Steuerung von Schrittmotoren mit dem ATmega32

Schrittmotoren besitzen einen Rotor mit einer ganzen Reihe permanent-magnetischer Pole. Die Bewegung des Rotors erfolgt durch eine Folge elektrischer Ummagnetisierungen der Statorpole. Bei jedem Ummagnetisierungsschritt bewegt sich der Rotor um einen definierten Drehwinkel weiter. Dieser Drehwinkel hängt von der Anzahl der Rotor- und Statorpolpaare ab.

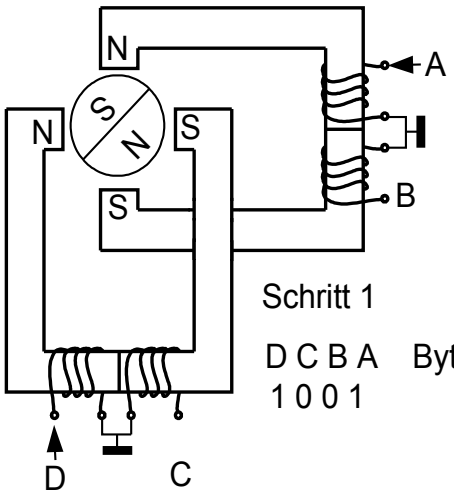
Man unterscheidet unipolare und bipolare Schrittmotoren.
Unipolare Schrittmotoren haben 2 Wicklungen mit Mittelanzapfung. Sie haben 5 bis 6 Anschlüsse.
Bipolare Schrittmotoren haben 2 Wicklungen ohne Mittelanzapfung. Sie haben 4 Anschlüsse.

Unipolare Schrittmotoren lassen sich einfach mit einer Spannungspolarität ansteuern, während bei bipolaren Schrittmotoren die Spannung zur Ansteuerung umgepolt werden muss.

Man unterscheidet Vollschritt- und Halbschrittbetrieb.

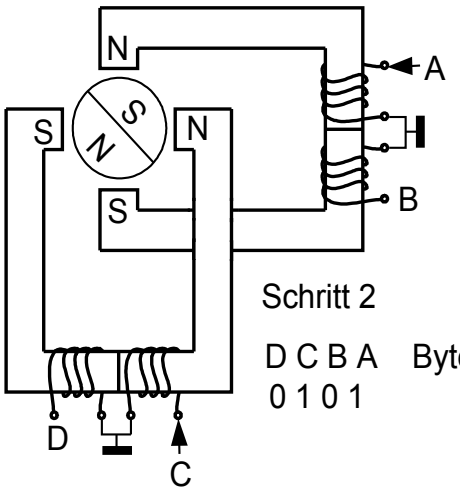
Unipolarer Schrittmotor: Vollschrittbetrieb

Die Abbildung veranschaulicht die Arbeitsweise. bei der Drehung des Motors werden periodisch 4 Schritte durchlaufen. Der vereinfacht dargestellte Motor mit einen Statorpolpaar dreht sich bei jedem Schaltschritt um 90 Grad nach rechts.
Für die Ansteuerung des Motors vom Mikrocontroller her ist ein Motortreiber-Baustein erforderlich. Hierzu eignet sich hervorragend das IC **ULN2803A**.



