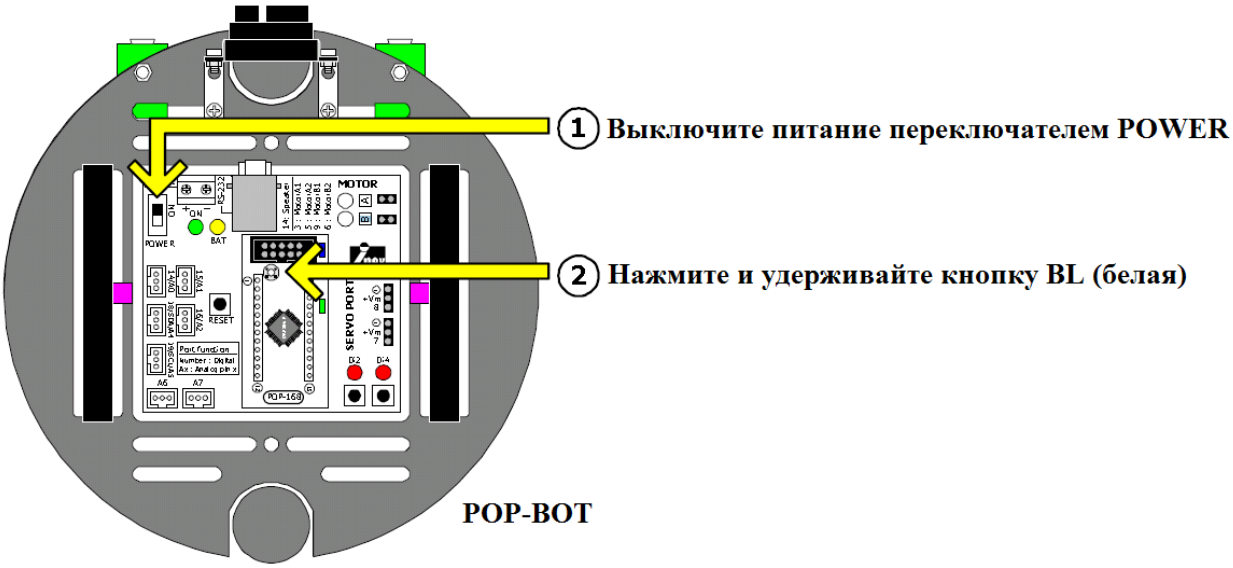
Если на POP-168 загорится и не будет мигать синий светодиод, то POP-168 успешно вошел в режим Начального загрузчика (Bootloader mode) и готов для загрузки машинных кодов.



(2.2) Используйте переключатель ПИТАНИЕ (POWER) на плате RBX-168 и кнопку BL в модуле POP-168

(2.2.1) Выключите POP-BOT.

(2.2.2) Нажмите и удерживаете кнопку BL.



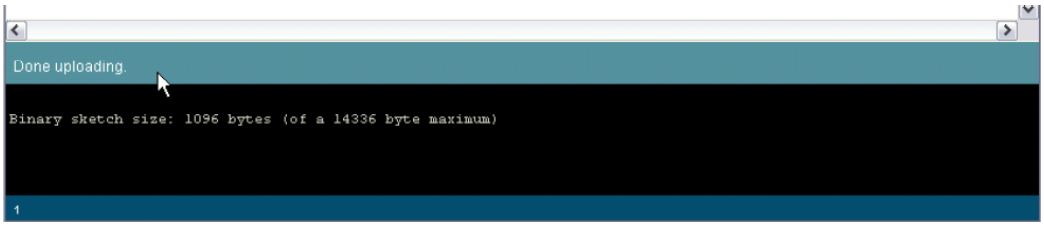
(2.2.3) Включите POP-BOT переключателем ***Питание (POWER)***

(2.2.4) Отпустите кнопку ***BL***

Если синий светодиод на плате POP-168 зажегся и не мигает, то POP-168 вошел в режим Начального Загрузчика (Bootloader mode) и готов к загрузке машинных кодов (прошивки).

(3) В среде Arduino IDE, выберите пункт меню **Файл Загрузить в плату Ввода/Вывода** (**File Upload to I/O Board**). Подождите завершения загрузки.

(4) После завершения загрузки, в строке состояния в нижней части основного окна будет отображаться сообщение **Загрузка завершена** (**Done Uploading**).



(5) После завершения загрузки, еще раз нажмите кнопку RESET (СБРОС). После этого POP-BOT начнет выполнение скетча.

Светодиод, подключенный к выводу Di13 (Синий светодиод) модуля POP-168, начнет мигать с интервалом в 1 секунду.

5 : Описание основных типов движения POP-BOT

Робот POP-BOT имеет 2 канала управления электродвигателями постоянного тока. В каждом из каналов можно независимо управлять скоростью и направлением вращения двигателя с помощью программного обеспечения. Электродвигатели постоянного тока управляются сигналами с ШИМ (широтно-импульсная модуляция). В данном разделе описано, как управлять электродвигателями постоянного тока с помощью ШИМ и как генерировать ШИМ-сигнал микроконтроллером Arduino POP-168 в программе на языке C.

5.1 Основы управления двигателем постоянного тока при помощи ШИМ

Изменяя (модулируя) ширину импульса, подаваемого на электродвигатель постоянного тока, можно увеличивать или уменьшать количество энергии, поступающей на электродвигатель в единицу времени, таким образом, увеличивая или уменьшая скорость (частоту вращения) электродвигателя. Заметим, что хотя напряжение, подаваемое на электродвигатель, имеет постоянную амплитуду, изменяется коэффициент заполнения импульсов управления. Это означает, что при большей ширине импульса, электродвигатель будет вращаться с большей частотой.

Обратимся к Рисунку 5-1, на котором показано напряжение питания Vs, с ШИМ, подаваемое на электродвигатель постоянного тока. Частота вращения электродвигателя будет зависеть от времени Ton (время включения мотора). В течение этого промежутка времени, на электродвигатель постоянного тока будет подаваться полное напряжение питания; Vm. Если ширина Ton будет больше, то на электродвигатель постоянного тока будет поступать больше напряжения. В результате, он станет вращаться с большей скоростью (частотой). Процентное соотношение между временем Ton и полной длительностью периода (T) называется Коэффициент заполнения (Duty cycle). Его можно вычислить следующим образом :

$$\text{коэффициент заполнения } (\%) = 100 \times \frac{T_{on}}{T_{on} + T_{off}} \qquad (5.1)$$

$$\text{частота ШИМ} = \frac{1}{T_{on} + T_{off}} = \frac{1}{T} \qquad (5.2)$$

Среднее падение напряжение на электродвигателе = Напряжение источника питания x Коэффициент заполнения (%) (5.3)

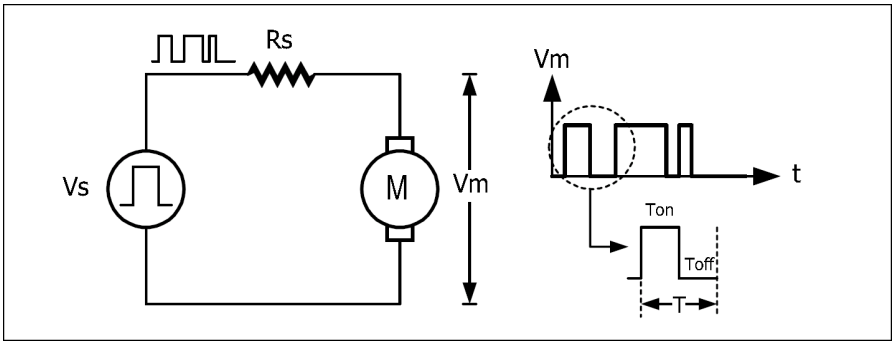


Рисунок 5-1. ШИМ-сигнал для управления электродвигателем

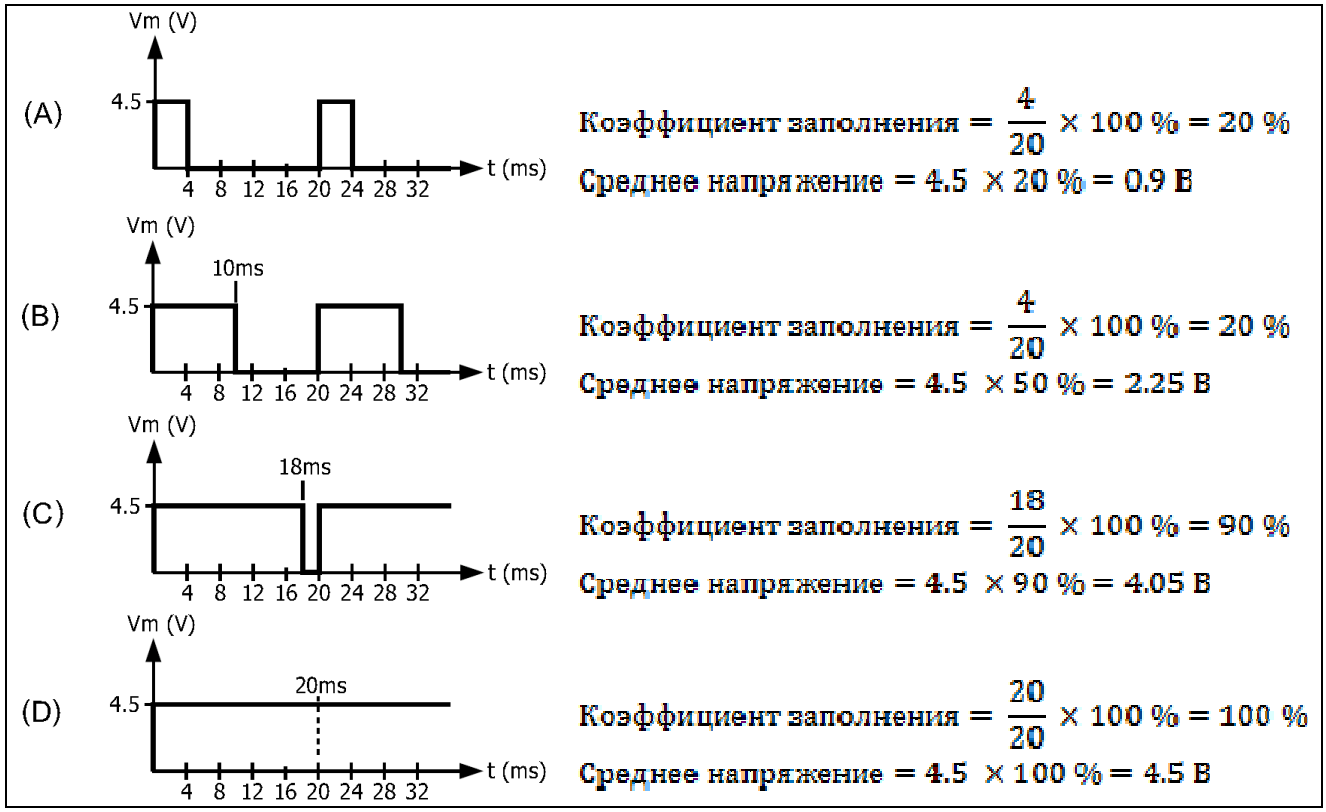


Рисунок 5-2. Показано соотношение между различными значениями коэффициента заполнения (скважности) импульсов управления и средним напряжением на электродвигателе.

Хотя коэффициент заполнения определяет частоту вращения вала электродвигателя, электродвигатель имеет предельную рабочую частоту. Если частота ШИМ-сигнала превышает этот порог, электродвигатель остановится, поскольку перейдет в точку насыщения. Пример сигнала с ШИМ на рисунке 5-2 имеет период 20 миллисекунд и частоту 50Гц.

На рисунке 5-2 (A) коэффициент заполнения сигнала с ШИМ составляет 20%. Электродвигатель будет вращаться с минимальной скоростью, поскольку среднее падение напряжения на нем составит всего 0.9В. При увеличении коэффициента заполнения на Рисунке 5-2 (B) и (C), среднее падение напряжения на электродвигателе увеличится. Это также приведет к увеличению скорости.

На Рисунке 5-2 (D) на электродвигатель постоянного тока подается полный уровень напряжения, поскольку коэффициент заполнения равен 100%. Таким образом, управление коэффициентом заполнения ШИМ-сигнала является методом регулирования скорости электродвигателя.

5.2 Arduino с ШИМ (PWM)

Среда Arduino имеет специальную функцию для генерации сигнала с ШИМ, который можно подать на любой из цифровых выходов. Это функция analogWrite(). Пользователь может изменять коэффициент заполнения ШИМ- сигнала от 0 до 100%, изменяя значение параметра функции от 0 до 255.

При значении = 0, ШИМ-сигнал отсутствует. Выходное напряжение равно 0В.

При значении = 51, Ширина импульса ШИМ-сигнала равна 20% периода. Коэффициент заполнения равен 20%.

При значении = 127, Ширина импульса ШИМ-сигнала равна половине периода. Коэффициент заполнения равен 50%.

При значении = 191, Ширина импульса ШИМ-сигнала равна 75% периода. Коэффициент заполнения равен 75%.

При значении = 255, Ширина импульса ШИМ-сигнала равна целому периоду. Коэффициент заполнения равен 100%.

На Рисунке 5-2 показан ШИМ-сигнал с различными значениями коэффициента заполнения.

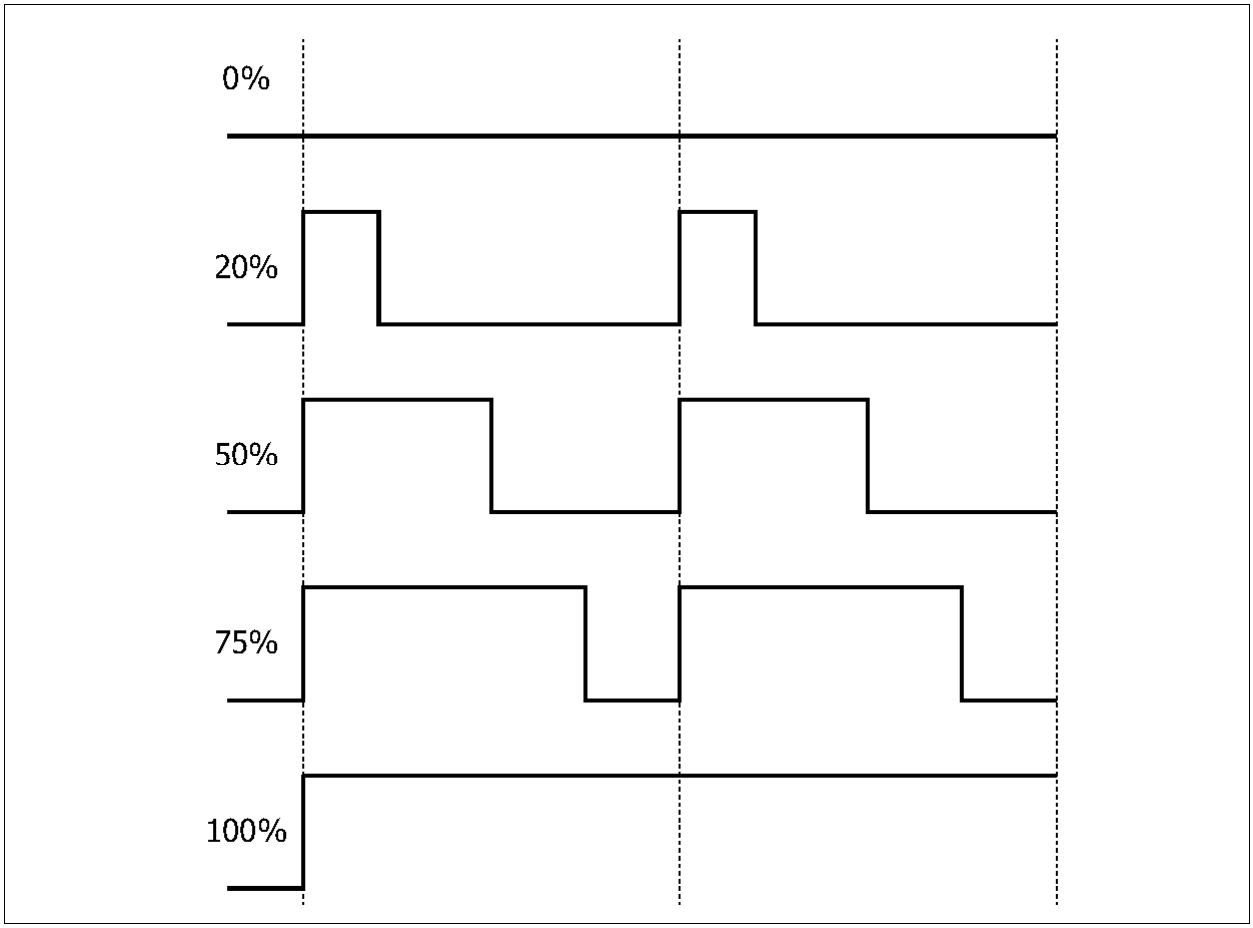


Рисунок 5-2. Форма выходного сигнала ШИМ при различных значениях коэффициента заполнения (скважности) импульсов.

Среднее значение выходного напряжение сигнала с ШИМ определяется его коэффициентом заполнения. Это среднее значение можно вычислить из приведенного ниже соотношения:

Выходное_напряжение = (время_включенного_состояния / время_выключенного_состояния) * максимальное_напряжение

ШИМ-сигнал, создаваемый функцией analogWrite(), можно использовать для регулирования яркости светодиода или применить для управления электродвигателем постоянного тока. Вывод Arduinons, который определен как выход сигнала с ШИМ будет выдавать сигнал с ШИМ непрерывно, до тех пор, пока не будет вызвана функция analogWrite() с новым значением периода и коэффициента заполнения, или не будут вызваны функции digitalRead или digitalWrite для этого же самого вывода.

Модуль Arduino POP-168 имеет 4 аналоговых выхода; они включают выводы 3, 5, 6 и 9 (Di3, Di5, Di6 и Di9)**.**

Функция analogWrite имеет формат

analogWrite(вывод, значение);

где ***вывод*** *- это номер вывода (3, 5, 6 и 9) порта Arduino*
значение *– это значение коэффициента заполнения от 0 до 255.*

Задание 1 : Базовые перемещения POP-BOT

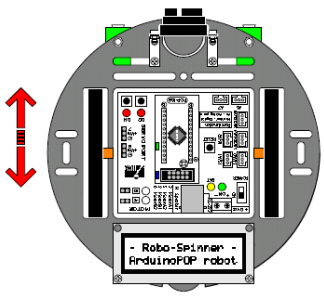
Задание 1-1 Движение вперед и назад

A1.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A1-1.
A1.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A1.3 Выключите напряжение питания робота и отключите загрузочный кабель.
A1.4 Убедитесь, что робот находится на плоской поверхности. Включите питание робота и наблюдайте за его работой.

Робот POP-BOT двигается вперед. Убедитесь, что оба светодиодных индикатора работы двигателя имеют зеленый цвет свечения. Через 1 секунду оба индикатора меняют свой цвет свечения на красный, и робот начинает двигаться назад.

Если робот функционирует неверно, *необходимо переподключить кабели электромоторов к противоположным портам / в противоположной полярности. Это необходимо делать до тех пор, пока робот не начнет двигаться правильно. Когда Вы получите нужный результат, запомните полученную конфигурацию подключения электродвигателей и используйте ее для выполнения всех заданий. Робот будет двигаться вперед-назад непрерывно, до тех пор, пока вы не выключите его питание.*

```
/*****************************************************************************
* POP-BOT V1.0
* Выполняет движение Вперед/Назад на полной скорости
*****************************************************************************/
void setup(){
      pinMode(3,OUTPUT); // Motor A1
      pinMode(5,OUTPUT); // Motor A2
      pinMode(6,OUTPUT); // Motor B2
      pinMode(9,OUTPUT); // Motor B1
}
void Forward(){ // Подпрограмма движения робота вперед
      digitalWrite(3,HIGH);
      digitalWrite(5,LOW);
      digitalWrite(6,HIGH);
      digitalWrite(9,LOW);
}
void Backward(){ // Подпрограмма движения робота вперед
      digitalWrite(3,LOW);
      digitalWrite(5,HIGH);
      digitalWrite(6,LOW);
      digitalWrite(9,HIGH);
}
void loop(){
      Forward();
      delay(1000);
      Backward();
      delay(1000);
}
/*****************************************************************************/
```



Листинг A1-1 : Файл Forward_Backward.pde; скетч-файл Arduino для управления движением POP-BOT вперед и назад

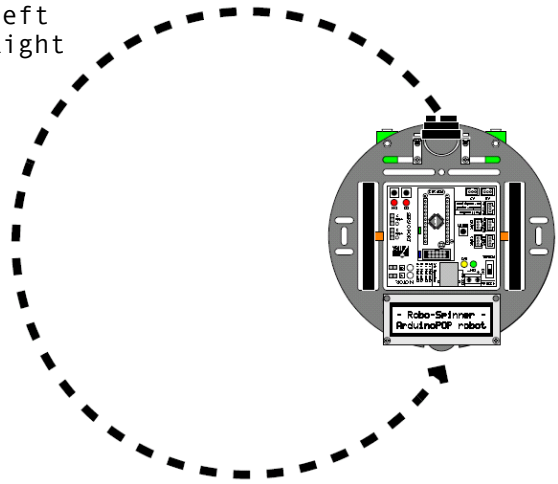
Задание 1-2 Управление движением робота по кругу

Если для каждого мотора установить различную скорость вращения, то это приведет к тому, что робот начнет двигаться по кругу. Это можно проверить следующим образом:

A1.5 Создайте новый скетч и запишите в него следующие коды на языке C, показанные на Листинге A1-2.
A1.6 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A1.7 Выключите напряжение питания робота и отключите загрузочный кабель.
A1.8 Убедитесь, что робот находится на плоской поверхности. Включите питание робота и наблюдайте за его работой.

Робот непрерывно двигается по кругу до тех пор, пока не нажата кнопка, подключенная к выводу Di4 платы управления POP-BOT, после чего робот останавливается.

```
/*********************************************************************
* POP-BOT V1.0
* Filename : MotorSpeedControl.pde
* Левый электродвигатель вращается с низкой скоростью, а правый
электродвигатель вращается
* с высокой скоростью, при этом Robo-Spinner обеспечивает движение движется
по кругу
*********************************************************************/
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(2,INPUT); // Электродвигатель Left
      pinMode(4,INPUT); // Электродвигатель Right
}
void Forward(int Lspeed,int Rspeed){
      analogWrite(3,Lspeed);
      digitalWrite(5,LOW);
      analogWrite(6,Rspeed);
      digitalWrite(9,LOW);
}
void Motor_Stop(){
      digitalWrite(5,LOW);
      digitalWrite(3,LOW);
      digitalWrite(6,LOW);
      digitalWrite(9,LOW);
}
void loop(){
      Forward(80,255);        // Выполняется циклически
      if(digitalRead(4)==0){  // пока не нажата кнопка
            Motor_Stop();     // Останов
            while(1);
      }
}
/*********************************************************************/
```



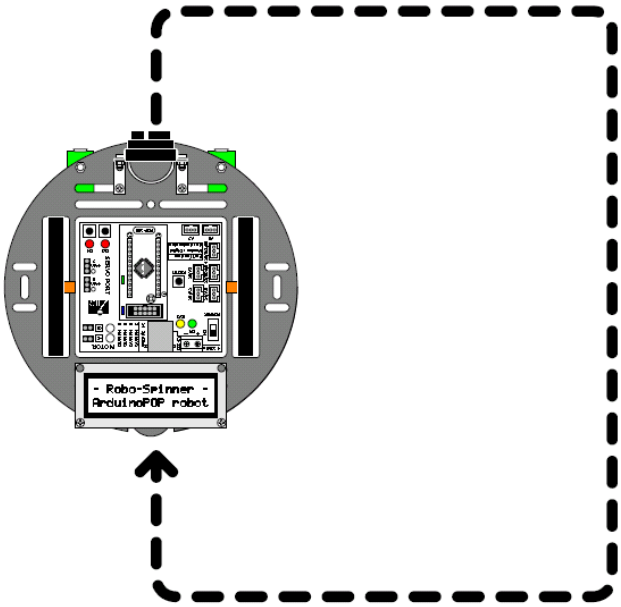
Листинг A1-2 : Файл MotorSpeedControl.pde; скетч-файл Arduino для управления движением POP-BOT по кругу

Задание 1-3 Управление движением робота по контуру прямоугольника

A1.9 Создайте новый скетч и запишите в него следующие коды на языке C, показанные на Листинге A1-3.
A 1.10 Загрузите скетч в робот. Выключите напряжение питания робота и отключите загрузочный кабель.
A1.11 Включите питание робота и наблюдайте за его работой.

Робот начинает движение после нажатия кнопки SW1 или SW2. ***При нажатии на кнопку SW1****, робот будет непрерывно двигаться вперед, и поворачивать налево, описывая своим движением квадрат.* ***При нажатии на кнопку SW2****, повороты будут выполняться в противоположном направлении.*

```
/*************************************************************************
* Robo-Spinner V1.0
* Filename : Rectangle_Running.pde
* Выполнение поворотов влево и вправо на 90 градусов
*************************************************************************/
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(2,INPUT); // Левая кнопка
      pinMode(4,INPUT); // Правая кнопка
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void loop(){
      if (digitalRead(2)==0){ // Нажата кнопка Di2
            while(1){
                  Forward(125); // Движение вперед
                  delay(900);
                  Spin_Left(125); // Поворот влево на 90 градусов
                  delay(400);
            }
      }
      if (digitalRead(4)==0){        // Нажата кнопка Di4
            while(1){
                  Forward(125);      // Движение вперед
                  delay(900);
                  Spin_Right(125);   // Поворот вправо на 90 градусов
                  delay(400);
            }
      }
}
/*************************************************************************/
```



Листинг A1-3 : Файл Rectangle_Running.pde; скетч-файл Arduino для управления движением POP-BOT по замкнутому прямоугольному контуру

Задание 2 : Бампер робота POP-BOT

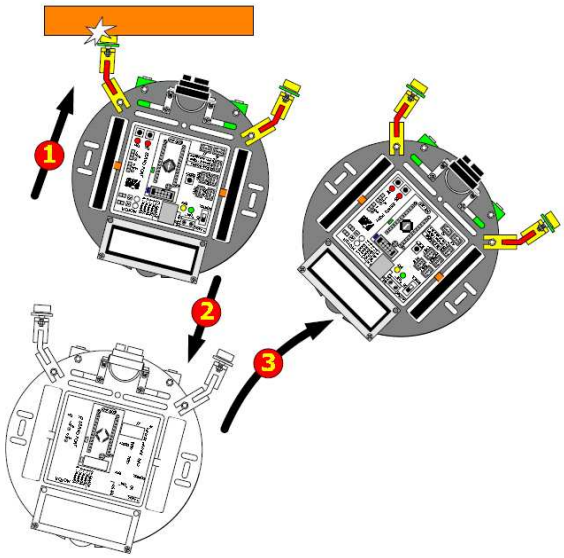
Задание 2-1 Простейший метод обнаружения препятствий

В этом задании робот будет запрограммирован таким образом, чтобы он реагировал на столкновения, приводящие к нажатию на любую из двух кнопок, расположенных в передней части POP-BOT на кронштейнах. После того, как столкновение произойдет, робот будет отходить назад и менять направление своего движения.

