ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

Курсовой проект
по дисциплине «Объектно-ориентированное программирование»
«Проектирование и программирование робота на базе Arduino для объезда препятствий»

Выполнил студент
гр. 3331506/10401 _____ Абраменко А. С.
(подпись)

Работу принял _____ Ананьевский М.С.
(подпись)

Санкт-Петербург
2024 г.

Оглавление

Введение

Цель проекта – Изучить конструктивные и программные особенности Arduino для решения задач навигации. Спроектировать и запрограммировать робота на базе Arduino для решения навигационных задач, связанных с объездом препятствий.

Актуальность: роботы, которые оборудованы датчиками расстояния (например, ультразвуковым датчиком) и другими элементами, такими как сервоприводы и драйверы двигателей, широко используются для решения навигационных задач робототехники, таких как объезд препятствий и формирование образа окружающей среды.

Задачи:

1. Произвести сборку робота, используя инструменты Arduino (самостоятельно подобрать микроконтроллеры, платы, драйверы, датчики, двигатели и др.) и другие необходимые элементы (провода, крепежные соединения и др.)
2. Изучить особенности программной реализации навигационных задач с помощью средств Arduino – ультразвукового модуля и сервопривода, а именно библиотеки, которые используются для управления данными элементами.

1. Среда разработки Arduino

Arduino — это комбинация аппаратной и программной частей для простой разработки электроники. Аппаратная часть включает в себя большое количество видов плат Arduino со встроенными программируемыми микроконтроллерами. Программная часть состоит из интегрированной среды разработки Arduino IDE - упрощенного языка программирования, огромного множества готовых функций и библиотек.

Все платы Arduino содержат основные компоненты, необходимые для программирования и совместной работы с другими схемами (см. рис. 1 в приложении):

- микроконтроллер Atmel;
- USB-интерфейс для программирования и передачи данных;
- стабилизатор напряжения и выводы питания;
- контакты входов ввода-вывода; индикаторные светодиоды (Debug, Power, Rx,
- Tx);
- кнопку сброса;
- встроенный последовательный интерфейс программирования (ICSP).

Основной элемент платы Arduino - микроконтроллер Atmel. На большинстве плат Arduino, включая Arduino Uno, установлен микроконтроллер ATmega.

Микроконтроллер исполняет весь скомпилированный код программы. Язык Arduino предоставляет доступ к периферийным устройствам микроконтроллера: аналого-цифровым преобразователям (ADCs), цифровым портам ввода-вывода, коммуникационным шинам и последовательным интерфейсам. На плате все эти порты выведены на штырьковые контакты. К тактовым контактам микроконтроллера A Tmega подключен кварцевый

резонатор на 16 МГц. С помощью кнопки сброса выполнение программы можно перезапустить.

У контроллеров Arduino к большинству контактов ввода-вывода можно подключить внешние схемы. Все контакты могут служить цифровыми входами и выходами. Часть контактов Arduino могут также действовать в качестве аналоговых входов. Многие из контактов работают в режиме мультиплексирования и выполняют дополнительные функции: различные коммуникационные интерфейсы, последовательные интерфейсы, широтно-импульсные модуляторы и внешние прерывания.

Возможно, самая важная особенность Arduino - непосредственное программирование через USB-порт, без дополнительного программатора. Эту функцию обеспечивает загрузчик Arduino, записанный в микроконтроллер ATmega на заводе-изготовителе, и позволяющий загружать пользовательскую программу на плату Arduino по последовательному порту USART.

2. Сборка робота – обзор составных частей

2.1 Плата Arduino Uno

Arduino Uno (см. рис. 1 в приложении) - основная плата линейки Arduino, она укомплектована микроконтроллером ATmega 328 и микросхемой l 6U2 преобразователя USB. Имеет продуманное исполнение, небольшой размер, множество библиотек и примеров кода.

Подключение к пинам микроконтроллера выполняется через штыревые линейки, распаянные по обе стороны платы. Таким образом разработчик может связать ATMega328 с внешними устройствами при помощи макетных проводов. Также под топологию Arduino Uno создано огромное количество шилдов (расширений), обеспечивающих дополнительный функционал путём их каскадного включения. В данном проекте я буду также использовать плату расширения Aduino sensor shield v5.0, о которой будет сказано позднее.

Для того, чтобы плата Arduino Uno могла функционировать, на неё необходимо подать питание. Сделать это можно несколькими способами, а именно:

1. Запитать непосредственно через USB-разъём с помощью шнура для программирования или связи с ПК;
2. Запитать от AC/DC адаптера с выходным напряжением 7-12В, подключившись через специальный разъём внешнего питания.
3. Подать напряжение 7-12В напрямую на вход Vin, который расположен на штыревой колодке питающей группы. При этом минусовой контакт источника питания следует соединить с одним из контактов GND платы.

Для реализации проекта я буду использовать способ 2. Непосредственно к плате Arduino Uno будет подведено питание, соединение с остальными устройствами робота будет осуществляться через плату расширения, о которой было упомянуто выше.

2.2 Плата расширения Aduino Sensor Shield v5.0

Коммутационная плата Arduino Sensor Shield V5.0 (см. рис. 2 в приложении) предназначена для расширения функциональности контроллеров на платформе Arduino UNO. Плата позволяет подключить различные вариации внешних устройств, таких как датчики, сервомашинки, реле, кнопки, потенциометры и т.д.

Питание платы осуществляется или с Arduino, или от внешних источников питания (блоков питания, батарей).

Соединение на штырьках с платой Arduino Uno, которое используется в проекте, представлено на рис.3 в приложении. Посредством именно этого шилда будет осуществляться подключение ультразвукового датчика расстояния и сервопривода.

2.3 Драйвер двигателей L298N

Для работы двигателей постоянного тока, осуществляющих перемещения различных частей роботов, нужны токи значительной величины. Однако управляющие сигналы платы Arduino имеют значения силы тока в 800 mA, а зачастую и меньше, что недостаточно для управления даже самым маленьким двигателем постоянного тока напрямую.

Чтобы подключить двигатель к Arduino, есть несколько способов:

1. Можно использовать реле. Двигатель включается в отдельную электрическую сеть, не связанную с платой Arduino. Реле по команде микроконтроллера замыкает или размыкает контакты, тем самым включает или выключает ток. Соответственно, двигатель включается или выключается. Это самая простая схема, но она не позволит управлять скоростью и направлением вращения.

2. Можно использовать силовой транзистор. В таком случае возможно управлять током, проходящим через двигатель, а значит и управлять скоростью вращения шпинделя. Но для смены направления вращения этот способ не подойдет.

3. И наконец, можно использовать специальную схему подключения, называемую Н-мостом, с помощью которой можно изменять направление движения шпинделя двигателя. Сегодня микросхемы и модули, содержащие два или больше Н-моста, получили широкое распространение. На этом способе управления двигателями постоянного тока я и остановлюсь.

Модуль, содержащий Н-мосты, называется Motor Shield или драйвером двигателей. Это – плата расширения для Ардуино, которая обеспечивает работу двигателей постоянного тока и шаговых двигателей. Самыми популярными платами Motor Shield являются схемы на базе чипов L298N и L293D, которые могут управлять несколькими двигателями.

Для простоты сборки, в проекте будет использован двухканальный драйвер на основе микросхемы L298N (см. рис. 4 в приложении). Она довольно мощная, имеет хороший запас по максимальному току (3А на канал).

Модуль используется для управления двигателями с напряжением от 5 до 35 В. При помощи одной платы L298N можно управлять сразу двумя двигателями.

На плате имеется возможность выбора источника напряжения – Motor Shield может питаться как от Ардуино, так и от внешнего источника. На плате имеется светодиод, который показывает, работает ли устройство. Все это делает использование драйвера очень простым и надежным.

2.3.1 Принцип работы драйвера на основе микросхемы L298N

Н-мост является электронной схемой, которая состоит из четырех ключей с нагрузкой.

Главная функция Н-моста — менять полярность на нагрузке, что является необходимым условием для многих примеров управления двигателями. В качестве нагрузки используем двигатель постоянного тока, поэтому при смене полярности, произойдет и смена направления вращения

двигателя (это зависит от того на какой контакт мотора подается плюс, а на какой минус).

На представленной схеме (см. рис. 5,6 в приложении) системы Н-моста, к верхней части конструкции присоединен положительный провод, а к нижней – отрицательный. Когда ключи S1 и S4 замкнуты, а S2 и S3 разомкнуты, мотор крутится в одну сторону, когда же S2 и S3 замкнуты, а S1 и S4 разомкнуты, мотор крутится в другую сторону.

Устройство драйвера на основе микросхемы L298N

- На клемму Motor power (мощность двигателя) можно подать напряжение от 5 до 35 В.
- Клемма Common ground – общая “масса”
- Винтовые клеммы Motor A Output и Motor B Output используются для подключения моторов
- Клемма +5V используется для электропитания датчиков и др.
- Перемычка 5V Select Jumper – джампер выбора (подача внешних 5В или внутренних)
- Выводы ENABLE A, B (ENA привязан к IN1, IN2; ENB к IN3, IN4) отвечают за раздельное управление каналами. Могут использоваться в двух режимах:

1)Условно "активном" режиме, когда ими будет управлять контроллер - высокий логический уровень (напряжение +5В, логическая “1”) разрешает вращение моторов, низкий (напряжение 0В, логический “0”) запрещает вне зависимости от состояния выводов "IN". Для регулировки скорости моторов, на "EN" выводы подается модулированный ШИМ (PWM) сигнал.

