|  |
| --- |
| ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»  КАФЕДРА «СИСТЕМЫ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ»  КУРСОВОЙ ПРОЕКТ ПО КУРСУ «ТЕХНИЧЕСКИЕ СРЕДСТВА СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ»  “Разработка системы управления актуатором поворотных колес мобильного робота.” |

# Введение.

**Актуальность**.

В настоящее время практически все крупные автомобильные и транспортные компании активно занимаются разработкой автоматических транспортных средств на колесном или гусеничном ходу, поскольку их использование позволит снизить себестоимость грузоперевозок за счет экономии топлива и заработной платы водителей, а также приведет к минимизации ДТП.

Лаборатория имитационного моделирования систем управления перспективных аэрокосмических летательных аппаратов и наземных подвижных объектов кафедры «Системы автоматического управления» МГТУ им. Н.Э.Баумана занимается разработкой автомобиля-робота на базе колесного шасси с электродвигателем.

Целью данной работы является разработка контроллера нижнего уровня, осуществляющего управление актуатором и мотором тормоза автомобиля-робота. Для достижения данной цели были поставлены следующие задачи:

* анализ предметной области, изучение особенностей проектирования подобных систем;
* определение компонентов контроллера нижнего уровня;
* разработка структурной схемы контроллера нижнего уровня;
* разработка принципиальной схемы контроллера нижнего уровня и ее практическая реализация;
* разработка систем автоматического управления с использованием различных регуляторов;
* сравнительный анализ систем.

**Постановка задачи.**

Задача робототехники – это создание и применение роботов и основанных на их использовании робототехнических систем различного назначения. Возникнув на основе механики и кибернетики, робототехника породила новые направления их развития. Для механики это, прежде всего, связанно с многозвенными механизмами типа манипуляторов, а для кибернетики – с интеллектуальным управлением, которое требуется для роботов с искусственным интеллектом.

Робота можно определить, как универсальный автомат для осуществления механических действий, подобных тем, которые производит человек, выполняющий физическую работу. При создании первых роботов и вплоть до сегодняшнего дня образцом для них служат физические возможности человека. Именно стремление заменить человека на тяжёлых работах и породило сначала идею робота, затем первые попытки её реализации (в средние века) и, наконец, обусловило возникновение и развитие робототехники и роботостроения.

В рамках общей системы реализуется следующий узел, представленный на рисунке 1.

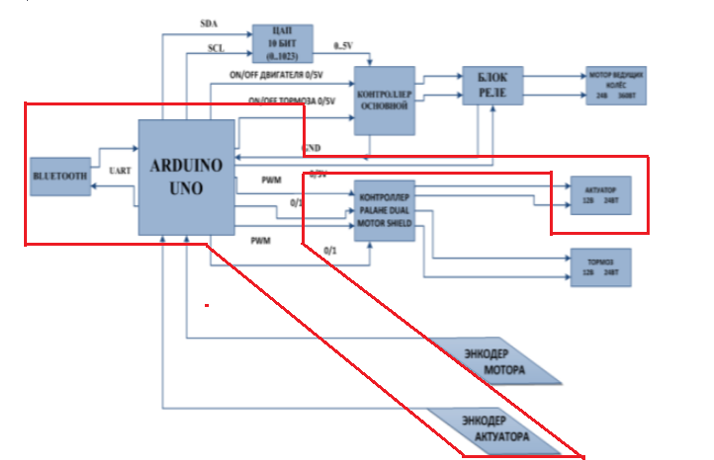


Рисунок 1. Состав реализуемого узла управления.

В данной работе решается задача управления направлением движения мобильного робота. В качестве элементов управления направлением движения используются поворотные колеса. Необходимо разработать аппаратную и программную части управления актуатором поворотных колес мобильного робота.

