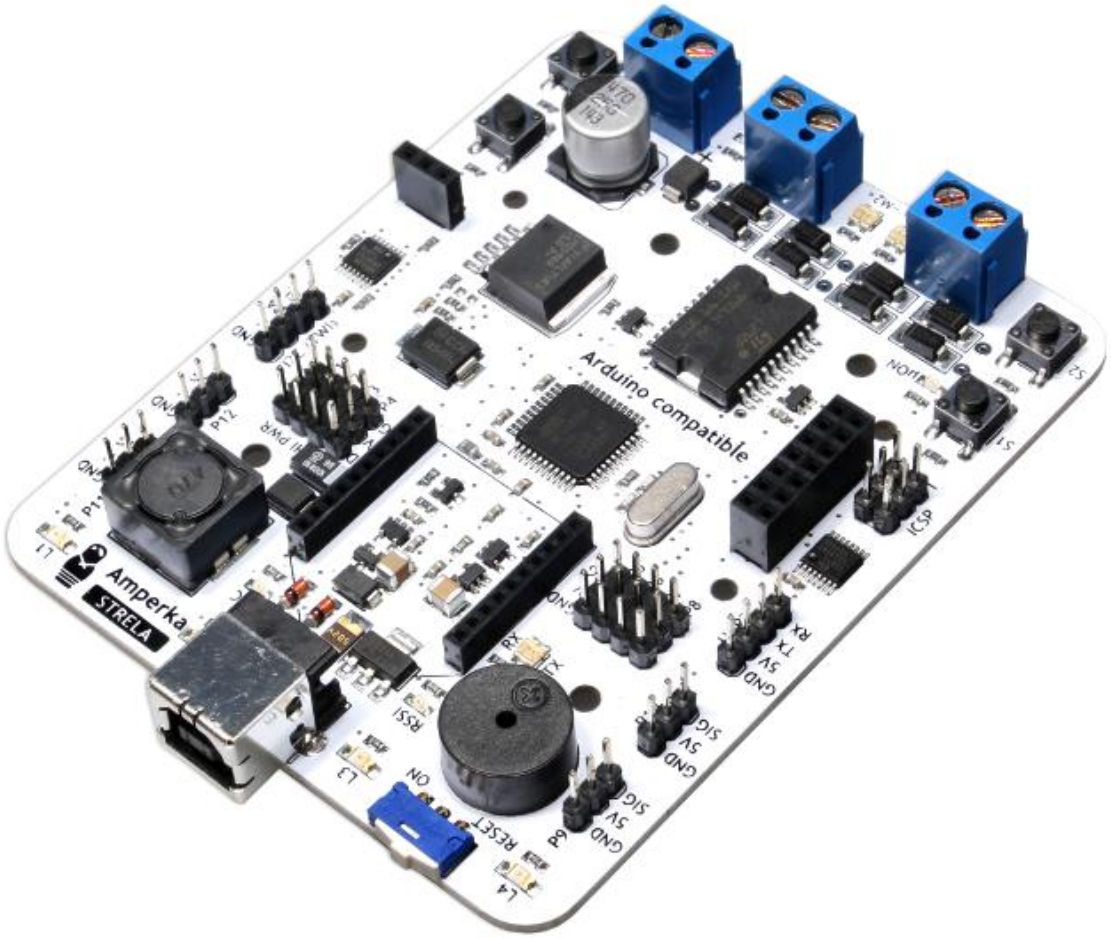
Strela

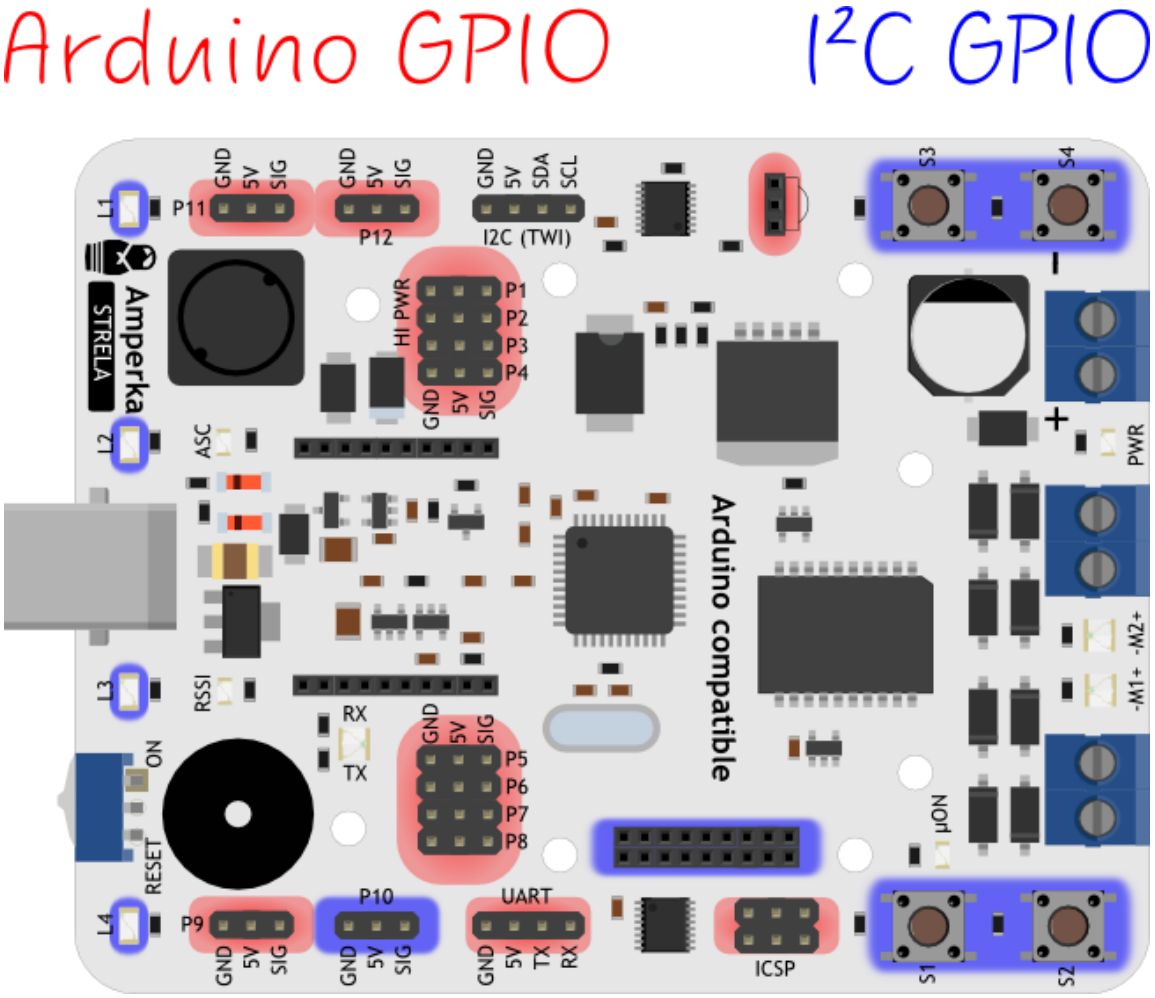
Strela — это Arduino-совместимая платформа разработанная Амперкой и предназначенная для уменьшения времени на постройку роботов и упрощения работы с ними. Она выполнена на основе микроконтроллера ATMega32u4 и обладает большим количеством цифровых и аналоговых входов/выходов, выведенных на трёхштырьковые контакты. Плата содержит DC-DC преобразователь, позволяет управлять двумя коллекторными моторами постоянного тока до 2 А на один канал с напряжением до 24 В, подключать модули беспроводной связи и LCD-экран.