2)Условно "пассивном" режиме, когда на выводы "EN" подается высокий логический уровень (+5V). В данном режиме нельзя регулировать скорость двигателей, они будут всегда вращаться на полную скорость. Направление вращения будет задаваться по-прежнему, а вот для остановки в данном варианте, состояние выводов будет уже играть роль. Для остановки нужно будет подавать одноименные сигналы на выводы "IN".

• Клеммы Input1/2 подключают к двигателям правой стороны робота, Input3/4 - к двигателям его левой стороны. Чередование разноименных сигналов (высокий логический уровень или низкий) на парах выводов IN1, IN2 и IN3, IN4 задают направление вращения моторов.

Режим	**IN1**	**IN2**
Вращение в одну сторону	1	0
Вращение в обратную сторону	0	1
Блокировка мотора	1	1
Отключение мотора	0	0

2.3.2 Управление с помощью ШИМ-сигнала

Контакты ENA и ENB позволяют управлять моторами с помощью ШИМ-сигнала.

Можно заметить, что при частом попеременном включении и выключении электромотора частота вращения его ротора изменяется. Происходит регулировка скорости вращения путём периодичного включения и отключения тока, подаваемого на моторчик. Если изменять при этом время в подключённом состоянии период подключений и длину паузы между ними, можно регулировать скорость вращения мотора.

Широтно-Импульсная модуляция - это операция получения изменяющегося аналогового значения посредством цифровых прямоугольных импульсов - сигнала, который постоянно переключается между максимальным и минимальным значениями. Такой сигнал моделирует напряжение между максимальным значением (5 В) и минимальным (0 В), изменяя при этом длительность времени включения 5 В относительно включения 0 В.

Другими словами, с помощью ШИМ (широтно-импульсной модуляции) можно управлять скоростью вращения двигателя, поскольку позволяет изменять время, в течение которого напряжение подается на двигатель (см. рис. 7 в приложении).

На практике, при более длительных паузах между замыканием ключа и более коротких сигналах двигатель будет работать медленнее и стремиться к нулю. При уменьшении паузы и увеличении длины сигнала двигатель будет ускоряться и стремиться к предельной скорости.

Если замкнуть ключ и не допускать его размыкания, двигатель будет работать на полной скорости. Если же прекратить замыкать ключ, двигатель перестанет работать.

Чтобы продемонстрировать принцип работы драйвера на основе микросхемы L298N, я проведу несколько экспериментов.

В следующих экспериментах я наглядно покажу зависимость характера работы двигателей от состояния выводов EN и IN.

2.3.3 Эксперимент 1.

Схема для эксперимента представлена на рис. 8 в приложении.

1. Чтобы подавать питание на драйвер необходимо заранее снять 5-ти вольтовой джампер (выбор внешних 5-ти вольт).

2. Подключаю два мотора, отвечающие за вращение колес на одной стороне робота, к первой паре винтовых клемм Motor A Output, два других мотора, отвечающих за вращение колес на другой стороне робота, ко второй паре клемм Motor B Output. Работа четырех двигателей с помощью двухканального драйвера возможна благодаря параллельному подключению пар двигателей, при котором мощность остается неизменной. Соответственно, за работу пары двигателей на одной стороне робота будет отвечать вывод ENA, а за работу пары двигателей на другой стороне - ENB

Подключаю драйвер с двигателями к монтажной плате следующим образом (см. рис.). В этой конструкции на IN1 и IN3 подается напряжение 5 В, а IN2 и IN4 подключены к “массе”. Как уже было сказано выше, при таком

подключении моторы на обеих сторонах робота должны вращаться, но этого не происходит из-за состояния ENA и ENB. В конструкции на ENA подается напряжение 5 В, а ENB подключен к “массе”. Поэтому мы можем наблюдать вращение двух колес на стороне, за которую отвечал вывод ENA.

2.3.4 Эксперимент 2.

Схема для эксперимента представлена на рис. 9 в приложении.

Подключаю драйвер с двигателями к монтажной плате. В этой конструкции на IN1 и IN3 подается напряжение 5 В, а IN2 и IN4 подключены к “массе”. Также необходимо обратить внимание на состояние ENA и ENB. Обе клеммы подается напряжение 5 В, поэтому мы можем наблюдать вращение всех четырех колес, за которые отвечали выводы ENA и ENB, в одну и ту же сторону.

2.3.5 Эксперимент 3.

Схема для эксперимента представлена на рис. 10 в приложении.

Подключаю драйвер с двигателями к монтажной плате. В этой конструкции на IN2 и IN3 подается напряжение 5 В, а IN1 и IN4 подключены к “массе”. При такой конструкции колеса должны двигаться в противоположных направлениях, так как значения IN1 и IN3 и парных с ними IN2 и IN4 различаются. Также необходимо обратить внимание на состояние ENA и ENB. На обе клеммы подается напряжение 5 В, поэтому мы можем наблюдать вращение двух колес, за которые отвечал вывод ENA, и двух колес, за которые отвечал вывод ENB, в разные стороны.

Таким образом, были рассмотрены основные положения в работе драйвера двигателей – схема Н-моста и управление с помощью ШИМ-сигнала. В проекте я буду использовать драйвер для управления колёсами робота с помощью программы и микроконтроллера.

2.4 Мотор постоянного тока с редуктором

Мотор постоянного тока представлен на рис. 11 в приложении. Применительно к данным моторам-редукторам, можно выделить следующие технические характеристики:

- Диапазон напряжений питания: 3В – 8В;
- Номинальный ток потребления при напряжении 3,6В: 240 мА;
- Передаточное число редуктора: 1/48;
- Скорость вращения при напряжении 3,6В без нагрузки: 170 об/мин.
- Крутящий момент при напряжении 6В: 800 г/см;
- Диаметр вала: 5.4 мм;
- Габариты (для прямой модификации): 64мм х 20мм х 20мм;
- Масса: 26 грамм.

Как упоминалось в предыдущем пункте, потребление данного мотора составляет 250 мА (при напряжении 3,6В). Это означает, что прямое управление с выводов Arduino здесь неуместно. А если учесть, что в представленной задаче необходимо четыре таких мотора, то задача сводится к использованию либо транзисторов, либо драйвера двигателей. Шагом ранее мы выбрали использование драйвера двигателей, так как полевой транзистор не будет позволять нам крутить колёса в обе стороны.

2.5 Сервопривод SG90 micro

Сервопривод - это механизм с электромотором с управлением. С его помощью можно вращать механический привод на заданный угол с заданной скоростью или усилием (см. рис. 12 в приложении). Включая и выключая электромотор, можно вращать выходной вал — конечную шестерню сервопривода, к которой можно прикрепить крестовину для передачи вращающего движения на рабочий орган.

Для контроля положения используется датчик обратной связи — энкодер, который будет преобразовывать угол поворота обратно в

электрический сигнал. Для этого часто используется потенциометр. При повороте бегунка потенциометра происходит изменение его сопротивления, пропорциональное углу поворота. Таким образом, с его помощью можно установить текущее положение механизма.

Сервопривод управляется ШИМ сигналом (см. рис.13 в приложении), точнее длиной импульса: минимальная (0 градусов) и максимальная (~180 градусов) длина импульса колеблется в зависимости от модели и производителя сервопривода.

Скорость сервопривода измеряется интервалом времени, который требуется рычагу сервопривода, чтобы повернуться на 60°. Характеристика 0,1 с/60° означает, что сервопривод поворачивается на 60° за 0,1 с.

В проекте я буду использовать компактный 9-ти граммовый сервопривод SG90 micro (см. рис. 14 в приложении). Сервопривод может быть подключен к Arduino непосредственно. Для этого от него идёт шлейф из трёх проводов:

- красный — питание; подключается к контакту 3.3/5V или напрямую к источнику питания
- коричневый или чёрный — земля
- жёлтый— сигнал; подключается к цифровому выходу Arduino

В данном проекте я использую сервопривод для поворота ультразвукового модуля, установленного с помощью специального пластмассового адаптера и поворачивающегося вместе с рабочим органом сервопривода.

2.6 Датчик расстояния HC-SR04 - ультразвуковой модуль

Ультразвуковой модуль HC-SR04 (см. рис. 15 в приложении) является прибором бесконтактного типа, и обеспечивает высокоточное измерение и стабильность. Работает по принципу эхолокации — посылает пучок ультразвука и получает его отражение с некоторой задержкой, с помощью которой и можно высчитать расстояние до объекта. Диапазон дальности его

измерения составляет от 2 до 400 см, работает при температурах от 0° до 60° С. Точность измерения составляет ± 1 см, рабочее напряжение датчика до 5,5 В.

Ультразвуковой дальномер HC SR04 имеет такие технические параметры:

- Питающее напряжение 5В;
- Рабочий параметр силы тока – 15 мА;
- Сила тока в пассивном состоянии < 2 мА;
- Обзорный угол – 15°;
- Сенсорное разрешение – 0,3 см;
- Измерительный угол – 30° (см. рис. 16 в приложении);
- Ширина импульса – 10-6 с.