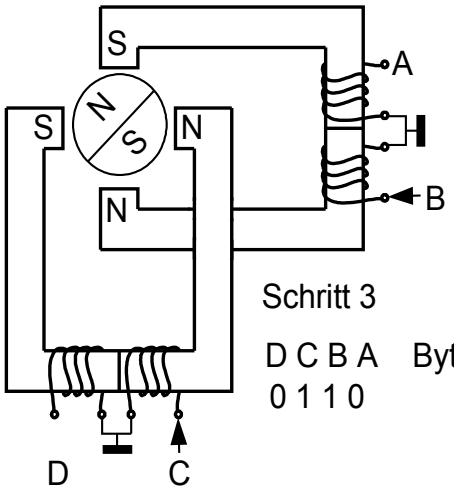
Schritt 1

D	C	B	A	Bytewert
1	0	0	1	9



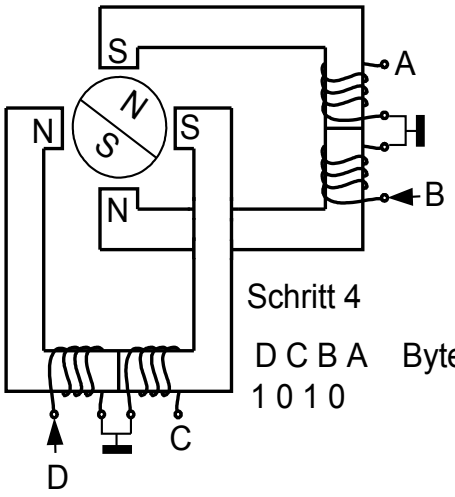
Schritt 2

D	C	B	A	Bytewert
0	1	0	1	5



Schritt 3

D	C	B	A	Bytewert
0	1	1	0	6

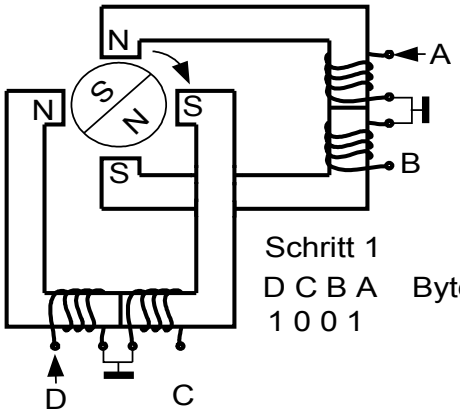


Schritt 4

D	C	B	A	Bytewert
1	0	1	0	10

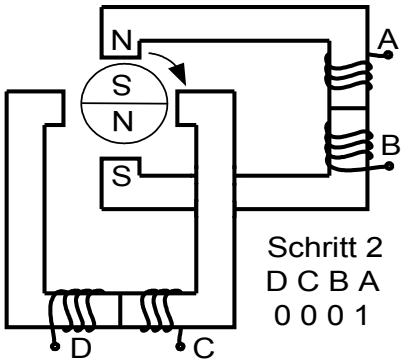
Unipolarer Schrittmotor: Halbschrittbetrieb

Während im Vollschrittbetrieb in den 4 Schritten immer beide Wicklungen stromdurchflossen sind, werden im Halbschrittbetrieb 4 weitere Schritte eingebaut, bei denen nur jeweils eine Wicklung eingeschaltet ist.
Bei jedem Schritt dreht sich unser Modellmotor um 45 Grad. Die Auflösung der Drehbewegung in Einzelschritte wird damit verdoppelt (8 Schritte statt 4 Schritte). Der Motor dreht sich deshalb bei gleicher Schrittgeschwindigkeit nur noch halb so schnell.
Bei der Ansteuerung eines Schrittmotors muss zwischen den Schaltschritten eine kurze Wartezeit eingebaut werden. Bei zu kleiner Wartezeit kann der Schrittmotor der Schrittgeschwindigkeit nicht mehr folgen. Die maximal mögliche Schrittgeschwindigkeit muss bei fehlendem Datenblatt experimentell ermittelt werden.