Краткое руководство по использованию

Плата программируется в Arduino IDE. Для начала работы с платой достаточно выбрать в меню *Инструменты* → *Плата* → *Arduino Leonardo* и соответствующий COM-порт.

Управление GPIO



Платформа Strela имеет на борту контакты подключённые к Arduino-микроконтроллеру и контакты подключённые к I²C-расширителю портов ввода-вывода. На аппаратном уровне доступ к этим портам ввода-вывода очень сильно различается, но вы можете не переживать по этому поводу. Для облегчения работы с платой из Arduino IDE была создана библиотека Strela. Все примеры будут с этой библиотекой.

Простой пример управления этими контактами:

StrelaPinOutput.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>
void setup() {

  uPinMode(P9, OUTPUT);  // это Arduino GPIO. Мы настроим его на выход
  uPinMode(P10, OUTPUT); // а это — I²C-GPIO. Его мы тоже настроим на
выход
}

void loop() {
  uDigitalWrite(P9, HIGH);  // подадим на P9 высокий уровень
  uDigitalWrite(P10, HIGH); // подадим на P10 высокий уровень

  delay(1000);
```

```
  uDigitalWrite(P9, LOW);   // подадим на P9 низкий уровень
  uDigitalWrite(P10, LOW);  // подадим на P10 низкий уровень

  delay(1000);
}
```

Легко можно обнаружить, что разница в управлении контактами Arduino и контактами I²C-расширителей портов отсутствует, если использовать функции `uPinMode`, `uDigitalWrite` и `uDigitalRead` из библиотеки Strela вместо привычных `pinMode`, `digitalWrite` и `digitalRead`. Конечно, можно использовать и привычные функции управления цифровыми контактами, но они будут работать только с Arduino-контактами. Остальные функции для работы с контактами, такие как `analogWrite` или `analogRead` остаются без изменения и работают только с предназначенными для них Arduino-контактами.

Обратите внимание на обозначение пинов в коде. Вместо привычных цифр которые используются в Arduino Leonardo, для обозначения номеров контактов используются значения P9 и P10. Это сделано для унификации доступа к контактам с различным способом управления, а так же для упорядочивания пинов.

Давайте теперь считаем уровень с ножек:

StrelaPinInput.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>
void setup() {

  uPinMode(P9, INPUT);  // это Arduino GPIO. Мы настроим его на вход
  uPinMode(P10, INPUT); // а это — I²C-GPIO. Его мы тоже настроим на
вход

  Serial.begin(9600); //Подготовим к работе Serial-порт

  while (!Serial) { // Так как Strela рботает на том же контроллере,
что и Arduino Leonardo
    ;               // необходимо дождаться подключения виртуального
Serial-порта
  }
}

void loop() {
  Serial.print("P9 - ");
  //Считаем значение с P9
  // и отправим его в последовательный порт
  Serial.println(uDigitalRead(P9));

  Serial.print("P10 - ");
  //Считаем значение с P10
  // и отправим его в последовательный порт
  Serial.println(uDigitalRead(P10));

  delay(500);
}
```

Если посмотреть в терминал, то можно увидеть первое отличие — порт P10 будет возвращать 1, а порт P9 — случайное значение. Это происходит потому, что I²C-

расширитель портов на своих входах имеет встроенный подтягивающий резистор на 100 кОм. Обычно это ни сколько не мешает.

Встроенные кнопки и светодиоды

Работать со встроенными кнопками и светодиодами можно при помощи тех же функций `uDigitalRead` и `uDigitalWrite`. Так как заранее известно, что светодиоды должны управляться цифровым выходом, а считывать нажатие кнопки нужно с цифрового входа, функция `uPinMode` для этих контактов ни к чему не приводит. Их даже не нужно указывать в секции `setup`, так как библиотека Strela инициализирует периферию автоматически.

StrelaBlink.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>
void setup() {
}

void loop() {

  uDigitalWrite(L1, HIGH); // Зажгём перывый светодиод
  delay(1000);

  uDigitalWrite(L1, LOW);  // И погасим его
  delay(1000);
}
```

В следующем примере мы будем считывать значение с кнопок и зажигать светодиоды. Будем зажигать светодиод под тем же номером, что и нажатая кнопка.

StrelaLedsAndButtons.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>
void setup() {
}

void loop() {
  for (byte i = S1; i <= S4; ++i) // Будем проверять состояние кнопок
по очереди
  {
    byte buttonNumber = i - S1;          //Определим номер текущей
кнопки
    bool buttonState = uDigitalRead(i); //Считаем положение кнопки

    //Подадим считанное значение с кнопки на светодиод с тем же номером
    uDigitalWrite(L1 + buttonNumber, buttonState);
  }
}
```

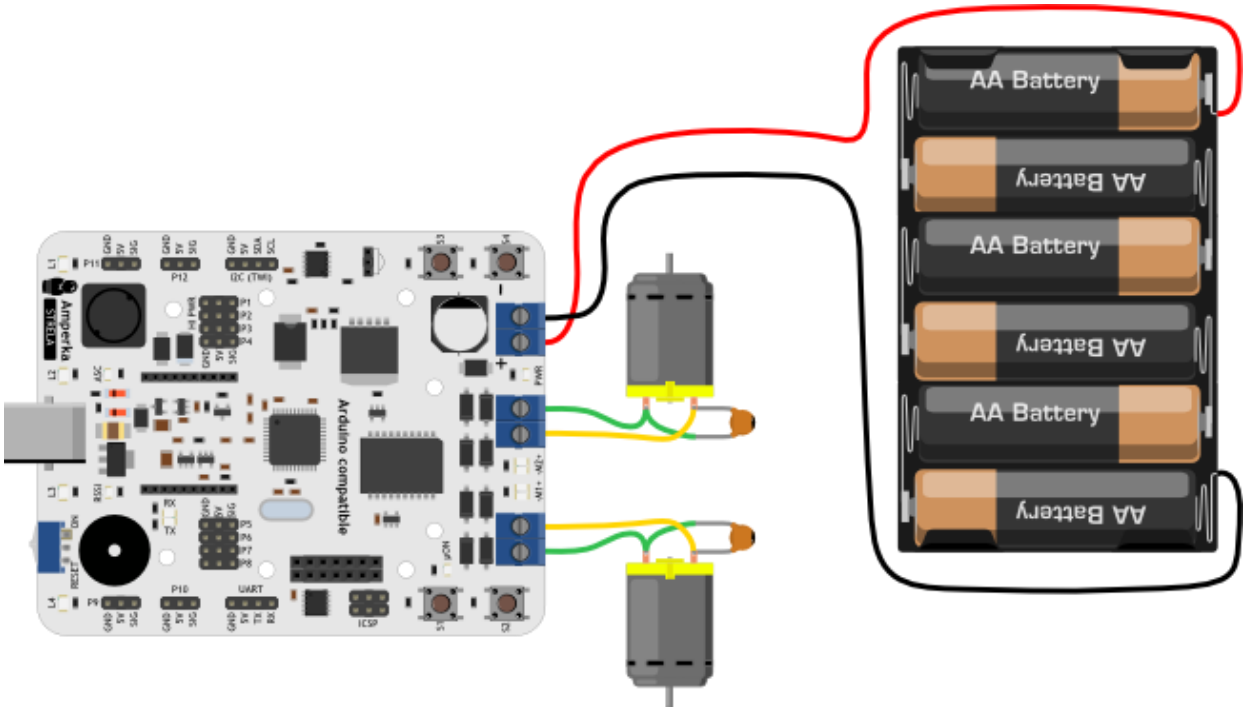
Пищалка

Добавим к предыдущему скетчу звуковое сопровождение:

StrelaLedsButtonsAndBuzzer.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>
void setup() {
}

void loop() {
  for (byte i = S1; i <= S4; ++i) // Будем проверять состояние кнопок
по очереди
  {
    byte buttonNumber = i - S1;          //Определим номер текущей
кнопки
    bool buttonState = uDigitalRead(i); //Считаем положение кнопки

    if (buttonState)           //Если она была нажата...
      tone(BUZZER, 1000, 50); //...пикнем пищалкой с частотой 1000 Гц,
50 мс

    //Подадим считанное значение с кнопки на светодиод с тем же номером
   uDigitalWrite(L1 + buttonNumber, buttonState);
  }
}
```

Управление коллекторными двигателями



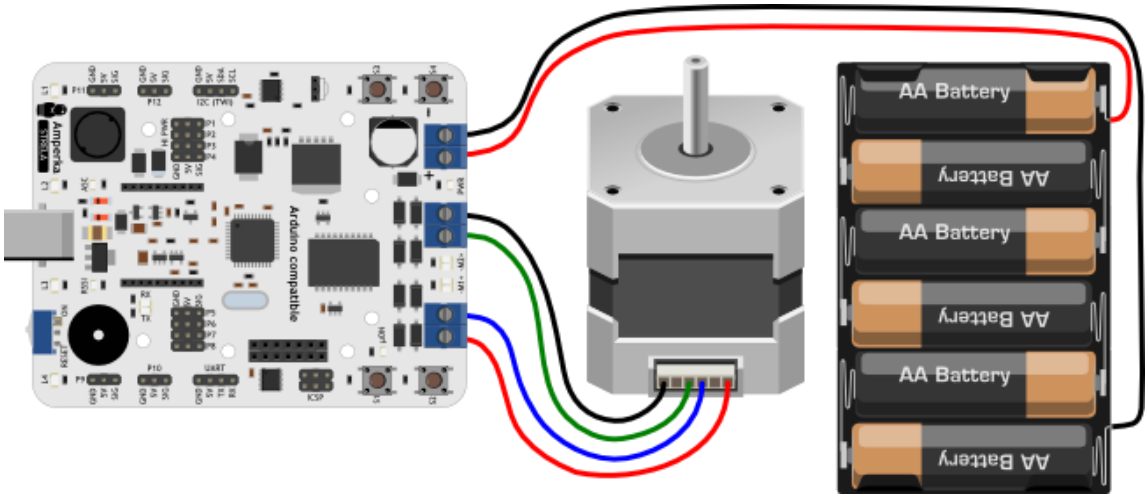
Теперь перейдём к работе с двигателями. О характеристиках рекомендуемых двигателей и источников питания можете прочитать в разделе «Силовая часть».

StrelaMotors.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>

void setup() {
  motorConnection(1, 0);// Я неправильно прикрутил один мотор
  //поэтому, чтобы их не перекручивать
```

```
  //можно воспользоваться этой функцией.
  //Направление вращения мотора 1 будет изменено.
}

void loop() {

  //Моторы управляются функцией drive(int m1, int m2).
  //m1 (m2) - это скорость вращения мотора 1 (2).
  //Скорость регулируется в пределах от -255 до 255
  //Если это число положительное - мотор будет вращаться вперёд,
  //если отрицательное - назад.

  //Фаза 1
  uDigitalWrite(L1, HIGH);    // Зажгли светодиод 1

  drive(127, 127);    //Средний ход вперёд
  delay(2000);        // в течении 2 секунд.

  //Фаза 2
  uDigitalWrite(L2, HIGH);    // Зажгли светодиод 2

  drive(127, 0);      //Поворот на право
  delay(1000);         // в течении 1 секунды

  //Фаза 3
  uDigitalWrite(L3, HIGH);    // Зажгли светодиод 3

  drive(-255, -255); //Полный назад
  delay(1000);         // в течении 1 секунды

  //Фаза 4
  uDigitalWrite(L4, HIGH);    // Зажгли светодиод 4

  for (int i = 0; i <= 255; ++i)
  {
    drive(i, -i);     //Разворот на месте с ускорением
    delay(1);
  }

  //Фаза 5

  drive(0, 0);        //Стоп

  for (int i = L1; i <= L4; i++)
  {
    uDigitalWrite(i, LOW); // Гасим все светодиоды
  }

  // Всё, приехали
  while (true)
    ;

}
```