Датчик оснащен четырьмя выводами (стандарт 2, 54 мм):

- Контакт питания положительного типа – +5В;
- Trig (T) – выход сигнала входа;
- Echo (R) – вывод сигнала выхода;
- GND – вывод «Земля».

При настройке ультразвукового датчика могут возникнуть трудности с определением расстояния до звукопоглощающих объектов, поскольку они способны полностью погасить излучаемый сигнал. Для идеальной точности измерения расстояния, поверхность изучаемого объекта должна быть ровной и гладкой.

В данном проекте я буду использовать ультразвуковой дальнометр для определения расстояния до ближайшего препятствия. Таким образом, необходимо установить его на «носу» робота, соединить с сервоприводом – таким образом датчик будет получать информацию о ближайших препятствиях на определенном диапазоне градусов.

2.7 Процесс сборки модели

Пошаговый процесс сборки модели, а также конечный результат представлены на рисунках 1- 6 в приложении 2.

Сборка модели происходила в таком порядке:

1. Закрепляю четыре двигателя постоянного тока с колёсами на нижнем корпусе робота, припаиваю к каждому из двигателей по два провода («+» и «-»). Для соединения этих двигателей с драйвером, установленном на верхней части нижнего корпуса, зажимаю провода на клеммах: двигатели по правой стороне – на MotorA, по левой стороне – на MotorB. Таким образом обеспечиваю раздельное управление колёсами на разных сторонах робота.
2. Устанавливаю на верхнем корпусе плату Arduino Uno с платой расширения, далее шестью проводами (на драйвере – на ENA, ENB, IN1-4) соединяю с пинами 1-7 на плате расширения. Таким образом обеспечиваю возможность управления двигателями с помощью программы
3. Питание платы Arduino Uno буду осуществлять с помощью двух аккумуляторов 18650 3500mAh, установленных на нижнем корпусе, с помощью AC/DC адаптера
4. Закрепляю сервопривод на «носу» верхнего корпуса, конструктивно устанавливаю ультразвуковой модуль. Обращаю внимание на ориентацию сервопривода – из центрального положения он крутится на 90 градусов влево и на 90 градусов вправо.
5. Подключаю сервопривод и ультразвуковой датчик к плате расширения. Сервопривод – на 7 пин, ультразвуковой модуль – на 8 (trigPin) и на 9 (echoPin).

Сборка ходового макета робота со встроенными электродвигателями, ультразвуковым датчиком закончена. Последний шаг – программная проверка исправности некоторых устройств и создание программы, обеспечивающей задачи проекта.

3 Создание программы

Перед тем, как писать программу для робота, следует разобраться в её особенностях, связанных с подключением к плате сервопривода и ультразвукового датчика.

3.1 Библиотека Servo

При работе с сервоприводами можно генерировать управляющие импульсы самостоятельно, но это настолько распространённая задача, что для её упрощения существует стандартная библиотека Servo.

Библиотека Servo позволяет осуществлять программное управление сервоприводами. Управление осуществляется следующими функциями:

attach() — присоединяет объект к конкретному выводу платы. Возможны два варианта синтаксиса для этой функции: servo.attach(pin) и servo.attach(pin, min, max). При этом pin — номер пина, к которому присоединяют сервопривод, min и max — длины импульсов в микросекундах, отвечающих за углы поворота 0° и 180°. По умолчанию выставляются равными 544 мкс и 2400 мкс соответственно. Возвращаемого значения нет.

write() — отдаёт команду сервоприводу принять некоторое значение параметра. Синтаксис: servo.write(angle), где angle — угол, на который должен повернуться сервопривод

writeMicroseconds() — отдаёт команду послать на сервопривод имульс определённой длины, является низкоуровневым аналогом предыдущей команды. Синтаксис следующий: servo.writeMicroseconds(uS), где uS — длина импульса в микросекундах. Возвращаемого значения нет.

read() — читает текущее значение угла, в котором находится сервопривод. Синтаксис: servo.read(), возвращается целое значение от 0 до 180;

attached() — проверка, была ли присоединён объект к конкретному пину. Синтаксис следующий: servo.attached(), возвращается логическая

истина, если объект была присоединён к какому-либо пину, или ложь в обратном случае;

detach() — производит действие, обратное действию attach(), то есть отсоединяет объект от пина, к которому был приписан. Синтаксис: servo.detach().

В библиотеке Servo для Arduino по умолчанию выставлены следующие значения длин импульса: 544 мкс — для 0° и 2400 мкс — для 180°.

Библиотеки для управления сервоприводами (Servo) и для работы с приёмниками/ передатчиками на 433 МГц VirtualWire используют одно и то же прерывание. Это означает, что их нельзя использовать в одном проекте одновременно. Существует альтернативная библиотека для управления сервомоторами — Servo2. Отличие заключается в том, что функция Servo.write(angle) задаёт не угол, а скорость вращения привода. В нашем случае ничего не мешает использовать библиотеку Servo.

Для проверки исправности сервопривода (вращение на 180 градусов, задержка в крайнем и в среднем положениях), а также для теста команд из библиотеки напишем и загрузим на плату простую программу:

```
#include <Servo.h> // подключаем библиотеку для работы с сервоприводом

Servo servo1; // объявляем переменную servo типа "servo1"

void setup() {
   servo1.attach(7); // привязываем сервопривод к аналоговому выходу 7
}

void loop() {
   servo1.write(0); // ставим угол поворота под 0
   delay(2000); // ждем 2 секунды

   servo1.write(90); // ставим угол поворота под 90
   delay(2000); // ждем 2 секунды

   servo1.write(180); // ставим угол поворота под 180
   delay(2000); // ждем 2 секунды
}
```

Сервопривод исправен – происходит последовательная смена положений рабочего органа, задержка в каждом положении на 2 секунды.

3.2 Управление ультразвуковым модулем

В данном проекте у ультразвукового датчика достаточно простая функция, поэтому я обойдусь без подключения сторонних библиотек.

Для проверки работоспособности датчика я использую простую программу, в которую заложена последовательность действий:

1. Коротким импульсом (2-5 микросекунды) переводим датчик расстояния в режим эхолокации, при котором в окружающее пространство высылаются ультразвуковые волны с частотой 40 КГц.
2. Ждем, пока датчик проанализирует отраженные сигналы и по задержке определит расстояние.
3. Получаем значение расстояния. Для этого ждем, пока HC SR04 выдаст на входе ECHO импульс, пропорциональный расстоянию. Мы определяем длительность импульса с помощью функции pulseIn, которая вернет нам время, прошедшее до изменения уровня сигнала (в нашем случае, до появления обратного фронта импульса).
4. Получив время, мы переводим его в расстояние в сантиметрах путем деления значения на константу (для датчика SR04 это 29.1 для сигнала «туда», столько же для сигнала «обратно», что в сумме даст 58.2).

   Далее – вывод значения на экран.

   Программа для проверки ультразвукового датчика:

```
#define PIN_TRIG 8
#define PIN_ECHO 9

long duration, cm;

void setup() {

  // Инициализируем взаимодействие по последовательному порту

  Serial.begin (9600);
  //Определяем вводы и выводы
```

```
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
}

void loop() {

  // Сначала генерируем короткий импульс длительностью 2-5 микросекунд.

  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(5);
  digitalWrite(PIN_TRIG, HIGH);

  // Выставив высокий уровень сигнала, ждем около 10 микросекунд. В этот момент
датчик будет посылать сигналы с частотой 40 КГц.
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  //  Время задержки акустического сигнала на эхолокаторе.
  duration = pulseIn(PIN_ECHO, HIGH);

  // Теперь осталось преобразовать время в расстояние
  cm = (duration / 2) / 29.1;

  Serial.print("Расстояние до объекта: ");
  Serial.print(cm);
  Serial.println(" см.");

  // Задержка между измерениями для корректной работы скетча
  delay(250);
}
```

Выведенное на экран значение расстояния в сантиметрах соответствует действительности, датчик работает исправно.

3.3 Итоговая программа для робота

Программа, используемая для корректной работы робота, представлена ниже. Она обеспечивает независимое движение колёс на разных сторонах машинки и непрерывную работу сервопривода – смену крайних и среднего положений. Если дистанция до ближайшего препятствия становится меньше 25 см, робот сдаёт назад и поворачивает направо.