**Цель работы**: разработать алгоритмы управления актуатором поворотных колес мобильного робота.

# Элементы системы

## Ардуино уно

В качестве управляющего контроллера был выбран Arduino Uno

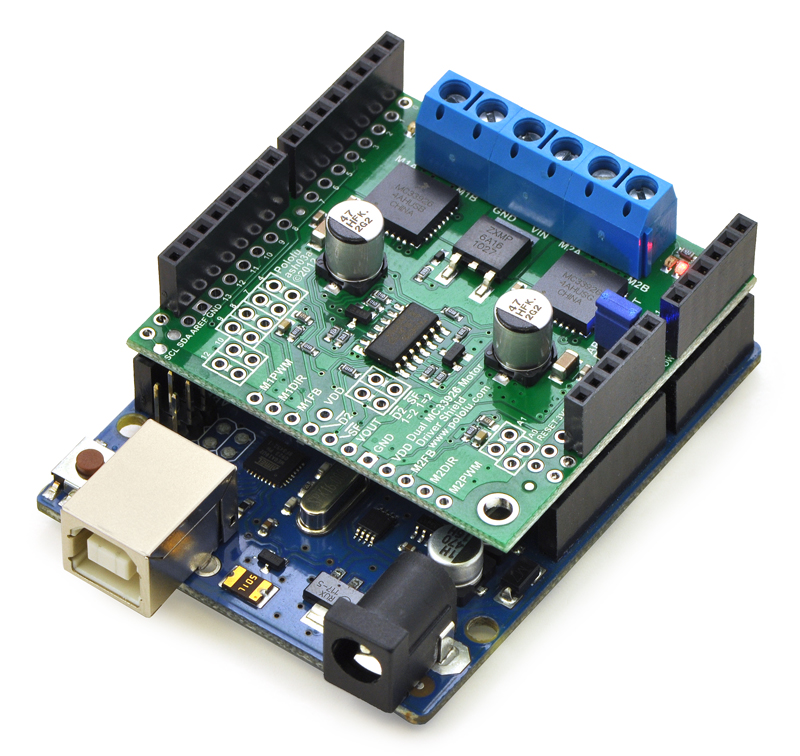


Рисунок 2 Arduino Uno

Данный контроллер контроллер построен на ATmega328. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

**Технические характеристики**

|  |  |
| --- | --- |
| Микроконтроллер | ATmega328 |
| Рабочее напряжение | 5 В |
| Входное напряжение (рекомендуемое) | 7-12 В |
| Входное напряжение (предельное) | 6-20 В |
| Цифровые Входы/Выходы | 14 (6 из которых могут использоваться как выходы [ШИМ](http://arduino.ru/Tutorial/PWM)) |
| Аналоговые входы | 6 |
| Постоянный ток через вход/выход | 40 мА |
| Постоянный ток для вывода 3.3 В | 50 мА |
| Флеш-память | 32 Кб (ATmega328) из которых 0.5 Кб используются для загрузчика |
| ОЗУ | 2 Кб (ATmega328) |
| EEPROM | 1 Кб (ATmega328) |
| Тактовая частота | 16 МГц |
| Диапазон рабочих температур | 0 – 75 С° |

**Входы и Выходы**

* **VIN**. Вход используется для подачи питания от внешнего источника (в отсутствие 5 В от разъема USB или другого регулируемого источника питания). Подача напряжения питания происходит через данный вывод.
* **5V**. Регулируемый источник напряжения, используемый для питания микроконтроллера и компонентов на плате. Питание может подаваться от вывода VIN через регулятор напряжения, или от разъема USB, или другого регулируемого источника напряжения 5 В.
* **3V3**. Напряжение на выводе 3.3 В генерируемое встроенным регулятором на плате. Максимальное потребление тока 50 мА.
* **GND**. Выводы заземления.
* **Последовательная шина: 0 (RX) и 1 (TX)**. Выводы используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы последовательной шины ATmega8U2
* **Внешнее прерывание: 2 и 3**. Данные выводы могут быть сконфигурированы на вызов прерывания либо на младшем значении, либо на переднем или заднем фронте, или при изменении значения.
* **ШИМ: 3, 5, 6, 9, 10, и 11.** Любой из выводов обеспечивает ШИМ с разрешением 8 бит.
* **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** Посредством данных выводов осуществляется связь SPI.
* **LED: 13.** Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий потенциал, то светодиод горит.