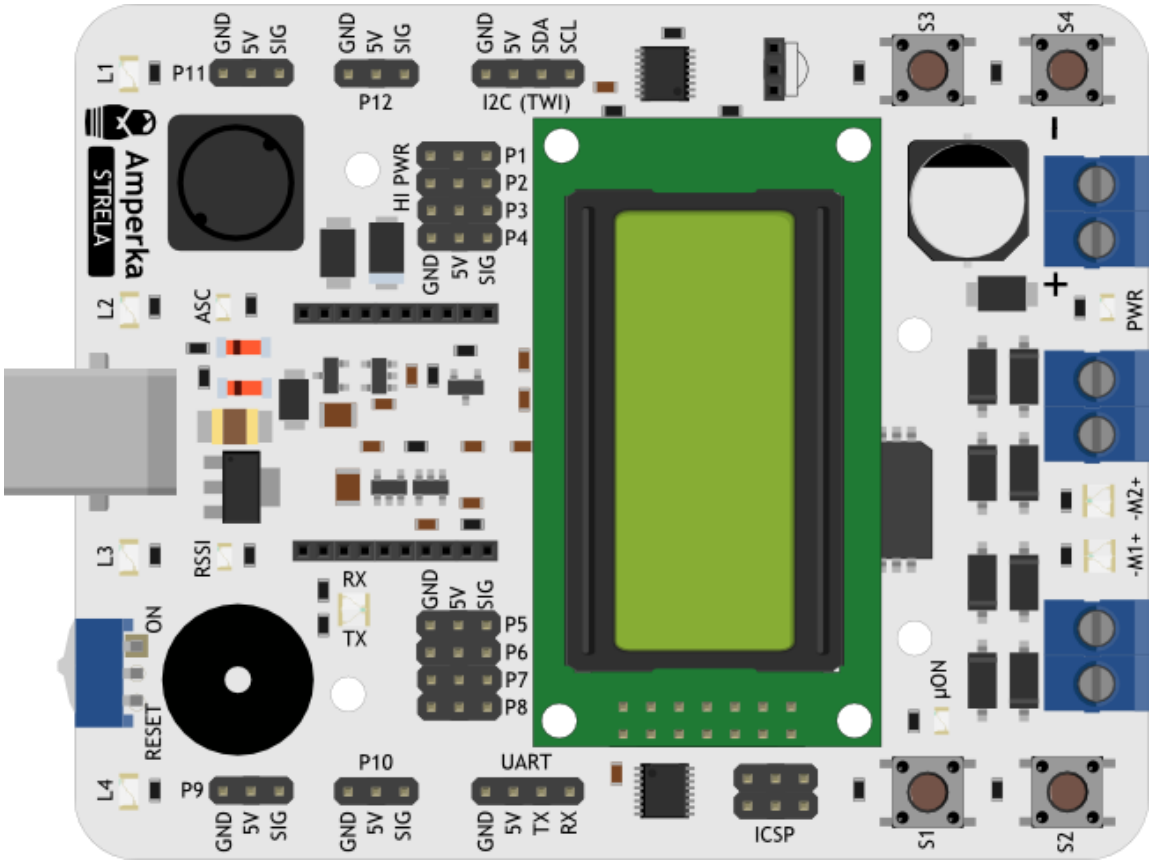
Управление шаговым двигателем



StrelaStepper.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>

void setup() {
}

void loop() {

  // Шаговый двигатель управляется при помощи
  // функции stepperMotor(int stepsToMove, int stepDelay);
  // stepsToMove — это количество шагов которые должен
  // сделать шаговый двигатель.
  // Если это число положительное — мотор крутится в одну сторону,
  // если отрицательное - в другую.
  // stepDelay — это пауза между шагами в миллисекундах.
  // От этой величины зависит скорость вращения мотора.
  // Чем больше пауза, тем медленнее вращается мотор
  // Функция stepperMotor() управляет врщением мотора только в
  // полушаговом режиме. В этом режиме шаговый двигатель
  // развивает наибольшую мощность и точность позиционирования.

  // Быстро шагаем 200 раз вперёд
  stepperMotor(200, 5);

  // Медленно шагаем 200 раз назад.
  // Мотор вернётся в исходное положение
  stepperMotor(-200, 10);

}
```

Жидкокристаллический дисплей



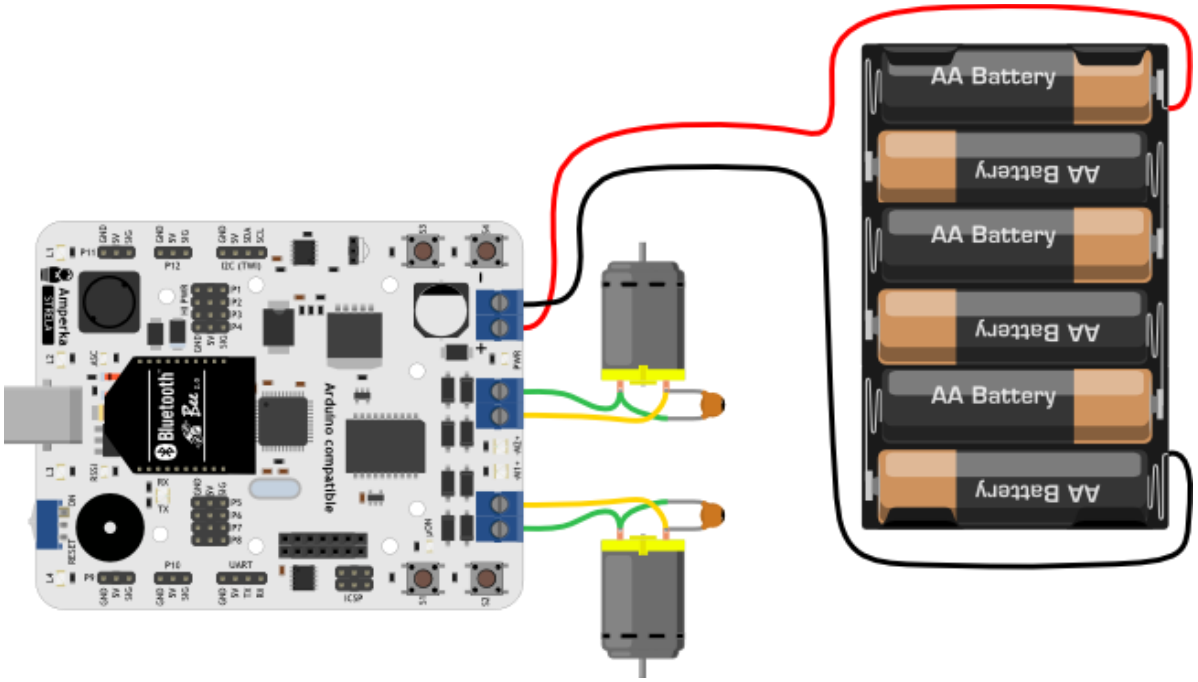
Подключим к плате LCD-экран MT-08S2A. Управлять мы им будем при помощи библиотеки LiquidCrystal_I2C. Мы уже добавили гарантированно работающую версию этой библиотеки в папку с библиотекой Strela, поэтому отдельно её устанавливать не нужно.

StrelaLiquidCrystal.ino

```
// Подключим библиотеку для работы с I2C-расширителем портов
#include <Wire.h>
// Подключим библиотеку Strela
#include <Strela.h>
// Подключим библиотеку для работы с LCD-эраном через I2C
#include <LiquidCrystal_I2C.h>

// Создадим объект lcd, который будет выводить текст на экран
LiquidCrystal_I2C lcd(LC_ADDR, LCEN, LCRW, LCRS, LC4, LC5, LC6, LC7);

void setup()
{
  //Инициализация экрана. У нашего экрана 8 символов 2 строки
  lcd.begin(8, 2);
  //Переместим курсор в начало первой стройки (символ 0, строка 0)
  lcd.home();
  //Печатаем строку
  lcd.print("Hello");
  //Переместим строку в начало второй строки (символ 0, строка 1)
  lcd.setCursor(0, 1);
  //Печатаем строку
  lcd.print("World!");
  //Пауза, чтобы успеть насладиться результатом
  delay(2000);
```

```
}

void loop()
{
  // Далее будем просто выводить время прошедшее со старта скетча

  //Очищаем экран
  lcd.clear();
  // выставляем курсор в положение 0, 0
  lcd.home();
  // Пишем первую строку
  lcd.print("Uptime:");
  // переместим курсор в начало второй строки
  lcd.setCursor(0, 1);
  // узнаем время прошедшее со старта скетча в миллисекундах
  unsigned long time = millis();
  // и выведем его на экран
  lcd.print(time);
  //Пауза, чтобы успеть насладиться результатом
  delay(500);
}
```

Управление роботом с мобильного телефона через Bluetooth-модуль



Подключим к плате Bluetooth Bee. Будем управлять нашим роботом дистанционно с помощью мобильного телефона на ОС Android.

Для Android существует огромное количество приложений, при помощи которых можно управлять Arduino при помощи Bluetooth. В данном случае для управления Strela мы использовали Arduino Bluetooth RC Car.

ArduinoBluetoothRCCarOnStrela.ino

```
//Arduino Bluetooth RC Car in Strela
//Original App:
//https://play.google.com/store/apps/details?id=braulio.calle.bluetooth
RCcontroller