Программа:

```
// Setup the servo motor
#include <Servo.h>
Servo myservo;            // initialization servo
```

```
int servposnum = 0;
int servpos = 0;

// Setup Motor A (front and rear) pins
int enableA = 1;
int pinA1 = 3;
int pinA2 = 2;

// Setup Motor B (front and rear) pins
int enableB = 6;
int pinB1 = 5;
int pinB2 = 4;

// Setup Ultrasonic Sensor pins
#define trigPin 8
#define echoPin 9


void setup() {
  // Configure the pin modes for each drive motor
   pinMode (enableA, OUTPUT);
   pinMode (pinA1, OUTPUT);
   pinMode (pinA2, OUTPUT);

   pinMode (enableB, OUTPUT);
   pinMode (pinB1, OUTPUT);
   pinMode (pinB2, OUTPUT);

   // Configure the pin modes for the Ultrasonic Sensor
   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);

   // Turn pin into servo driver.
   myservo.attach(7);
}

void loop() {
  // Main code goes here and will run repeatedly:

     car(); // function keeps moving car forward while distance > 25 cm
     avoid(); // function makes car go back, turn slightly right to move forward
in new direction

}

// Create motor functions
void motorAforward() {
digitalWrite (pinA1, HIGH);
digitalWrite (pinA2, LOW);
```

```
}
void motorBforward() {
digitalWrite (pinB1, LOW);
digitalWrite (pinB2, HIGH);
}
void motorAbackward() {
digitalWrite (pinA1, LOW);
digitalWrite (pinA2, HIGH);
}
void motorBbackward() {
digitalWrite (pinB1, HIGH);
digitalWrite (pinB2, LOW);
}
void motorAstop() {
digitalWrite (pinA1, HIGH);
digitalWrite (pinA2, HIGH);
}
void motorBstop() {
digitalWrite (pinB1, HIGH);
digitalWrite (pinB2, HIGH);
}
void motorAcoast() {
digitalWrite (pinA1, LOW);
digitalWrite (pinA2, LOW);
}
void motorBcoast() {
digitalWrite (pinB1, LOW);
digitalWrite (pinB2, LOW);
}
void motorAon() {
digitalWrite (enableA, HIGH);
}
void motorBon() {
digitalWrite (enableB, HIGH);
}
void motorAoff() {
digitalWrite (enableA, LOW);
}
void motorBoff() {
digitalWrite (enableB, LOW);
}

// Setup movement functions
void forward (int duration) {
motorAforward();
motorBforward();
delay (duration);
}
void backward (int duration) {
motorAbackward();
```

```
motorBbackward();
delay (duration);
}
void right (int duration) {
motorAbackward();
motorBforward();
delay (duration);
}
void left (int duration) {
motorAforward();
motorBbackward();
delay (duration);
}
void coast (int duration) {
motorAcoast();
motorBcoast();
delay (duration);
}
void breakRobot (int duration) {
motorAstop();
motorBstop();
delay (duration);
}
void disableMotors() {
motorAoff();
motorBoff();
}
void enableMotors() {
motorAon();
motorBon();
}

// Setup Ultrasonic Sensor distance measuring
int distance() {
  int duration, distance;
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(1000);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;
  return distance;
}

// Setup the main car function
void car() {
int distance_0;
distance_0 = distance();
  // Keep moving forward in a straight line while distance of objects > 25cm
  while(distance_0 > 25)
  {
```

```
      // Keep moving servo motor back and forth to scan surroundings
      // This allows the ultrasonic sensor to see more to its left and right
      if(servposnum == 0)
      {
       myservo.writeMicroseconds (1500);   //  set middle - 1500
       servposnum = 1;
       delay(100);
      }
      else if(servposnum == 1)
      {
       myservo.writeMicroseconds (1800);   // set left like “+300” by middle
       servposnum = 2;
      delay(100);
      }
      else if(servposnum  == 2)
      {
       myservo.writeMicroseconds (1500);   //  middle
       servposnum = 3;
       delay(100);
      }
      else if(servposnum == 3)
      {
       myservo.writeMicroseconds (1200);   // set right like “-300” by middle
       servposnum = 1;
       delay(100);
      }
      motorAon();
      motorBon();
      forward(1);
      distance_0 = distance();

  }
  breakRobot(0);

}
void avoid()
{
    // Go back and turn slightly right to move car in new direction if object
detected < 25cm away
    backward(320);
    right(100);
}
```

Ключевые константы в программе:

- Минимальное расстояние до ближайшего объекта – 25 см
- Среднее положение сервопривода -1500
- Крайнее левое положение сервопривода – 1800 (поворот на 45 градусов относительно среднего положения)

- Крайнее правое положение сервопривода – 1200 (поворот на -45 градусов относительно среднего положения)
- При обнаружении препятствия – назад на 320, направо на 100.

Заключение

В ходе работы были подробно изучены элементы, которые используются в конструкции робота для решения различных задач, такие как платы, сервоприводы, датчики расстояния на базе платформы Arduino. Сборка робота сопровождалась описанием взаимодействия всех составляющих элементов системы.

С программной стороны проекта были заранее затестированы и проверены некоторые элементы, такие как сервопривод и ультразвуковой датчик, и библиотека Servo. Также после конечной сборки робота была написана комплексная программа, отвечающая поставленным цели и задачам проекта.

Список литературы

1. Блум Джереми. Б71. Изучаем Arduino: инструметы и методы технического волшебства: Пер. с англ. - СПб.: БХВ-Петербург, 2015. - 336 с.: ил. ISBN 978-5-9775-3585

2. Монк С. Программируем Arduino: Основы работы со скетчами / С. Монк. - Санкт-Петербург : Питер, 2016. - 176 с. - ISBN 978-5-496-01956-9.

3. Соммер У. С61. Программирование микроконтроллерных плат Arduino/Freeduino. — СПб.: БХВ-. Петербург, 2012. — 256 с.: ил. — (Электроника). ISBN 978-5-9775-0727-1.

4. Сайт Александра Климова [Электронный ресурс] // Работаем с сервоприводами. URL: https://developer.alexanderklimov.ru/arduino/servo.php