На платформе Uno установлены 6 аналоговых входов (обозначенных как A0 .. A5), каждый разрешением 10 бит (т.е. может принимать 1024 различных значения). Стандартно выводы имеют диапазон измерения до 5 В относительно земли, тем не менее имеется возможность изменить верхний предел посредством вывода AREF. Некоторые выводы имеют дополнительные функции:

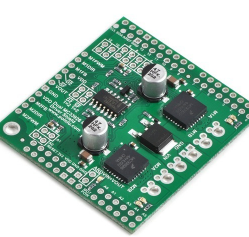
* **I2C: 4 (SDA) и 5 (SCL).** Посредством выводов осуществляется связь I2C.

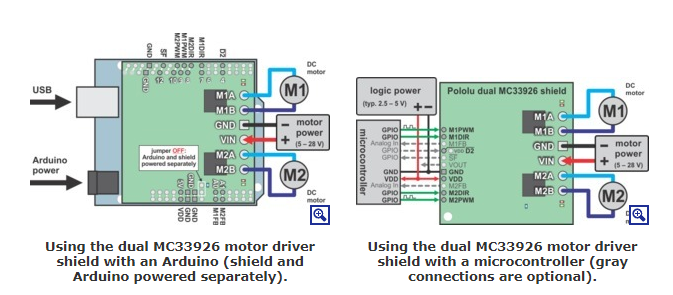
Дополнительная пара выводов платформы:

* **AREF.** Опорное напряжение для аналоговых входов. Используется с функцией.
* **Reset.** Низкий уровень сигнала на выводе перезагружает микроконтроллер. Обычно применяется для подключения кнопки перезагрузки на плате расширения, закрывающей доступ к кнопке на самой плате Arduino.

# Контроллер Pololu

В качестве драйвера для управления мотором актуатора был выбран контроллер Pololu.





|  |  |
| --- | --- |
| **Минимальное рабочее напряжение:** | 5 V[**2**](https://www.pololu.com/product/2503/specs#note2) |
| **Максимальное рабочее напряжение:** | 28 V[**3**](https://www.pololu.com/product/2503/specs#note3) |
| **Выходной ток на канал:** | 3 A[**4**](https://www.pololu.com/product/2503/specs#note4) |
| **Current sense:** | 0.525 V/A |
| **Максимальная частота ШИМ:** | 20 kHz |
| **Минимальное логическое напряжение:** | 2.5 V |
| **Максимальное логическое напряжение:** | 5.5 V |

# Актуатор

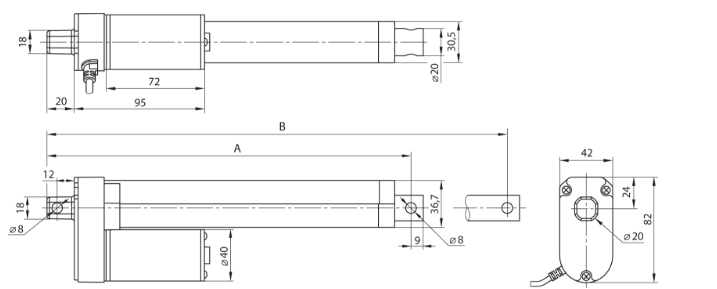
В робототехнике и механике для организации линейного движения, а не вращательного как у двигателей с валом, зачастую применяют устройства, называемые линейные актуаторы. В основном актуаторы подразделяют на три типа: пневматические, гидравлические и электрические. В роботах и различных радиолюбительских проектах в основном применяется третий тип актуаторов благодаря невысокой цене и простоте подключения к электрическому оборудованию системы. В основе электрического линейного актуатора находится обычный электродвигатель. Он может быть как постоянного, так и переменного тока. Зачастую применяют двигатель постоянного тока. Механизм линейного актуатора осуществляет преобразование вращательного момента двигателя в поступательное движение рабочего механизма актуатора. Для этого в составе линейного актуатора должен быть редуктор и ходовой винт.