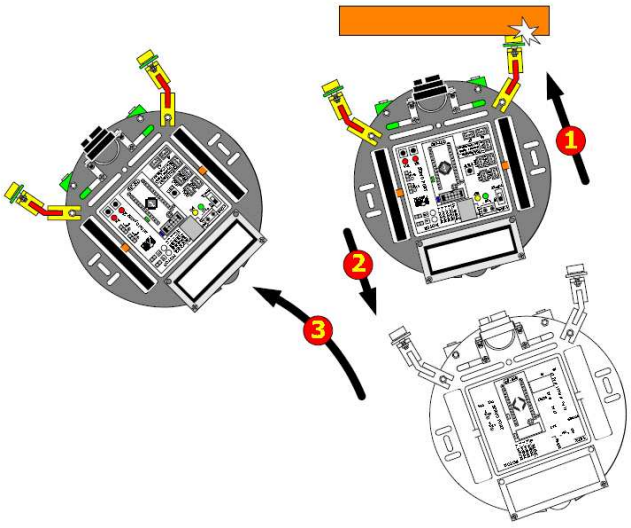
A2.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A2-1.
A2.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A2.3 Выключите напряжение питания робота и отключите загрузочный кабель.
A2.4 Подготовьте демонстрационное поле, разместив на нем несколько коробок или других объектов.
A2.5 Поместите робот на демонстрационное поле. Включите питание робота и наблюдайте за его работой.

Робот POP-BOT будет читать состояние обеих кнопок, подключенных к портам **15/A1** и **17/A3**. При нажатии на кнопку или касании ею некоторого объекта, на соответствующем выходе появится логический “0”.

При нормальной работе *робот будет непрерывно двигаться вперед.*
Если робот прикоснулся к какому-либо объекту левой кнопкой*, то робот будет двигаться назад и изменит направление своего движения, повернув направо, чтобы избежать столкновения с объектом.*
Если робот прикоснулся к какому-либо объекту правой кнопкой*, то робот будет двигаться назад и изменит направление своего движения, повернув налево, чтобы избежать столкновения с объектом.*



Робот сталкивается с объектом слева



Робот сталкивается с объектом справа

```
/*************************************************************************
* POP-BOT V1.0
* Filename : BumperRobot.pde
* POP-BOT с датчиками на бамперах
*************************************************************************/
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(15,INPUT); // Левая кнопка
      pinMode(17,INPUT); // Правая кнопка
}
void loop(){
      Forward(150);
      if (digitalRead(15)==0){ // Проверка состояния  кнопки левого бампера
            Backward(150);delay(500);
            Spin_Right(200);delay(400);
      }
      if (digitalRead(17)==0){ // Проверка состояния  кнопки правого бампера
            Backward(150);delay(400);
            Spin_Left(200);delay(400);
      }
}
/*************************************************************************/
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Backward(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*************************************************************************/
```

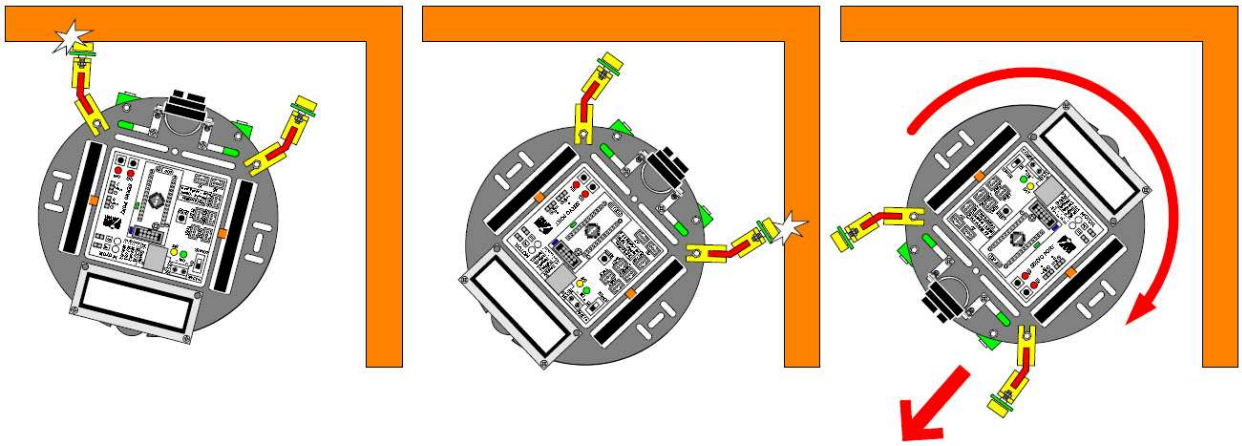
Листинг А2-1 : Файл BumperRobot.pde; скетч-файл Arduino для управления бамперами POP-BOT

Задание 2-2 Выход из ловушки в углу комнаты

Когда POP-BOT находится в углу, он оказывается зажатым между правой и левой стенами. Это вызывает непрерывное касание бамперами робота стен и, таким образом, робот оказывается захваченным в ловушку в этом углу. Решением является изменение кода на языке C, который был показан на Листинге A2-1, таким образом, чтобы он соответствовал Листингу A2-2.

A2.6 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A2-2.
A2.7 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A2.8 Выключите напряжение питания робота и отключите загрузочный кабель.
A2.9 Подготовьте демонстрационное поле, разместив на нем несколько коробок или других объектов, как было сделано в Задании 2-1.
A2.10 Поместите робот на демонстрационное поле. Включите питание робота и наблюдайте за его работой.

Робот будет двигаться вперед и проверять наличие препятствий. Если случится более 5 столкновений подряд, робот повернется на 180 градусов, чтобы изменить направление своего движения.



```
/*****************************************************************************
* POP-BOT V1.0
* Filename : CornerEscape.pde
* Робот сбегает из угла комнаты
*****************************************************************************/
int Count=0;
int Flag_=0;
void setup(){
      pinMode(3,OUTPUT); // Двигатель A1
      pinMode(5,OUTPUT); // Двигатель A2
      pinMode(6,OUTPUT); // Двигатель B2
      pinMode(9,OUTPUT); // Двигатель B1
      pinMode(15,INPUT); // Левая кнопка
      pinMode(17,INPUT); // Правая кнопка
}
void loop(){
      Forward(150);

      if (Count>5){           // Оказывается в ловушке в углу более 5 раз ?
            Count=0;
            Backward(150);          // Сбегает из угла
```

```
            delay(2000);
            Spin_Right(200);
            delay(800);
      }
      if (digitalRead(15)==0){      // Проверка состояния кнопки левого
      бампера
            if(Flag_==1){           // Проверка предыдущего состояния
                  Count++;
            }
            else{
                  Count=0;
            }
            Flag_=0;
            Backward(150);          // Нормальная работа
            delay(500);
            Spin_Right(200);
            delay(400);
      }
      if (digitalRead(17)==0){      // Проверка состояния кнопки правого
      бампера
            if(Flag_==0){           // Проверка предыдущего состояния
                  Count++;
            }
            else{
                  Count=0;
            }
            Flag_=1;
            Backward(150); // Нормальная работа
            delay(400);
            Spin_Left(200);
            delay(400);
      }
}
/*****************************************************************************/
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Backward(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*****************************************************************************/
```

Листинг А2-2 : Файл CornerEscape.pde; скетч-файл Arduino для управления действиями POP-BOT в углах

6 : POP-BOT и ЖКИ с последовательным интерфейсом

ЖКИ SLCD16x2 представляет собой модуль ЖКИ содержащий 2 строки по 16 символов, который обменивается данными с помощью последовательного интерфейса. Это позволяет получать данные последовательно и отображать их на экране ЖКИ. Последовательные данные могут приниматься при скорости обмена от 2400 до 9600 Бод (бит/сек). Обычный интерфейс для управления ЖКИ требует не меньше 6 сигнальных линий, но для SLCD16x2 необходима только одна сигнальная линия. Такой дисплейный модуль является удобным для использования в роботе POP-BOT.

6.1 Информация об SLCD16x2

6.1.1 Особенности

- Последовательный ввод или ввод с помощью Инвертированных/Неинвертированных логических сигналов с уровнями TTL/CMOS
- Выбираемый перемычкой режим работы 1/8 или 1/16
- Совместим с набором команд Scott Edwardshs LCD Serial Backpack™ . Добавлены расширенные команды, упрощающие управление ЖКИ.
- Простота организации обмена данными с микроконтроллером
- Работает от единственного источника питания с напряжением +5 В постоянного тока

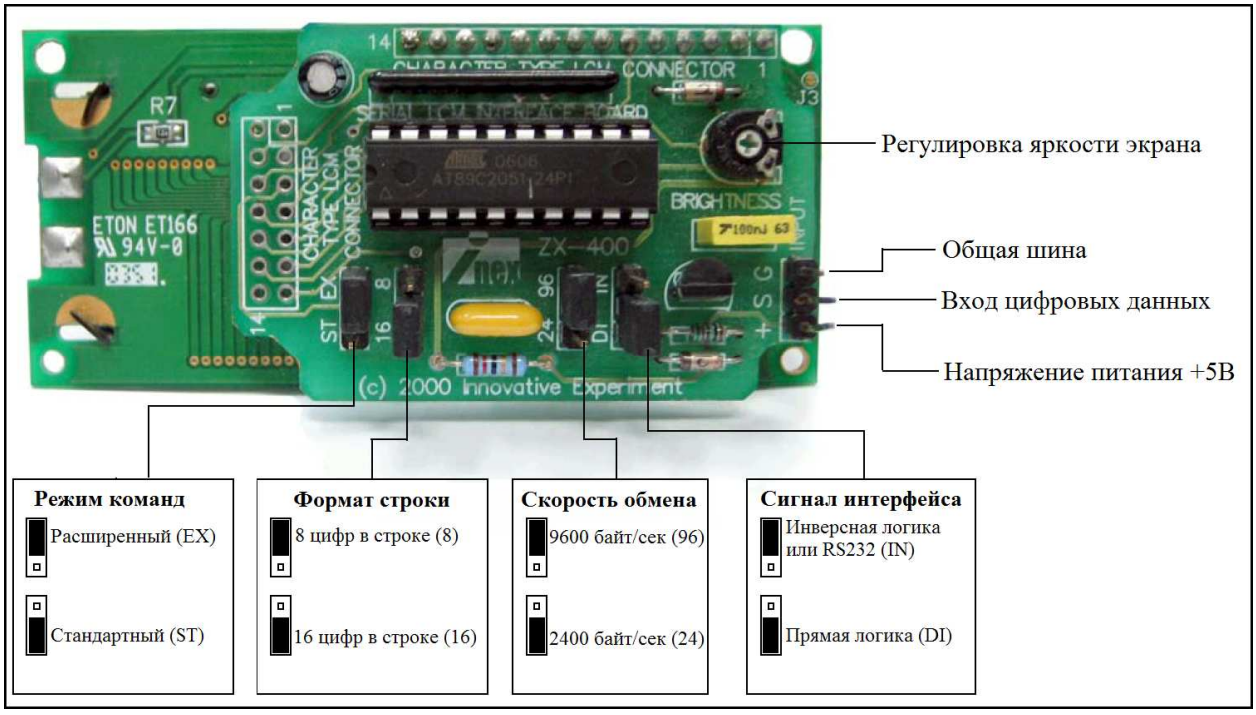


Рисунок 6-1. Детальное описание положения перемычек на модуле ЖКИ SLCD16x2

6.1.2 Настройки

На Рисунке 6-1, показан детальный вид задней стороны SLCD16x2. Пользователь может увидеть 4 двухпозиционных перемычки, которые конфигурируются следующим образом :

(1) **Перемычка режима команд (Mode command jumper) :** Переключает режимы команд. SLCD16x2 имеет 2 режима команд. Первым является режим Стандартных команд (ST). Этот режим полностью совместим с набором команд Scott Edwardshs LCD Serial Backpack™. Вторым режимом является режим Расширенных команд (EX). ***Для всех Заданий по изучению POP-BOT, размещенных в данном руководстве, необходимо выбрать режим Стандартных команд (ST).***

(2) **Перемычка настройки длины строки (Lines jumper)** : Выбирает режим работы строк дисплея: 1/8 или 1/16. Режим 1/8 означает, что в каждой строке будет отображаться 8 знаков. Режим 1/16 означает, что в каждой строке будет отображаться 16 или более. В стандартном режиме перемычка устанавливается в положение ***1/16.***

(3) **Перемычка выбора скорости обмена (Baudrate select jumper)**: Выбирается одна из 2 возможных скоростей обмена - 2400 или 9600 бит/сек, с форматом данных 8N1 (8-битные данные, нет бита четности и 1 стоповый бит). ***Для POP-BOT необходимо установить в положение 9600.***

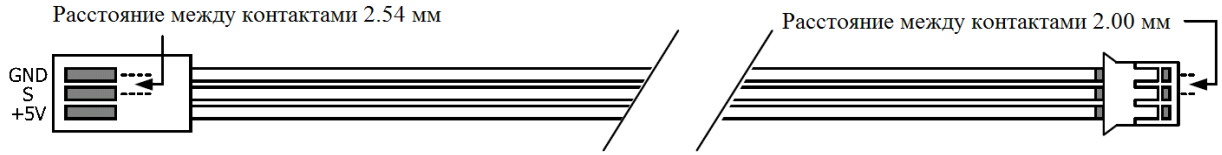
(4) **Перемычка выбора сигнального интерфейса (Interface signal jumper) :** Выбирается один из 2 возможных вариантов – Инвертированная логика с TTL/CMOS уровнями сигналов (IN) или Прямая логика с TTL/CMOS уровнями сигналов (DI). ***Для POP-BOT необходимо установить в положение DI.***

ЖКИ SLCD16x2 обеспечивает регулировку яркости переменным резистором, отмеченным надписью **BRIGHTNESS** (**ЯРКОСТЬ**).

Соединительный разъем имеет 3 вывода : Напряжение питания +5В (+), Вход последовательных данных (**S**) и Общая шина (**G**).

6.1.3 Организация обмена данными между SLCD16x2 и POP-BOT

Для соединения обмена данными между SLCD и платой управления POP-BOT необходим кабель JST3AA-8. Назначение выводов разъемов кабеля показано на следующем рисунке.



В отличие от показанного на рисунке, кабель JST3AA-8 имеет на обоих концах разъемы с шагом между контактами 2.00мм. Это позволяет подключить его к любому разъему JST любого порта платы управления POP-BOT и к входному разъему SLCD16x2

После подключения убедитесь, что все перемычки находятся в следующих положениях :

- Выбор режима команд в положении **Стандартные** (**ST**)
- Выбор длины строки дисплея в положении **16 символов в строке** (**16**)
- Выбор скорости обмена в положении **9600 бит/сек** (**96**)
- Выбор сигнального интерфейса в положении **Прямой** (**DI**)

6.1.4 Обмен данными и командами

После надлежащего подключения и конфигурирования ЖКИ SLCD16x2, ему по последовательному интерфейсу можно посылать команды и данные. Для отправки данных достаточно послать любое сообщение, такое как “Hello”, непосредственно через последовательный интерфейс Ввода/Вывода, в результате чего сообщение “Hello” будет показано на экране ЖКИ.

Для отправки команды, достаточно отправить инструкцию из стандартного набора на ЖКИ (см Рисунок 8-2), предварительно послав символ префикса команды, ASCII 254 (0FE в шестнадцатеричном коде или 11111110 в двоичном коде). ЖКИ SLCD16x2 рассматривает байт, поступивший непосредственно после префикса, как командную инструкцию, после чего сразу переходит в обычный режим приема данных.

Например: чтобы очистить экран ЖКИ, команда очистки в двоичном формате 00000001 (или ASCII 1), необходимо послать в SLCD16x2 вначале [254], а затем [1] (где заключенные в квадратные скобки [ ] символы означают одиночный байт со значением, соответствующим помещенной в скобках величине)

Команда\Бит данных	D7	D6	D5	D4	D3	D2	D1	D0
1.Инициализация ЖКИ	0	0	0	0	0	0	0	0
2.Очистка ЖКИ	0	0	0	0	0	0	0	1
3.Курсор в начало	0	0	0	0	0	0	1	*
4.Выбор режима	0	0	0	0	0	1	I / D	S
5.Показ настроек	0	0	0	0	1	D	C	B
6.Сдвиг дисплея	0	0	0	1	S / C	R / L	*	*
7.Настройка функций	0	0	1	*	N	F	*	*
8.Установка адреса CGRAM	0	1	A5	A4	A3	A2	A1	A0
9.Установка адреса DDRAM	1	A6	A5	A4	A3	A2	A1	A0

* Безразличное состояние бита

S 0=Автоматический сдвиг курсора после приема байта
1=Курсор не передвигается

I/D 0=После приема байта, позиция курсора уменьшается на 1
1=После приема байта, позиция курсора увеличивается на 1

При S=1 курсор не сдвигается

D 0=Дисплей выключен, 1= Дисплей включен

C 0=Курсор выключен, 1=Курсор включен

B 0=Курсор не мигает, 1=Курсор мигает

S/C 0=Сдвиг курсора, 1=Сдвиг дисплея

R/L 0=Сдвиг влево, 1=Сдвиг вправо

N 0=Длина строки 1/8, 1=Длина строки 1/16
(не рекомендуется устанавливать этот бит,
вместо этого используйте установку перемычкой)

F 0=Размер символа 5x7 точек, 1=Размер символа 5x10 точек

A0...A7 - адрес CGRAM или DDRAM

Обзор стандартного набора команд
(За исключением команды Инициализации ЖКИ.
При инициализации устанавливаются следующие значения:
I/D=1, S=0, D=1, C=0, B=0, N=1, F=0, Адрес DDRAM=00)

Временная диаграмма последовательного обмена данными

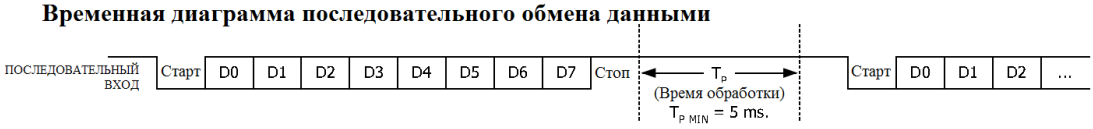


Рисунок 6-2. Список команд и временные диаграммы работы ЖКИ SLCD16x2

6.1.5 Набор символов ЖКИ

Большинство символов ЖКИ (Рисунок E) не могут быть изменены, поскольку они записаны в ПЗУ (ROM). Однако, первые восемь символов, соответствующие номерам от 0 до 7 набора символов ASCII, запоминаются в ОЗУ (RAM). Путем записи новых значений в ОЗУ генератора символов (CGRAM) можно как угодно изменять эти символы, основываясь на матрице размером 5x8 точек.

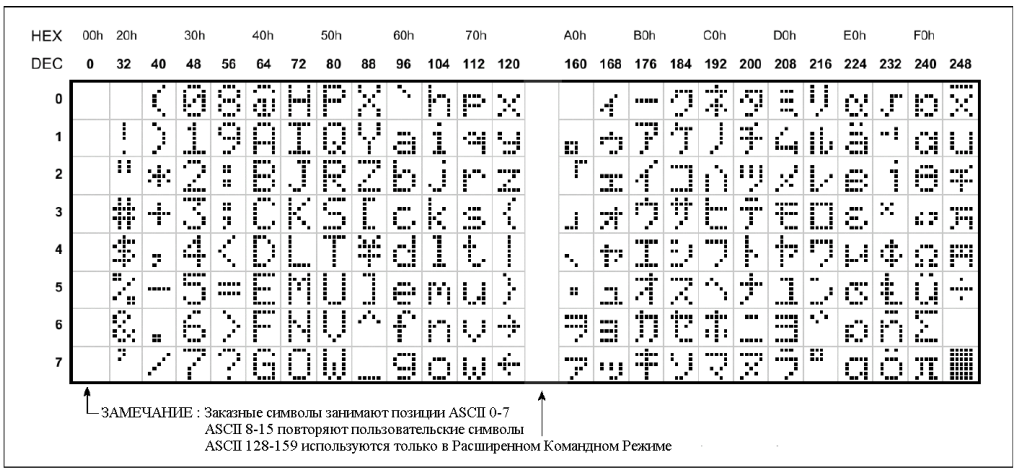


Рисунок Е. Таблица символов ЖКИ. (Встроенные символы контроллеров HD44780 или SED1278F0A)

Чтобы создать собственные символы, необходимо указать начальный адрес CGRAM, по которому будет записана первая строка символа, затем записать первую строку битов символа, после чего установить следующий адрес CGRAM и записать следующий байт. После повторения этой процедуры 8 раз (для каждого символа), можно использовать записанный символ. Область CGRAM 0 расположена по адресам CGRAM от 0x00H до 0x07H, CGRAM 1 - по адресам от 0x08H до 0x0FH, CGRAM 2 - по адресам от 0x10H до 0x17H, ...вплоть до CGRAM 7 - по адресам от 0x38H до 0x3FH. См. следующий рисунок.

Схема размещения элементов в растровом изображении

	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Значения Байта Двоичное	Десятичное	**Определение пользовательских символов** Пример : Загрузка символа **СТРЕЛКА** в CGRAM 3. Программа должна послать следующий набор байтов в SLCD контроллер
Байт 0						XXX00000	0	[254],[01011000 b].[0],
Байт 1			■			XXX00100	4	[254],[01011001 b].[4],
Байт 2				■		XXX00010	2	[254],[01011010 b].[2],
Байт 3	■	■	■	■	■	XXX11111	31	[254],[01011011 b].[31],
Байт 4				■		XXX00010	2	[254],[01011100 b].[2],
Байт 5			■			XXX00100	4	[254],[01011101 b].[4],
Байт 6						XXX000000	0	[254],[01011110 b].[0],
Байт 7						XXX00000	0	[254],[01011111 b].[0],

Стандартный Набор Команд ЖКИ

От внешнего MCU могут управляться только регистр инструкций (IR) и регистр данных (DR) ЖКИ. Перед началом любой внутренней операции ЖКИ, управляющая информация временно записывается в эти регистры, чтобы иметь возможность работы с различными MCU, которые работают с разной скоростью обмена данными, или разными устройствами управления периферией. Внутренние операции ЖКИ определяются сигналами, посланными внешним MCU. Эти сигналы, которые включают в свой состав сигнал выбора регистра (RS), сигнал чтения/записи (R/W) и сигналы на шине данных (от DB0 до DB7), представляют собой набор инструкций ЖКИ (Таблица 3). Имеются четыре категории инструкций, которые:

- Задают функции ЖКИ, такие как формат дисплея, длина данных и т. п.
- Устанавливают адреса внутреннего ОЗУ (RAM)
- Выполняют обмен данными с внутренним ОЗУ (RAM)
- Выполняют множество других функций

После ознакомления с таблицей вы можете создавать свои собственные команды и затем их протестировать. Ниже приведен краткий список полезных команд, которые наиболее часто используются при работе с ЖКИ.

Команда	**Шестнадца-теричное**	**Десятичное**
Функциональный набор: 8 бит, 1 строка, 5x7 точек	0x30	48
Функциональный набор: 8 бит, 2 строки, 5x7 точек	0x38	56
Функциональный набор: 4 бита, 1 строка, 5x7 точек	0x20	32
Функциональный набор: 4 бита, 2 строки, 5x7 точек	0x28	40
Режим ввода	0x06	6
Дисплей выключен, Курсор выключен (очистка дисплея без очистки содержимого DDRAM)	0x08	8
Дисплей включен, Курсор включен	0x0E	14
Дисплей включен, Курсор выключен	0x0C	12
Дисплей включен, Курсор мигает	0x0F	15
Сдвиг всего дисплея влево	0x18	24
Сдвиг всего дисплея вправо	0x1C	30
Перемещение курсора на один символ влево	0x10	16
Перемещение курсора на один символ вправо	0x14	20
Очистка дисплея (также очищается содержимое DDRAM)	0x01	1
Установка адреса DDRAM или позиции курсора на дисплее	0x80+доп*	128+доп*
Установка адреса CGRAM или установка указателя на положение CGRAM	0x40+ доп**	64+доп**

**** Адреса DDRAM даны для основной секции ЖКИ***
*****Диапазон адресов CGRAM от 0x00 до 0x3F; от 0x00 до 0x07 для символа1 и так далее…***

6.2 Что нужно знать об обмене данными между Arduino и SLCD16x2

Обмен данными между Arduino POP-168 и ЖКИ с последовательным интерфейсом происходит следующим образом :

(1) В текст скетча необходимо включить заголовочный файл библиотеки ***SoftwareSerial.h***, используя команду препроцессора #include
(2) Назначить вывод порта POP-168 командой #define следующим образом:

```
#define rxPin 3   // Назначить Di 3 в качестве вывода приема
                  // последовательных данных или rxPin
#define txPin 2   // Назначить Di 2 в качестве вывода передачи
                  // последовательных данных или txPin
SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);
```

(3) В процедуре **setup()** необходимо установить на передающем выводе высокий логический уровень, подождать некоторое время и установить скорость обмена 9600 бит/сек с помощью команды mySerial.begin(9600). Ниже показан пример кода, реализующего все вышеописанные действия.

```
// Установить высокий уровень на выводе txPin (как рекомендовано)
digitalWrite(txPin, HIGH);
delay(1000); //Подождать некоторое время
// Задать режим работы для выводов tx и rx:
pinMode(rxPin, INPUT);
pinMode(txPin, OUTPUT);
mySerial.begin(9600); // Задать скорость обмена
```

Мы рассмотрели рекомендованные коды настройки обмена данными между Arduino и модулем ЖКИ с последовательным интерфейсом SLCD16x2.

Следующие команды имеют важное значение для обмена данными между модулем SLCD16x2 и POP-BOT:

(1) Команда инициализации :
mySerial.print(0xFE,BYTE);
mySerial.print(Command,BYTE);
(2) Очистить экран ЖКИ :
mySerial.print(0x01,BYTE);
(3) Переместить курсор в левую верхнюю позицию, или ДОМОЙ (HOME position):
mySerial.print(0x80,BYTE);
(4) Переместить курсор в левую позицию нижней строки :
mySerial.print(0xC0,BYTE);
(5) Для отображения сообщения, пользователь должен ввести сообщение, заключенное в кавычки “ ”.
mySerial.print(“Hello”); // Отображает “HELLO”

Задание 3 : Простейшее программирование ЖКИ SLCD16 x 2

A3.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A3-1.
A3.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A3.3 Выполните операцию Сброса POP-BOT и понаблюдайте за работой SLCD16 x 2.