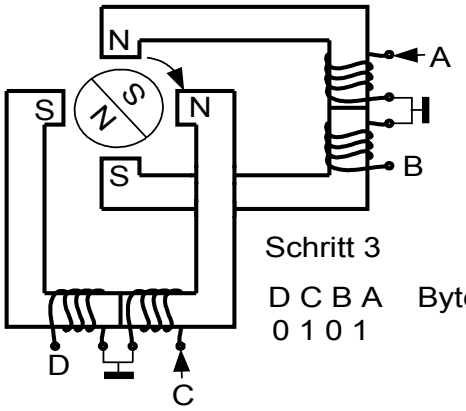
Schritt 1

D	C	B	A	Bytewert
1	0	0	1	9



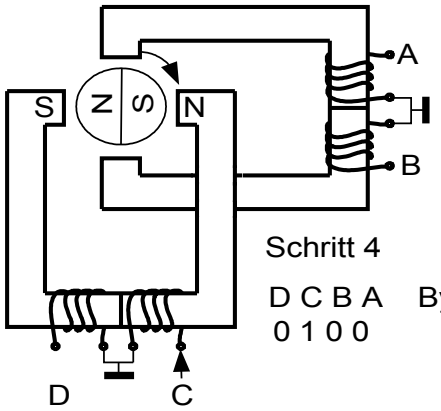
Schritt 2

D	C	B	A	Bytewert
0	0	0	1	1



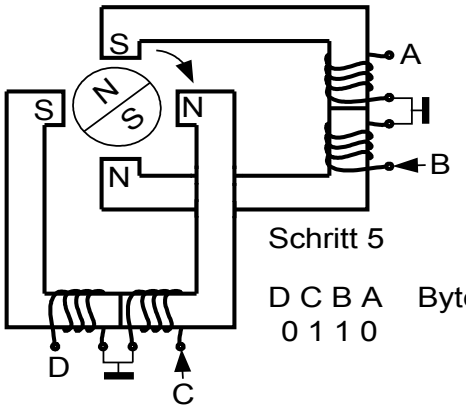
Schritt 3

D	C	B	A	Bytewert
0	1	0	1	5



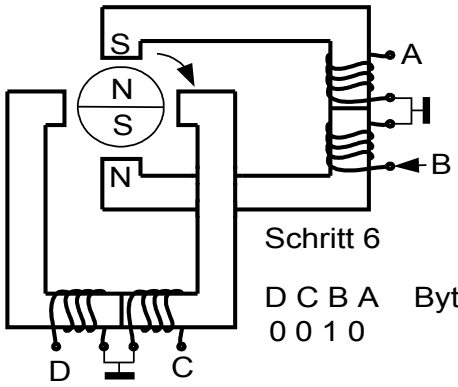
Schritt 4

D	C	B	A	Bytewert
0	1	0	0	4



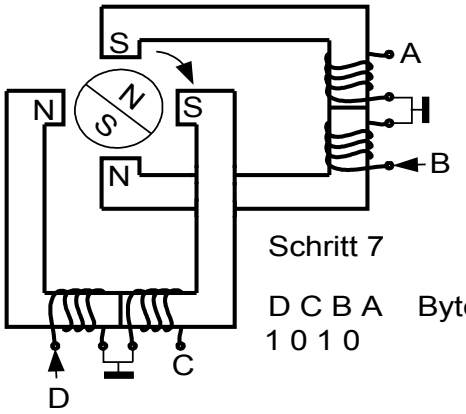
Schritt 5

D	C	B	A	Bytewert
0	1	1	0	6



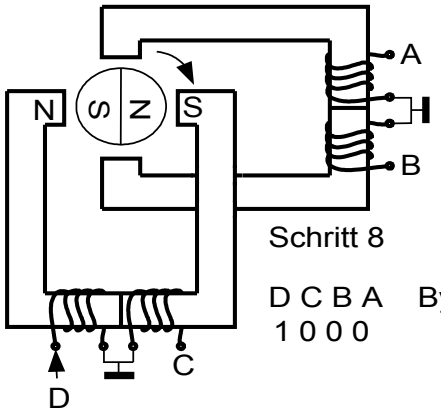
Schritt 6

D	C	B	A	Bytewert
0	0	1	0	2



Schritt 7

D	C	B	A	Bytewert
1	0	1	0	10

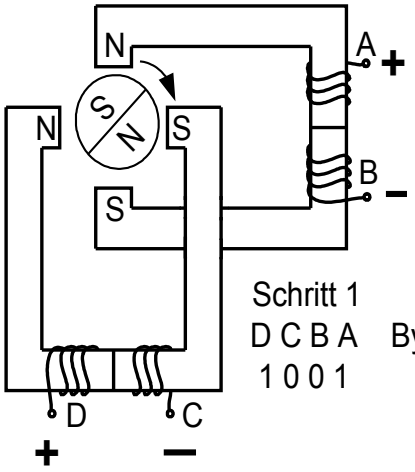


Schritt 8

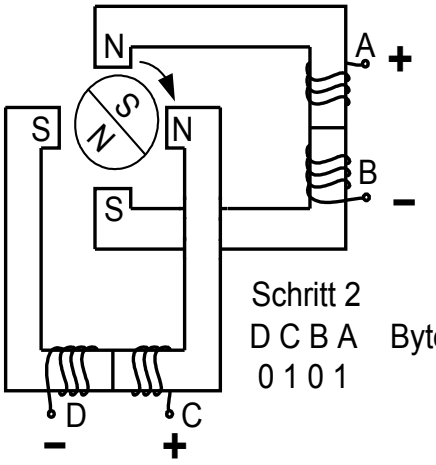
D	C	B	A	Bytewert
1	0	0	0	8

Bipolarer Schrittmotor: Vollschrittbetrieb

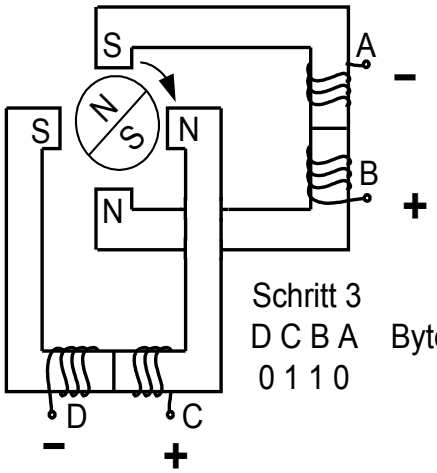