#include <Wire.h> // Библиотека для работы с I2C
```

```
#include <Strela.h> // Библиотека для работы со Стрелой

int velocity = 0;  //Здесь будет храниться значение скорости

int defaultSpeed = 100; // это число мы будем использовать в логике
поворотов

void setup()
{
  Serial1.begin(9600);      //Bluetooth Bee по умолчанию использует эту
скорость
  motorConnection(1, 0);    // Я неправильно прикрутил один мотор
  //поэтому, чтобы их не перекручивать
  //можно воспользоваться этой функцией.
  //Направление вращения мотора 1 будет изменено.
}

void loop()
{
  if (Serial1.available() > 0)     //Если появились новые команды
  {
    control();                     //вызываем функцию управления
  }
  //Здесь можно написать ещё много своего кода
}

void control()  // функция управления
{
  char dataIn = Serial1.read();  //Считаем значение пришедшей команды

  if (dataIn == 'F')           //Если пришла команда "F"
    drive(velocity, velocity);   //едем вперёд

  else if (dataIn == 'B')     //или если пришла команда "B"
    drive(-velocity, -velocity); //едем назад

  else if (dataIn == 'L')     //или если пришла команда "L"
    drive(-velocity, velocity);  //поворачиваем налево на месте

  else if (dataIn == 'R')     //или если пришла команда "R"
    drive(velocity, -velocity);  //поворачиваем направо на месте

  else if (dataIn == 'I')     //или если пришла команда "I", едем вперёд
и направо
    drive(defaultSpeed + velocity, defaultSpeed - velocity);

  else if (dataIn == 'J')     //или если пришла команда "J", едем назад
и направо
    drive(-defaultSpeed - velocity, -defaultSpeed + velocity);

  else if (dataIn == 'G')    //или если пришла команда "I", едем вперёд
и налево
    drive(defaultSpeed - velocity, defaultSpeed + velocity);

  else if (dataIn == 'H')    //или если пришла команда "H", едем назад и
налево
    drive(-defaultSpeed + velocity, -defaultSpeed - velocity);

  else if (dataIn == 'S')    //или если пришла команда "S", стоим
    drive(0, 0);

  else if (dataIn == 'U')    //или если "U", зажигаем "передние фары"
  {
    uDigitalWrite(L2, HIGH);
```

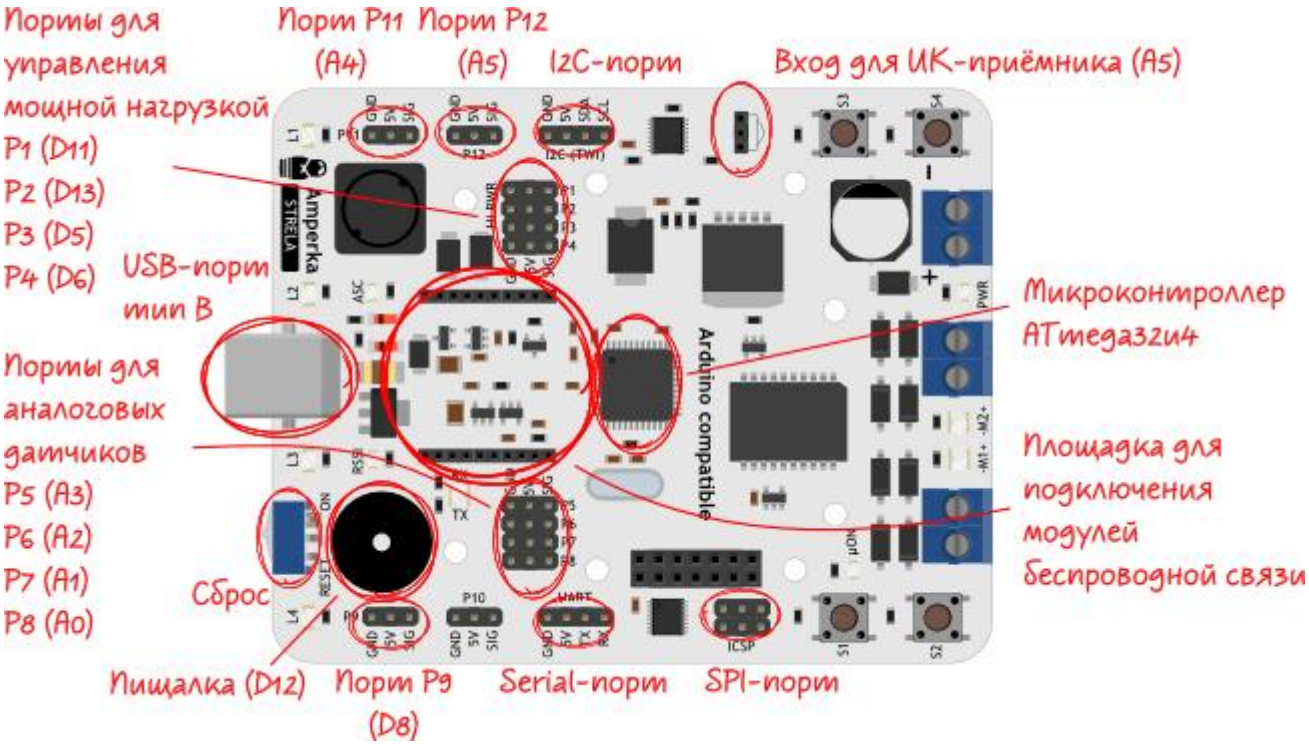
```
    uDigitalWrite(L3, HIGH);
  }
  else if (dataIn == 'u')    //или если "u", гасим "передние фары"
  {
    uDigitalWrite(L2, LOW);
    uDigitalWrite(L3, LOW);
  }
  else if (dataIn == 'W')    //или если "W", зажигаем "задние фары"
  {
    uDigitalWrite(L1, HIGH);
    uDigitalWrite(L4, HIGH);
  }
  else if (dataIn == 'w')    ////или если "w", гасим "задние фары"
  {
    uDigitalWrite(L1, LOW);
    uDigitalWrite(L4, LOW);
  }

  // если к нам пришло значение от 0 до 9
  else if (((dataIn - '0') >= 0) && ((dataIn - '0') <= 9))
    velocity = (dataIn - '0') * 25; //сохраняем новое значение скорости