SLCD16x2 покажет следующее сообщение:


```
/*************************************************************************
* POP-BOT V1.0
* Filename : SimpleLCD.pde
* Просмотр сообщения на экране SLCD
*************************************************************************/
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
void setup(){
      digitalWrite(txPin,HIGH);
      delay(1000);
      pinMode(txPin,OUTPUT);
      MySerial.begin(9600);
      delay(1000);
}
void loop(){
      MySerial.print(0xFE,BYTE);
      MySerial.print(0x80,BYTE);
      MySerial.print("POP-BOT");
      MySerial.print(0xFE,BYTE);
      MySerial.print(0xC0,BYTE);
      MySerial.print("Hello World !");
      while(1);
}
/*************************************************************************/
```

Листинг A3-1. Файл SimpleLCD.pde; скетч-файл Arduino для демонстрации простейших приемов работы POP-BOT с ЖКИ

Задание 4 : Управление ЖКИ SLCD16 x 2 с помощью команд

Вы можете многими операциями отображения данных на SLCD16 x 2, такими как выбор строки дисплея, очистка экрана, выбор формата дисплея и т.д., посылая в SLCD16 x 2 команду управления. Для режима Стандартных команд, первым необходимо послать байт 0 x FEH, а вслед за ним команду. Пользователь может выбрать необходимую команду управления SLCD, воспользовавшись информацией, приведенной в этом разделе.
A4. 1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A4-1.
A4.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A4.3 Выполните операцию Сброса POP-BOT и понаблюдайте за работой SLCD16 x 2.

SLCD16 x 2 показывает множество программно задаваемых вариантов отображение сообщения

```
/*************************************************************************
* POP-BOT V1.0
* Filename : SLCDrunningText.pde
* Демонстрирует отображение  Show текста и цифр на экране ЖКИ
* с последовательным интерфейсом
*************************************************************************/
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
void setup(){
      digitalWrite(txPin,HIGH);
      delay(1000);
      pinMode(txPin,OUTPUT);
      MySerial.begin(9600);
      delay(1000);
}
void LCD_CMD(int Command){
      MySerial.print(0xFE,BYTE); // Команда
      MySerial.print(Command,BYTE);
}
void loop(){
      int i;
      LCD_CMD(0x80); // Первая строка
      MySerial.print("POP-BOT");
      LCD_CMD(0xC0); // Вторая строка
      MySerial.print("Hello World !");
      delay(2000);
      LCD_CMD(0x01); // Команда очистки экрана
      LCD_CMD(0x85); // ROW 1,COL 5
      MySerial.print("From");
      delay(500);
      LCD_CMD(0x07); // Сдвиг текста влево
      for(i=0;i<9;i++){
            MySerial.print(" ");
            delay(200);
      }
```

```
   LCD_CMD(0x05); // Сдвиг текста вправо
      for(i=0;i<9;i++){
            MySerial.print(" ");
            delay(200);
      }
      for(i=0;i<9;i++){ // Мигание текста
            LCD_CMD(0x08);
            delay(200);
            LCD_CMD(0x0C);
            delay(200);
      }
      LCD_CMD(0x00); // Отображение текста
      MySerial.print("Innovative"); // Строка 1
      LCD_CMD(0xC0);
      MySerial.print("Experiment"); // Строка 2
      delay(5000);
      i=0;
      // Show Number
      LCD_CMD(0x01); // Очистка экрана
      MySerial.print("Counter"); // Строка 1
      while(1){
            i++;
            LCD_CMD(0xC5); // Строка 2
            MySerial.print(i,DEC);
            delay(100);
      }
}
```

Листинг A4-1 : Файл SLCDrunningText.pde; скетч-файл Arduino для демонстрации дополнительных функций ЖКИ с последовательным интерфейсом

Описание программы

Часть 1 Инициализация модуля обмена данными микроконтроллера и SLCD16x2

Часть 2 Выбор строки дисплея, в которой будет выводиться текст. Верхняя строка (0x80) выбрана для отображения сообщения **POP-BOT**. Нижняя строка (0xC0) выбрана для отображения сообщения **Hello World !**.

Часть 3 Отправляется команда Очистки экрана (0x01) и задается пятая позиция в первой строке ЖКИ (0x85) в качестве начальной позиции для вывода **From**.

Часть 4 Отправляется команда сдвига влево (0x07) и организуется бесконечный цикл сдвига сообщения **From** в левом направлении.

Часть 5 Отправляется команда сдвига вправо (0x05) и организуется бесконечный цикл сдвига сообщения **From** обратно к начальному положению.

Часть 6 Цикл для последовательной отправки команды Гашения дисплея (0x08) и команды Включения дисплея (0x0C) и наоборот. Это приводит к миганию сообщения **From**.

Часть 7 Операция напоминает реализованную в Части 2, но с измененными сообщениями: в верхней строке - **Innovative**, а в нижней строке - **Experiment**

7 : Движение POP-BOT вдоль линии

Следование линии, или наблюдение за линией является популярным заданием при изучении робототехники. Целью выполнения такого задания является изучение работы с аналоговыми датчиками. В наборе по изучению робототехники POP-BOT имеется пара инфракрасных отражательных датчиков для выполнения такого рода задания. Два отражательных ИК-датчика будут установлены в нижней части POP-BOT таким образом, что они смогут обнаруживать как белые, так и черные линии.

7.1 ZX-03 : Инфракрасный отражательный датчик

Сердцем этого датчика является оптопара с открытым оптическим каналом TCRT5000. Датчик создан для обнаружения близких объектов с помощью инфракрасного (ИК) излучения. Позади прозрачного синего окошка датчика расположен инфракрасный излучающий диод, а позади черного окошка расположен инфракрасный фототранзистор. Когда инфракрасное излучение, испущенное светодиодом, отражается от поверхности и возвращается обратно, попадая на черное окошко, оно воздействует на базу инфракрасного фототранзистора, заставляя его проводить электрический ток. Чем более сильный поток ИК-излучения падает на базу фототранзистора, тем больший ток течет через него. На рисунке 7-1 показана работа датчика ZX-03.

При использовании в режиме аналогового датчика, ZX-03 может обнаруживать оттенки серого на бумаге и расстояние до близкорасположенных объектов, если освещенность в комнате остается постоянной.

Рабочее расстояние от датчика до линии, нарисованной на полу или листе бумаги, должно составлять от 3 до 8 мм. Напряжение на выходе датчика может меняться от 0.1 до 4.8В, а цифровое значение, полученное после преобразования в АЦП в диапазоне от 20 до 1 000. Таким образом, ZX-03 будет вполне подходящим датчиком для использования в качестве датчика отслеживания линий line tracking sensor.

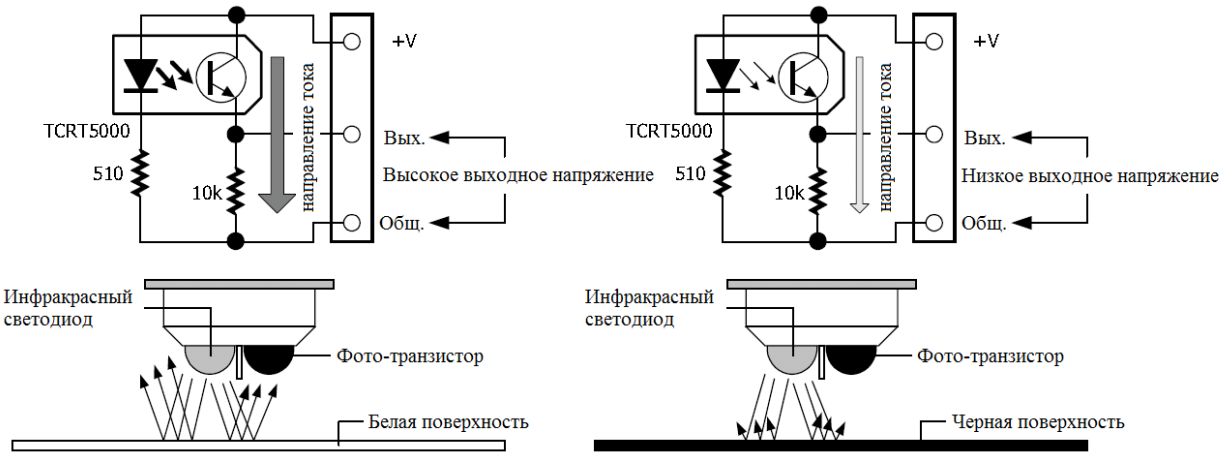


Рисунок 7-1. Работа инфракрасного отражательного датчика ZX-03 над белой и черной поверхностью

7.2 Подготовка к выполнению задания по отслеживанию линий

7.2.1 Подготовка компонент демонстрационного поля

Во всех заданиях, описанных в этом разделе, используется “самодельное демонстрационное поле“. Оно состоит из белой поверхности с нанесенной на нее черной линией, и черной поверхности с нанесенной на нее белой линией. Вы должны создать собственное демонстрационное поле (не включенное в данный набор), используя приведенное ниже описание:

1. Полипропиленовая пластина (PP пластина) белого и черного цвета. Размер 90 x 60 см. Однако размер можно изменить в зависимости от целей и ресурсов.
2. Черная и белая изолента шириной 1 дюйм (2.5 см), по 2 рулона каждого цвета. Рекомендуется изолента марки 3M.
3. Ножницы или резак.

7.2.2 Установка порогового значения для выполнения задания по отслеживанию линий с использованием функции analogRead()

Робот POP-BOT может обнаруживать различия между линией и поверхностью, считывая показания с инфракрасного отражательного датчика через аналоговый входной порт. Для чтения величины сигнала на аналоговом порту POP-BOT использует функцию **analogRead()** предоставляемую средой разработки Arduino.

POP-BOT считывает показания датчика, расположенного над черной линией или черной поверхностью, при маленьком значении измеряемой величины (меньше, чем 400, при минимальном значении равном 0) и считывает показания датчика, расположенного над белой линией или белой поверхностью, при большом значении измеряемой величины (больше, чем 500, при максимальном значении 1023). Пороговое значение для принятия решения о том, является ли цвет линии или поверхности черным или белым, можно вычислить, как полусумму ответа от белой и черной поверхностей следующим образом:

Пороговое значение = (Значение над белой поверхностью + значение над черной поверхностью) / 2

В Задании 5 детально описано вычисление порогового значения для задачи слежения за линиями.

Задание 5 : Обнаружение белых и черных участков поверхности

Для выполнения задания POP-BOT необходимо наличие двух модулей инфракрасных отражательных датчиков, которые уже прикреплены с нижней стороны основания робота. Таким образом, для решения задачи необходимы только операции, связанные с программированием.

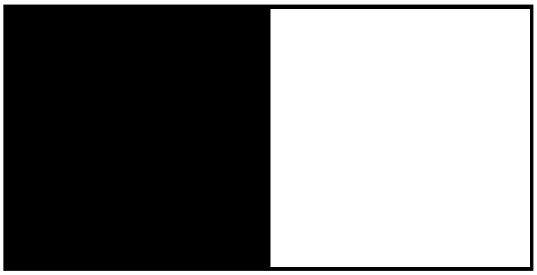
Прежде чем разрабатывать робот, способный отслеживать линии, конструктору необходимо запрограммировать робот таким образом, чтобы он мог обнаруживать различие между белой и черной поверхностью.

A5.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A5-1.
A5.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A5.3 Отсоедините загрузочный кабель.

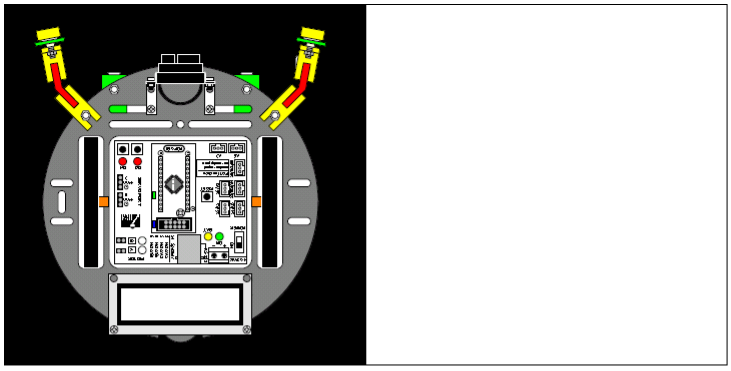
```
/*****************************************************************************
* POP-BOT V1.0
* Filename : AnalogRead.pde
* Чтение аналогового сигнала с инфракрасного отражательного датчика и
отображение
* в цифровом виде на экране ЖКИ SLCD
*****************************************************************************/
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int LeftSensor;
int RightSensor;
void setup(){
      digitalWrite(txPin,HIGH);
      pinMode(txPin,OUTPUT);
      MySerial.begin(9600);
      delay(1000);
}
void LCD_CMD(int Command){
      MySerial.print(0xFE,BYTE); // Команда
      MySerial.print(Command,BYTE);
}
void loop(){
      LeftSensor = analogRead(7); // Чтение значения с левого датчика
      RightSensor = analogRead(6); // Чтение значения с правого датчика
      LCD_CMD(0x80); // Установка курсора дисплея в начало первой строки
      MySerial.print("L Sensor= "); // Отображение показаний левого датчика в
      1-й строке
      LCD_CMD(0x8A);
      MySerial.print(LeftSensor,DEC);
      LCD_CMD(0xC0); // Установка курсора дисплея в начало второй строки
      MySerial.print("R Sensor= "); // Отображение показаний правого датчика
      во 2-й строке
      LCD_CMD(0xCA);
      MySerial.print(RightSensor,DEC);
      delay(200);
}
/*****************************************************************************/
```

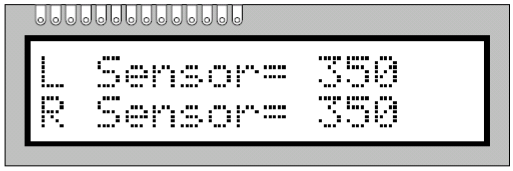
Листинг A5-1 : Файл AnalogRead.pde; скетч-файл Arduino для чтения состояния инфракрасного датчика, чтобы отобразить его на экране ЖКИ с последовательным интерфейсом POP-BOT

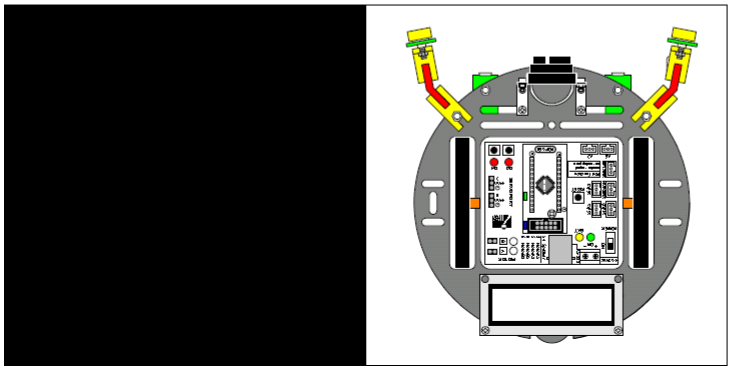
A5.4 Создайте тестовый лист с белым и черным полем, как показано на рисунке. Белое поле должно иметь размер 30 x 30 см, и черное поле должно иметь такой же размер - 30 x 30см (рекомендуемый).

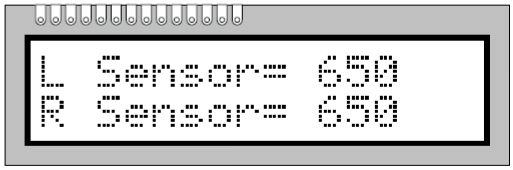


A5.5 Поместите POP-BOT, который уже был должным образом запрограммирован на шаге A5.3 над белой поверхностью тестового листа. Включите робота. Прочитайте значение на экране SLCD и запишите его. После этого поместите робота над черной поверхностью, прочитайте появившееся значение и также запишите его.









В результате получится следующее :

Значение над белой поверхностью будет лежать в диапазоне между 500 и 950

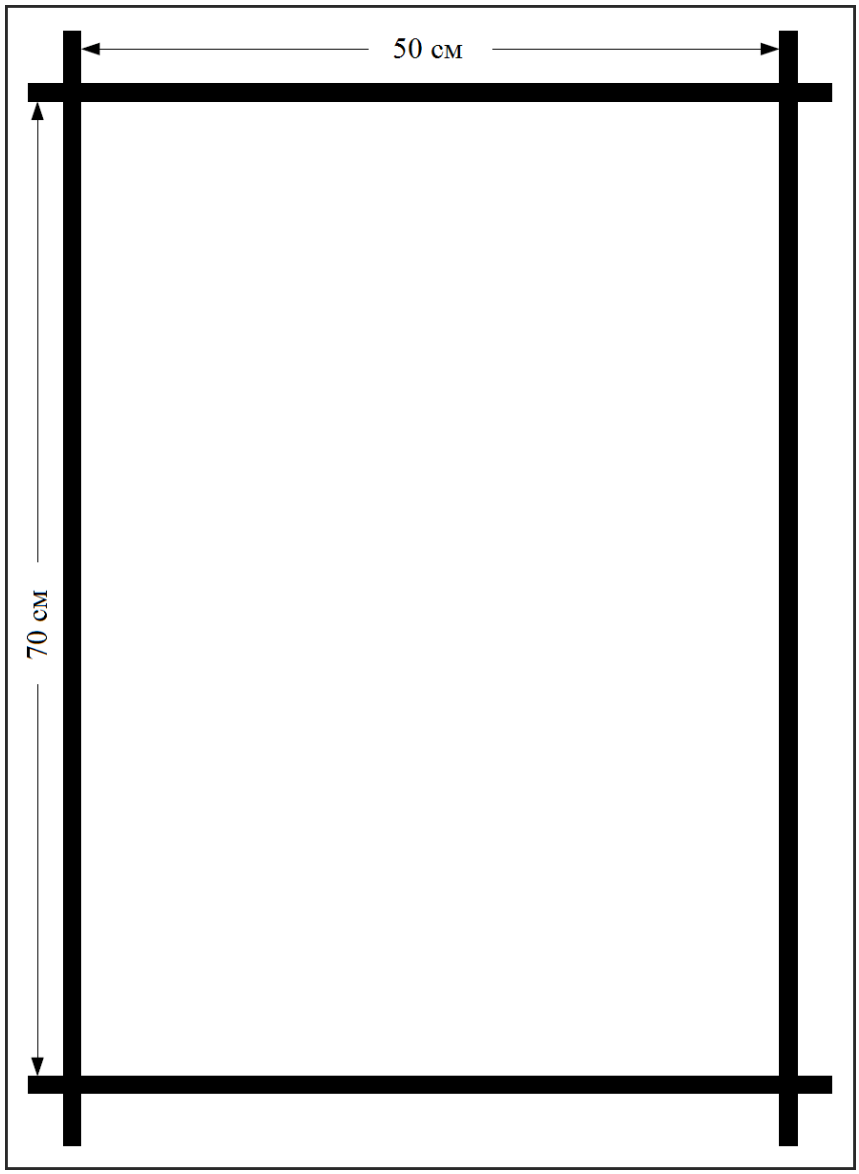
Значение над черной поверхность будет лежать в диапазоне между 100 и 400

Примерное пороговое значение для обнаружения линии будет (650+350) /2 = 500.

Задание 6 : Движение POP-BOT в пределах границы

После того, как стали известны значения показаний датчика над белой и черной поверхностью, в следующем задании будет затронут вопрос перемещения POP-BOT в пределах границы, заданной черной линией.

A6.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A6-1.
A6.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A6.3 Отсоедините загрузочный кабель.
A6.4 Создайте границу, подобную показанной на рисунке ниже. Белая поверхность будет иметь размер 90 x 60 см, а черные линии ширину 1 дюйм (2.5 см)



```
/*************************************************************************
* POP-BOT V1.0
* Filename : BlackBorderMove.pde
* POP-BOT движется на черной линией границы
*************************************************************************/
int Ref=500;
int Left,Right;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
}
void loop(){
      Left = analogRead(7); // Чтение значения с левого датчика ZX-03
      Right = analogRead(6); // Чтение значения с правого датчика ZX-03
      if (Left>Ref && Right>Ref){ // Оба датчика обнаружили белую поверхность
            Forward(150);
      }
      else if (Left<Ref && Right<Ref){ // Оба датчика обнаружили черную линию
            Backward(150);delay(100);
      }
      else if (Left<Ref) { // Только левый датчик обнаружил черную линию
            Backward(150);delay(300);
            Spin_Right(150);delay(500);
      }
      else if (Right<Ref){ // Только правый датчик обнаружил черную линию
            Backward(150);delay(300);
            Spin_Left(150);delay(400);
      }
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Backward(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*************************************************************************/
```

Листинг А6-1 : Файл BlackBorderMove.pde; скетч-файл Arduino для демонстрации движениея POP-BOT по замкнутому черному контуру

А6.5 Положите POP-BOT внутри контура, образованного черной линией границы. Включите робот. Наблюдайте за перемещением робота.

POP-BOT движется вперед по белой поверхности, до тех пор, пока один из датчиков не обнаружит черную границу. Дальнейшее поведение робота:

***Если оба сенсора обнаружили черную полосу** : POP-BOT в течение короткого промежутка времени двигается назад, затем снова двигается вперед.*

***Если черную полосу обнаружил только левый датчик** : POP-BOT в течение короткого промежутка времени двигается назад, затем поворачивает направо, затем снова двигается вперед.*

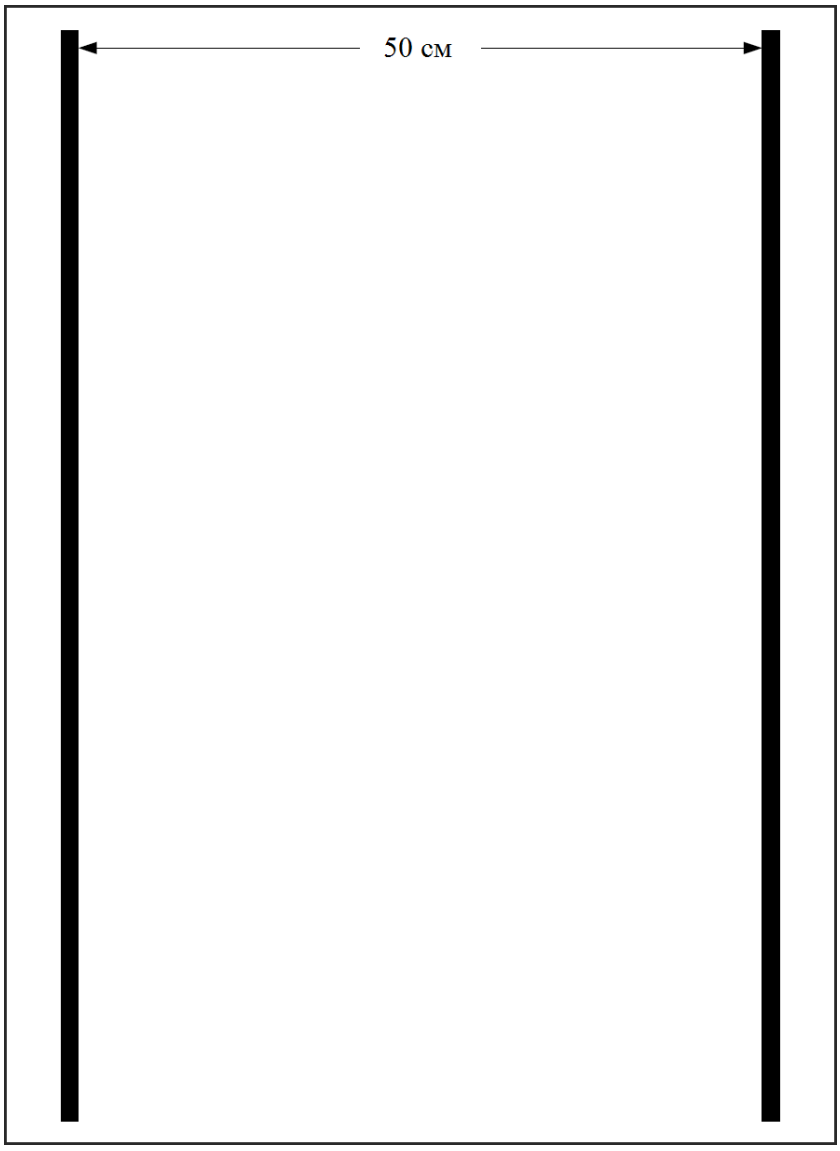
***Если черную полосу обнаружил только правый датчик** : POP-BOT в течение короткого промежутка времени двигается назад, затем поворачивает налево, затем снова двигается вперед.*

Окончательно POP-BOT будет непрерывно перемещаться в пределах черной границы.

Задание 7 : Движение POP-BOT между двумя параллельными линиями (наподобие шарика от пинг-понга)

При выполнении этого задания будет показано, как POP-BOT движется в форме зигзага. Черные линии будут поворотными точками. Робот POP-BOT движется вперед до тех пор, пока не обнаружит черную линию, чтобы изменить направление движения. Эта операция всегда повторяется. Таким образом, робот будет двигаться внутри пространства, ограниченного черными линиями.

A7.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A7-1.
A7.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робота.
A7.3 Отсоедините загрузочный кабель.
A7.4 Создайте поле для пинг-понга, следуя приведенному ниже рисунку. Белая поверхность имеет размер 90 x 60 см, а черная линия имеет ширину 1 дюйм (2.5 см).



```
/*************************************************************************
* POP-BOTV1.0
* Filename : PingPong.pde
* POP-BOT движется зигзагообразно между двумя черными линиями
*************************************************************************/
int Ref=500;
int Left,Right;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
}
void loop(){
      Left = analogRead(7); // Чтение значения с левого датчика ZX-03
      Right = analogRead(6); // Чтение значения с правого датчика ZX-03
      if (Left>Ref && Right>Ref){ // Оба датчика обнаружили белую поверхность
            Forward(150);
      }
      else if (Left<Ref && Right<Ref){ // Оба датчика обнаружили черную линию
            Backward(150);delay(100);
      }
      else if (Left<Ref) { // Только левый датчик обнаружил черную линию
            Spin_Right(150);
            delay(420);
      }
      else if (Right<Ref){ // Только правый датчик обнаружил черную линию
            Spin_Left(150);
            delay(420);
      }
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Backward(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*************************************************************************/
```

Листинг А7-1 : Файл PingPong.pde; скетч-файл Arduino для демонстрации движения POP-BOT между двумя стенами (пинг-понг)

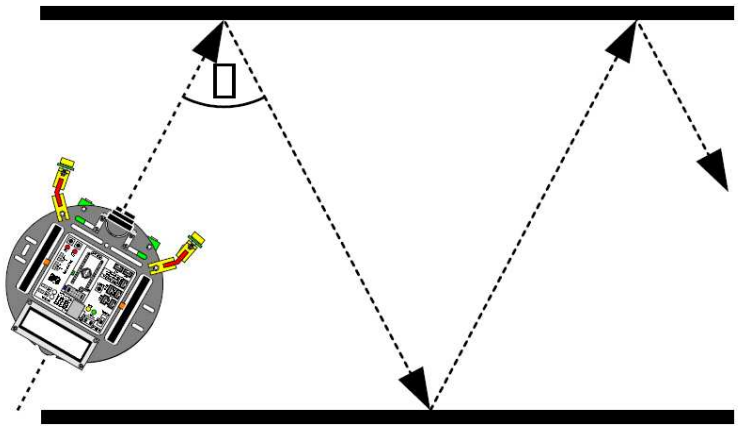
Концепция программирования

Программирование в данном задании можно разделить на следующие шаги:

(1) Обнаружение разницы значений между белой поверхностью и черной линией.
(2) Чтение значения датчика и запоминание для последующего сравнения.
(3) Проверка выполнения одного из следующих условий:
Случай #1 : Оба датчика обнаруживают белую поверхность.
Действие *: Робот движется вперед.*
Случай #2 : Только левый датчик обнаруживает черную линию.
Действие*: Робот поворачивает вправо, чтобы изменить направление движения для того, чтобы двигаться в сторону, противоположную черной линии.*
Случай #3 : Только правый датчик обнаруживает черную линию.
Действие*: Робот поворачивает влево, чтобы изменить направление движения для того, чтобы двигаться в сторону, противоположную черной линии.*
Случай #4 *:* Не выполняется ни одно из приведенных выше условий
Действие*: Зарезервировано для нужд программиста.*
(4) Назад к шагу (2).