5. GyverKit [Электронный ресурс] // Arduino и сервопривод. URL: https://kit.alexgyver.ru/tutorials/servo/

Приложение 1

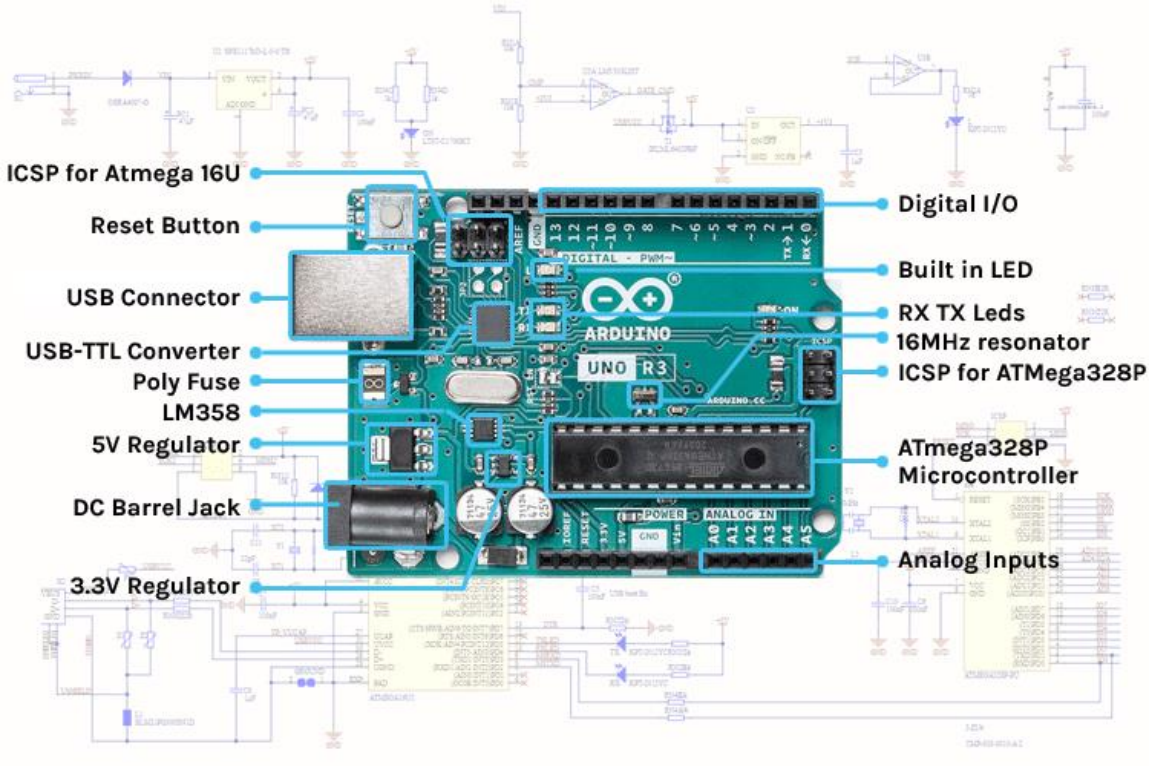


Рисунок 1 – плата Arduino Uno

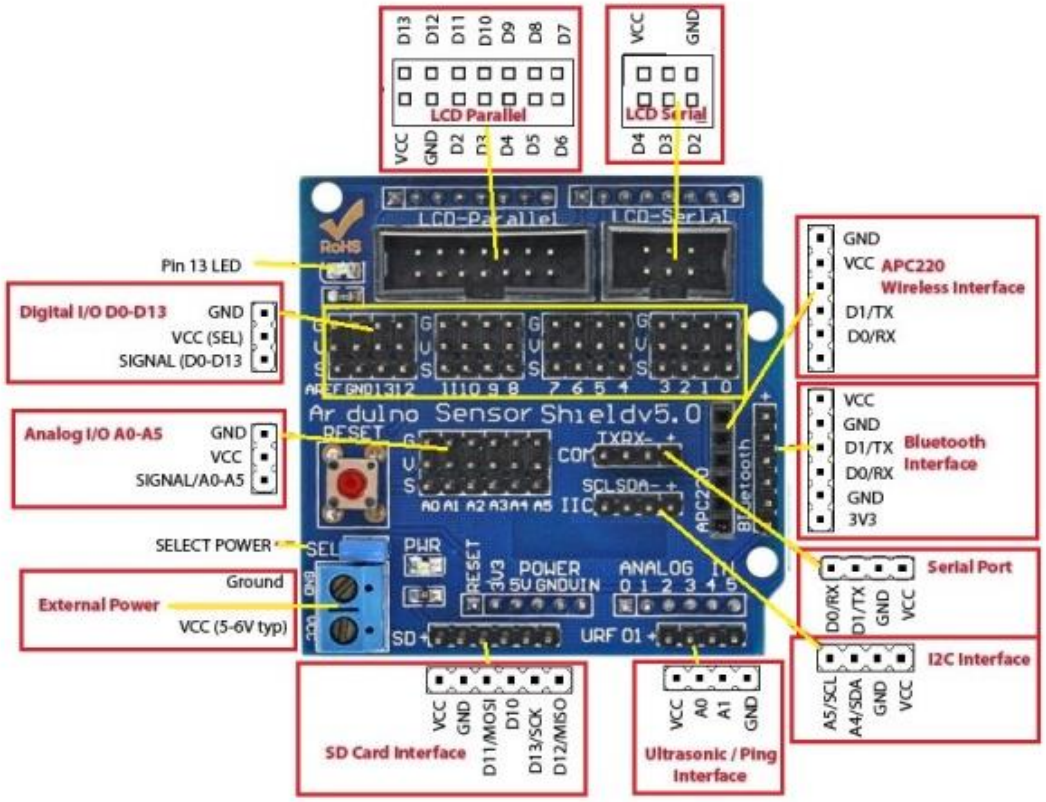


Рисунок 2 – плата расширения Arduino Sensor Shield V5.0

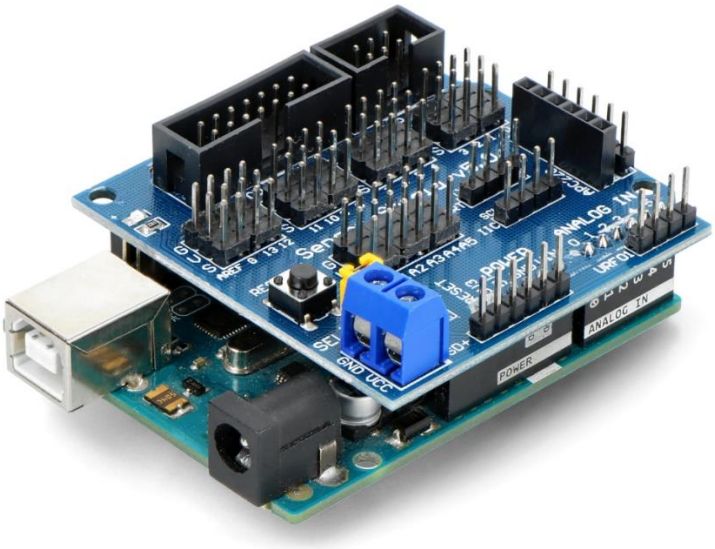
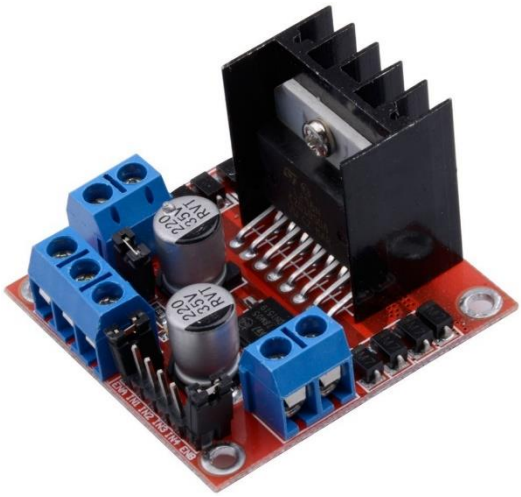