Beim bipolaren Schrittmotor müssen die Spannungen an den beiden Statorwicklungen bei der Schrittsteuerung umgepolt werden können. Das macht die elektronische Ansteuerung etwas komplizierter.
Bei geeigneter Ansteuerelektronik zum Beispiel mit den IC **L298N Muliwatt-15** kann der bipolare Schrittmotor mit derselben Bytefolge angesteuert werden wie der unipolare Schrittmotor:



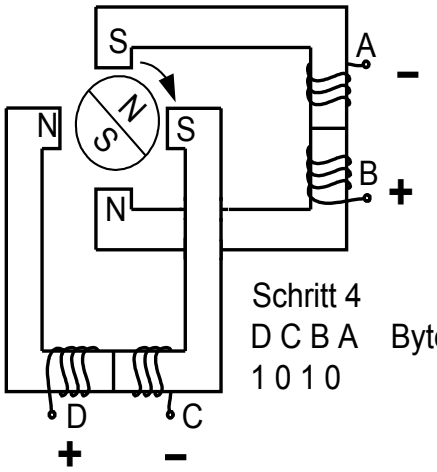
Schritt 1
D C B A Bytewert
1 0 0 1 9



Schritt 2
D C B A Bytewert
0 1 0 1 5



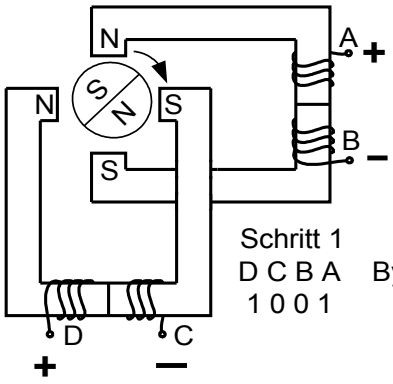
Schritt 3
D C B A Bytewert
0 1 1 0 6



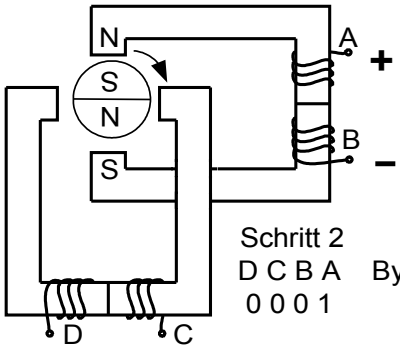
Schritt 4
D C B A Bytewert
1 0 1 0 10

Bipolarer Schrittmotor: Halbschrittbetrieb

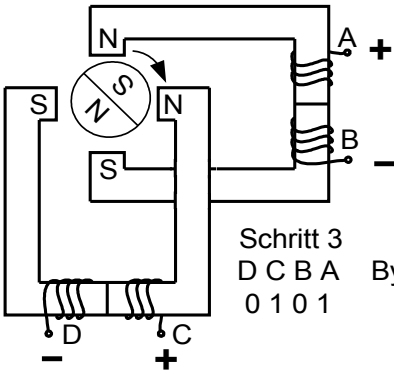
Auch hier ergibt sich durch die 4 zusätzlichen Zwischenschritte eine Halbierung des Schrittwinkels und eine Halbierung der Drehzahl.