A7.5 Положите POP-BOT на поле для Пинг-Понга. Включите робот. Наблюдайте за движением робота.

В этой программе время поворота задано равным примерно 150 миллисекундам. Это вызывает поворот на определенный угол (см. приведенную ниже иллюстрацию). Программист может изменить значение этого времени, чтобы получить заданный угол для правильного управления направлением движения.



Если это значение будет больше, то угол будет более острым. Это может привести к тому, что робот будет возвращаться обратно по тому же самому пути. С другой стороны, если значение будет меньше, то робот может проскочить линию, или даже будет двигаться параллельно линии, не замечая ее.

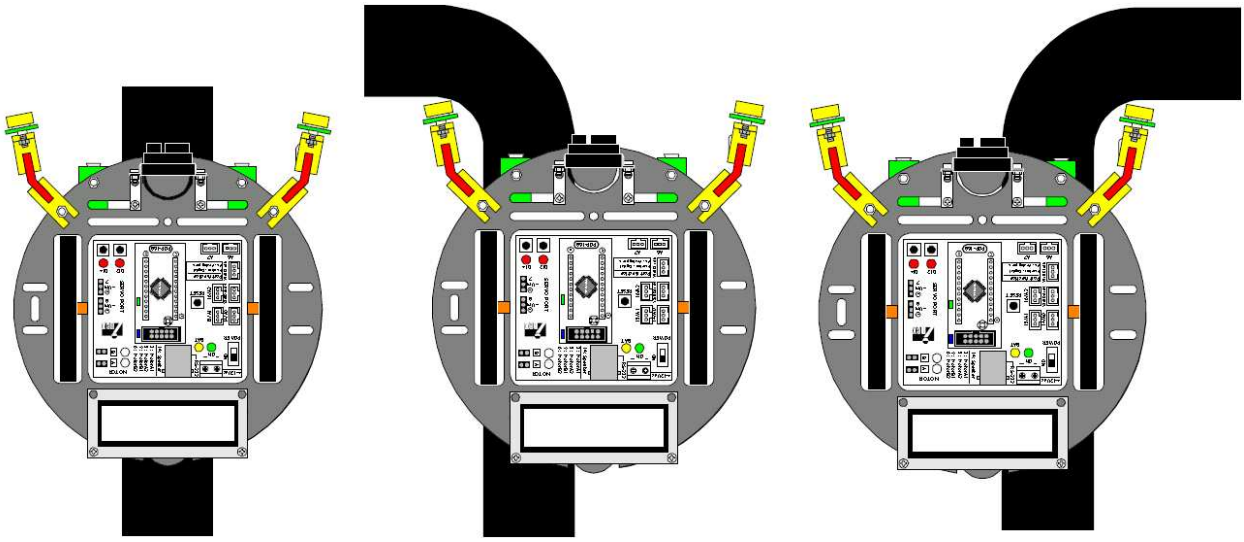
Задание 8 : Движение робота вдоль черной линии

Движение робота вдоль линии может проходить по трем различным сценариям.

(1) Оба датчика на выходе имеют значения, которые соответствуют белому цвету : робот будет двигаться вперед. Таким образом, данная программа написана так, чтобы в нормальном режиме робот постоянно двигался вперед.

(2) Левый датчик обнаруживает черную линию : эта ситуация возникает, когда робот медленно отклоняется вправо. Таким образом, данная программа написана так, чтобы робот возвращался обратно влево, для продолжения своего нормального движения.

(3) Правый датчик обнаруживает черную линию : эта ситуация возникает, когда робот медленно отклоняется влево. Таким образом, данная программа написана так, чтобы робот возвращался обратно вправо, для продолжения своего нормального движения.



Сценарий #1
Оба датчика обнаружили белую поверхность

Сценарий #2
Левый датчик обнаружил черную линию

Сценарий #3
Правый датчик обнаружил черную линию

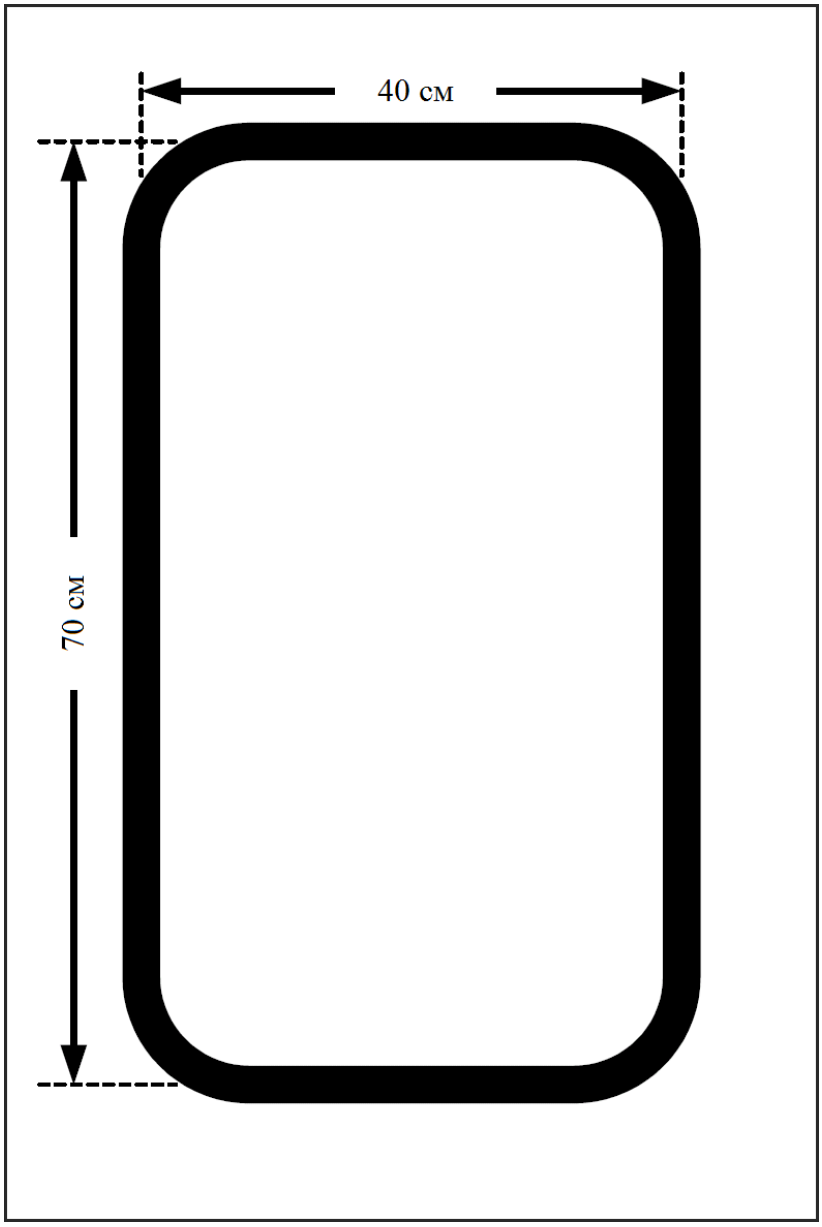
Для каждого из рассмотренных сценариев, можно создать подпрограмму на языке C, как это показано на Листинге A8-1

A8.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A8-1.
A8.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A8.3 Отсоедините загрузочный кабель.

```
/*************************************************************************
* POP-BOT V1.0
* Filename : SimpleLineTracking.pde
* POP-BOT движется по черной линии
*************************************************************************/
int Ref=500;
int Left,Right;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
}
void loop(){
      Left = analogRead(7); // Чтение значения с левого датчика ZX-03
      Right = analogRead(6); // Чтение значения с правого датчика ZX-03
      if (Left>Ref && Right>Ref){ // Оба датчика обнаружили белую поверхность
            Forward(150);
      }
      else if (Left<Ref) { // Только левый датчик обнаружил черную линию
            Spin_Left(250);
      }
      else if (Right<Ref){ // Только правый датчик обнаружил черную линию
            Spin_Right(250);
      }
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}/*************************************************************************/
```

Листинг A8-1 : Файл SimpleLineTracking.pde; скетч-файл Arduino для управления движением POP-BOT вдоль черной линии

A8.4 Создайте простейшее белое поле с черной линией, как это показано на иллюстрации ниже. Белая поверхность имеет размер 90 x 60 см, а черная линия имеет ширину 1 дюйм (2.5 см)



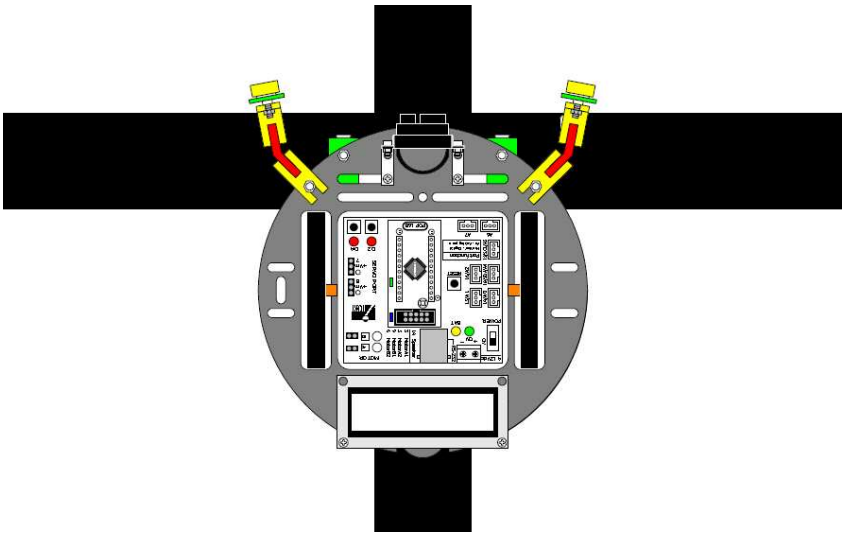
A8.5 Положите POP-BOT в области черной линии (или над ней). Включите робот. Наблюдайте за движением робота.

Робот POP-BOT будет двигаться вдоль черной линии. Возможно, что робот убежит с черной линии. В таком случае можно увеличить точность отслеживания линии, отредактировав программу путем изменения порогового значения для датчика и подкорректировав положение инфракрасных отражательных датчиков на корпусе робота.

Задание 9 : Обнаружение пересечения линий

Уяснив сущность задания 8, можно несколько улучшить поведение робота POP-BOT таким образом, что он будет двигаться вдоль черной линии и обнаруживать пересечение линий, используя те же самые 2 датчика. Все, что необходимо для этого сделать – это немного отредактировать код программы из предыдущего задания.

Когда робот подходит к черной линии, которая образует Т-пересечение с линией, по которой робот движется в настоящий момент, оба датчика зафиксируют появление черной линии. Для отработки этого сценария в главную программу необходимо добавить соответствующую функцию. Улучшенная программа на языке С показана на Листинге А9-1.

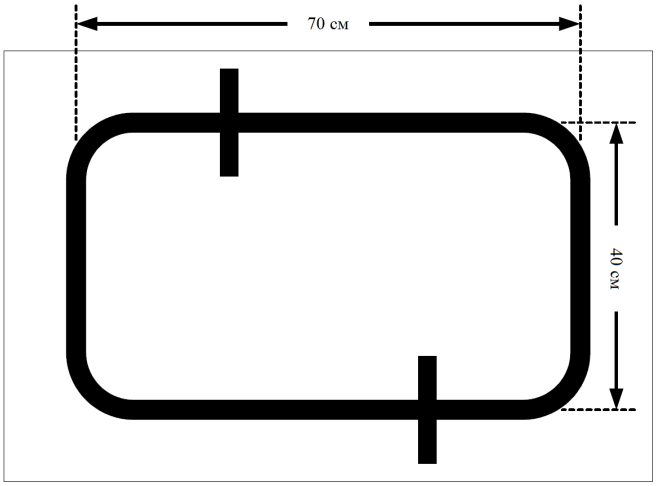


```
/*****************************************************************************
* Filename : CrossingLineDetect.pde
* POP-BOT tracks the black line and beep when detect the crossing line
*****************************************************************************/
int Ref=700;
int Left,Right;
int Cnt=0,j;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(14,OUTPUT); // Пьезоизлучатель
}
void loop(){
      Left = analogRead(7); // Чтение значения с левого датчика ZX-03
      Right = analogRead(6); // Чтение значения с правого датчика ZX-03
      if (Left>Ref && Right>Ref){ // Оба датчика обнаружили белую поверхность
            Forward(150);
      }
      else if (Left<Ref && Right<Ref){ // Оба датчика обнаружили черную линию
      // It's mean crossing line.
            Cnt++;
            Motor_Stop();
            for (j=0;j<Cnt;j++){
```

```
                  Beep();
                  delay(100);
            }
            Forward(150);
            delay(100);
      }
      else if (Left<Ref) { // Только левый датчик обнаружил черную линию
            Spin_Left(255);
      }
      else if (Right<Ref){ // Только правый датчик обнаружил черную линию
            Spin_Right(255);
      }
}
void Beep(){ // Beep routine
      int i;
      for (i=0;i<600;i++){
            digitalWrite(14,HIGH);
            delayMicroseconds(150);
            digitalWrite(14,LOW);
            delayMicroseconds(150);
      }
}
void sound(int freq ,int duration){
      unsigned long us;
      int duration_,i;
      us=(1000000/(freq*2));
      duration_ = (duration/(us*2));
      for (i=0;i<duration_;i++){
            digitalWrite(14,HIGH);
            delayMicroseconds(us);
            digitalWrite(14,LOW);
            delayMicroseconds(us);
      }
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Motor_Stop(){
      digitalWrite(3,LOW);
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*********************************************************************/
```

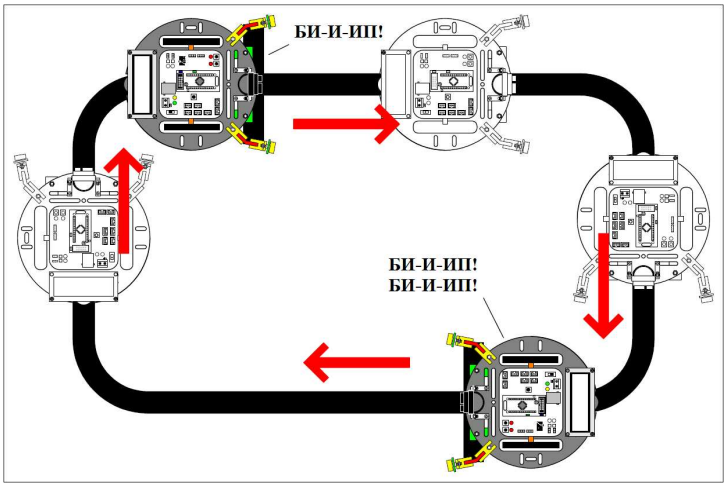
Листинг А9-1 : Файл CrossingLineDetect.pde; скетч-файл Arduino для управления движением POP-BOT вдоль черной линии и обнаружение пересечения черных линий

A9.1 Улучшим поле с простой черной линией из Задания 8. Добавим несколько линий пересекающих основную линию. Добавим столько пересечений, сколько нам понравится. Однако, убедитесь, что расстояния между соседними пересечениями больше, чем 2 ширины робота.



A9.2 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A9-1.
A9.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A9.3 Отсоедините загрузочный кабель.
A9.4 Поместите робота на демонстрационное поле. Включите питание робота. Наблюдайте за его движениями.

Робот будет двигаться вдоль черной линии. Когда робот обнаружит пересечение, он притормозит и подаст одиночный звуковой сигнал. Когда робот обнаружит второе пересечение, он издаст звуковой сигнал дважды, и количество звуковых сигналов будет увеличиваться на единицу при обнаружении каждого нового пересечения.

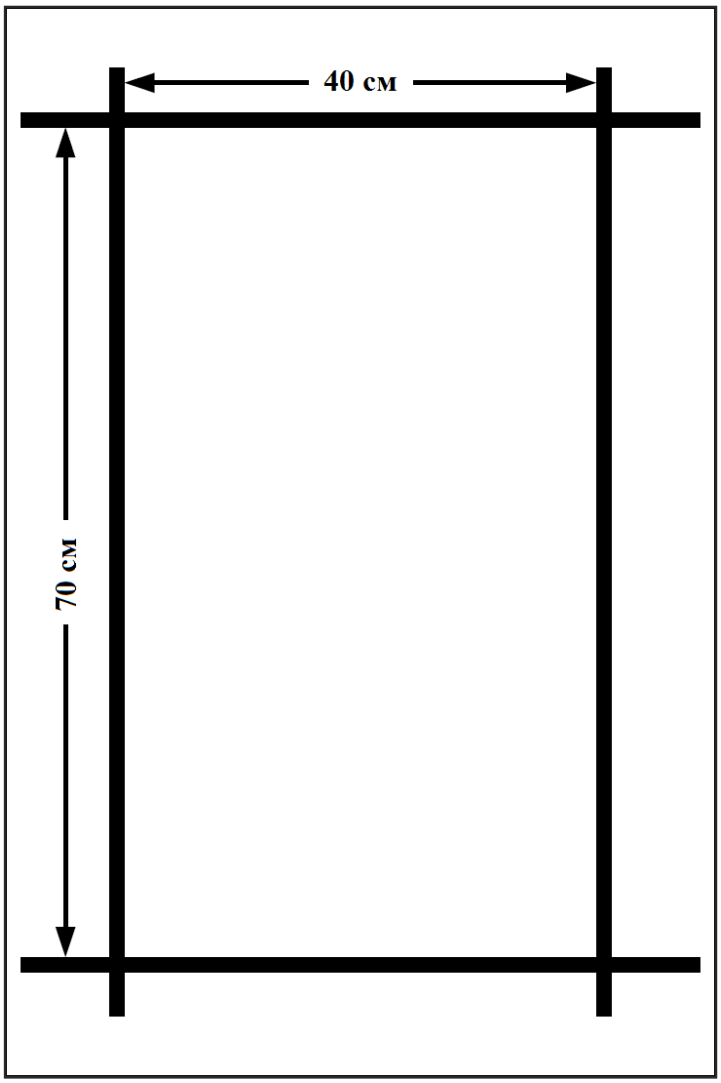


Замечание : При выполнении операции торможения электродвигателя, робот будет останавливаться и немедленно тормозить вал электродвигателя. Но иногда этого недостаточно. Необходимо запрограммировать робота, чтобы он двигался назад в течение короткого промежутка времени. Это вызовет надежную остановку робота в необходимом положении.

Задание 10 : Движение POP-BOT вдоль линий, пересекающихся под углом 90^0

В этом задании будет рассмотрено, как организовать поворот робота на 90 градусов при обнаружении точки соединения или пересечения. Эта технология является очень важной в реальных задачах автоматизации. Для многих задач отслеживания линий характерно наличие множества точек пересечения и наложения. Робот должен обнаружить пересечение и переместиться достаточно точно для того, чтобы его движение оставалось стабильным.

A10.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A10-1
A10.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A10.3 Отсоедините загрузочный кабель.
A10.4 A8.4 Создайте простейшее белое поле с черной линией, как это показано на иллюстрации ниже. Белая поверхность имеет размер 90 x 60 см, а черная линия имеет ширину 1 дюйм (2.5 см)

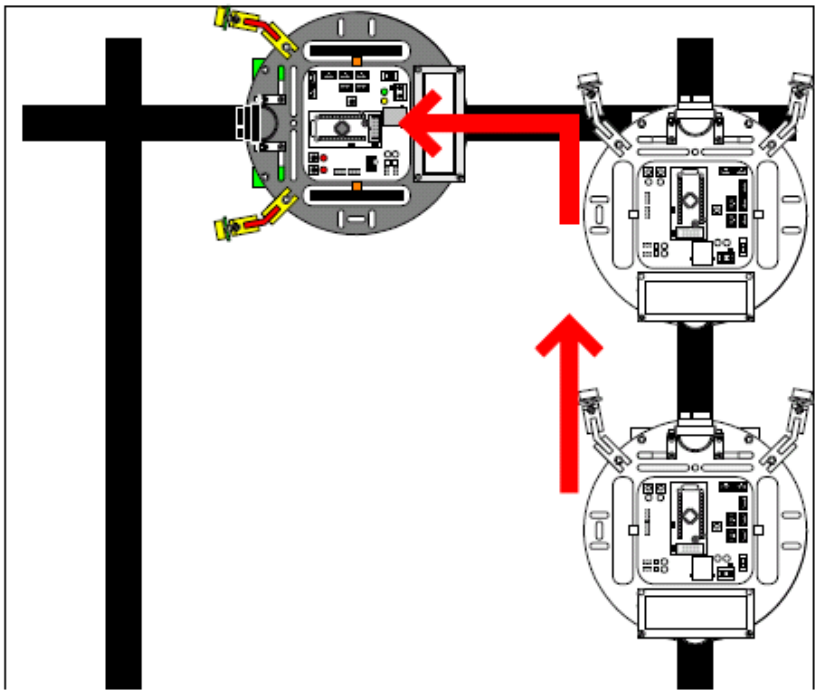


```
/*************************************************************************
* POP-BOT V1.0
* Filename : RightTurnLineTracking.pde
* POP-BOT движется вдоль линии и поворачивает вправо на 90 градусов,
* когда обнаруживает пересечение линий
*************************************************************************/
int Ref=700;
int Left,Right;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(14,OUTPUT); // Пьезоизлучатель
}
void loop(){
      Left = analogRead(7); // Чтение значения с левого датчика ZX-03
      Right = analogRead(6); // Чтение значения с правого датчика ZX-03
      if ((Left<Ref) && (Right<Ref)){ // Обнаружение пересечения линий
            Right90();
      }
      else if ((Left>Ref) && (Right>Ref)){ // Над линией
            Forward(150);
      }
      else if (Left<Ref) { // Только левый датчик обнаружил черную линию
            Spin_Left(150);
      }
      else if (Right<Ref){ // Только правый датчик обнаружил черную линию
            Spin_Right(150);
      }
}
/*Turn right 90 degree function */
void Right90(){
      Forward(150);
      delay(50);
      Spin_Right(200);
      delay(100);
      while(analogRead(6)>Ref);
      delay(50);
}
/*Movement function*/
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*************************************************************************/
```

Листинг A10-1 : Файл RightTurnLineTracking.pde; скетч-файл Arduino для управления движением POP-BOT черной линии и обнаружения пересечения линий

A10.5 Поместите POP-BOT над линией. Включите робота и наблюдайте за его движением.

POP-BOT движется вдоль линии. Каждый раз, когда робот обнаруживает пересечение линий, он поворачивает вправо на угол 90 градусов и продолжает следовать линии.



Наиболее важной в данном задании является функция **Right90**. Она написана для Arduino на языке программирования C/C++. Мы можем описать работу этой функции следующим образом:

1. После того, как датчик робота обнаруживает пересечение линий, POP-BOT должен двигаться вперед, 0.05 секунды для установки положения робота в центре пересекающихся линий.
2. Повернуть вправо и подождать 0.1 секунды
3. Повторять чтение значения на выходе правого датчика до тех пор, пока он не обнаружит черную линию.
4. Подождать 0.05 перед возвратом в главный цикл.

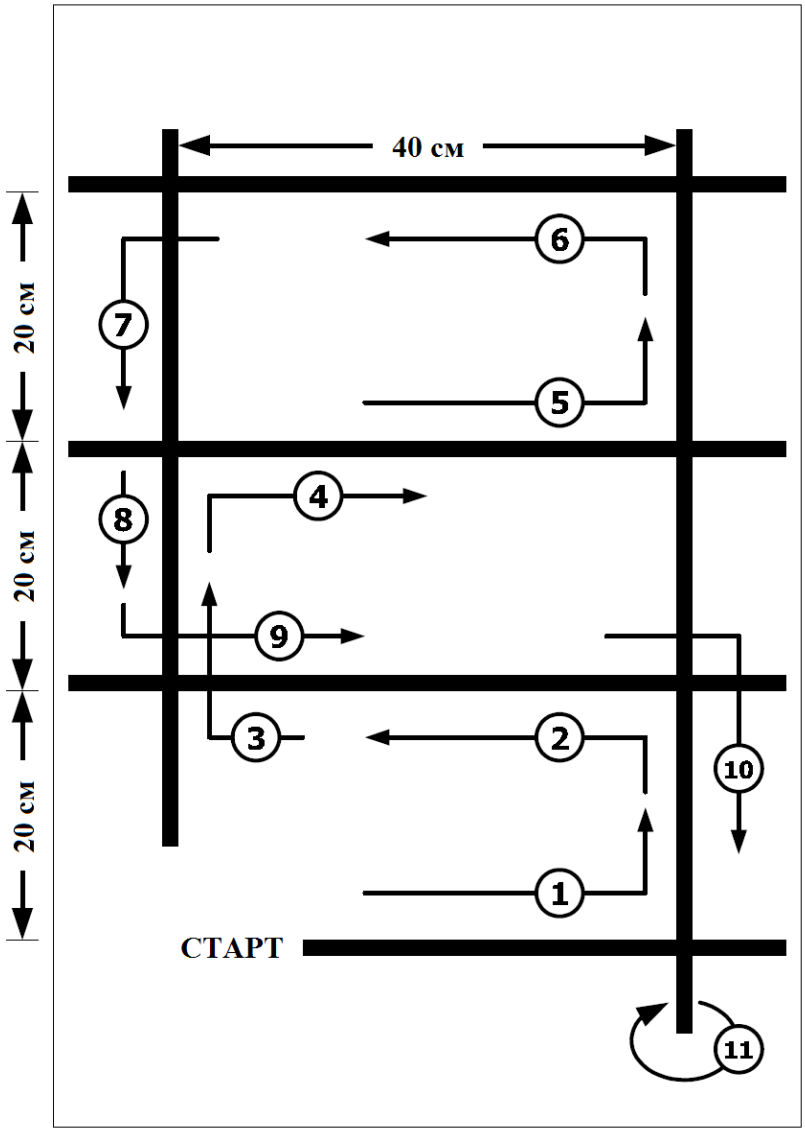
Ниже приведен исходный код этой функции.

```
/*Функция поворота вправо на 90 градусов */
void Right90(){
      Forward(150);
      delay(50);
      Spin_Right(200);
      delay(100);
      while(analogRead(6)>Ref);
      delay(50);
}
```

Задание 11 : Движение по участку с большим количеством пересекающихся линий

В этом задании рассматривается одна из наиболее популярных во многих робототехнических играх миссий - отслеживание линии. Демонстрационное поле будет содержать множество пересекающихся линий. Робот должен двигаться, следуя линии, обнаруживать каждую точку пересечения или наложения и принимать решение двигаться ли дальше вперед или назад, либо поворачивать вправо или влево.

Используя фрагменты кода из Заданий с 8 по 10, следует объединить их все вместе, чтобы создать законченный код для решения поставленной задачи движения на сильно пересеченной местности. Ниже приведен пример расположения линий на поле и движения по ним.