Итак, для работы линейного актуатора используется ходовой винт, приводимый в движение двигателем постоянного тока. Благодаря этому и создается линейное движение устройства постоянного напряжения. При отключении питания линейный привод зафиксирует свое положение.

Линейные актуаторы LAM3

Линейный актуатор LAM3 состоит из двигателя постоянного тока (12В или 24В), редуктора и винта, интегрированные в единый механизм компактного размера для совершения линейных перемещений. Актуаторы имеют функцию самоблокировки.





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Основные характеристики линейных актуаторов LAM3:** | | | | |
|  | **LAM3-S0** | **LAM3-S1** | **LAM3-S2** | **LAM3-S3** |
| Напряжение питания двигателя | 24В постоянного тока | | | |
| Усилие, Н | 120 | 240 | 500 | 750 |
| Максимальная длина хода, мм | 300 | | | |
| Скорость движения, мм/с | 45 - 57 | 22 - 30 | 12 - 17 | 8 - 10 |
| Номинальный ток, А | 2,2 | 1,9 | 1,8 | 1,8 |
| Температура окружающей среды, °C | -20...+65 | | | |
| Класс защиты | IP54 | | | |

## Энкодер актуатора

В качестве датчика линейного положения актуатора был выбран датчик линейного положения ДП-100.

Датчик обеспечивает выполнение следующих функций:

• Измерение углового перемещения вала датчика и выдача счетных импульсов

• Выдача сигнала направления вращения вала датчика





**Технические данные**

• Число импульсов на 1 оборот ДП-10/50 10 или 50 ДП-100 20 или 100

• Выбор положительного направления вращения

• Напряжение питания, В +5 / +12

• Ток потребления, мA 15

• Рабочий диапазон температур, °С -40 … +55

• Относительная влажность воздуха, без конденсации влаги %, при температуре +25°С 95

• Атмосферное давление, КПа 84-107

• Габариты датчика, мм, не более ДП-10/50 ∅56x110 ДП-100 52х75х60

• Масса, кг, не более ДП-10/50 0.9 ДП-100 0.4 Паспорт, руководство по эксплуатации датчиков ДП-10/50, ДП-100 5 4. Параметры выходных сигналов