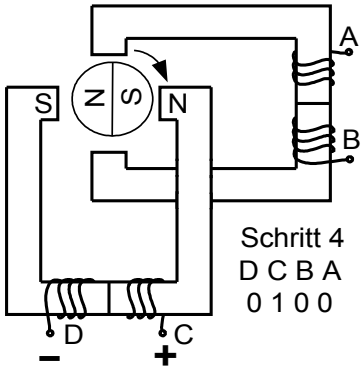
Schritt 1
D C B A Bytewert
1 0 0 1 9



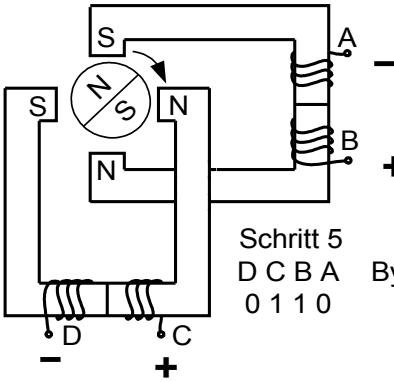
Schritt 2
D C B A Bytewert
0 0 0 1 1



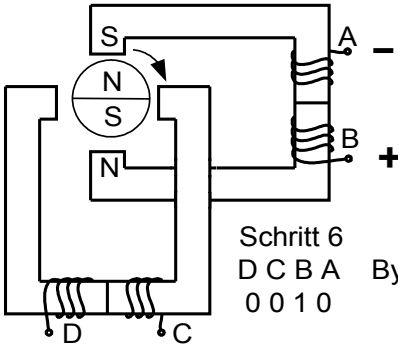
Schritt 3
D C B A Bytewert
0 1 0 1 5



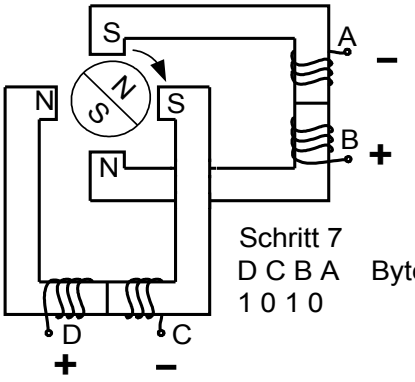
Schritt 4
D C B A Bytewert
0 1 0 0 4



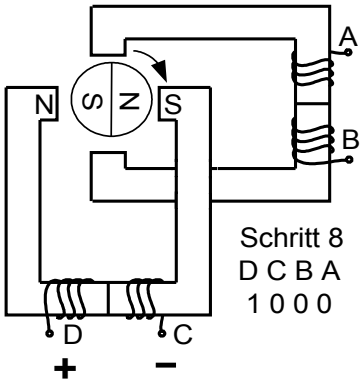
Schritt 5
D C B A Bytewert
0 1 1 0 6



Schritt 6
D C B A Bytewert
0 0 1 0 2



Schritt 7
D C B A Bytewert
1 0 1 0 10



Schritt 8
D C B A Bytewert
1 0 0 0 8

Schrittmotor-Steuerprogramm in C

In C schreibt man die Folge der Bytewerte zum Ansteuern des Schrittmotor am besten in Arrays:

```
uint8_t vs[4]={9,5,6,10},              //Tabelle Vollschritt Rechtslauf
        hs[8]={9,1,5,4,6,2,10,8};  //Tabelle Halbschritt Rechtslauf
```