A11.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A11-1
A11.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A11.3 Отсоедините загрузочный кабель.

```
/*************************************************************************
* POP-BOT V1.0
* Filename : MultiCrossingLineDetect.pde
* POP-BOT движение вдоль линии и проверка наличия пересечения линий
* для реализации сложного алгоритма движения
*************************************************************************/
int Ref=700;
int Left,Right;
int Cnt=0;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(14,OUTPUT); // Пьезоизлучатель
}
void loop(){
      while(Cnt<11){
            Left = analogRead(7); // Чтение значения с левого датчика ZX-03
            Right = analogRead(6); // Чтение значения с правого датчика ZX-03
            if ((Left<Ref) && (Right<Ref)){ // Обнаружена точка пересечения
                  Cross();
            }
            else if ((Left>Ref) && (Right>Ref)){// Кратковременное движение
      вдоль линии
                  Forward(150);
            }
            else if (Left<Ref) { // Только левый датчик обнаружил черную линию
                  Spin_Left(150); // Кратковременный поворот влево
            }
            else if (Right<Ref){    // Только правый датчик обнаружил черную
      линию
                  Spin_Right(150);  // Кратковременный поворот вправо
            }
      }
      Forward(200);
      delay(200);
      Spin_Left(200); // Поворот вокруг
      delay(2000);
      Motor_Stop();
      Beep();
      while(1);
}
void Cross(){
      Cnt++;
      if (Cnt==11){
            Motor_Stop();
      }
      else if (Cnt==8){ // Проверка для движения вперед
            Forward(200);
            delay(300);
      }
      else if(Cnt==3 || Cnt==4 || Cnt==10 ){ // Проверка для поворота вправо
на 90 градусов
            Right90();
      }
      else{ // else Turn Left
            Left90();
      }
}
```

```
   /*Функция поворота вправо на 90 градусов*/
void Right90(){
      Forward(150);
      delay(50);
      Spin_Right(200);
      delay(100);
      while(analogRead(6)>Ref);
      delay(50);
}
/* Функция поворота влево на 90 градусов */
void Left90(){
      Forward(150);
      delay(50);
      Spin_Left(200);
      delay(100);
      while(analogRead(7)>Ref);
      delay(50);
}
void Beep(){
      int i;
      for (i=0;i<600;i++){
            digitalWrite(14,HIGH);
            delayMicroseconds(150);
            digitalWrite(14,LOW);
            delayMicroseconds(150);
      }
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Motor_Stop(){
      digitalWrite(3,LOW);
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(9,LOW);
}
/******************************************************************************/
```

Листинг A11-1 : Файл MultiCrossingLineDetect.pde; скетч-файл Arduino для управления движением POP-BOT вдоль черной линии и обнаружения пересечения линий для выполнения миссии движения по пересеченной местности

A11.4 Разместите POP-BOT в точке СТАРТ (START) (см. на иллюстрацию демонстрационного поля). Включите робота и наблюдайте за его движением.

POP-BOT движется вдоль линии, следуя маршруту, показанному на иллюстрации демонстрационного поля. POP-BOT будет выполнять 3 сценария при обнаружении пересечения линий:

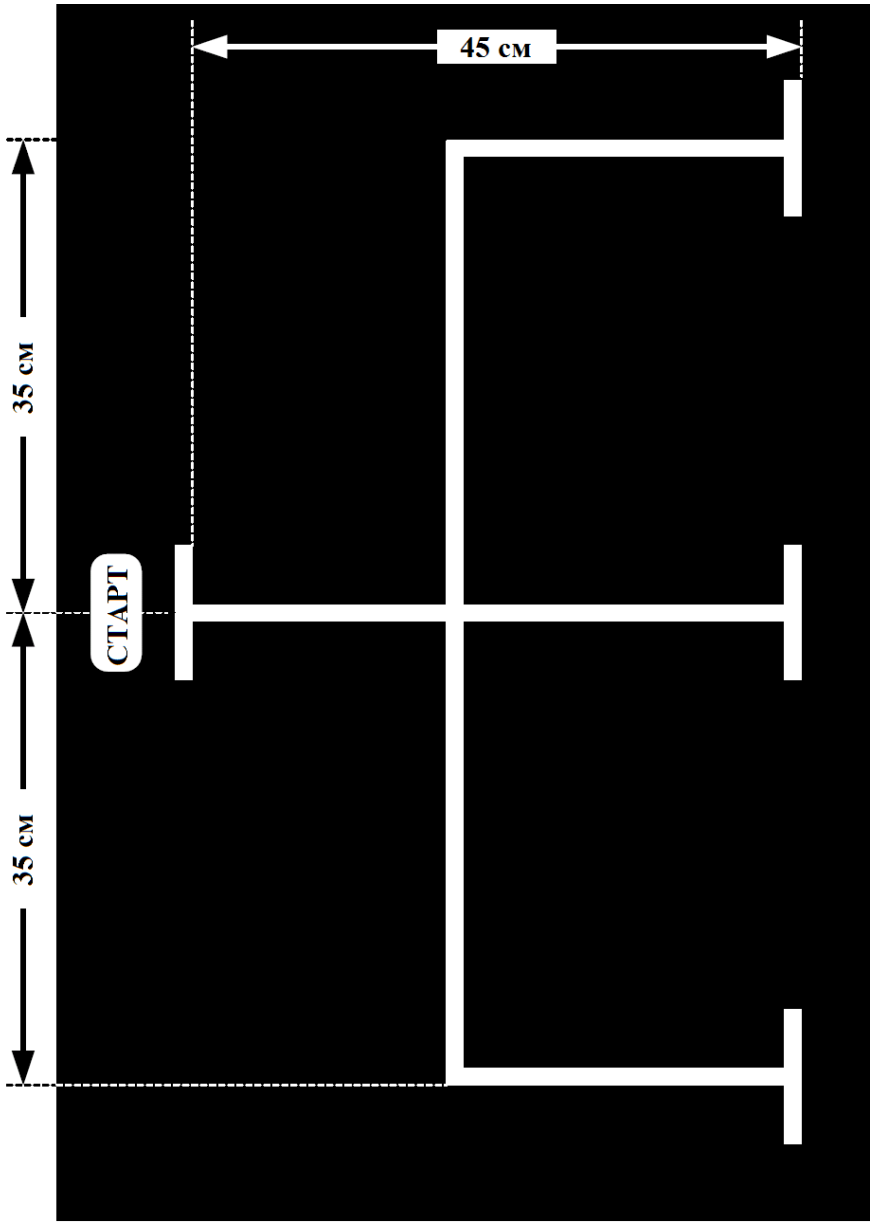
Сценарий 1 : Движение вперед, после обнаружения пересечения
Сценарий 2 : Поворот влево, после обнаружения пересечения
Сценарий 3 : Поворот вправо, после обнаружения пересечения

Как видно из маршрута движения, Сценарий 1 выполняется только у 8-го пересечения, Сценарий 2 выполняется 3 раза при 3-м, 4-м и 10-м пересечении. Сценарий 3 POP-BOT будет выполнять у последнего пересечения.

После того, как POP-BOT пройдет последнее пересечение (11-е), POP-BOT будет разворачиваться 2 секунды и затем остановится.

Задание 12 : Бросаем вызов белой линии

Это задание было взято из одной из миссий в робототехнических играх. Отличием от Задания 11 является цвет линии и путь движения. В этом задании будет использоваться белая линия на черной поверхности. Демонстрационное поле изображено на рисунке ниже. Миссия будет состоять в том, чтобы начиная от пересечения с отметкой СТАРТ (START) дойти до конца каждого из трех других пересечений. В каждом из пересечений можно закрепить воздушный шарик. Робот должен будет проколоть все эти шарики. Робот, который сможет выполнить задачу быстрее всех, будет считаться победителем.



A12.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A12-1.
A12.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A12.3 Отсоедините загрузочный кабель.

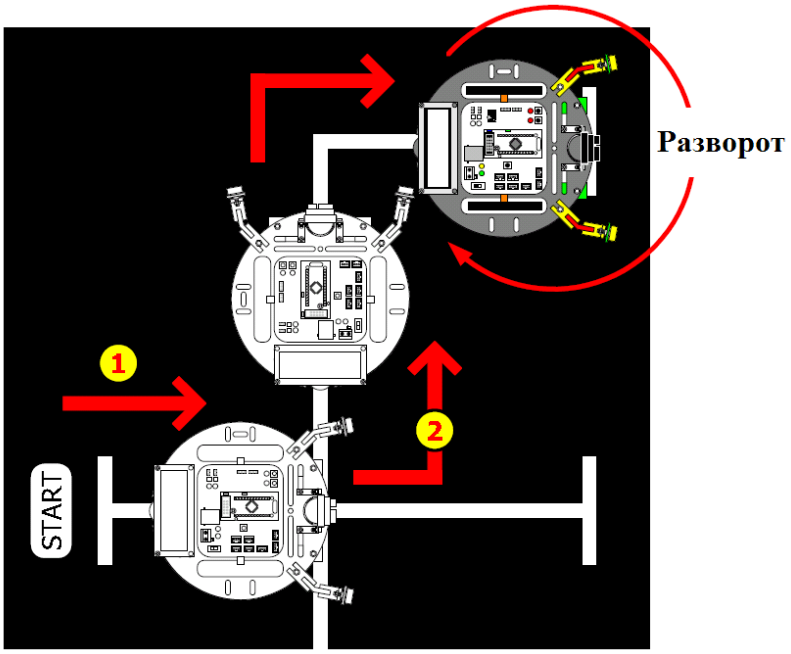
```
/*****************************************************************************
* Filename : WhiteLineDetect.pde
* POP-BOT движется следуя белой линии и проверяет наличие пересечений,
* чтобы совершить сложное движение для прокалывания шариков в 3 конечных
точках
*****************************************************************************/
int Ref=400;
int Left,Right;
int Cnt=0;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(14,OUTPUT); // Пьезоизлучатель
}
void loop(){
      while(Cnt<12){
            Left = analogRead(7); // Чтение значения с левого датчика ZX-03
            Right = analogRead(6); // Чтение значения с правого датчика ZX-03
            if ((Left<Ref) && (Right<Ref)){ // Обнаруживает черную линию
            пересечения
                  Cross();
            }
            else if ((Left>Ref) && (Right>Ref)){ // Обнаруживает белую линию
                  Forward(150);
            }
            else if (Left<Ref) { // Только левый датчик обнаружил черную линию
                  Spin_Right(150); // Поворот вправо
            }
            else if (Right<Ref){ // Только правый датчик обнаружил черную
            линию
                  Spin_Left(150); // Поворот влево
            }
      }
      while(1); // Endless loop
}
void Cross(){
      Cnt++;
      if (Cnt==12){
            Motor_Stop();
      }
      else if (Cnt==2 || Cnt==10 ){ // Проверка условия поворота вправо на 90
      градусов.
            Right90();
      }
      else if(Cnt==3 || Cnt==6 || Cnt==9 ){ // Проверка условия поворота влево
на 180 градусов.
            Left180();
      }
      else{ // иначе, поворот влево на 90 градусов.
            Left90();
      }
}
```

```
/*Функция поворота вправо на 90 градусов*/
void Right90(){
      Forward(150);
      delay(50);
      Spin_Right(200);
      delay(100);
      while(analogRead(6)<Ref);
      delay(50);
}
/* Функция поворота влево на 90 градусов */
void Left90(){
      Forward(150);
      delay(50);
      Spin_Left(200);
      delay(100);
      while(analogRead(7)<Ref);
      delay(50);
}
/* Функция поворота влево на 180 градусов */
void Left180(){
      Spin_Left(200);
      delay(300);
      while(analogRead(7)<Ref);
      delay(50);
}
/*Функция движения вперед*/
      void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Motor_Stop(){
      digitalWrite(3,LOW);
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(9,LOW);
}/*****************************************************************************/
```

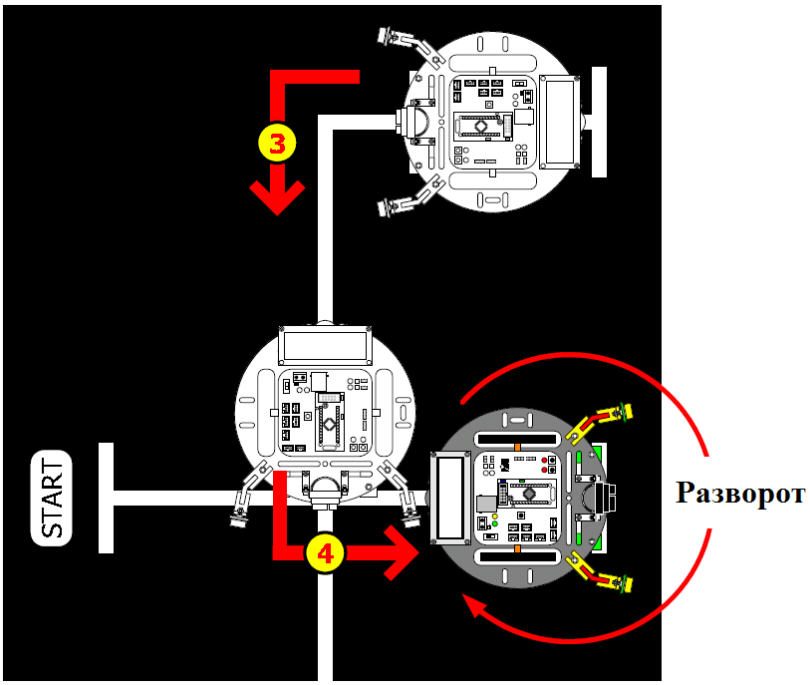
Листинг А12-1 : Файл WhiteLineDetect.pde; скетч-файл Arduino для управления движением POP-BOT вдоль белой линии и обнаружения пересечения линии с «пристанью» в конечных пунктах

A12.4 Поместите POP-BOT в точку СТАРТ (START) (см. иллюстрацию демонстрационного поля). Включите робота и наблюдайте за его движением.

Шаг 1 : POP-BOT движется вдоль линии от точки СТАРТ (START)
Шаг 2 : POP-BOT обнаруживает пересечение первый раз и поворачивает влево, чтобы попасть в первую точку назначения в левой части демонстрационного поля.

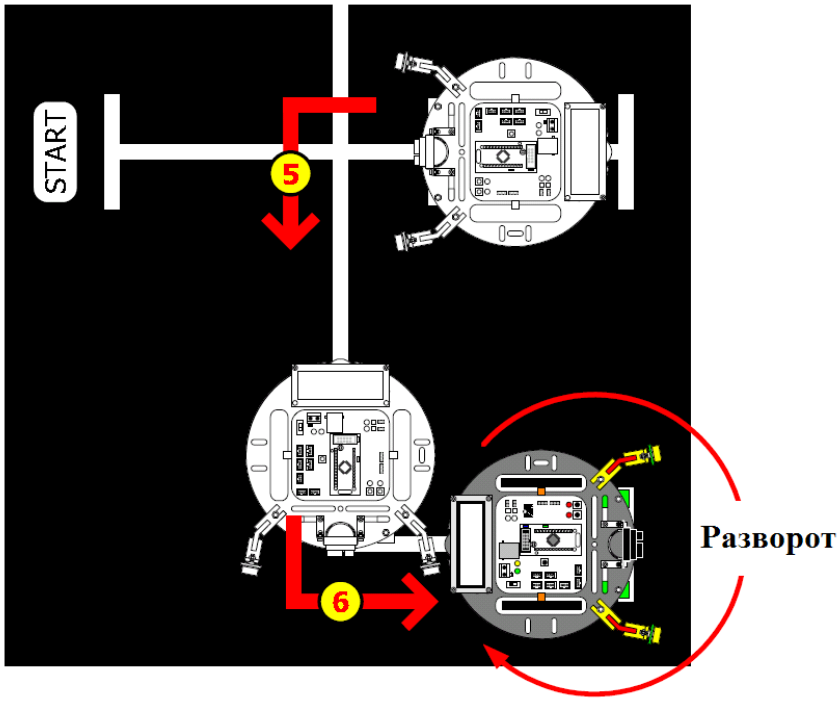


Шаг 3 : POP-BOT движется обратно из первой точки назначения, еще раз проходя пересечение.
Шаг 4 : POP-BOT обнаруживает пересечение второй раз и поворачивает влево, чтобы попасть во вторую точку назначения в середине демонстрационного поля.



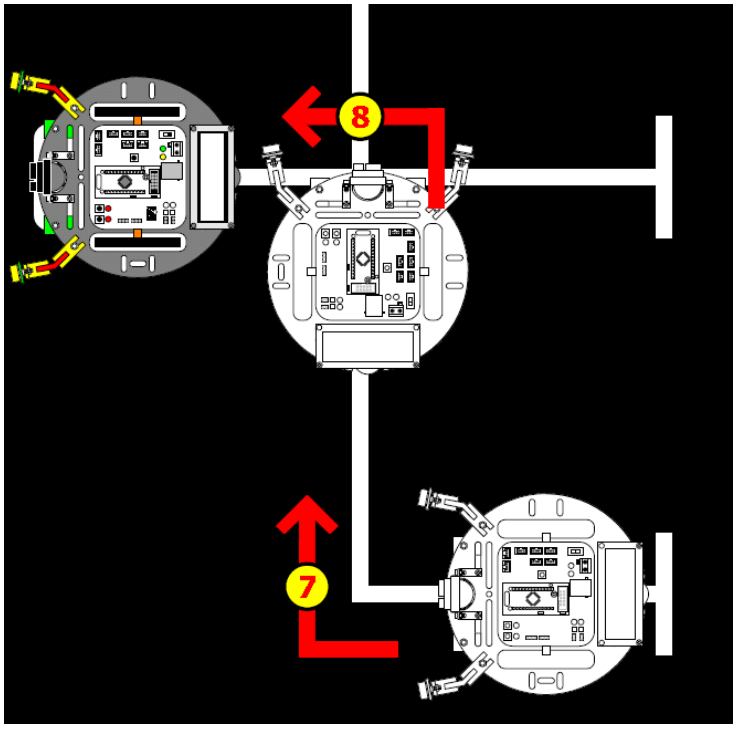
Шаг 5 : POP-BOT движется обратно из второй точки назначения, еще раз проходя пересечение.

Шаг 6 : POP-BOT обнаруживает пересечение третий раз и поворачивает влево, чтобы попасть в третью точку назначения в правой части демонстрационного поля.



Шаг 7 : POP-BOT движется обратно из последней точки назначения, еще раз проходя пересечение линий.

Step 8 : POP-BOT последний раз обнаруживает пересечение, поворачивает налево и движется назад к точке СТАРТ (START), чтобы завершить миссию.



8 : POP-BOT обнаруживает края плоских поверхностей

В разделе 7 мы использовали инфракрасный отражательный датчик для обнаружения линий. Знаете ли Вы, для чего еще можно использовать этот датчик? В данном разделе будет показана возможность использования инфракрасных отражательных датчиков для обнаружения края поверхности, чтобы управлять движением робота по поверхности стола и не допускать его падения со стола!

Путем простейшего изменения положения датчиков и простой программы, мы сможем приспособить POP-BOT для обнаружения краев стола. Начнем со сборки механических частей, расположения датчиков в правильных местах, после чего создадим скетч файл Arduino для проверки приближения к краю поверхности стола.

8.1 Список дополнительных деталей

Прямая крепежная планка с 12 отверстиями x 2

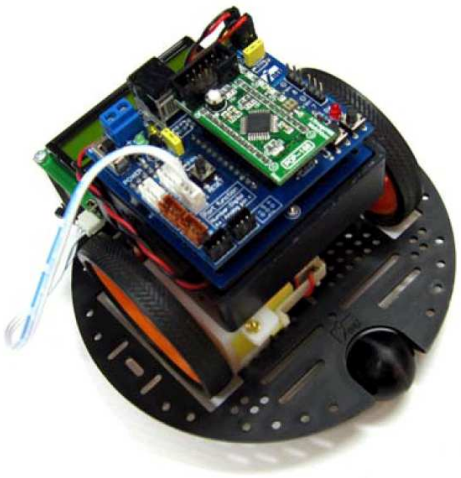
Винты M3X10 мм x 4

Гайки M3 x 4

Пластиковые втулки 3 мм x 2

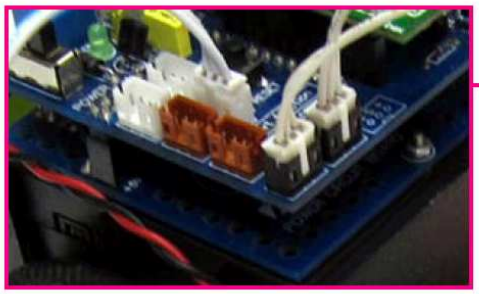
8.2 Процедура изменения конструкции робота

(1) Снимите все датчики прикосновения и датчики отслеживания линий с корпуса POP-BOT. Теперь перед нами простейшая форма мобильного робота POP-BOT.

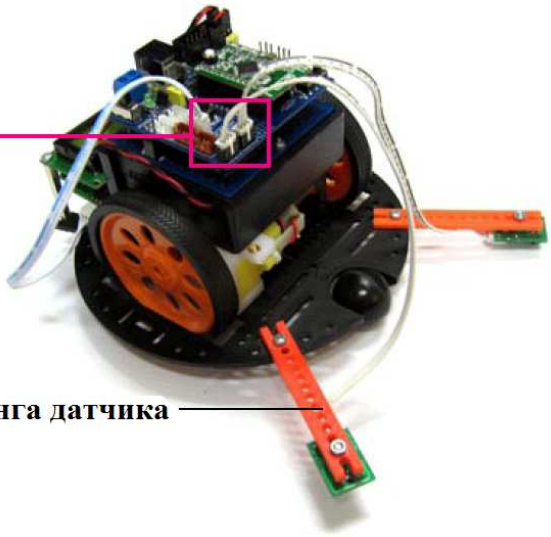


(2) Закрепите инфракрасный отражательный датчик ZX-03 в первом отверстии прямой крепежной планки 12 отверстиями, используя винт M3 x 10 мм, 3 мм пластиковую втулку и гайку M3, следуя приведенной ниже фотографии. Сделайте 2 таких субмодуля.

(3) Закрепите обе структуры, собранные на шаге (2) с левой и правой стороны в передней части корпуса POP-BOT, используя винт M3x10мм и гайку M3, следуя приведенной ниже фотографии. Далее присоедините кабелем левый датчик к порту A7, а правый датчик к порту A6. Вы можете подобрать положение штанги датчика применительно к конкретным условиям выполнения заданий.



На фотографии показано подключение датчика к плате управления POP-BOT



Задание 13 : POP-BOT обнаруживает край стола

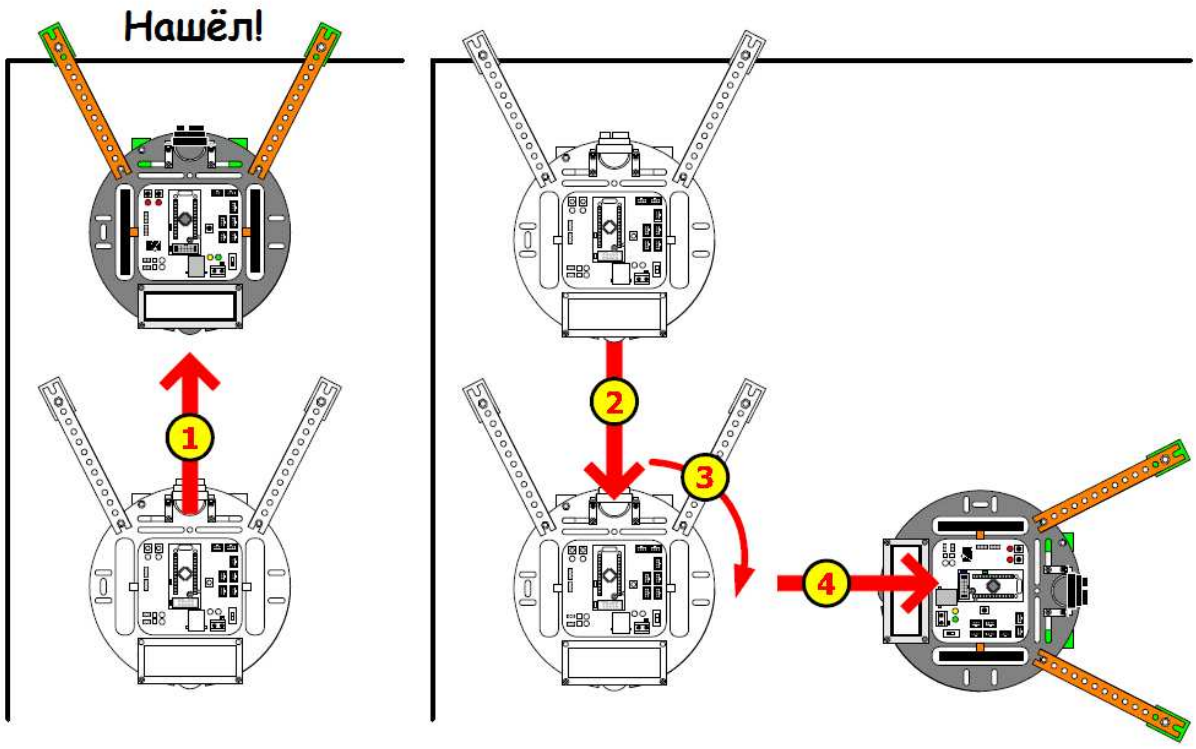
В этом задании демонстрируется интересное поведение робота: POP-BOT двигается по поверхности стола и не падает со стола. Используя 2 инфракрасных отражательных датчика, которые закреплены в передней части робота, можно обнаружить окончание поверхности стола. Код данного задания похож на код, который был создан для отслеживания линий. Если датчик обнаруживает присутствие поверхности, аналоговый сигнал на его выходе имеет высокий уровень. Как только датчик оказывается за пределами поверхности стола, инфракрасное излучение перестает отражаться от поверхности стола и перестает попадать на датчик, поэтому выходной аналоговый сигнал датчика будет иметь низкий, почти нулевой уровень.

Такое поведение датчика можно использовать, чтобы создать код, который будет управлять движением POP-BOT по поверхности стола и препятствовать падению POP-BOT со стола.

A13.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A13-1.
A13.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робота.
A13.3 Отсоедините загрузочный кабель.
A13.4 Поместите на поверхность стола. Для чистоты эксперимента необходимо убрать с поверхности стола все предметы. Включите питание POP-BOT и наблюдайте за движением робота.