  else if (dataIn == 'q') //если "q" - полный газ!
    velocity = 255;
}
```

Вот что из этого получилось:

Элементы платы

Теперь рассмотрим подробнее все компоненты и разъёмы, которые можно встретить на плате с точки зрения схемотехники.

Микроконтроллер ATmega32u4



В плате использован микроконтроллер **ATMega32u4** с прошитым загрузчиком Arduino Leonardo. Микроконтроллер программируется через USB-порт в Arduino IDE. Работая с платой в Arduino IDE, необходимо выбирать плату Arduino Leonardo.

Контакты для управления мощной нагрузкой

Контакты питания пинов **P1–P4** соединены короткими и широкими контактными дорожками с DC–DC преобразователем. Это позволяет подключать к этим контактам достаточно мощную нагрузку, например сервоприводы. Так же они подходят для подключения цифровых сенсоров.

Контакты для аналоговых датчиков

Пины **P4-P9**, а так же **P11** и **P12** могут быть использованы не только как цифровые входы/выходы. Они так же подходят для подключения сенсоров с аналоговым сигналом.

Соответствие контактов Strela и Arduino Leonardo

Strela	Arduino Leonardo	Доступные функции
P1	11	Цифровой вход/выход, ШИМ.
P2	13	Цифровой вход/выход, ШИМ.
P3	5	Цифровой вход/выход, ШИМ.
P4	6, A7	Цифровой вход/выход, ШИМ, аналоговый вход.
P5	A3	Цифровой вход/выход, аналоговый вход.
P6	A2	Цифровой вход/выход, аналоговый вход.
P7	A1	Цифровой вход/выход, аналоговый вход.
P8	A0	Цифровой вход/выход, аналоговый вход.
P9	8, A8	Цифровой вход/выход, аналоговый вход.
P11	A4	Цифровой вход/выход, аналоговый вход.
P12 (IR)	A5	Цифровой вход/выход, аналоговый вход. Strela использует этот же контакт как **вход для ИК-приёмника**. Управление с бытового ИК-пульта управления возможно с помощью библиотеки IRremote.

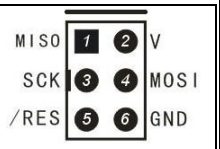
Контакты специального назначения

Некоторые стандартные контакты имеют специальное назначение, поскольку именно они управляют установленным на Strela оборудованием, или отвечают за коммуникацию.

Контакты Arduino Leonardo	Конфигурация	Функция
12	OUTPUT	Пищалка — встроенный пьезодинамик, который воспроизводит звук с помощью стандартной функции tone().
7	INPUT	Этот контакт подключён к I^2C-расширителям портов, для обнаружения изменения напряжения на их входах. Этот контакт можно использовать для прерывания основной

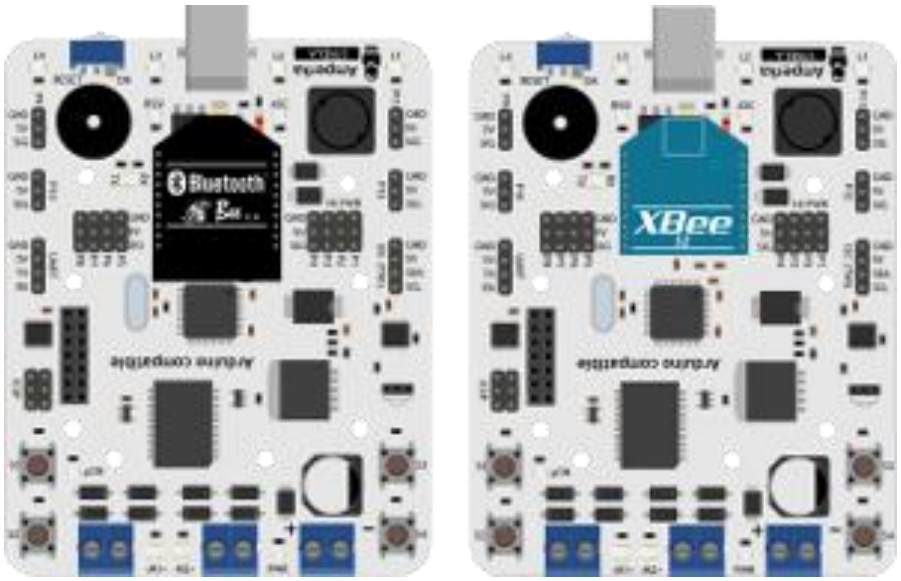
Контакты Arduino Leonardo	Конфигурация	Функция
		программы по изменению уровня.
9	OUTPUT	Подключён к драйверу двигателей. Используется для задания скорости вращения мотора M1 при помощи ШИМ.
10	OUTPUT	Подключён к драйверу двигателей. Используется для задания скорости вращения мотора M2 при помощи ШИМ.
4	OUTPUT	Подключён к драйверу двигателей. Используется для задания направления вращения мотора M1.
~ (PE2)	OUTPUT	Подключён к драйверу двигателей. Используется для задания направления вращения мотора M2.

Коммуникация

Порт связи	Контакты Arduino Leonardo	Функция	Использование
Serial	0, 1	RX, TX	Используется для связи по последовательному протоколу с модулем беспроводной связи, если он подключён, или с другими устройствами. Программно передача происходит с помощью Serial1. Если в вашем проекте не используются устройства с последовательным протоколом, эти контакты могут быть использованы как обычные цифровые пины.
SPI	14, 15, 16	MISO, SCK, MOSI	 SPI-порт доступен на колодке ICSP. Используется для связи по протоколу SPI. Передача данных происходит с помощью библиотеки SPI. Если в вашем проекте не используются периферийные устройства с подключением по SPI, эти контакты могут быть использованы как обычные цифровые пины. Контактная колодка ICSP может использоваться для программирования платы с помощью программатора.
I^2C(TWI)	2, 3	SDA, SCL	Используются для коммуникации по шине I^2C. По этим пинам управляются встроенные I^2C-расширители портов, поэтому использовать их в других целях не получится. К ним может быть подключено и другое оборудование, управляемое по I^2C. Передача данных происходит с помощью библиотеки Wire.