Рисунок 3 – конструкция из основной платы и платы расширения



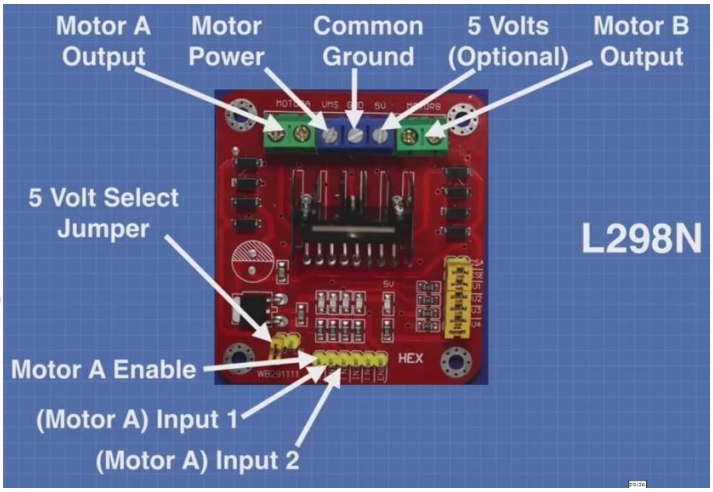


Рисунок 4 - Драйвер L298N

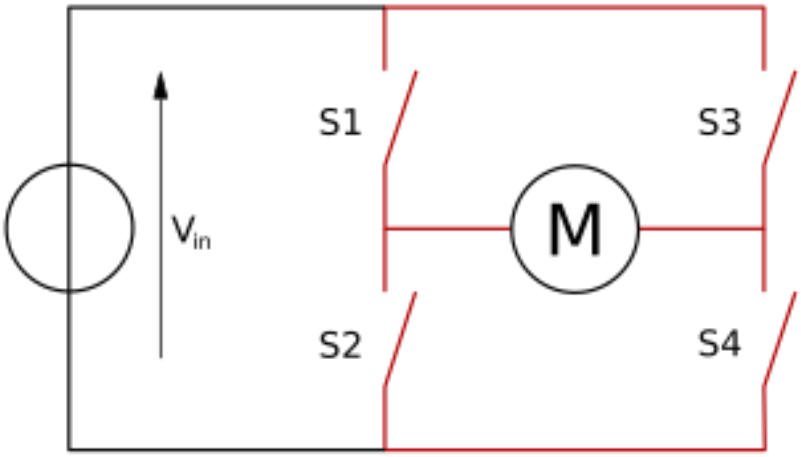


Рисунок 5 – Н-мост

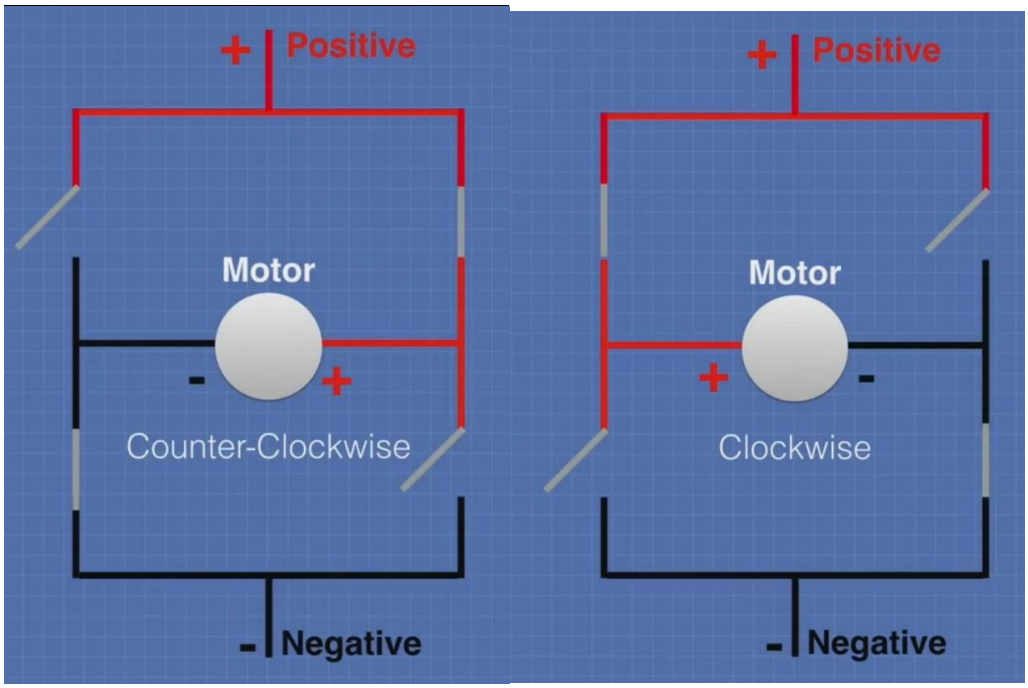


Рисунок 6 - Н-мост

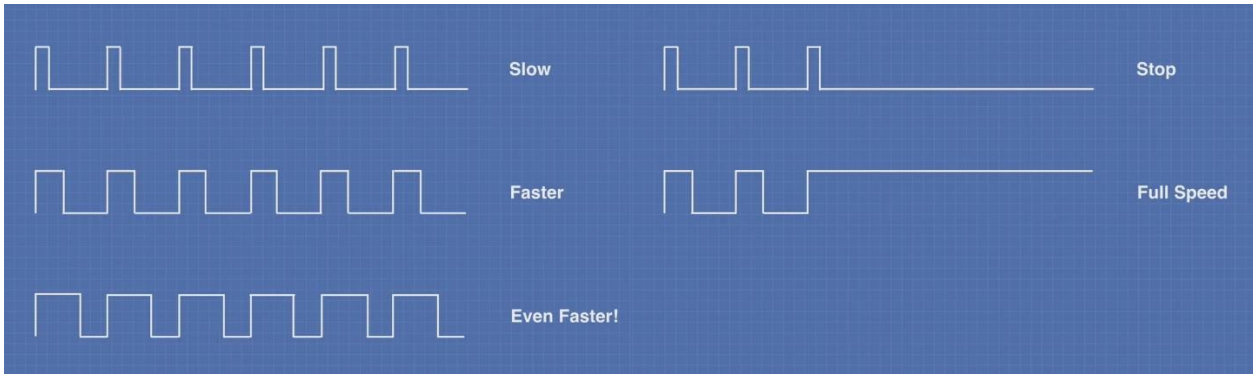


Рисунок 7 – ШИМ-модуляция

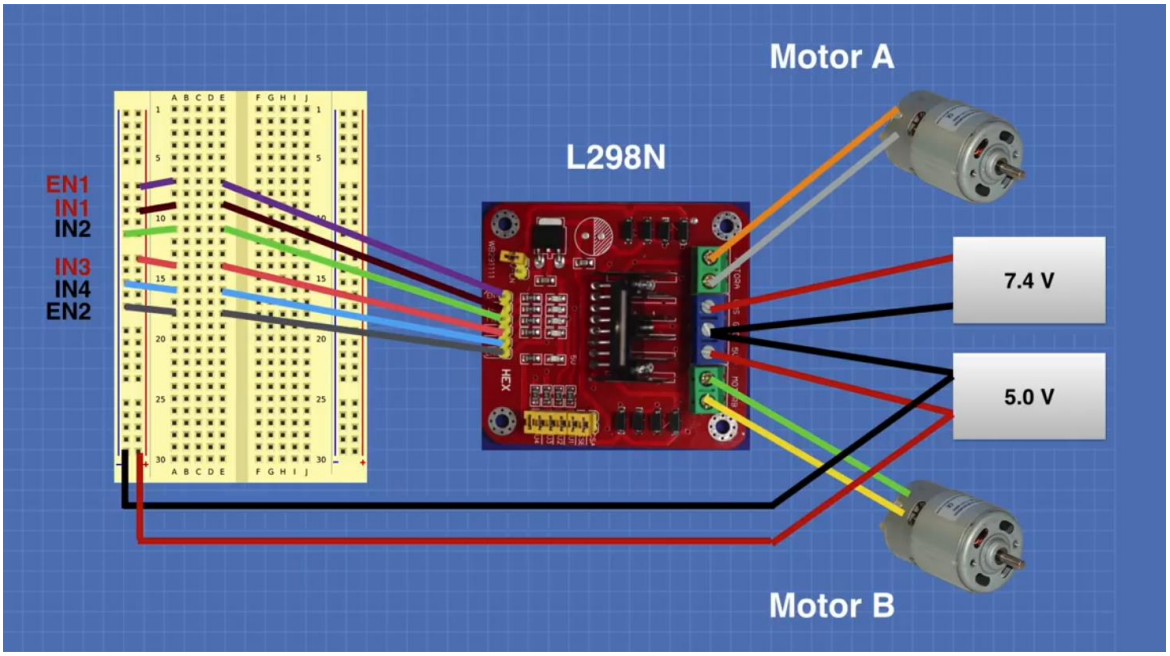


Рисунок 8 – Эксперимент 1

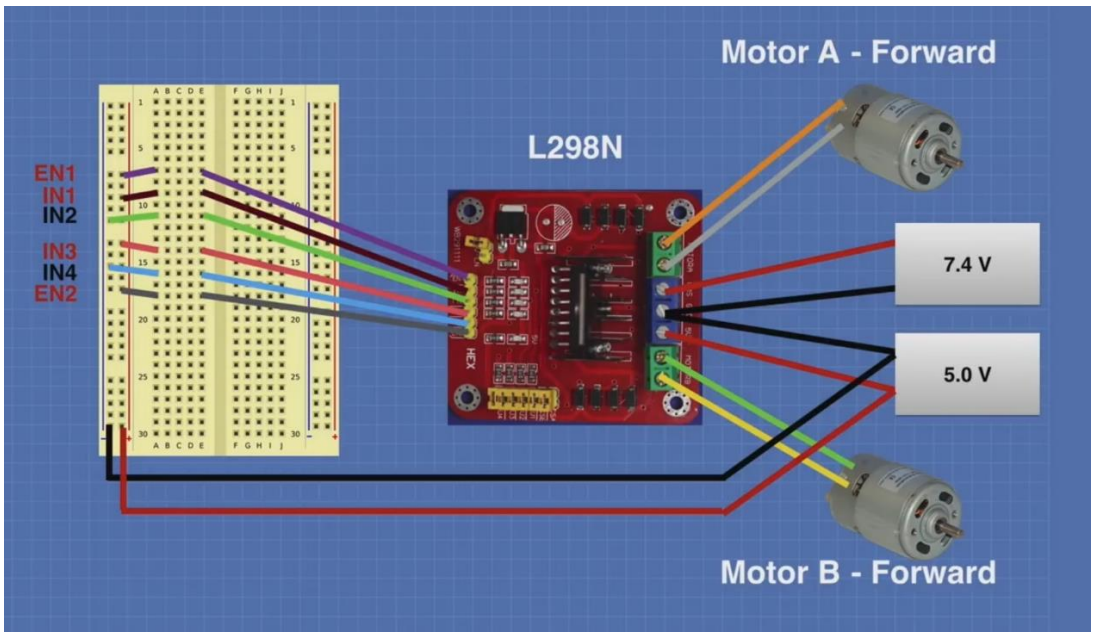


Рисунок 9 – Эксперимент 2

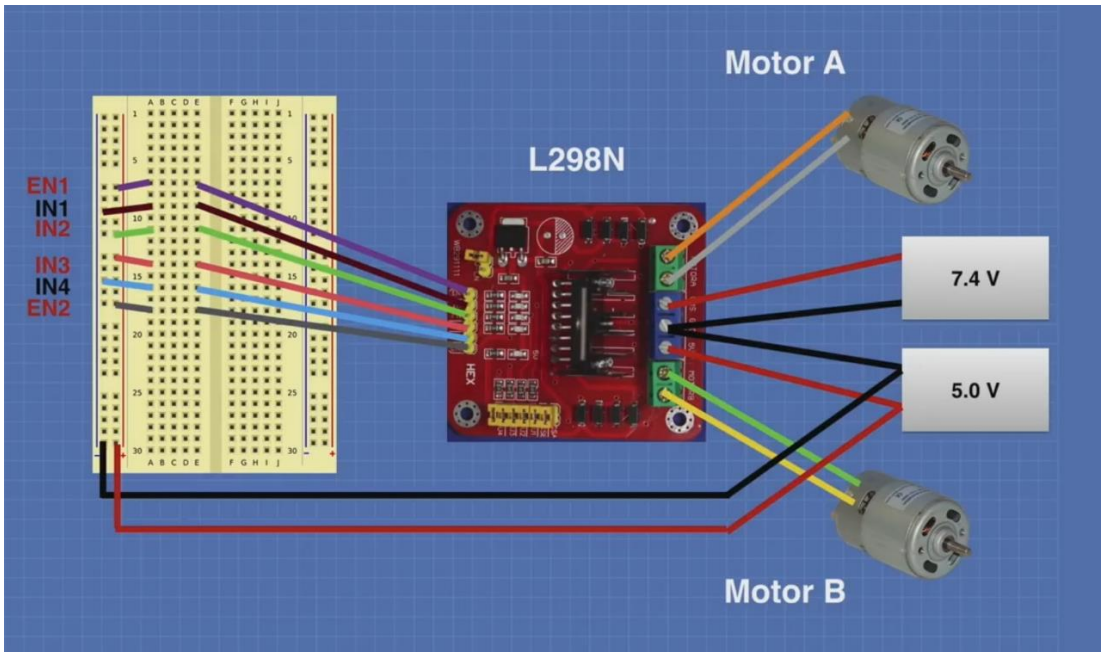
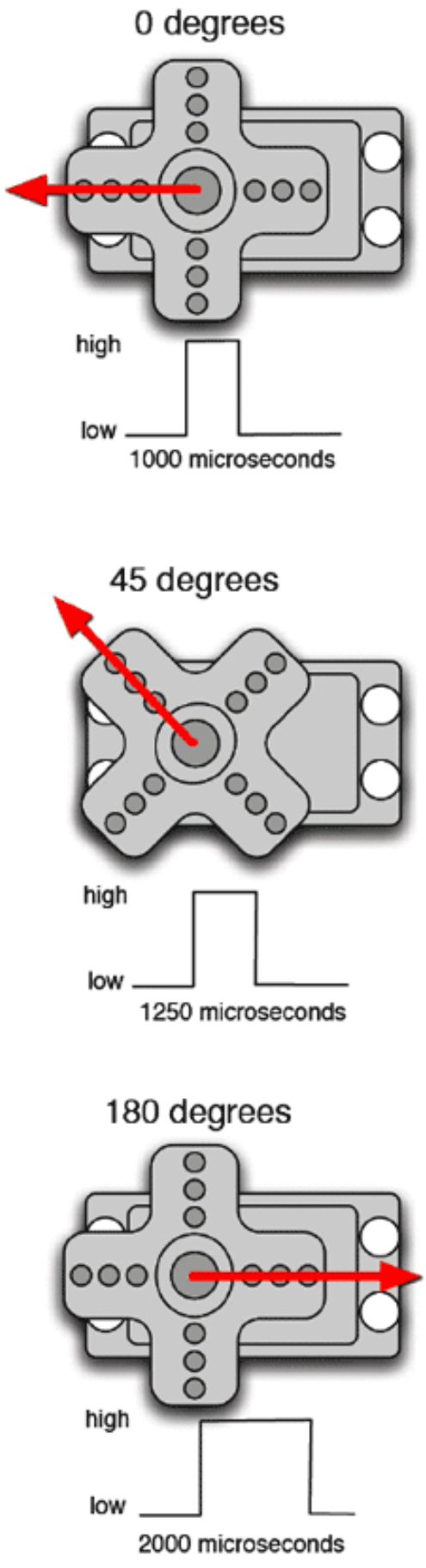