POP-BOT движется вперед до тех пор, пока датчик не окажется за краем стола. POP-BOT будет изменять направление своего движения, следуя данному сценарию:

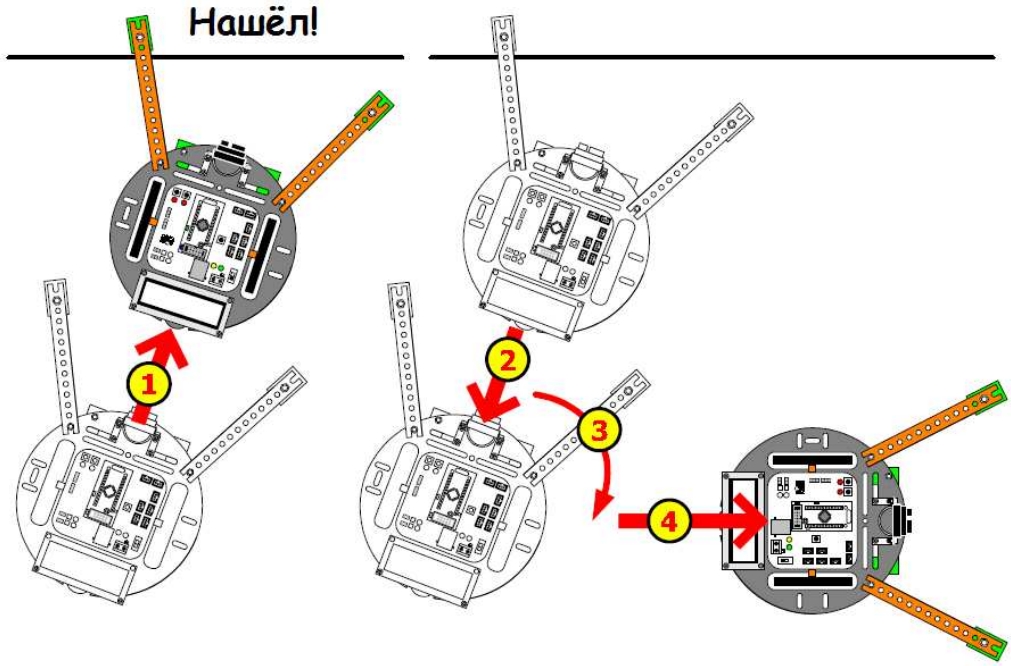
1.Оба датчика оказались за пределами края стола: POP-BOT двигается назад, поворачивает вправо и затем продолжает двигаться вперед.



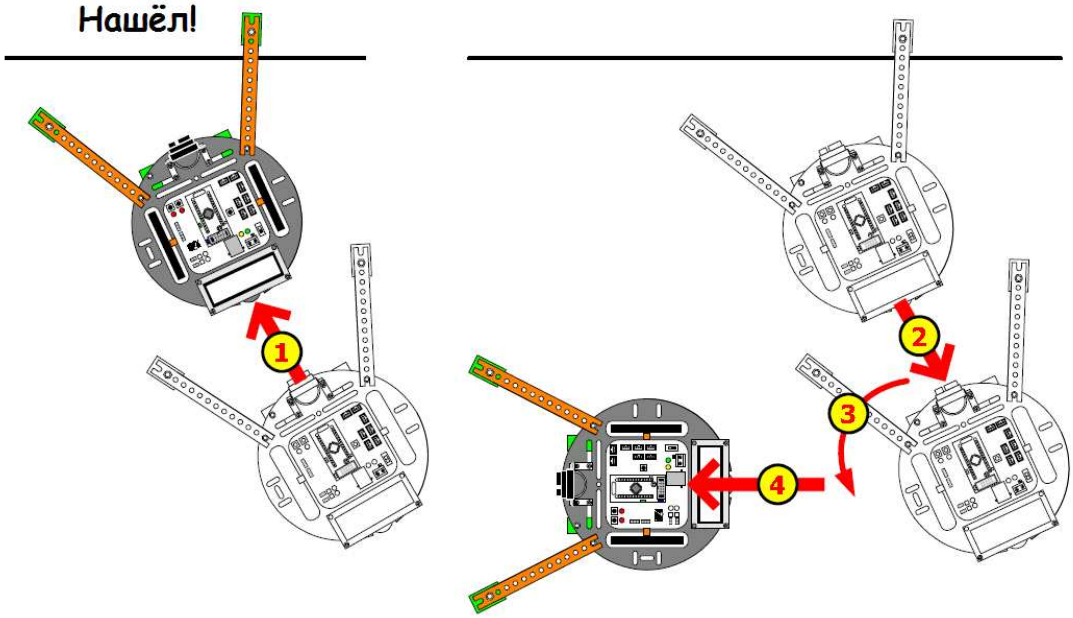
```
/*****************************************************************************
* POP-BOT V1.0
* Filename : EdgeDetect.pde
* POP-BOT избегает падения с края стола
*****************************************************************************/
int Ref=300;
int Left,Right;
void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
}
void loop(){
      Left = analogRead(7); // Чтение значения с левого датчика ZX-03
      Right = analogRead(6); // Чтение значения с правого датчика ZX-03
      if (Left>Ref && Right>Ref){ // Оба датчика в пределах поверхности стола
            Forward(150);
      }
      else if (Left<Ref && Right<Ref){ // Оба датчика за пределами поверхности стола
            Backward(150);
            delay(200);
            Spin_Right(150);
            delay(500);
      }
      else if (Left<Ref) { // Только левый датчик за пределами поверхности стола
            Backward(150);
            delay(300);
            Spin_Right(150);
            delay(400);
      }
      else if (Right<Ref){ // Только правый датчик за пределами поверхности стола
            Backward(150);
            delay(300);
            Spin_Left(150);
            delay(300);
      }
}
/*Movement function*/
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Backward(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/*****************************************************************************/
```

Листинг A13-1 : Файл EdgeDetect.pde; скетч-файл Arduino для управления движением POP-BOT по столу, чтобы обнаружить его край и избежать падения со стола

2. Левый датчик оказался за пределами поверхности стола : POP-BOT движется назад, поворачивает вправо, после чего продолжает двигаться вперед.



3. Правый датчик оказался за пределами поверхности стола : POP-BOT движется назад, поворачивает влево, после чего продолжает двигаться вперед.



9 : POP-BOT избегает столкновения с предметами бесконтактным способом

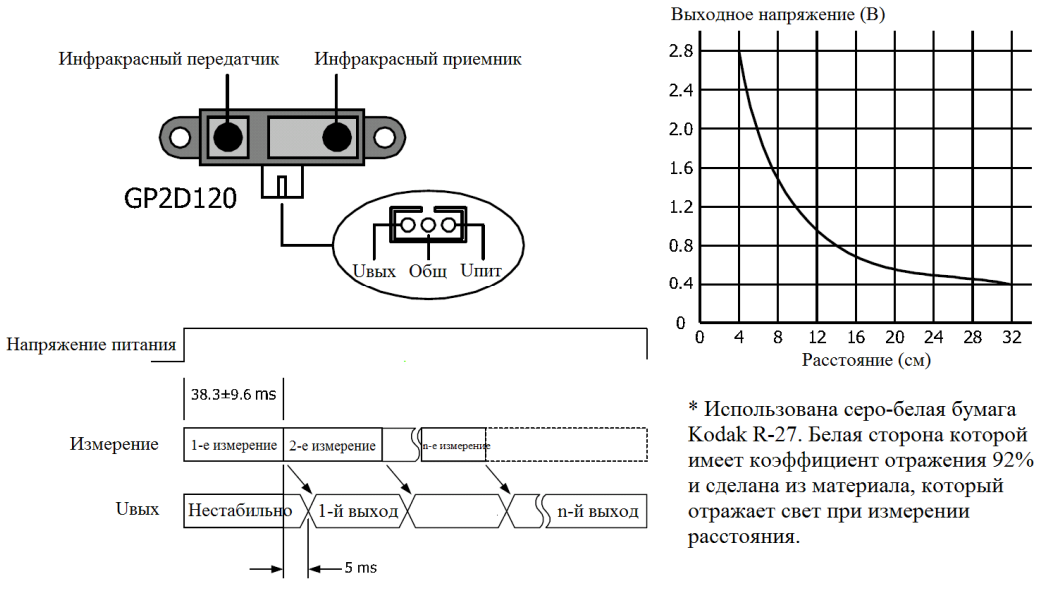
9.1 GP2D120 : Инфракрасный датчик расстояния с дальностью обнаружения от 4 до 30 см

В разделе 7 было рассмотрено множество примеров об обмене данными с инфракрасными отражательными датчиками. Они представляют собой один из видов аналоговых датчиков. В этом разделе мы будем концентрировать свое внимание на другом типе аналоговых датчиков. Это – инфракрасные датчики расстояния, или инфракрасные локаторы GP2D120. Имеется несколько примеров использования этих датчиков и их применения.

Одним из специальных датчиков в робототехнике является инфракрасный датчик расстояния. Некоторые называют его ИК-локатор. Используя модуль GP2D120, POP-BOT получает возможность измерения расстояний и бесконтактного обнаружения препятствий, используя инфракрасное излучение. Ваш POP-BOT сможет предотвратить столкновения без использования физического контакта с препятствиями.

9.1.1 Особенности GP2D120

- Для измерения расстояния используется отражение инфракрасного излучения
- Может измерять расстояния в диапазоне от 4 до 30 см.
- Напряжение питания от 4.5 до 5В при потребляемом токе 33мА
- Выходное напряжение в диапазоне от 0.4 до 2.4В при напряжении питания +5В



* Использована серо-белая бумага Kodak R-27. Белая сторона которой имеет коэффициент отражения 92% и сделана из материала, который отражает свет при измерении расстояния.

Рисунок 9-1 Назначение выводов модуля GP2D120, принцип работы и кривая чувствительности

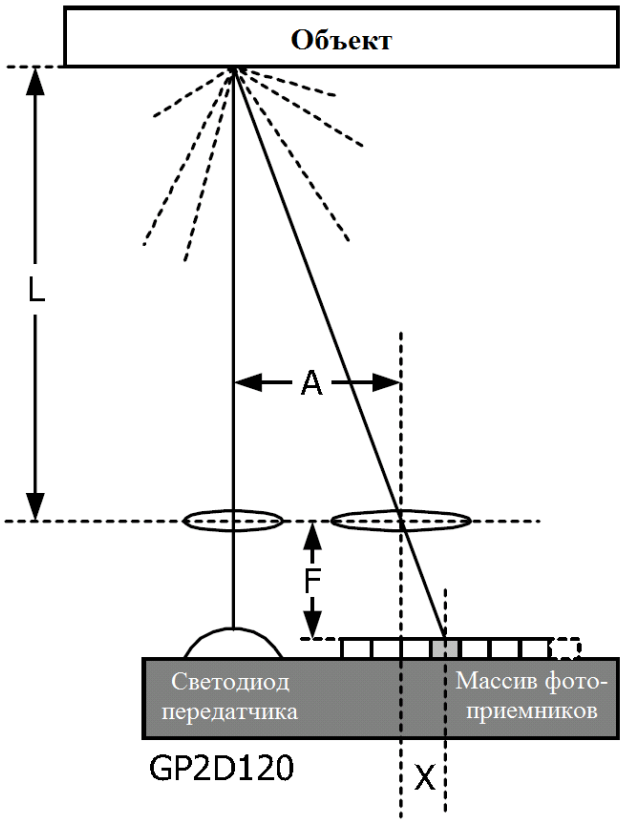
Модуль инфракрасного локатора GP2D120 имеет 3 вывода : вход напряжения питания (Vcc), общая шина (GND) и выход аналогового сигнала (Vout). Чтобы прочитать величину напряжения на выходе датчика GP2D120, необходимо подождать окончания переходного периода, который составляет, примерно, от 32 до 52.9 мсек.

Выходное напряжение GP2D120 при расстоянии до объекта 30 см и напряжении источника питания +5В находится между 0.25 и 0.55В, со средним значением 0.4В. При расстоянии до объекта, равном 4 см, выходное напряжение будет изменяться в пределах 2.25В± 0.3В.

9.1.2 Как работает модуль ИК-локатора?

Измерения расстояния можно организовать разными способами. Простейшим для понимания является измерение с помощью ультразвука, когда в сторону объекта посылают звуковые волны и измеряется время, через которое приходит отраженная волна. Такой метод возможен потому, что звуковые волны перемещаются не слишком быстро, и измерения времени полета волны можно производить с помощью достаточно простого оборудования. Однако, в случае инфракрасного излучения время, которое требуется волне, чтобы достичь препятствия и вернуться назад, не может быть измерено простыми способами, поскольку инфракрасное излучение передвигается очень быстро – со скоростью света. Оборудование для таких измерений пока еще не доступно. Поэтому следует использовать следующую теорию.

Инфракрасное излучение, которое посылают из передатчика на объект, находящийся впереди, проходит через фокусирующую линзу таким образом, чтобы интенсивность светового потока была сфокусирована в некоторой точке. Как только свет достигает поверхности объекта, происходит его отражение. Часть отраженного света будет отражена обратно на приемное устройство, в котором другая линза сфокусирует этот свет и будет определена точка его падения на приемный датчик. Затем свет поступит на массив фототранзисторов. Позиция фототранзистора, на которой приходится максимум светового потока, может быть использована для определения расстояния (L) от излучателя света до препятствия по следующей формуле:



$$\frac{L}{A} = \frac{F}{X}$$

Откуда, L равно

$$L = \frac{F \times A}{X}$$

Таким образом, перед тем, как будет изменено напряжение, в соответствии с измеренным расстоянием, значение расстояния от массива фототранзисторов передается в Модуль Обработки Сигнала.

9.1.3 Взаимодействие GP2D120 с АЦП

Выходное напряжение модуля GP2D120 будет меняться согласно измеренному расстоянию до препятствия. Например, выходное напряжение Vout = 0.5В соответствует расстоянию 26см, а выходное напряжение Vout = 2В соответствует расстоянию 6см. В Таблице 9-1 показано обобщенное соотношение между выходным напряжением Vout и измеренным расстоянием для GP2D120.

При измерении выходного напряжения модулем встроенного в микроконтроллер АЦП, результатом будут сырые данные от аналого-цифрового преобразования. Пользователю необходимо будет использовать программное обеспечение для преобразования этих сырых данных в соответствующее им расстояние. Примерное значение расстояния можно вычислить по следующей формуле.

$$\mathrm{R} = \frac{2914}{\mathrm{V} + 5} - 1$$

Предупреждение относительно сигнального кабеля GP2D120

Модуль GP2D120 имеет назначение выводов, отличающееся от назначения выводов платы управления POP-BOT, несмотря на внешнюю похожесть разъемов. Поэтому к модулю GP2D120 уже подключен специальный сигнальный кабель. Пользователю необходимо всего лишь присоединить другой конец соединительного кабеля к разъему на плате управления POP-BOT. Не отключайте кабель от модуля и не заменяйте его на соединительные кабели от других модулей!

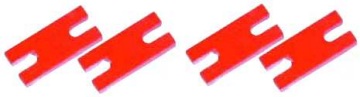
Выходное напряжение модуля GP2D120 (В)	Результат 10-разрядного аналого-цифрового преобразования		Расстояние (см)
	Шестнадцатеричный	Десятичный	
0.4	0x052	82	32
0.5	0x066	102	26
0.6	0x07B	123	22
0.7	0x08F	143	19
0.8	0x0A4	164	16
0.9	0x0B8	184	14
1.0	0x0CD	205	13
1.1	0x0E1	225	12
1.2	0x0F6	246	11
1.3	0x104	266	10
1.4	0x11F	287	9
1.5	0x133	307	8
1.6	0x148	328	8
1.7	0x15C	348	7
1.8	0x171	369	7
1.9	0x185	389	6
2.0	0x19A	410	6
2.1	0x1AE	430	6
2.2	0x1C3	451	5
2.3	0x1D7	471	5
2.4	0x1EC	492	5
2.5	0x200	512	5
2.6	0x214	532	4

Таблица 9-1 : Соотношение между выходным напряжением датчика расстояния GP2D120, результатом аналого-цифрового преобразования и измеряемым расстоянием до объекта

9.2 Доработка POP-BOT для работы с модулем GP2D120

9.2.1 Список дополнительных деталей

Модуль GP2D120 с кабелем



Прямой крепеж x 4

Тупоугольный крепеж x 2

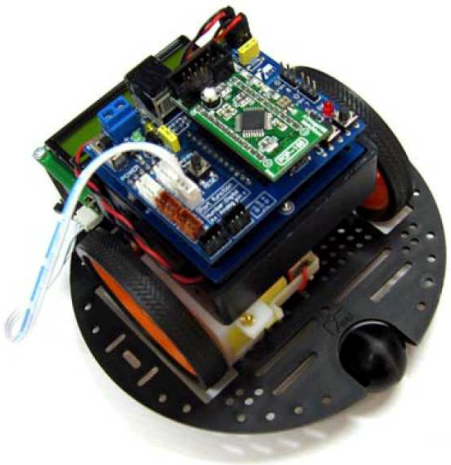
Прямоугольный крепеж x 2

Винты M3x10мм x 4

Гайки M3 x 4

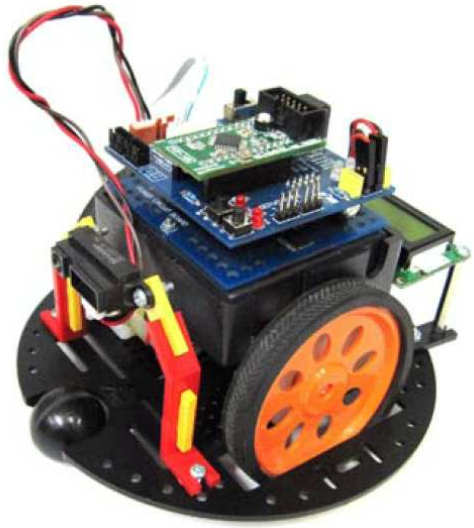
9.2.2 Процедура изменения конструкции робота

(1) Снимите все датчики с корпуса POP-BOT. Теперь перед нами простейшая форма мобильного робота POP-BOT.



(2) Присоедините 2 прямых крепежных элемента к отверстиям в корпусе модуля GP2D120, используя винты M3x10мм и гайки M3. Далее прикрепите тупоугольные крепежные элементы к свободным концам прямых крепежных элементов, затем опять прямые крепежные элементы и, наконец, прямоугольные крепежные элементы.

(3) Закрепите структуру модуля GP2D120, полученную на шаге (2), в передней части корпуса POP-BOT, используя винты M3x10мм и гайки M3 в положении, соответствующем рисунку ниже. Присоедините кабель от GP2D120 к порту **19/SCL/A5** платы управления POP-BOT.
Теперь POP-BOT с ИК-локатором готов для программирования.



9.3 Как прочитать данные с модуля GP2D120 робота POP-BOT

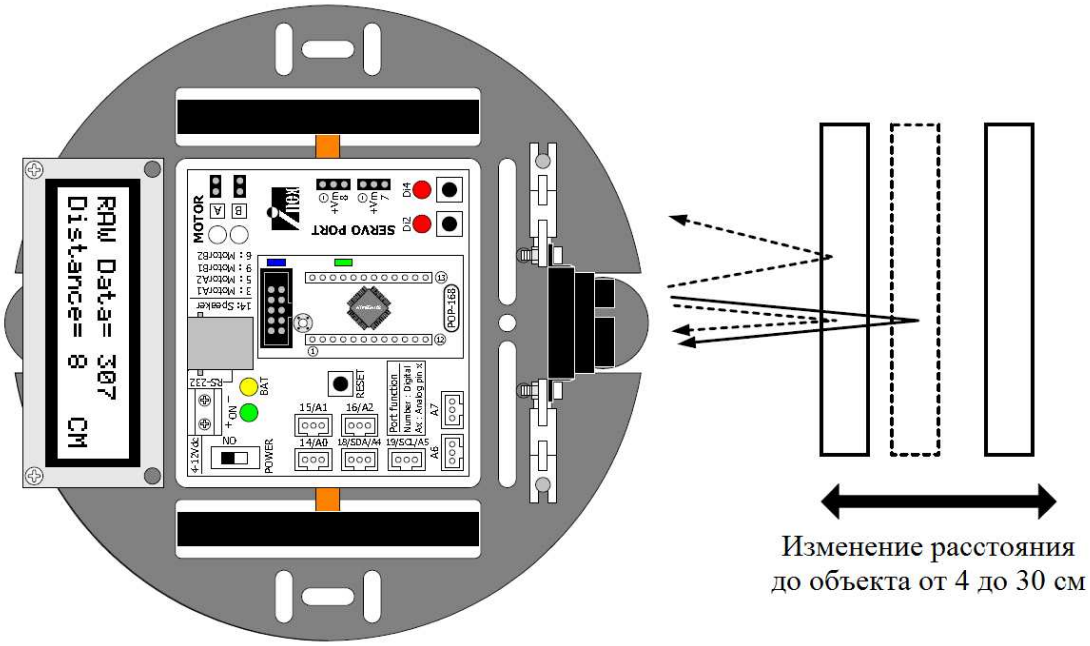
Робот POP-BOT имеет модуль микроконтроллера POP-168. Он работает с программным обеспечением Arduino. Библиотека подпрограмм Arduino имеет специальную функцию чтения значения из аналогового порта ввода. Это - функция **analogRead()**. Значение в скобках является номером аналогового входа (от 0 до 7). Для POP-BOT допустимы только значения от 3 до 7. Функция **analogRead()** возвращает целые значения от 0 до 1023. Это – результат 10-разрядного аналого-цифрового преобразования.

Сырые данные можно преобразовать в единицы напряжения (Вольты) по следующей формуле:

Вольты = Сырые_данные x 5 /1023

Задание 14 : Чтение данных с модуля GP2D120

A14.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A14-1.
A14.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A14.3 Отсоедините загрузочный кабель.
A14.4 Поместите робот POP-BOT на стол. Положите перед датчиком GP2D120 какой-либо предмет. Включите робот POP-BOT. передвигайте объект относительно датчика GP2D120. Наблюдайте за изменениями результатов на экране SLCD.



```
/*************************************************************************
* POP-BOT V1.0
* Filename : GP2D120withSLCD.pde
* наблюдение данных от модуля GP2D120 на экране модуля SLCD16x2
*************************************************************************
***/
#include <SoftwareSerial.h>
#define rxPin 16 // SLCD pin
#define txPin 16 // Same pin
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int gp2;
float distance;
void setup(){
      pinMode(txPin,OUTPUT);
      MySerial.begin(9600);
      delay(1000);
}
void LCD_CMD(int Command){
      MySerial.print(0xFE,BYTE); // Команда
      MySerial.print(Command,BYTE);
}
void loop(){
      gp2=analogRead(5);
      distance=(2914/(gp2+5))-1; // Преобразование в сантиметры
      LCD_CMD(0x80); // Select LCD first Line
      MySerial.print("RAW Data= "); // Отображение сырых данных
      LCD_CMD(0x8A);
      MySerial.print(gp2,DEC);
      LCD_CMD(0xC0); // Установка курсора в начало второй строки ЖКИ
      MySerial.print("Distance= "); // Отображение расстояния в сантиметрах
      LCD_CMD(0xCA);
      MySerial.print(distance,DEC);
      LCD_CMD(0xCE);
      MySerial.print("CM");
      delay(200);
}
/*************************************************************************/
```

Листинг A14-1 : Файл GP2D120withSLCD.pde; скетч-файл Arduino для чтения POP-BOT информации с датчика расстояния GP2D120 (инфракрасный локатор)

Описание программы

(1) Начальный обмен данными по последовательному интерфейсу. Установит перемычку 16/A2 для последовательного порта.

(2) Цикл чтения аналогового сигнала через вывод An5 порта ввода платы управления POP-BOT и отображение на экране ЖКИ.

(3) Преобразование сырых данных о расстоянии в сантиметры, используя следующую формулу:
см = (2914 / (gp2 + 5)) - 1

(4) Преобразование результата в текстовую форму и отправка в модуль SLCD16x2 для отображения.

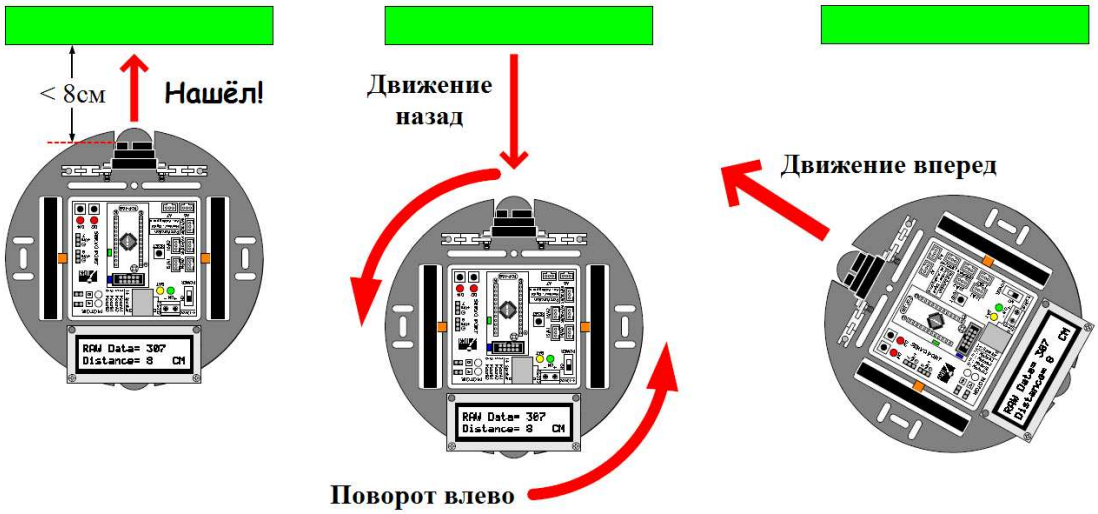
(5) Цикл опроса GP2D120 с интервалом 0.2 секунды.

Задание 15 : Бесконтактная система предотвращения столкновений

С помощью модуля GP2D120, используя инфракрасное излучение, нашему роботу можно добавить функции измерения расстояния и обнаружения препятствий. Наш POP-BOT сможет избегать столкновений без какого бы то ни было физического контакта с препятствием.

A15.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A15-1.
A15.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A15.3 Отсоедините загрузочный кабель.
A14.4 Поместите POP-BOT на пол. Положите перед роботом любой предмет и наблюдайте за его действиями.

Робот будет измерять расстояние до объекта пока оно не достигнет значения 8 см. при отсутствии препятствий, робот будет непрерывно двигаться вперед. После обнаружения объекта на предельной дистанции 8 см, он отъедет назад, повернет влево и снова будет продолжать двигаться вперед.



```
/*************************************************************************
* POP-BOT V1.0
* Filename : TouchlessObjectRobot.pde
*************************************************************************/
int gp2;
void setup(){
      pinMode(3,OUTPUT); // Установить цифровые линии как выходы
      pinMode(5,OUTPUT);
      pinMode(6,OUTPUT);
      pinMode(9,OUTPUT);
}
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Backward(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void Spin_Left(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
void loop(){
      int i;
      for (i=0;i<5;i++){ // Повторить 5 раз для фильтрации шума
            gp2=(gp2+analogRead(5));
      }
      gp2=gp2/5;
      if (gp2>290){ // Found object
            Backward(200); // Отъехать назад для изменения направления движения
            delay(300);
            Spin_Left(200); delay(350); // Изменить направление движения
      }
      else{
            Forward(200); // Двигаться вперед
      }
}
/*************************************************************************/
```

Листинг A15-1 : Файл TouchlessObjectRobot.pde; скетч-файл Arduino для демонстрации бесконтактного избегания роботом POP-BOT препятствий.

Описание программы

(1) Стартуя, POP-BOT издает одиночный сигнал. Этот сигнал можно использовать для проверки, что робот находится в состоянии RESET (СБРОС) при снижении напряжения питания батареи.