• Количество выходных сигналов 2

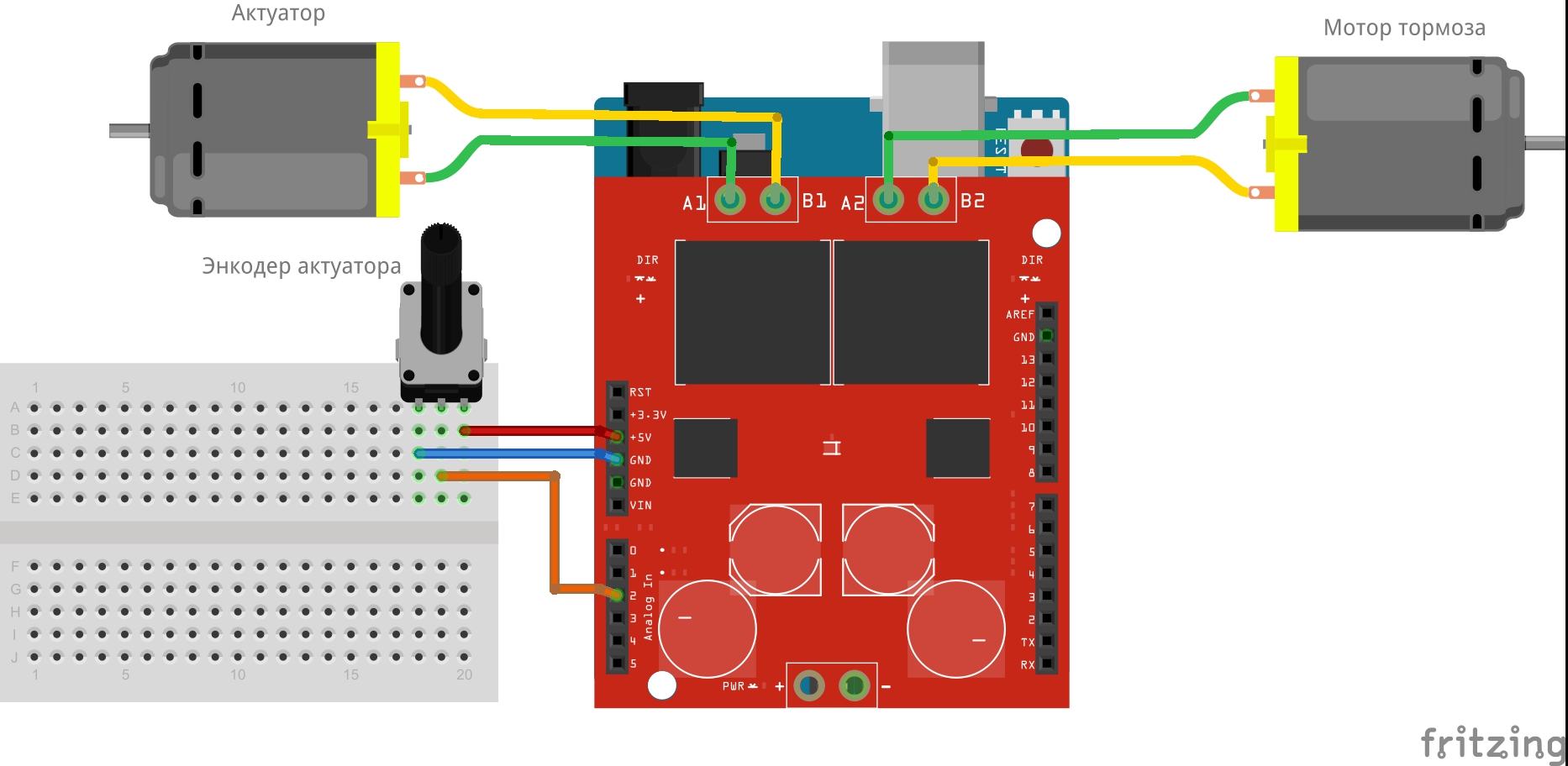
• Амплитуда выходных сигналов, В 12

• Длительность счетных импульсов, мкс 100

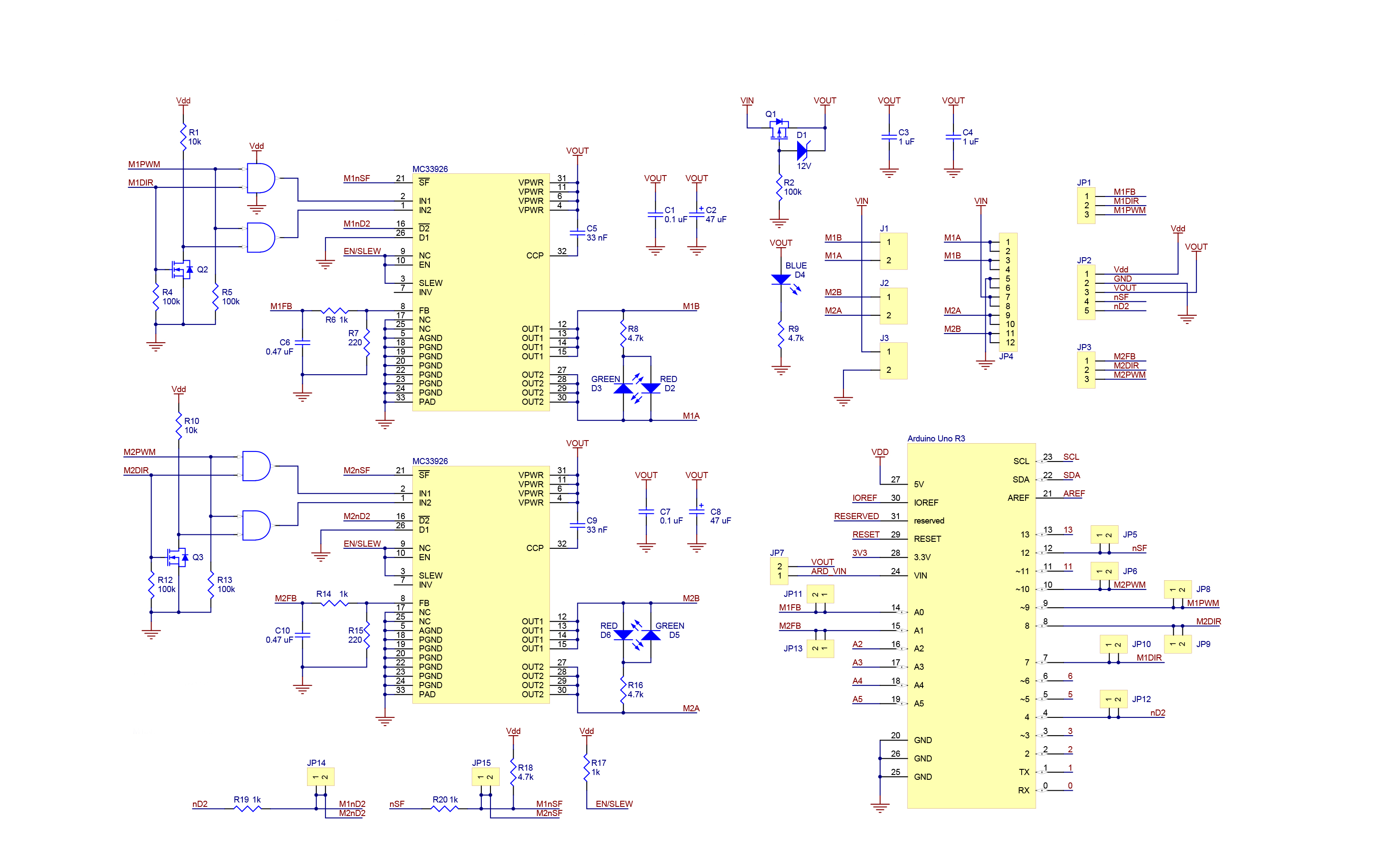
• Сопротивление нагрузки, ом 240 … 10000

• Максимальный ток нагрузки, мА 50

# Схема соединений



# Принципиальная схема контроллера актуатора



# Алгоритм работы

Arduino выдает 2 ШИМ-сигнала и 2 цифровых (по одному каждого типа на один мотор). Силовой модуль (схема которого есть) эти сигналы преобразует в выходные сигналы на моторы (max. 5A, 28 V).

Актуатор имеет два контакта для подключения. Направление движения штока (выдвигание/задвигание) осуществляется изменением полярности. С подачей напряжения начинается движение штока. Внутри актуатора стоят два концевых микропереключателя, так что в крайних положениях штока двигатель останавливается даже при подведённом питании.

Тормоз не имеет обратной связи, актуатор – имеет. У актуатора в качестве обратной связи используется энкодер. Фактически, этот энкодер – переменный резистор хорошего качества.

В листинге пример П-регулятора (пропорционального).