Рисунок 10 – Эксперимент 3

Рисунок 11 – Двигатель постоянного тока



Риунок 12 – схема различных положений сервопривода

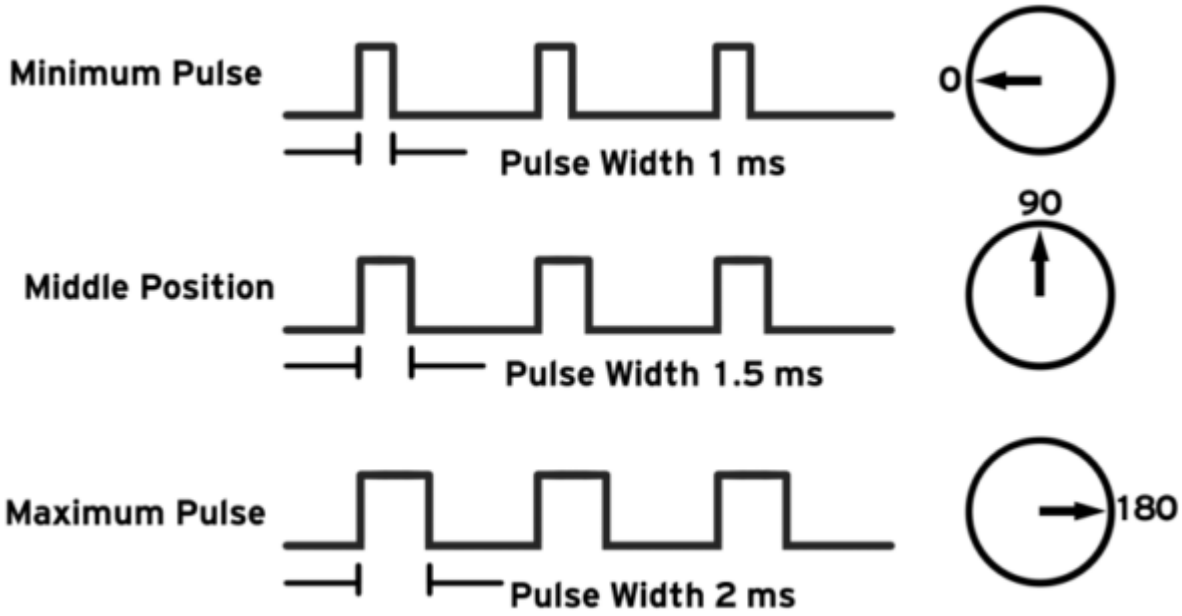


Рисунок 13 – управление сервопривода с помощью ШИМ-сигнала

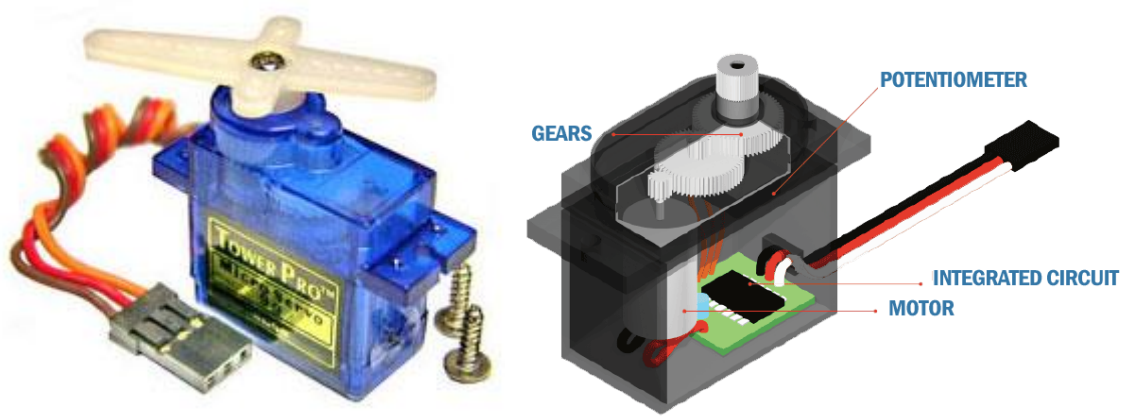


Рисунок 14 – сервопривод SG90 micro

Рисунок 15 – Ультразвуковой датчик HC-SR04

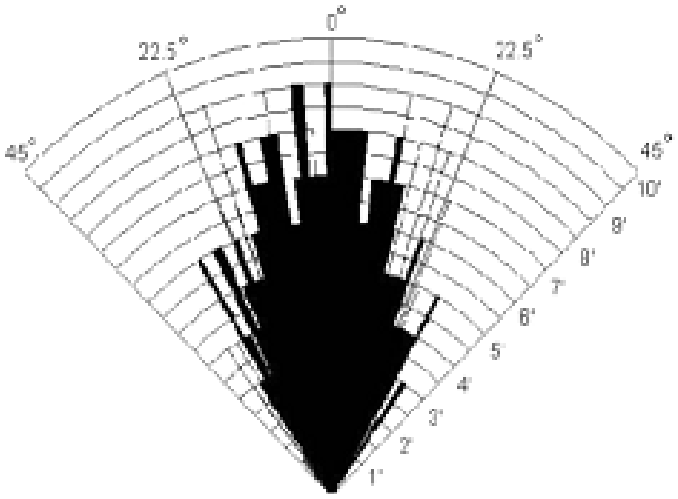