(2) Чтение очередного значения с модуля GP2D120 и пятикратное запоминание в переменной **gp2**. Вычисление среднего значения для защиты от ошибок чтения при движении робота.

(3) Проверка, имеет ли переменная **gp2** значение 290 или нет? Если да, то это значит, что расстояние от робота до объекта меньше, чем 8 см (примерно). Программа начнет управлять роботом таким образом, что он начнет двигаться назад в течение 0.25 секунды, и затем поворачивать налево в течение 0.5 секунды, чтобы изменить направление движения и предотвратить столкновение с объектом.

(4) Если значение переменной **gp2** меньше, чем 290, то робот будет продолжать движение вперед.

10 : Работа POP-BOT с сервомотором

Робот POP-BOT имеет выходы для подключения RC-сервомоторов. Робот POP-BOT может независимо управлять двумя маленькими RC-сервомоторами. Плата управления POP-BOT уже обеспечивает напряжение питания сервомотора на выводах разъема Servo. Для подключения сервомотора пользователю не требуется использовать дополнительные батареи. Это является важной особенностью робота POP-BOT. Робот POP-BOT может управлять 4 моторами: 2 электродвигателями постоянного тока и 2 сервомоторами.

10.1 Введение в принципы работы сервомотора

На рисунке 10-1 показан внешний вид Стандартного Сервомотора. Кабель с разъемом используется для подключения сервомотора к источнику питания (Vdd и Vss) и к источнику сигналов управления (линия Ввода/Вывода микроконтроллера). Кабель передает напряжения Vdd, Vss и сигнала от разъема к сервомотору. Кронштейн является частью сервомотора, которая по внешнему виду очень сильно напоминает четырехлучевую звезду. При работе сервомотора кронштейн является движущейся частью, которой управляет микроконтроллер. В корпусе сервомотора расположена схема управления, электродвигатель постоянного тока и редуктор. Все эти части работают совместно, принимая сигнал высокого/низкого уровня от микроконтроллера, и преобразуют его затем в положение кронштейна сервомотора.

На Рисунке 10-2 показано назначение проводов кабеля сервомотора. Он имеет 3 провода различного цвета; **Черный** для сигналов GND или Vss или Отрицательного полюса источника питания, **Красный** для сигнала Vdd или Напряжения питания сервомотора и **Белый** (иногда **Желтый** или **Коричневый**) для сигнала управления.

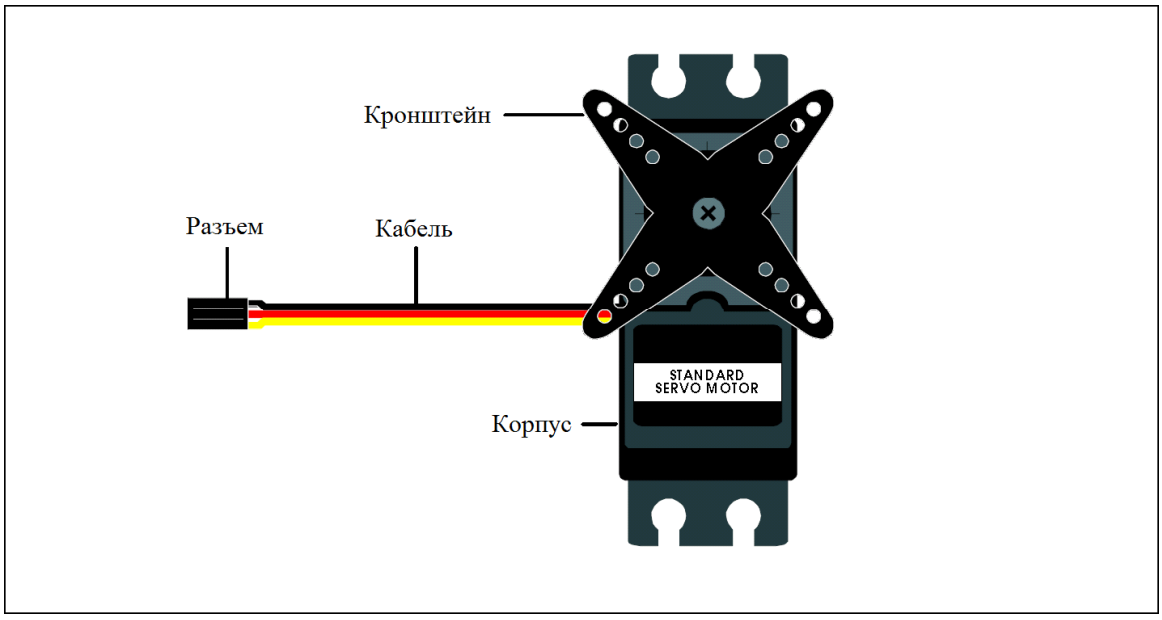


Рисунок 10-1. Внешний вид стандартного сервомотора

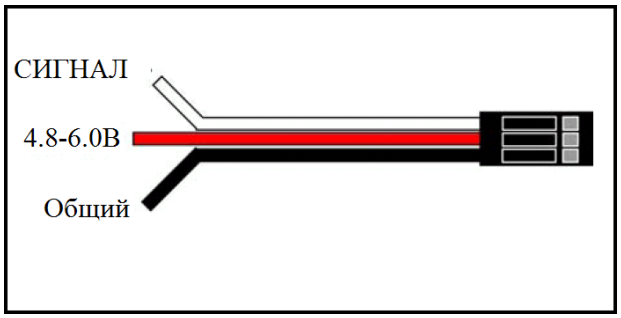


Рисунок 10-2. Стандартное назначение выводов кабеля сервомотора

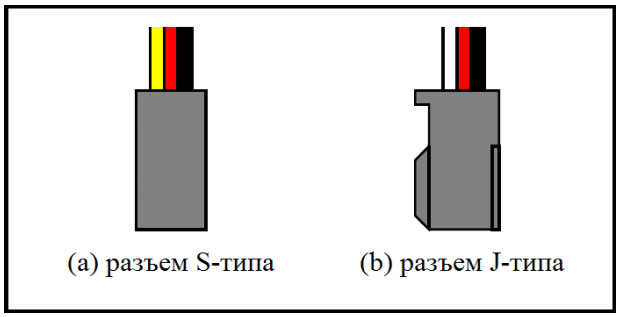


Рисунок 10-3. Стандартные типы разъемов сервомотора

Разъемы кабеля сервомотора могут быть двух типов : S-типа и J-типа, как показано на Рисунке 10-3.

Для управления сервомотором используются импульсы переменной длительности. Управляющие импульсы представляют собой импульсы положительной полярности длительностью от 1 до 2 миллисекунд с частотой повторения от 50 до 60 импульсов в секунду. Детали можно изучить по Рисунку 10-4. Начинается процесс управления с генерации импульса с периодом 20 миллисекунд и шириной положительного импульса длительностью 1 миллисекунда. Сервомотор передвинет кронштейн в крайнее левое положение. Импульс длительностью 1.5 миллисекунды вызовет перемещение кронштейна сервомотора в среднее положение, а импульс длительностью 2 миллисекунды вызовет перемещение кронштейна сервомотора в крайнее правое положение.

*Важной характеристикой сервомотора являются 2 параметра, такие как **Быстродействие** или коэффициент сервопередачи или время отработки команды и **Момент**. Коэффициент серво передачи или время отработки команды, используется для определения скорости серво вращения. Этот параметр представляет собой время, за которое кронштейн сервомотора поворачивается на заданный угол, обычно 60 градусов. Например, предположим, что имеется сервомотор с временем отработки команда 0.17сек/60 градусов, без нагрузки. Это значит, что для поворота на 180 градусов понадобится время, равное половине секунды. И еще большее время, если сервомотор находится под нагрузкой. Эта информация является очень важной для определения максимального времени отработки команды для Вашего робототехнического приложения. Она также полезна для определения максимальной скорости движения робота, если сервомотор переделан под работу с полнооборотным вращением кронштейна. Запомните, в самом скверном случае, время вращения будет максимальным, когда кронштейн сервомотора находится в одном из крайних положений, и сервомотор получает команду перевести кронштейн в другое крайнее положение. Для сервомоторов с очень большим нагрузочным моментом это может занять несколько секунд.

Вращательный момент – это стремление силы повернуть предмет вокруг некоторой оси. Единицей измерения момента является **унция на дюйм (oz-in**) или **килограмм на сантиметр (kg-cm)**. Это говорит Вам о сервомоторе то, что он может управлять нагрузкой в 1 унцию, чтобы передвинуть ее на 1 дюйм, или нагрузкой в 1кг веса, чтобы передвинуть ее на 1 сантиметр (1унция = 0.028кг, или 1кг = 25.274унции). Обычно RC-сервомоторы имеют вращающий момент порядка 3.40 кг см/47oz-in.

---

* http://www.societyofrobots.com/actuators_servos.shtm

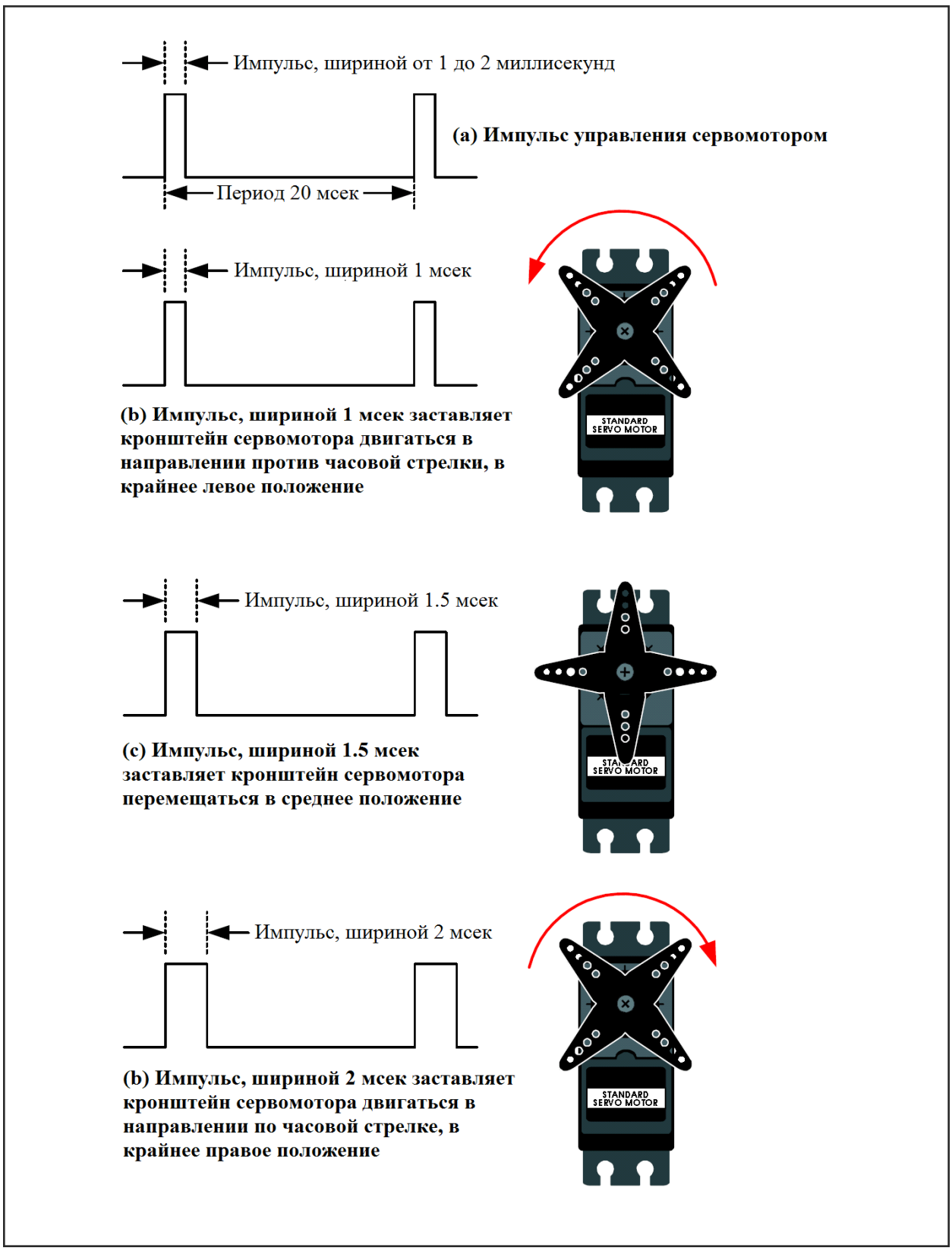


Рисунок 10-4. Временные диаграммы импульсного управления сервомотором

10.2 Arduino управляет сервомотором *

Сердцем управления сервомотором для Arduino POP-168 является библиотека SoftwareServo. Поскольку в аппаратной части POP-BOT отсутствуют выходы ШИМ для организации управления сервомотором, то мы будет использовать цифровые порты общего назначения Di7 и Di8 для создания выходных сигналов управления сервомоторами.

Библиотека SoftwareServo может независимо управлять сервомоторами, подключенными ко всем нашим выходам. Формат вызова функций библиотеки (API) создан по образцу библиотеки управления сервомоторами с сайта wiring.org, но исходный код отличается от имеющегося на сайте. Вы не ограничены 8 сервомоторами, но должны вызывать метод **SoftwareServo :: refresh(),** как минимум, 1 раз в 50 мсек или чаще, чтобы обновлять сигналы, управляющие сервомоторами.

10.2.1 Стандартные методы

attach(int)

Превращает вывод порта в выход управления сервомотором. Вызывает функцию pinMode. Возвращает 0 в случае неудачного завершения.

detach()

Исключает вывод из работы в режиме управления сервомотором.

write(int)

Устанавливает угол поворота сервомотора в градусах: от 0 до 180.

read()

Возвращает значение, использованное при последнем вызове функции write().

attached()

Возвращает 1, если в настоящее время присоединен сервомотор.

10.2.2 Дополнительные методы

refresh()

Необходимо вызывать каждые 50мсек для обновления информации управления сервомоторами. Этот метод можно вызывать так часто, как это необходимо, но нет большого смысла делать эти вызовы чаще, чем 1 раз в 20мсек. При вызове метода, он будет отрабатываться от 0.5 до 2.5, но не будет запрещать обработку прерываний.

setMinimumPulse(uint16_t)

задает продолжительность импульса, соответствующего углу 0 градусов в микросекундах (по умолчанию минимальное значение равно 544 микросекунды).

setMaximumPulse(uint16_t)

задает продолжительность импульса, соответствующего углу 180 градусов в микросекундах (по умолчанию максимальное значение равно 2400 микросекунды).

---

* http://www.arduino.cc/playground/ComponentLib/Servo

10.2.3 Безопасный поворот

Даже когда будет подключен сервомотор, он не будет получать никаких управляющих команд до тех пор, пока вы не переведете его в начальное состояние методом **write(),** чтобы воспрепятствовать переходу сервомотора в неопределенное сотояние.

10.2.4 Размер

Библиотека занимает в flash-памяти примерно 850 байт и 6+(8 х количество сервомоторов) байт SRAM-памяти.

10.2.5 Ограничения

Библиотека не останавливает обработку прерываний, таким образом, функция **millis()** будет продолжать нормальную работу, и Вы не потеряете последовательные входные данные, но конец импульса может быть расширен на время, необходимое на обработку прерываний, что может вызвать небольшую ошибку в отработке команды сервомотором. Если имеется большое количество сервомоторов, то может появиться небольшое (от 1 до 3 градусов) отклонение положения исполнительного органа сервомотора при малых значениях позиционного угла.

10.2.6 Пример

Следующий пример показывает, как можно управлять сервомотором, подключенным к выводу 7 с помощью потенциометра, подключенного к аналоговому входу 2

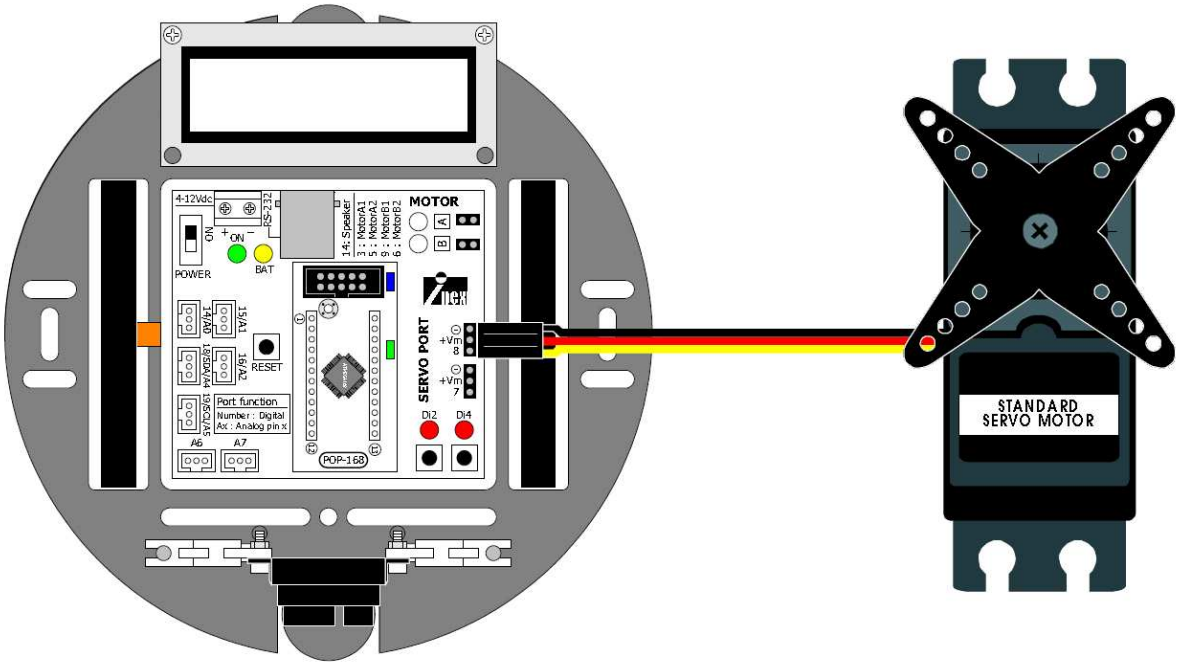
```
#include <SoftwareServo.h>
SoftwareServo myservo; // создание сервообъекта для управления
сервомотором
int potpin = 2; // аналоговый вход, используемый для подключения
потенциометра
int val; // переменная, в которой будет сохраняться значение,
прочитанное с аналогового входа
void setup() {
      myservo.attach(7);
      // добавить сервомотор, подключенный к выводу 7 к сервообъекту
}
void loop() {
      val = analogRead(potpin);
      // чтение значения с потенциометра (значение между 0 о 1023)
      val = map(val, 0, 1023, 0, 179);
      // масштабирование для использования с сервомотором (значение
между 0 и 180)
      myservo.write(val);
      // установить позицию сервомотора, согласно отмасштабированному
значению
      delay(15); // подождать, пока сервомотор выполнит команду
      SoftwareServo::refresh();
}
```

Задание 16 : POP-BOT управляет сервомотором

В этом задании рассмотрен простейший пример управления стандартным RC-сервомотором от платы управления POP-BOT.

Задание 16-1: Простейший пример управления сервомотором

A16.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A16-1.
A16.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A16.3 Отсоедините загрузочный кабель.
A16.4 Подключите стандартный RC-сервомотор к выходу SERVO PORT 7 или 8 платы управления POP-BOT.



A16.5 Включите питание робота и наблюдайте за его работой.

После включения питания сервомотор управляется от POP-BOT. Кронштейн сервомотора непрерывно передвигается из крайнего левого положения в крайнее правое и обратно.

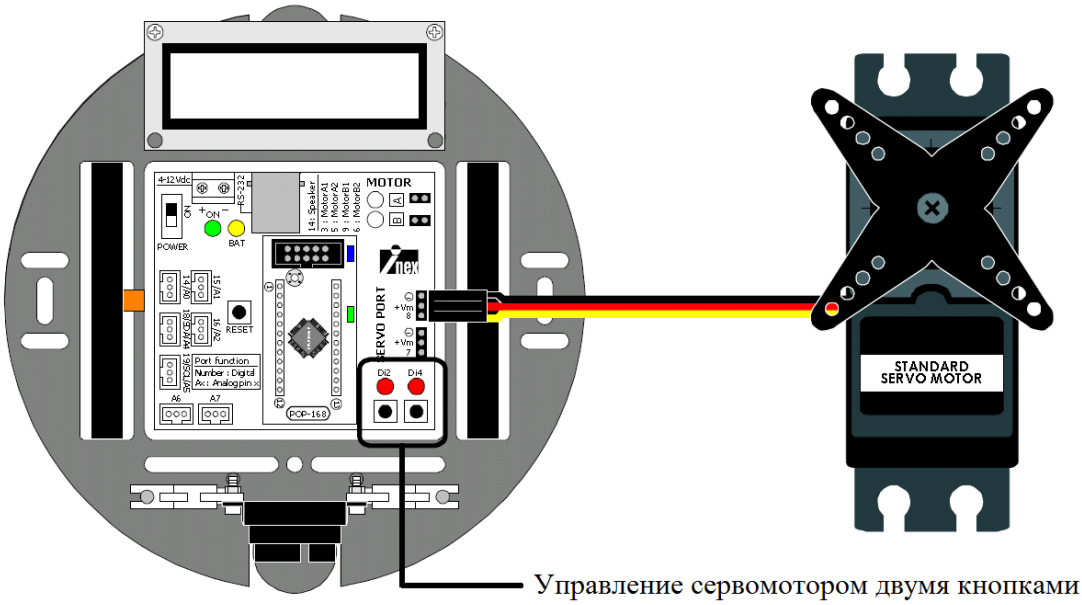
```
/*************************************************************************
* POP-BOT V1.0
* Filename : SimpleServo.pde
* Простейшее управление сервомотором
*************************************************************************/
int i;
void setup(){
      //---- Servo Motor ----//
      pinMode(8,OUTPUT); // Сервомотор
      pinMode(7,OUTPUT); // Сервомотор
}
void loop(){
      for (i=0;i<100;i++){
            digitalWrite(7, HIGH); // Сконфигурировать вывод Di7 как выход
            управления сервомотором
            digitalWrite(8, HIGH); // Сконфигурировать вывод Di8 как выход
            управления сервомотором
            delayMicroseconds(500); // Задержка для формирования
            положительного импульса
            digitalWrite(7,LOW);
            digitalWrite(8,LOW);
            delay(20); // Задержка для формирования отрицательного импульса
      }
      for (i=0;i<100;i++){
            digitalWrite(7, HIGH); // Сконфигурировать вывод Di7 как выход
            управления сервомотором
            digitalWrite(8, HIGH); // Сконфигурировать вывод Di8 как выход
            управления сервомотором
            delayMicroseconds(2300); // Задержка для формирования
            положительного импульса
            digitalWrite(7,LOW);
            digitalWrite(8,LOW);
            delay(20); // Задержка для формирования отрицательного импульса
      }
}
/*************************************************************************/
```

Листинг A16-1 : Файл SimpleServo.pde; скетч-файл Arduino для демонстрации того, как POP-BOT управляет сервомотором

Задание 16-2 : Управление сервомотором при помощи кнопки POP-BOT

В этом задании к коду предыдущего задания добавляется еще несколько строк, чтобы реализовать управление сервомотором с помощью кнопки платы управления POP-BOT.

A16.6 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A16-2.
A16.7 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A16.8 Отсоедините загрузочный кабель.
A16.9 Подключите стандартный RC-сервомотор к выходу SERVO PORT 7 или 8 платы управления POP-BOT.



A16.10 Включите POP-BOT. Нажимайте кнопки на выходах Di2 и Di4 и наблюдайте за работой робота.

Di2 используется для подачи на сервомотор команды перемещения в крайнее правое положение.
Di4 используется для подачи на сервомотор команды перемещения в крайнее левое положение.

Когда кронштейн сервомотора переместится в последнее крайнее положение, POP-BOT будет издавать звуковой сигнал, чтобы сообщить пользователю о завершении операции.

Чтобы управлять положением кронштейна сервомотора, можно нажать на кнопку и удерживать ее, или нажать и отпустить.

```
/*************************************************************************
* POP-BOT V1.0
* Filename : SwitchControlServo.pde
* Управление сервомотором с помощью 2 кнопок, подключенных к портам Di2 и Di4.
* Отображение действий на экране SLCD
*************************************************************************/
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int Old_i,i=1500,j=0,k;
void setup(){
      pinMode(8,OUTPUT); // Сервомотор
      pinMode(7,OUTPUT); // Сервомотор
      pinMode(14,OUTPUT); // Пьезоизлучатель
      pinMode(2,INPUT); // Левая кнопка
      pinMode(4,INPUT); // Правая кнопка
      pinMode(txPin,OUTPUT); // Вывод обмена с SLCD
      MySerial.begin(9600); // Обмен с SLCD
      delay(1000);
      i=1500; // Центральное значение для сервомотора
}
void loop(){
      if(digitalRead(2)==0){ // Нажата кнопка увеличения угла
            if(i<2500){ // Проверка на максимальное значение
                  i+=20; // Если меньше, увеличить значение переменной
            }
            else {Beep();}
      }
      if(digitalRead(4)==0){ // Нажата кнопка уменьшения угла
            if(i>400){ // Проверка на минимальное значение
                  i-=20; // Если больше, уменьшить значение переменной
            }
            else {Beep();}
      }
      if (i!=Old_i){
            MySerial.print(0xFE,BYTE); // Очистка экрана ЖКИ
            MySerial.print(0x01,BYTE);
            MySerial.print(i,DEC); // Отображение положения на ЖКИ
            Old_i =i ;
      }
      digitalWrite(7, HIGH); // Сконфигурировать вывод Di7 как выход управления
      сервомотором
      digitalWrite(8, HIGH); // Сконфигурировать вывод Di8 как выход управления
      сервомотором
      delayMicroseconds(i); // Задержка для формирования положительного импульса
      digitalWrite(7,LOW);
      digitalWrite(8,LOW);
      delay(20); // Задержка для формирования отрицательного импульса
}
void Beep(){
      int i;
      for (i=0;i<600;i++){
            digitalWrite(14,HIGH);
            delayMicroseconds(150);
            digitalWrite(14,LOW);
            delayMicroseconds(150);
      }
}
/*************************************************************************/
```

Листинг A16-2 : Файл SwitchControlServo.pde; скетч-файл Arduino для демонстрации того, как POP-BOT управляет сервомотором с помощью двух кнопок

Операция программирования

Целью этого задания было продемонстрировать возможность управления положением исполнительного органа сервомотора с помощью кнопок управления. Нам понадобилось добавить 2 кнопки, чтобы изменять положение сервомотора и передавать значение положения на дисплей ЖКИ с последовательным управлением робота POP-BOT. Эти значения можно использовать в качестве образцовых в задачах управления сервомоторами.

В коде проверялось, нажаты ли кнопки, подключенные к портам Di2 и Di4. Если была нажата кнопка, подключенная к порту Di2, переменная i увеличивала свое значение на 20. Если была нажата кнопка, подключенная к порту Di4, переменная i уменьшала свое значение на 20. Значение этой переменной использовалось для определения ширины импульса управления сервомотором.

При достижении величиной переменной крайних значений (как максимального, так и минимального), вызывалась функция подачи звукового сигнала, чтобы информировать пользователя о завершении операции.