# Реализация системы

#include "DualMC33926MotorShield.h"

DualMC33926MotorShield md;

void stopIfFault()

{

if (md.getFault())

{

Serial.println("fault");

while(1);

}

}

void setup()

{

Serial.begin(115200);

Serial.println("Dual MC33926 Motor Shield");

md.init();

}

void loop()

{

int min\_pos = 0;

int max\_pos = 255;

int target\_pos = 128;

float p = 10;

int min\_sensorValue = 0;

int max\_sensorValue = 580;

int sensorValue = analogRead(A2) ;

float a = (sensorValue - min\_sensorValue) / float(max\_sensorValue - min\_sensorValue);

float current\_pos = a \* (max\_pos - min\_pos) + min\_pos;

float motor\_signal = (target\_pos - current\_pos) \* p;

if(motor\_signal > 400) motor\_signal = 400;

if(motor\_signal < -400) motor\_signal = -400;

/\*Serial.print(sensorValue);

Serial.print("\t");

Serial.print(a);

Serial.print("\t");

Serial.println(current\_pos);

delay(200);\*/

md.setM1Speed(round(motor\_signal));

}

//При интервале таймера счетчика импульсов энкодера в 200000 (0.2 мc)

//Скорость Мотор 1 Мотор 2

//

//При скорости 255 8-12 27-32

//

//При скорости 150 4-11 16-20

//При интервале таймера счетчика импульсов энкодера в 1000000 (1 c)

//Скорость Мотор 1 Мотор 2

//

//При скорости 255 55-60 138-141

//

//При скорости 150 38 84

//

//При скорости 100 стоит 45-50

//Скорости выравниваются при 200 и 100

#include <AFMotor.h>

#include "TimerOne.h"

// ---------------------- motors ----------------------------

const int PIN\_ENCODER\_LEFT = 2;

const int PIN\_ENCODER\_RIGHT = 3;

AF\_DCMotor motor2(4);

AF\_DCMotor motor1(3);

// ---------------------- encoders ----------------------------

// 20 .. 50

volatile unsigned int pulsesLeft;

volatile unsigned int pulsesRight;

volatile int pulsesLeftPerTimeUnit;

volatile int pulsesRightPerTimeUnit;

// ---------------------- wifi ----------------------------

char command;

// -------------------- timer --------------------------

long timerInterval = 200000; // 0.2 секунды

// -------------------- pi-regular --------------------------

// Диапазон сигнала управления

float minControlActionLeft = 100;

float maxControlActionLeft = 150;

float maxErrorLeft = 30;

float minControlActionRight = 200;

float maxControlActionRight = 255;

float maxErrorRight = 50;

struct PI\_CONTROLLER {

float Kp;

float Ki;

float integral;

float min\_integral;

float max\_integral;

PI\_CONTROLLER(float Kp0, float Ki0, float mini, float maxi) {

Kp = Kp0;

Ki = Ki0;

integral = 0;

max\_integral = maxi;

min\_integral = mini;

}

float Eval(float err) {

integral = integral + err;

if (integral > max\_integral) integral = max\_integral;

if (integral < min\_integral) integral = min\_integral;

return Kp \* err + Ki \* integral;

}

float y2u(float y,float maxError, float minControlAction, float maxControlAction) {

float u;

float maxY = Kp \* maxError;

// Serial.print("maxY: ");

// Serial.print(maxY, DEC);

float minY = -maxY;

u = (y - minY) / (maxY - minY) \* (maxControlAction - minControlAction) + minControlAction;

// Serial.print("u: ");

// Serial.print(u, DEC);

if (u > maxControlAction) u = maxControlAction;

if (u < minControlAction) u = minControlAction;

// Serial.print("u: ");

// Serial.print(u, DEC);

return u;

}

};

PI\_CONTROLLER piControllerLeft(2, 0.2, -100, 100);

PI\_CONTROLLER piControllerRight(2, 0.2, -100, 100);

void setup() {

Serial.println("log.info: setup start");

Serial3.begin(115200);