Will man nun z.B. 1000 Schritte benötigt man eine Variable z zum Zählen der Anzahl der durchgeführten Schritte und eine Variable n für die Schrittnummer:

```
uint16_t    z, n=0;
```

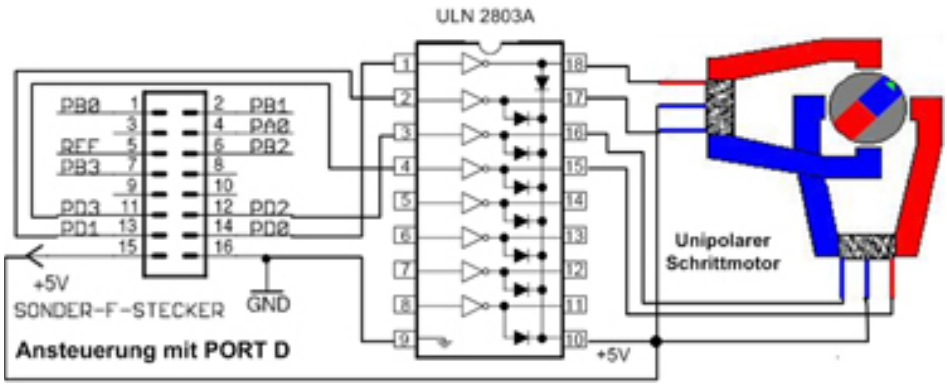
Das Programm für Vollschrittbetrieb kann dann so aussehen:

```
DDRA=0x0f; //Motor ist an Bit 0,1, 2 und 3 von Port A angeschlossen.
for (z=0; z<1000; z++){  PortA=vs[n & 3]; warte10ms(); n++;}
```

Mit der Und-Verknüpfung n & 3 werden aus der Schrittvariablen n die letzten beiden Bits ausmaskiert. n & 3 durchläuft so beim Hochzählen von n periodisch die Werte 0, 1, 2 und 3.
Mit PortA=vs[n & 3] wird der Bytewert aus der Tabelle vz gelesen und zum Schrittmotor übertragen.
Die Funktion warte10ms() sorgt dann für notwendige die Schaltverzögerung von 10 ms.
Mit n++; wird der nächste Rechtslaufschritt vorbereitet. Setzt man hier n--; ein läuft der Motor im Linksrichtung.

Oben wurde der Startwert der Schrittvariablen n einfach auf 0 gesetzt. Das ist nicht ganz korrekt.
Denn man weiß nie, wie der Schrittmotor beim Start steht. Deshalb steuert man üblicherweise nach dem Start des Programms einen Rererenzpunkt als Nullmarke an, bei dem ein Endschalter betätigt wird. Danach hat die Schrittvariable n einen exakten Startwert.

Meine Schaltung



Das vollständige Programm

```
#include <avr/io.h>
#include "warten.h"

//Steuerung eines unipolaren Schrttmotors
(3.6 Grad/Vollschritt)
// W.Tschallener 4.10.2006

//Tabelle Vollschritt
uint8_t vs[4]={9,5,6,10},

//Tabelle Halbschritt
        hs[8]={9,1,5,4,6,2,10,8},
            n;

void rechtslauf_Vschritte(uint16_t ns){
uint16_t z;
for ( z=0;z<ns;z++){
 n++;
 PORTD=vs[n&3];
 warten10ms();
 }
}
void linkslauf_Vschritte(uint16_t ns){
uint16_t z;
for(z=0;z<ns;z++){
 n--;
 PORTD=vs[n&3];
 warten10ms();
 }
}
void rechtslauf_Hschritte(uint16_t ns){
uint16_t z;
for (z=0;z<ns;z++){
 n++;
 PORTD=hs[n&7];
 warten10ms();
 }
}
void linkslauf_Hschritte(uint16_t ns){
uint16_t z;
for(z=0;z<ns;z++){
 n--;
 PORTD=hs[n&7];
 warten10ms();
 }
}
void main(){
n=0;
DDRD=15;
PORTB=0x0f;

do{
 rechtslauf_Hschritte(200);
 warten1s();
 linkslauf_Hschritte(200);
 warten1s();
}while(1);
}
```