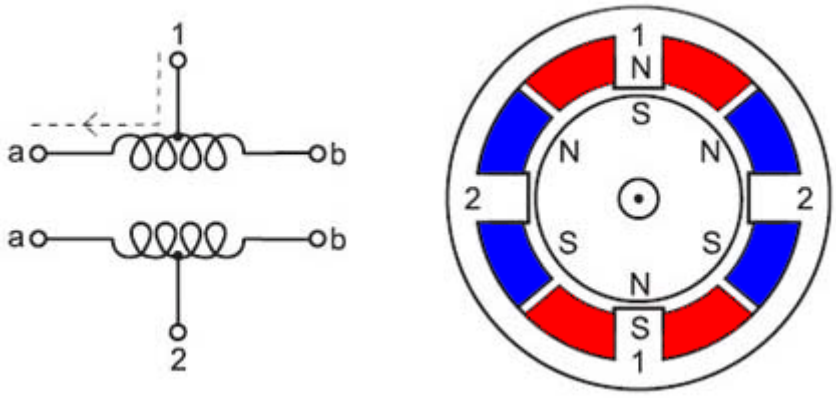
Подключение униполярного шагового двигателя к микроконтроллерам AVR

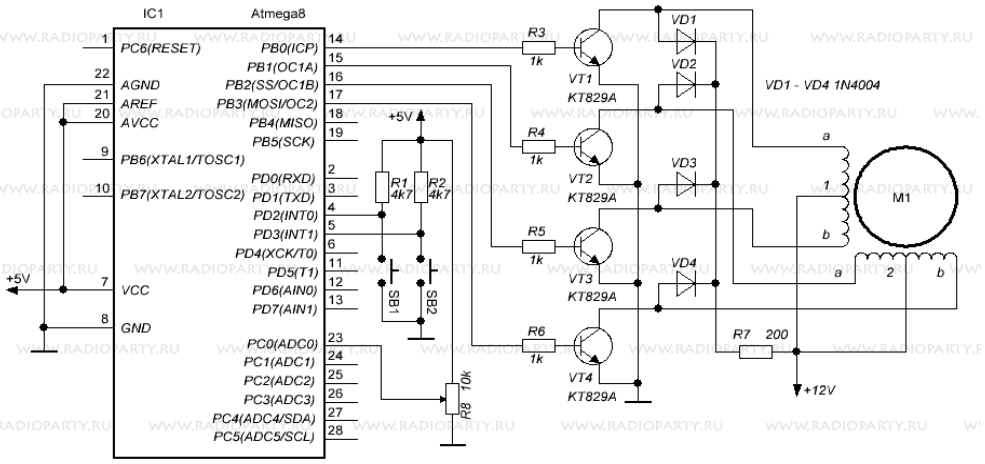
Цель: Изучить работу униполярного шагового двигателя, реализовать подключение двигателя к микроконтроллеру ATMEGA8.

Униполярный двигатель имеет одну обмотку в каждой фазе, от середины обмотки сделан отвод. Это позволяет изменять направление магнитного поля, создаваемого обмоткой, простым переключением половинок обмотки. При этом существенно упрощается схема драйвера. Драйвер должен иметь только 4 простых ключа. Средние выводы обмоток могут быть объединены внутри двигателя, поэтому такой двигатель может иметь 5 или 6 выводов. Иногда униполярные двигатели имеют раздельные 4 обмотки, по этой причине их ошибочно называют 4-х фазными двигателями. Каждая обмотка имеет отдельные выводы, поэтому всего выводов 8. Из-за простоты подключения и управления униполярного двигателя рассмотрим именно этот тип шагового двигателя.



Для управления 6-выводным униполярным шаговым двигателем нужно 4 независимых управляющих элемента - транзистора VT1 - VT4. Транзисторы используются составные типа КТ829, КТ973 с защитными диодами внутри или импортные аналоги. Можно взять отдельные транзисторы, можно использовать специальную микросхему ULN2003A, которая содержит целых

семь транзисторов, и кроме того, там есть еще 7 защитных диодов, которые пропускают через себя ток самоиндукции при переключении обмоток.



Управляющий контроллер Atmega8, тактируется от внутреннего генератора частотой 8МГц. В программе используем два внешних прерывания и прерывание по переполнению таймера T0. Все прерывания определяем и настраиваем в главной функции, также настраиваем порты ввода/вывода микроконтроллера.

Для запуска двигателя в ту или иную сторону необходимо подать на его обмотки последовательность импульсов, сдвинутых по фазе. Эти последовательности импульсов определим в массивах **cw_dir[]** и **ccw_dir[],** соответственно по часовой стрелке и против часовой стрелки. Указатель направления вращения **status** меняет свое состояние с лог.0 на лог.1 и наоборот при нажатии на одну из кнопок SB1 и SB2. Если status = 1 двигатель вращается против часовой, если status = 0 двигатель вращается по часовой стрелке. Переменная **status** меняет свое значение при наступлении внешних прерываний на входах INT0 и INT1. Чтобы происходило внешнее прерывание подтягиваем входы INT0 и INT1 через резисторы к плюсу питания.

Сигналы управления обмотками двигателя формируются на портах PB3 – PB0 программно. Формирование импульсных последовательностей выполняется в обработчике прерывания таймера 0. Переключение фаз происходит только при переполнении программного таймера. Переменная **ovftimes** определяет величину задержки между чередованиями импульсов. Ее значение связано со значением АЦП, чем больше значение АЦП, тем медленнее вращается вал двигателя и наоборот. Регулировка скорости вращения осуществляется переменным резистором R8, средний вывод которого подключен к каналу ADC0.