Serial.begin(9600);

Timer1.initialize(timerInterval);

Timer1.attachInterrupt(timerInterceptor);

pinMode(PIN\_ENCODER\_LEFT, INPUT);

pinMode(PIN\_ENCODER\_RIGHT, INPUT);

pulsesLeft = 0;

pulsesRight = 0;

attachInterrupt(digitalPinToInterrupt(PIN\_ENCODER\_LEFT), incrementPulsesLeft, FALLING);

attachInterrupt(digitalPinToInterrupt(PIN\_ENCODER\_RIGHT), incrementPulsesRight, FALLING);

motor2.run(RELEASE);

motor1.run(RELEASE);

Serial.println("log.info: setup end");

}

int goalSpeedLeft = 40;

int goalSpeedRight = 40;

volatile int leftSpeedCurrent;

void loop() {

float errorLeft = goalSpeedLeft - pulsesLeftPerTimeUnit;

float YLeft = piControllerLeft.Eval(errorLeft);

float ULeft = piControllerLeft.y2u(YLeft, maxErrorLeft, minControlActionLeft, maxControlActionLeft);

motor2.setSpeed(ULeft);

motor2.run(FORWARD);

float errorRight = goalSpeedRight - pulsesRightPerTimeUnit;

float YRight = piControllerRight.Eval(errorRight);

float URight = piControllerRight.y2u(YRight, maxErrorLeft, minControlActionRight, maxControlActionRight);

motor1.setSpeed(URight);

motor1.run(FORWARD);

Serial.print("g: ");

Serial.print(goalSpeedLeft, DEC);

Serial.print(" speed. ");

Serial.print("left: ");

Serial.print(pulsesLeftPerTimeUnit, DEC);

Serial.print(" right: ");

Serial.print(pulsesRightPerTimeUnit, DEC);

Serial.print(" ");

Serial.print(YRight, DEC);

Serial.print(" errorLeft ");

Serial.print(errorLeft, DEC);

Serial.print(" errorRight ");

Serial.print(errorRight, DEC);

Serial.print(" ul ");

Serial.print(ULeft, DEC);

Serial.print(" ur ");

Serial.print(URight, DEC);

receiveCommand();

doAction(command);

Serial.println();

delay(50);

}

void timerInterceptor() {

pulsesLeftPerTimeUnit = pulsesLeft;

pulsesLeft = 0;

pulsesRightPerTimeUnit = pulsesRight;

pulsesRight = 0;

}

void incrementPulsesLeft() {

pulsesLeft++;

}

void incrementPulsesRight() {

pulsesRight++;

}

void doAction(const char direction) {

// Serial.println("log.debug: doAction " + direction);

switch (direction) {

case 'f':

moveForward();

break;

case 'b':

moveBack();

break;

case 'r':

turnRight();

break;

case 'l':

turnLeft();

break;

case '!':

stopAction();

break;

}

}

void receiveCommand() {

while (Serial3.available() > 0) {

command = Serial3.read();

Serial.println("log.debug: receiveCommand");

}

}

void moveForward() {

Serial.println("log.debug: moveForward");

motor2.run(FORWARD);

motor1.run(FORWARD);

}

void moveBack() {

Serial.println("log.debug: moveBack");

motor2.run(BACKWARD);

motor1.run(BACKWARD);

}

void turnRight() {

Serial.println("log.debug: turnRight");

}

void turnLeft() {

Serial.println("log.debug: turnLeft");

}

void stopAction() {

Serial.println("log.debug: stopAction");

motor2.run(RELEASE);

motor1.run(RELEASE);

}

int calculateDistanceToFirstTurn()

int calculateDistanceAfterFirstTurn()

bool receiveGoalCoordinates(int x, int y) {

if(x > 200 || x < 1 || y > 200 || x < 1) {

Serial.println("wrong x or y");

return false;

}

goalX = x;

goalY = y;

return true;

}