USB	USB-разъём	Виртуальный	Используется для прошивки платы из Arduino IDE

Порт связи	Контакты Arduino Leonardo	Функция	Использование
	тип B	Serial-порт	и связи по последовательному протоколу с компьютером. Программно передача происходит с помощью Serial.

Площадка для подключения модулей беспроводной связи



На плате присутствует площадка для подключения модулей беспроводной связи формата XBee. Это могут быть модули для связи по протоколу XBee, Bluetooth или Wi-Fi. На площадке установлен линейный регулятор напряжения на 3,3 В. Ведь именно это напряжение является традиционным для XBee-модулей. Общение между модулем беспроводной связи и платой происходит через последовательный интерфейс. При написании кода этот последовательный интерфейс доступен через объект `Serial1`.

При необходимости модули XBee могут перезагружать микроконтроллер ATmega32u4, выставляя на контакте 12 площадки логическую единицу.

Сброс

На плате присутствует переключатель, который бывает полезен при отладке или при подготовке к старту на соревнованиях. В положении «RESET» контакт Reset микроконтроллера замыкается на землю. В таком положении на микроконтроллер подано напряжение, но он не работает. Чтобы запустить микроконтроллер необходимо перевести рычажок в положение «ON».

Силовая часть

Питание

Клеммники под винт для подключения питания на плате обозначены как **PWR**. К ним подключается источник питания, который будет использоваться для питания моторов. Напряжение питания должно быть в пределах 7–24 В постоянного тока.

Внимание!

При подключении питания соблюдайте полярность. Неправильное подключение может привести к непредсказуемому поведению или выходу из строя платы или источника питания.

На плате три контура питания.

- Первый — *силовой*, напряжение на который приходит с клеммника PWR. От этого контура запитана микросхема **H-моста L298P**, моторы и **DC-DC-преобразователь LM2596-5.0**.

 ***Важно!***

Не все источники питания подходят для работы с моторами.

Для корректной работы DC-DC преобразователя источники питания должны быть способны обеспечить напряжение выше 7 В при резких скачках нагрузки или при включении моторов на полную мощность. Если источник питания не рассчитан на мощность, необходимую двигателям, просадка напряжения при росте нагрузки ниже 7 В может привести к перезагрузке управляющего контроллера, и связанным с этим неадекватным поведением платы. Этому требованию соответствуют только **литий-ионные** и **никель-металлгидридные** аккумуляторы, или блоки питания необходимой мощности.

Поэтому питать робота от обычной "Кроны" не стоит.

- Источником напряжения для второго контура питания может быть DC-DC преобразователь или USB. Это контур *управления*. Напряжение в этом контуре — 5 В. От него запитаны микроконтроллер, I^2C-расширители, логика Н-моста и преобразователь напряжения на 3,3 В для питания модулей беспроводной связи. Это же напряжение подано на все трёхштырьковые контакты. Напряжение в 5 В может быть подано как с USB, так и с клеммника PWR. При этом при подключении по USB будет работать только контур управления, напряжение на Н-мосту будет недостаточным для работы двигателей. При одновременном подключении платы по USB и через колодки PWR питание будет подаваться только от PWR.
- Третий контур питания используется для питания модуля беспроводной связи и его обвязки через площадку для подключения модулей связи. Напряжение питания в этом контуре — 3,3 В.

Клеммники для подключения двигателей

Клеммники под винт для подключения нагрузки на плате обозначены как **-M1+ -M2+.**

Нагрузка разделена на 2 канала. Первый канал на плате имеет обозначение «M1», второй канал имеет обозначение «M2». Каждый канал управляется независимо.