Рисунок 16 – Диаграмма направленности ультразвукового модуля

Приложение 2

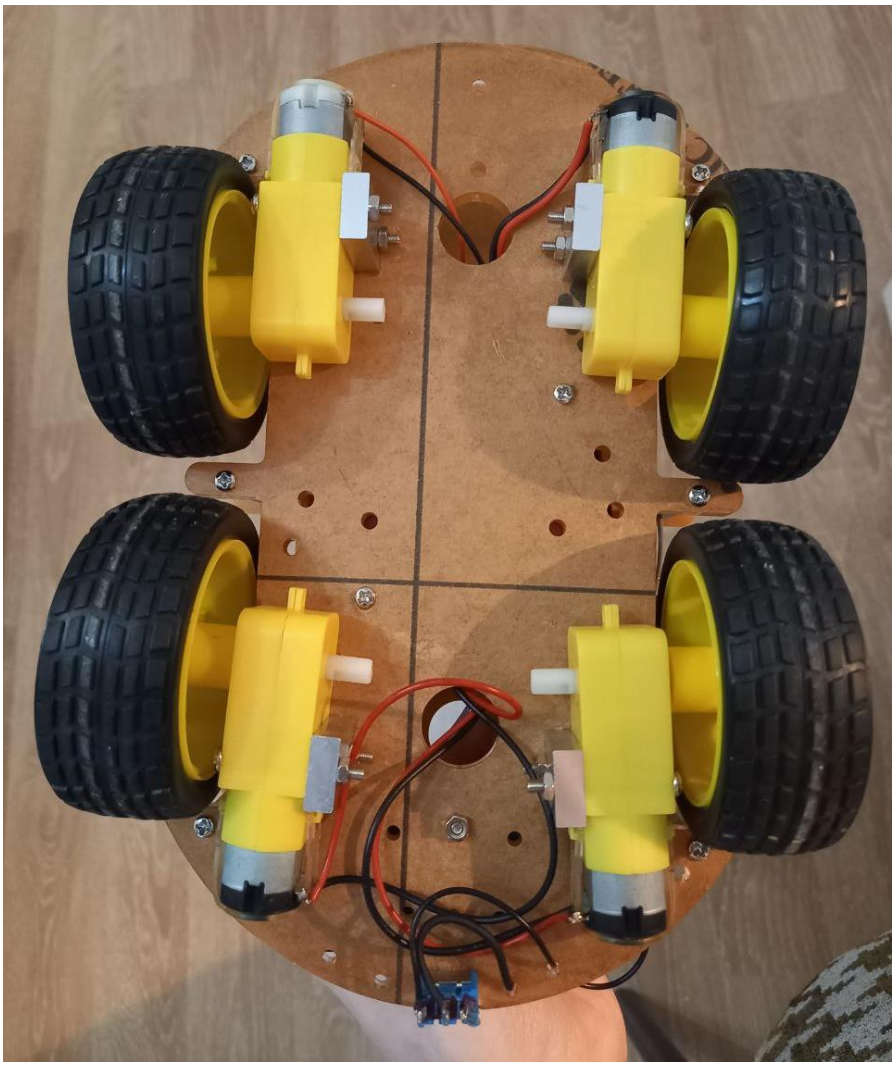


Рисунок 1

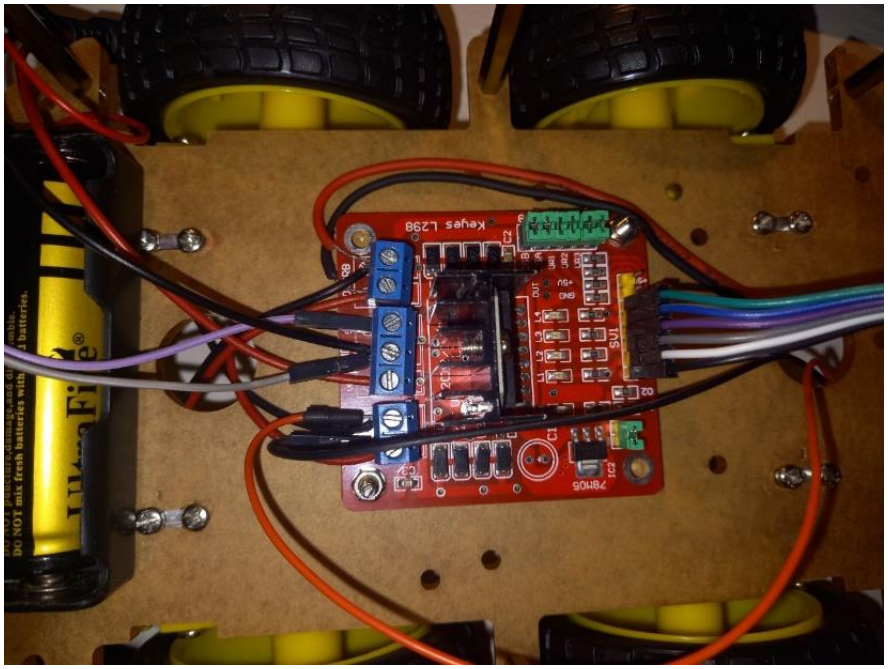


Рисунок 2

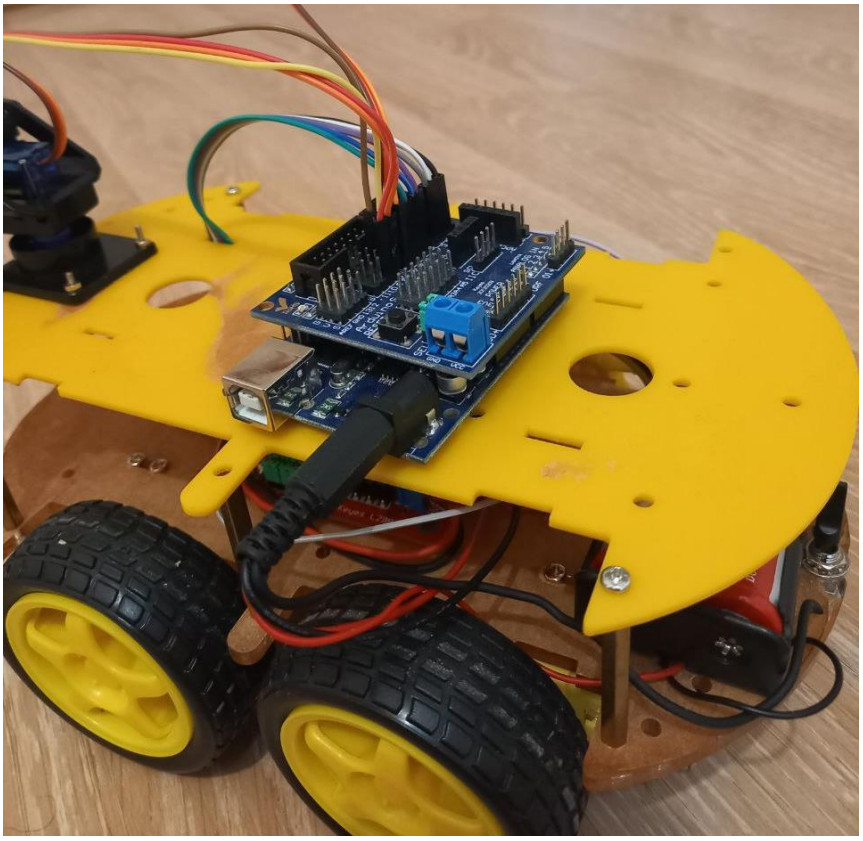


Рисунок 3

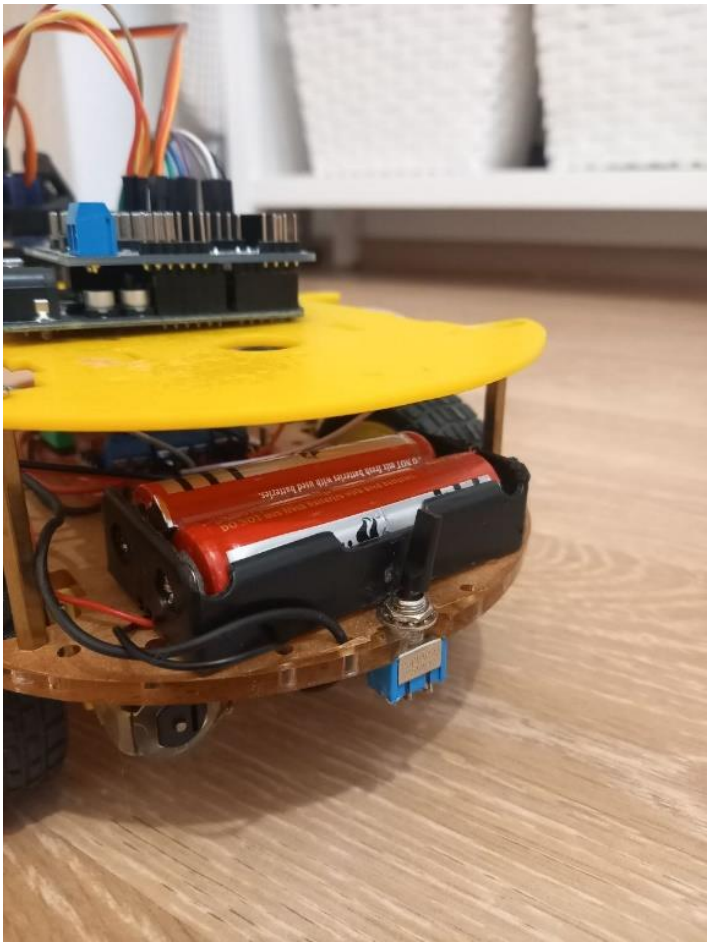


Рисунок 4

Рисунок 5

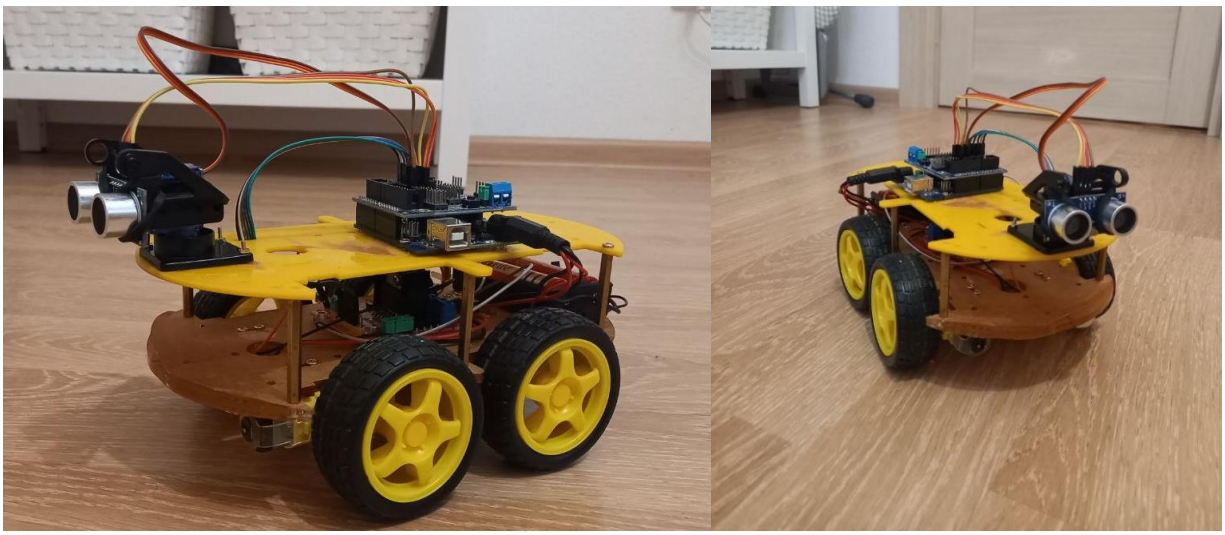


Рисунок 6