11 : Возможности робота POP-BOT по обнаружению объектов

В разделе 10 мы изучили методы управления сервомотором с помощью нашего робота POP-BOT. Важной составной частью такого управления явилась библиотека SoftwareServo. В этом разделе мы соберем вместе все знания об управлении сервомоторами и о считывании информации с датчиков. В этом разделе мы будем использовать датчик GP2D120. Мы доработаем наш POP-BOT, таким образом, что он превратится в мобильного робота для поиска объектов.

11.1 Превращение POP-BOT в мобильного робота для поиска объектов.

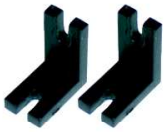
11.1.1 Список дополнительных комплектующих

GP2D120 с кабелем

Прямой крепеж x 4

Винты M3x15мм x 2

Винты M3x10мм x 4



Прямоугольный крепеж x 2

2мм саморезы x 2

Гайки M3 x 4

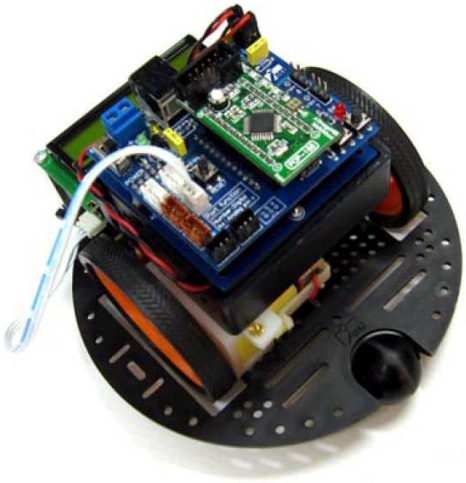
Стандартный серводвигатель

10мм пластиковая втулка x 2

33мм металлическая стойка x 2

11.1.2 Процедура изменения конструкции робота

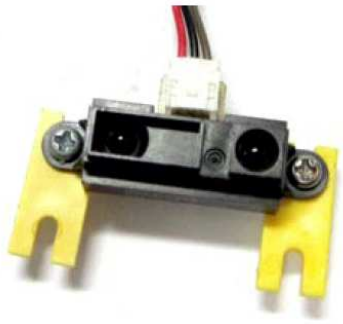
(1) Снимите все датчики с корпуса POP-BOT. Теперь перед нами простейшая форма мобильного робота POP-BOT.



(2) Снимите исполнительный кронштейн с сервомотора. Присоедините 2 металлических стойки длиной 33 мм к крепежным отверстиям сервомотора с помощью 10 мм пластиковых втулок и винтов M3 x 15 мм, согласно фотографии, помещенной ниже.

(3) Смонтируйте сервомотор с помощью присоединенных на шаге (2) стоек в передней части корпуса робота, используя винты M3 x 10 мм. Винты необходимо затянуть снизу.

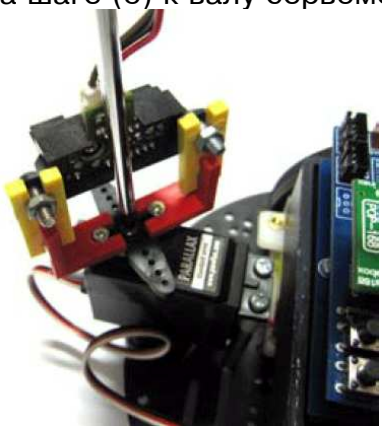
(4) Присоедините 2 прямых крепежных элемента к крепежным отверстиям модуля GP2D120, используя винты M3 x 10 мм и гайки M3.



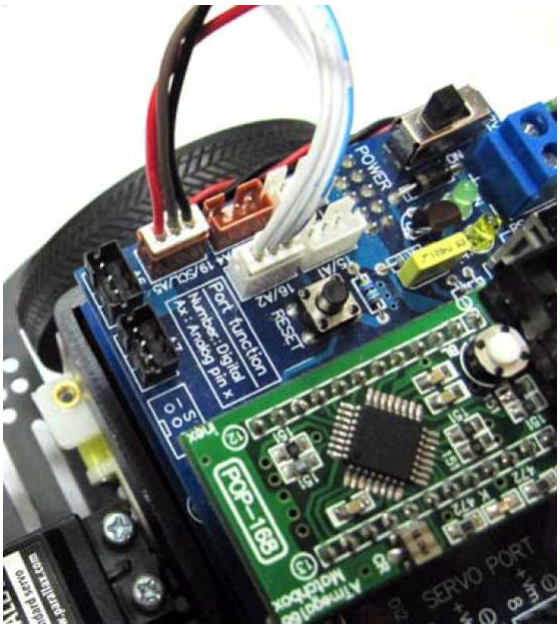
(5) Присоедините 2 прямоугольных крепежных элемента к кронштейну сервомотора, используя самые внутренние отверстия и 2-мм саморезы.

(6) Присоедините структуру GP2D120, созданную на шаге (4) к концам прямоугольных крепежных элементов.

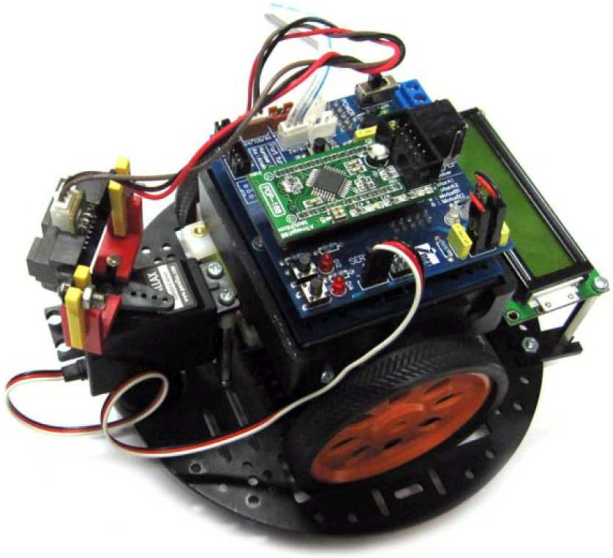
(7) Присоедините структуру кронштейна сервомотора, с датчиком GP2D120, созданную на шаге (6) к валу сервомотора. Затяните соединение винтом.



(8) Присоедините кабель сервомотора к выходу Servo порта 7. Убедитесь, что сервомотор подключен корректно. И наконец, подключите кабель GP2D120 к порту 19/SCL/A5 робота POP-BOT.

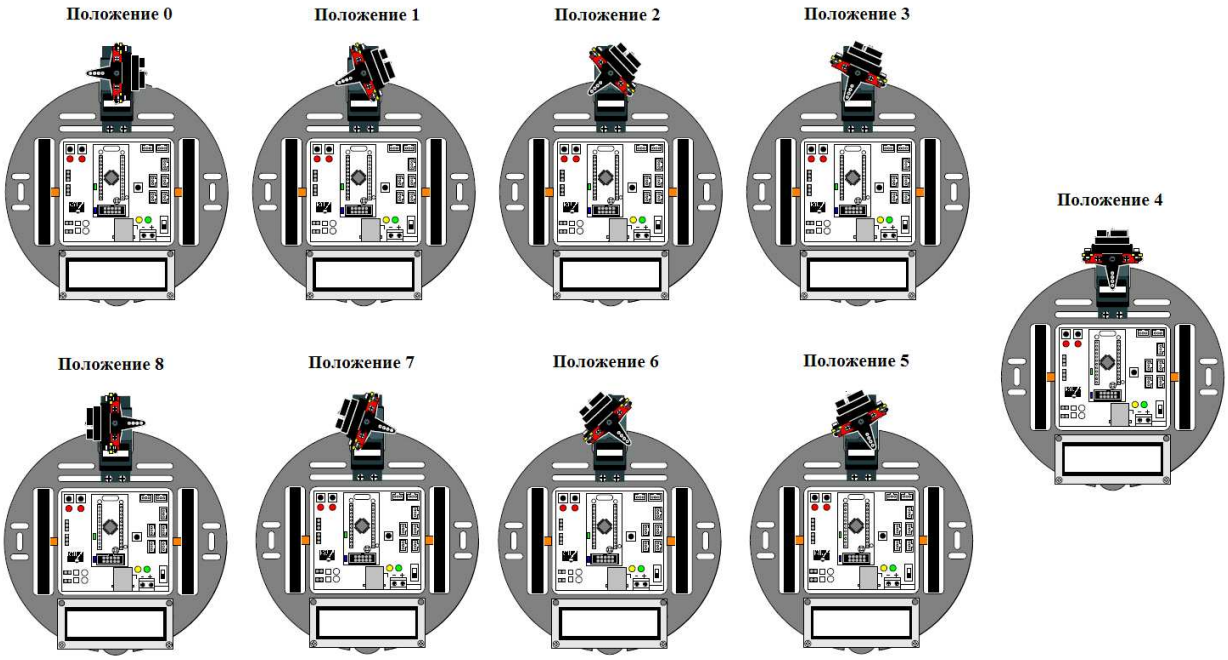


(9) Теперь изменение конструкции POP-BOT закончено, и он готов к программированию.



Задание 17 : POP-BOT ищет объекты

В этом задании демонстрируется, как искать объекты, перемещая кронштейн сервомотора. Робот POP-BOT, к которому подключен датчик GP2D120, размещенный на кронштейне сервомотора, будет перемещать кронштейн и проверять расстояние от датчика до попадающихся на пути его движения объектов. Имеются 9 фиксированных положений, в которых робот будет производить поиск объектов, как показано на рисунке ниже.



Робот POP-BOT будет читать значение показаний датчика на каждом шаге и отображать их на экране SLCD16x2. После прохождения всех 9 положений, микроконтроллер выберет максимальное значение, как результат поиска. Поскольку датчик дает самое высокое значение для самого близкорасположенного объекта, то в результате выполнения этого задания POP-BOT сможет определить правильное направление на целевой объект.

A17.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A17-1.
A17.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A17.3 Отсоедините загрузочный кабель.
A17.4 Задайте положение целевого объекта. Например, угол 67.5 градусов и расстояние 15 см от POP-BOT.

```
/*************************************************************************
* Filename : SeekingObject.pde
* Премещение кронштейна серводвигателя и поиск объектов, для отображения
* их положения на экране SCLD
*************************************************************************/
#include <SoftwareSerial.h>
#define rxPin 16
#define txPin 16
SoftwareSerial MySerial = SoftwareSerial(rxPin,txPin);
int PosValue[] = {460,610,760,1050,1340,1500,1660,1940,2220};
// Значения положения вала сервомотора
int GP2[9];
int j,Maximum,MAX_Point;
void setup(){
      //---- Servo Motor ----//
      pinMode(7,OUTPUT); // Сервомотор
      pinMode(14,OUTPUT); // Пьезоизлучатель
      pinMode(txPin,OUTPUT); // Вывод управления SLCD
      MySerial.begin(9600); // Обмен данными с SLCD
      delay(1000);
      LCD_Clear(); // LCD Clear Screen
}
void loop(){
      for(j=0;j<9;j++){ // Прохождение 9 положений
            Servo_Move(PosValue[j]); // Движение сервомотора
            GP2[j]=analogRead(5); // Чтение значения с GP2D120
            LCD_Clear();
            LCD_Show_Text(0x80,"Position");
            LCD_Show(0x89,j);
            LCD_Show_Text(0x8A,":= ");
            LCD_Show(0x8D,GP2[j]); // Отображение значения на экране SLCD
            delay(1000);
      }
      MAX_Point = GET_Point();
      Servo_Move(PosValue[MAX_Point]);
      // перемешение сервомотора в сторону цели.
      // Выбор максимального из 9 считанных значений
      LCD_Clear();
      LCD_Show_Text(0x80,"Selected :"); // Отображение максимального значения
      на экране ЖКИ
      LCD_Show(0x8A,MAX_Point);
      LCD_Show_Text(0xC0,"Value = ");
      LCD_Show(0xC8,Maximum);
      Beep();delay(5000);
      LCD_Clear(); // Очистка экрана ЖКИ
}
/** Вычисление положения цели **/
int GET_Point(){
      int i,Old=0,max_;
      for(i=0;i<9;i++){
            if(GP2[i]>Old){
                  Old=GP2[i];
                  max_=i;
            }
      }
      Maximum=Old;
      return(max_);
}
```

```
/** Управление положением сервомотора **/
void Servo_Move(int val){
      int i;
      for(i=0;i<20;i++){
            digitalWrite(7, HIGH); // Сконфигурировать порт 7 как сервопорт
            delayMicroseconds(val); // Задержка для формирования
            положительного импульса
            digitalWrite(7,LOW);
            delay(20); // Задержка для формирования отрицательного импульса
      }
}
/** Функция генерации звукового сигнала **/
void Beep(){
      int i;
      for (i=0;i<600;i++){
            digitalWrite(14,HIGH);
            delayMicroseconds(150);
            digitalWrite(14,LOW);
            delayMicroseconds(150);
      }
}
/** Функции для SLCD **/
void LCD_Show(int Position,int x){
      MySerial.print(0xFE,BYTE); // Очистить экран ЖКИ
      MySerial.print(Position,BYTE);
      MySerial.print(x,DEC); // Отобразить данные в десятичной форме
}
void LCD_Show_Text(int Position,char* x){
      MySerial.print(0xFE,BYTE); // Очистить экран ЖКИ
      MySerial.print(Position,BYTE);
      MySerial.print(x); // Show text
}
void LCD_Clear(){
      MySerial.print(0xFE,BYTE); // Очистить экран ЖКИ
      MySerial.print(0x01,BYTE);
}
/*********************************************************************/
```

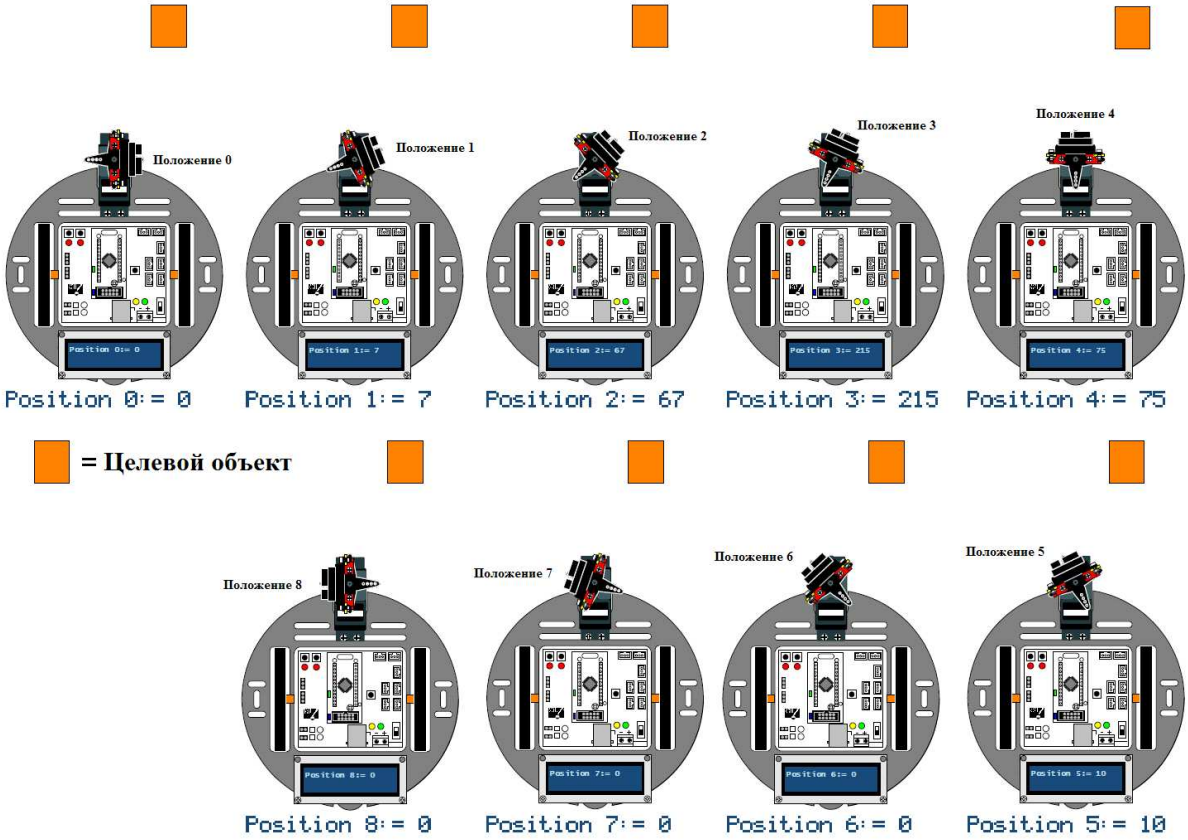
Листинг A17-1 : Файл SeekingObject.pde; скетч-файл Arduino для тестирования возможностей POP-BOT по обнаружению объктов

A17.5 Включите питание робота и наблюдайте за его работой.

После включения питания POP-BOT будет управлять сервомотором, чтобы переместить датчик GP2D120 в крайнее правое положение; Положение 0. Оно соответствует углу 0 градусов. Контроллер POP-BOT считывает данные с GP2D120 и отображает их на экране SLCD16x2 в следующем виде :

Position 0:= 0 *(для некоторых роботов это значение может быть другим)*

Далее POP-BOT перемещает датчик GP2D120 в Положение 1 (угол 22.5 градуса), снова считывает данные и отображает их на экране SLCD. Далее робот будет выполнять те же действия до достижения Положения 8.



После этого микроконтроллер выберет максимальное из 9 значений положения и отобразит его на экране SLCD в следующей форме:

Selected : 3

Value = 215

Это значит, что POP-BOT обнаружил объект в положении 3. Угол будет составлять примерно 67.5 градусов.

Задание 18 : POP-BOT ищет мяч

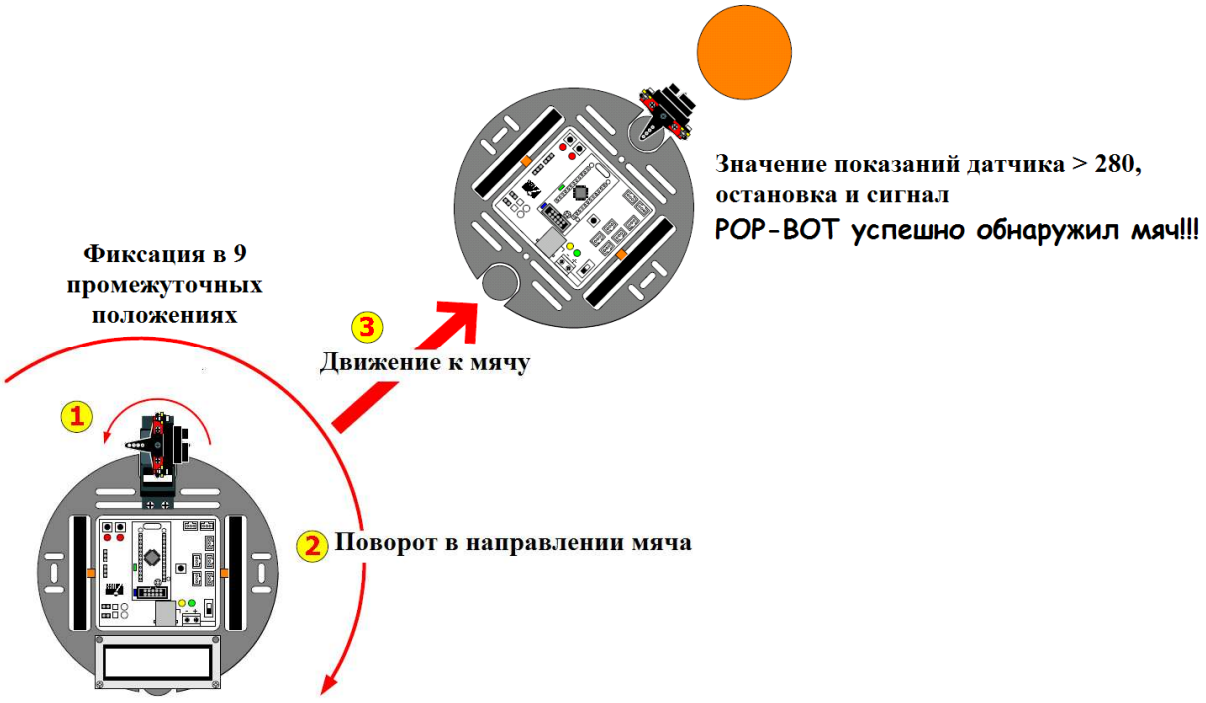
Это задание является модификацией Задания 17. Мы будем использовать наш код для решения реальной задачи. Робот POP-BOT будет перемещаться и искать целевой объект – мяч. Миссия будет считаться завершенной, когда POP-BOT приблизится к мячу, остановится и издаст звуковой сигнал.

A18.1 Откройте IDE Arduino и создайте скетч с кодом, показанным на Листинге A18-1.
A18.2 Переведите POP-BOT в режим Программирования. Загрузите скетч в робот.
A18.3 Отсоедините загрузочный кабель.
A18.4 Расположите целевой объект в любом удобном месте. Поместите робот POP-BOT на поле. Включите робот и наблюдайте за его работой.

Робот POP-BOT начинает управлять сервомотором, чтобы найти мяч. Поиск напоминает операцию, рассмотренную в Задании 17, но происходит быстрее, поскольку нет необходимости отображать результаты на экране SLCD. Робот POP-BOT будет двигаться в направлении, которое определит из максимальных показаний датчика.

Если все показания датчика оказались меньше 20, то это значит, что в данном направлении, на максимальном расстоянии обнаружения нет никаких объектов. Робот POP-BOT повернется вокруг, чтобы изменить направление поиска на противоположное.

Кроме того POP-BOT будет сравнивать показания датчика со значением 280, это будет означать, что POPBOT считает мячик найденным. Поскольку значение 280 указывает на весьма незначительное расстояние между POP-BOT и мячом. Если показание датчика не достигнет значения 280, POP-BOT будет продолжать искать мяч, повторяя рассмотренные операции.



```
/*****************************************************************************
* POP-BOT V1.0
* Filename : BallSeekerRobot.pde
* Поиск мяча и перемещение к нему.
*****************************************************************************/
int PosValue[] = {460,610,760,1050,1340,1500,1660,1940,2220}; // Положения
сервомотора
      int GP2[9];
      int j,Maximum,Position;
      void setup(){
      pinMode(3,OUTPUT); // Электродвигатель A1
      pinMode(5,OUTPUT); // Электродвигатель A2
      pinMode(6,OUTPUT); // Электродвигатель B2
      pinMode(9,OUTPUT); // Электродвигатель B1
      pinMode(7,OUTPUT); // Сервомотор
      pinMode(14,OUTPUT); // Пьезоизлучатель
      Beep();
      delay(2000);
}
void loop(){
      Motor_Stop();
      Servo_Home();
      for(j=0;j<9;j++){ // Установка положения 9 точек поиска
            Servo_Move(PosValue[j]); // Движение сервомотора
            GP2[j]=analogRead(5); // Чтение значения с GP2D120
      }
      Position=MAX_Point(); // Проверка условия отсутствия объектов
      if(Maximum<20){
            Spin_Right(150);delay(600); // Разворот, если значение меньше <
      20.
      }
      else if(Maximum>280){ // Определение положения мяча
            Servo_Move(PosValue[MAX_Point()]);
            Beep(); // Финишный гудок
            while(1);
      }
      else{
            switch(Position){ // Подпрограмма поиска
                  case 0:     Spin_Right(200);
                              delay(320);
                              Forward(200);
                              delay(400-Maximum);
                        break;
                  case 1:     Spin_Right(200);
                              delay(240);
                              Forward(200);
                              delay(400-Maximum);
                        break;
            case 2:     Spin_Right(200);
                        delay(160);
                        Forward(200);
                        delay(400-Maximum);
                  break;
            case 3:     Spin_Right(200);
                        delay(80);
                        Forward(200);
                        delay(400-Maximum);
                  break;
```

```
          case 4:     Forward(200);
                        delay(400-Maximum);
                break;
            case 5:     Spin_Left(200);
                        delay(80);
                        Forward(200);
                        delay(400-Maximum);
                break;
            case 6:     Spin_Left(200);
                        delay(160);
                        Forward(200);
                        delay(400-Maximum);
                break;
            case 7:     Spin_Left(200);
                        delay(240);
                        Forward(200);
                        delay(400-Maximum);
                        break;
            case 8:     Spin_Left(200);
                        delay(320);
                        Forward(200);
                        delay(400-Maximum);
            }
      }
}
/** Вычисление выбранного положения **/
int MAX_Point(){
      int i,Old=0,max_;
      for(i=0;i<9;i++){
            if(GP2[i]>Old){
                  Old=GP2[i];
                  max_=i;
            }
      }Maximum=Old;
      return(max_);
}
/** Управление положением сервомотора **/
void Servo_Home(){
      Servo_Move(460);
      Servo_Move(460);
      Servo_Move(460);
      Servo_Move(460);
}
void Servo_Move(int val){
      int i;
      for(i=0;i<5;i++){
            digitalWrite(7, HIGH); // Сконфигурировать порт 7 как выход  Servo
            delayMicroseconds(val); // Задержка для формирования
            положительного импульса
            digitalWrite(7,LOW);
            delay(20); // Задержка для формирования отрицательного импульса
      }
}
```

```
void Forward(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Motor_Stop(){
      digitalWrite(3,LOW);
      digitalWrite(5,LOW);
      digitalWrite(6,LOW);
      digitalWrite(9,LOW);
}
void Spin_Left(int speed){
      analogWrite(5,speed);
      digitalWrite(3,LOW);
      analogWrite(6,speed);
      digitalWrite(9,LOW);
}
void Spin_Right(int speed){
      analogWrite(3,speed);
      digitalWrite(5,LOW);
      analogWrite(9,speed);
      digitalWrite(6,LOW);
}
/** Функция генерации звукового сигнала **/
void Beep(){
      int i;
      for (i=0;i<600;i++){
            digitalWrite(14,HIGH);
            delayMicroseconds(150);
            digitalWrite(14,LOW);
            delayMicroseconds(150);
      }
}

/*************************************************************************/
```

Листинг А1-1 : Файл BallSeekerRobot.pde; скетч-файл Arduino для демонстрации того, как POP-BOT ищет и находит мяч

Таблица сокращений, использованных в руководстве

Сокращение		Расшифровка
Русское	**Английское**	
АЦП	ADC	Аналогово-цифровой преобразователь
ЖКИ	LCD	Жидкокристалический индикатор
ИК	IR	Инфракрасный
ОЗУ	RAM	Оперативное запоминающее устройство (запоминающее устройство с произвольной выборкой)
ПЗУ	ROM	Постоянное запоминающее устройство
СОЗУ	SRAM	Статическое оперативное запоминающее устройство
ШИМ	PWM	Широтно-импульсная модуляция
ЭППЗУ	EEPROM	Электрически перепрограммируемое постоянное запоминающее устройство
	AC	Переменный ток
	API	Прикладной пользовательский интерфейс
	DC	Постоянный ток
	ISP	Внутрисхемное программирование
	LED	Светоизлучающий диод, светодиод
	RPM	Обороты в минуту
	SLCD	Жидкокристаллический индикатор с последовательным интерфейсом
	TTL	Транзисторно-транзисторная логика