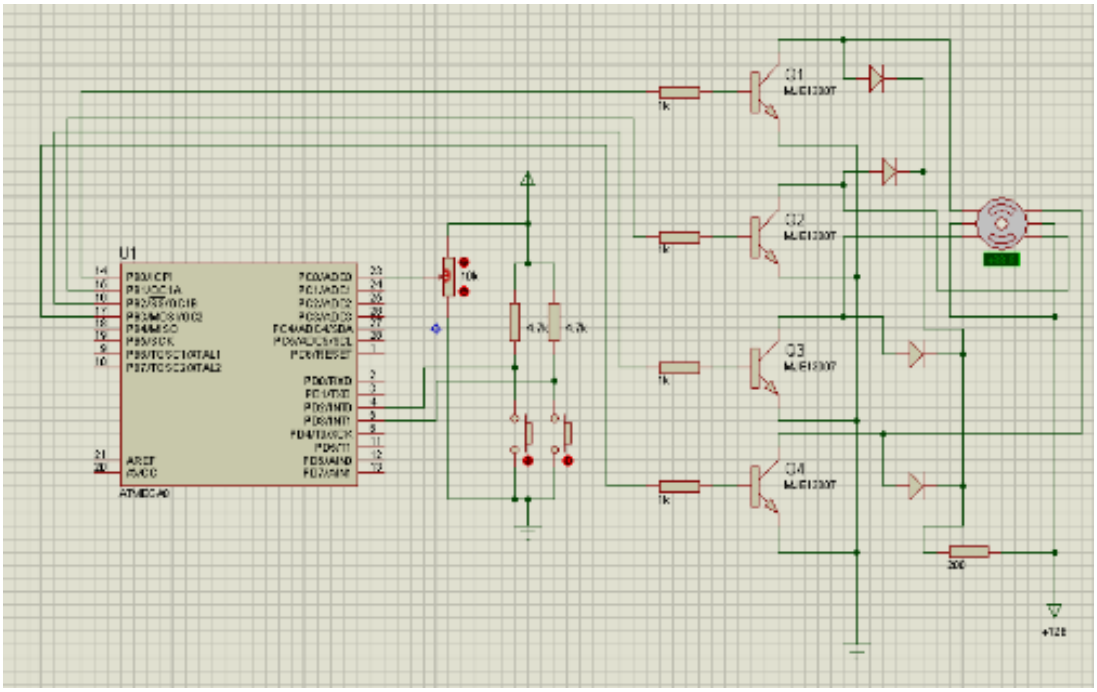
Код управляющей программы:

```
// Подключение униполярного шагового двигателя к AVR
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
// Полношаговый режим 2 фазы
// Направление вращения по часовой стрелке
unsigned char cw_dir[4]=
{
0b00000001,
0b00000010,
0b00000100,
0b00001000
};
// Направление вращения против часовой стрелки
unsigned char ccw_dir[4]=
{
0b00001000,
0b00000100,
0b00000010,
0b00000001
};
volatile unsigned char step_index;
volatile unsigned int ovftimes;
volatile unsigned char status;
// Прерывание по переполнению T0
ISR(TIMER0_OVF_vect)
{
static unsigned int count = 1;
count++;
if(count >= ovftimes) // Применяем задержку
{
cli(); // Запрещаем прерывания
if(status) // если status == 1 крутим против часовой
PORTB = ccw_dir[step_index++];
else       // иначе крутим по часовой
PORTB = cw_dir[step_index++];
if (step_index >= 4)
step_index=0;
count = 0; // Сброс счетчика
TCNT0 = 0; // Старт счетчика с нуля
sei(); // Глобально разрешаем прерывания
}
```

```
}
// прерывание по вектору INT0
ISR(INT0_vect)
{
status = 0; // по часовой
}
// прерывание по вектору INT1
ISR(INT1_vect)
{
status = 1; // против часовой
}
int main(void)
{
DDRB = 0b00001111; // PB0, PB1, PB2, PB3 - выходы
PORTB = 0x00; // Лог. нули на выходе
ADCSRA = (1<<ADEN) // разрешение АЦП
    | (1<<ADPS2) // предделитель на 64 (частота АЦП 125kHz)
    | (1<<ADPS1);
ADMUX = 0x00; // ADC0 - вход, внешний ИОН 5 Вольт

GICR |= (1<<INT1)|(1<<INT0); // Разрешаем внешние прерывания
MCUCR |= (1<<ISC11) // Прерывание по заднему фронту INT1
      |(1<<ISC01); // Прерывание по заднему фронту INT0
TCCR0 |= (1<<CS01); // Предделитель на 8
TCNT0 = 0; // Старт счетчика с нуля
TIMSK |= (1<<TOIE0); // Разрешаем прерывания по переполнению T0
step_index = 0;
ovftimes = 10; // первоначальная задержка
status = 0; // при включении вращение по часовой
sei(); // Глобально разрешаем прерывания
while(1)
{
ADCSRA |= (1<<ADSC); // Начинаем преобразование
while (ADCSRA & (1<<ADSC)); // Ждем пока завершится
преобразование
ovftimes = ADCW; // Значение временной задержки
}
}
```

Пример реализации в Proteus:



Задания для выполнения:

1. Изучить принцип работы униполярного шагового двигателя и законспектировать.
2. Смоделировать представленную программу в среде Proteus.

Результаты выполнения отправить на e-mail: rasov@rambler.ru с темой STEP_MOTOR_ФИО