Обозначения «+» и «-» показывают воображаемые начало и конец обмотки: если подключить два коллекторных двигателя таким образом, чтобы их одноимённые контакты щёточного узла соответствовали одному и тому же обозначению на плате, то, при подаче на Н-мост L298P одинаковых управляющих импульсов, моторы будут вращаться в одну и ту же сторону.

Характеристики драйвера двигателей L298p

Параметр	Мин.	Макс.	Ед. изм.
Напряжение питания двигателей	7	24	В
Продолжительный ток на канал *	—	2	А
Пиковый прерывистый ток на канал (80% включение, до 10 мс)	—	2,5	А

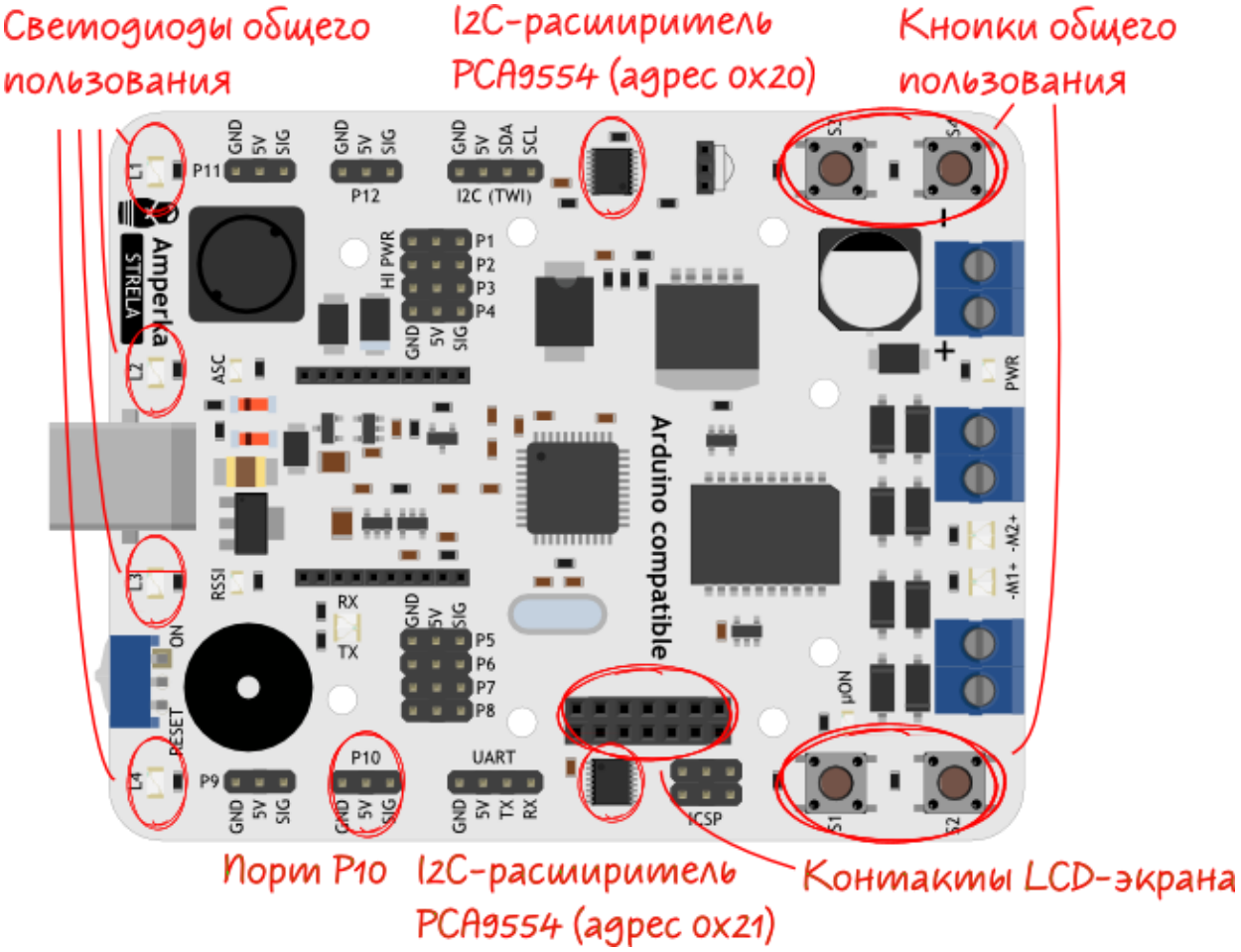
Параметр	Мин.	Макс.	Ед. изм.
Пиковый непрерывный ток на канал (до 100 мкс)	—	3	A

* для достижения максимума необходимо дополнительное охлаждение чипа L298P

Характеристики DC-DC преобразователя LM2596-5.0

Параметр	Мин.	Номинал.	Макс.	Ед. изм.
Напряжение питания силовой части	7	—	24	В
Выходное напряжение	4,75	5	5,25	В
Ток нагрузки	—	—	3	А

Расширители портов ввода-вывода



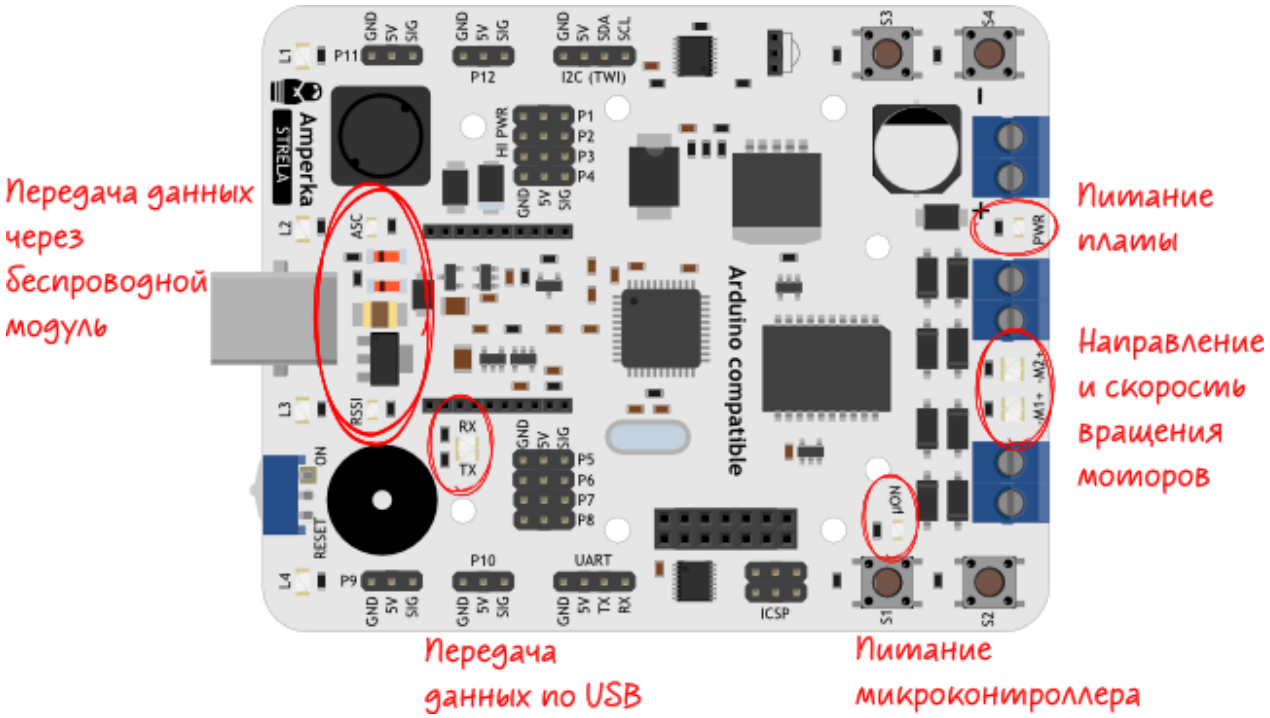
На плате установлены два I^{2}C-расширителя портов ввода-вывода PCA9554.

Адрес	Функция
0×20	Управление светодиодами L1-L4, получение информации о нажатии кнопок S1-S4.
0×21	Управление контактом P10 и LCD-экраном MT-08S2A при помощи библиотеки LiquidCrystal_I2C. Если LCD-экран не используется, можно использовать свободные контакты как дополнительные порты ввода-вывода.

Характеристики I²C-расширителей портов ввода-вывода PCA9554

Параметр	Номинал.	Ед. изм.
Напряжение питания	5	В
Максимальный входной ток одного контакта	20	мА
Максимальный выходной ток одного контакта	50	мА
Максимальный ток через контакты питания	85	мА

Индикация



Индикация питания платы

При правильном подключении питания через клеммник PWR, загорается светодиод индикации питания. Если полярность питания перепутана, или питание по какой-то причине не подано, светодиод гореть не будет.

Из-за большой ёмкости фильтрующего конденсатора, установленного на плате, светодиод индикации питания в некоторых случаях может кратковременно продолжать гореть и после отключения питания. Этот светодиод может тускло гореть при питании платы от USB.

Индикация питания микроконтроллера

Индикатор питания микроконтроллера горит, если на микроконтроллер ATmega32u4 подано напряжение.

Индикация направления и скорости вращения двигателей

При высоком логическом уровне на пине управления направлением вращения, т.е. вращении вперёд, индикатор светится зелёным светом. При этом полярность напряжения на клеммниках для подключения двигателей соответствует обозначению «+» и «-» на

плате. При низком уровне, т.е. при реверсе светодиод светится красным цветом. Яркость свечения светодиодов зависит от скорости вращения соответствующих моторов — чем выше скорость, тем ярче светится светодиод.

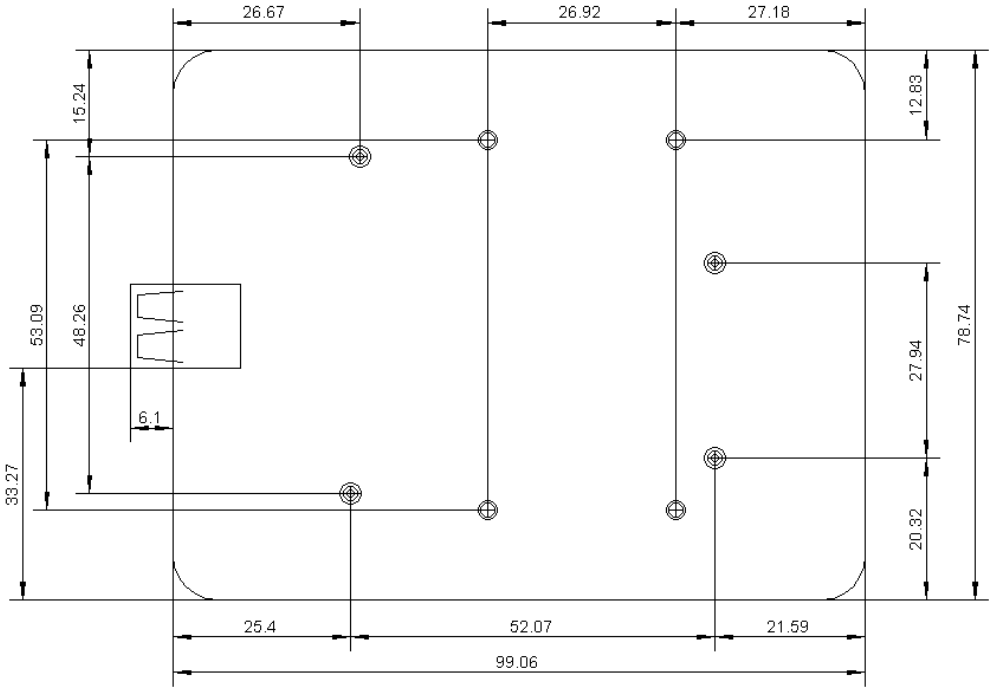
Индикатор передачи данных по USB

Светодиоды **RX** и **TX** светятся во время передачи данных через виртуальный последовательный интерфейс. При получении данных светится зелёный светодиод RX, при передаче — красный светодиод TX.

Индикаторы передачи данных через беспроводной модуль

- **RSSI** — индикатор мощности сигнала XBee-модулей. Чем ярче горит индикатор, тем больше мощность сигнала принимаемого XBee-модулем. Этот светодиод выполняет свою функцию только с модулями XBee.
- **ASC** — индикатор подключения к сети XBee-модулей. Если модуль не подключён, светодиод постоянно горит. Светодиод мигает при подключении к сети. Частота мигания зависит от роли модуля. Подробнее можно узнать из технического описания XBee.

Размеры платы



Ссылки

- Видеообзор
- Принципиальная схема платформы Strela
- Библиотека Strela для удобства работы с платой
- Техническое описание ATmega32u4
- Техническое описание DC-DC преобразователя LM2596-5.0
- Техническое описание H-моста L298P
- Техническое описание расширителя цифровых портов PCA9554
- Техническое описание LCD-экрана MT-08S2A-2YLG