
6. Stepper motor demo

A standard brushed or brushless motor can rotate; the bigger the applied voltage the faster it rotates. It is difficult to move a standard motor for a given angle or to make it stay in a certain position; there is no torque to hold the rotor in a stationary position. However, this can easily be achieved by a stepper motor. These motors are used where positioning is a primary goal.

A stepper motor basically consists of four stationary electromagnets (A, B, C, D) and a permanent magnet (R) which serves as a rotor. Consider a current is flowing through only one of the electromagnets (A, B, C or D) at a time. Figure 1 gives corresponding orientations of the rotor made of permanent magnet in the middle. It is important that the rotor remains in a given orientation as long as the current is flowing through the same electromagnet, and that the user knows the orientation of the rotor since the user is the one to turn-on a selected electromagnet.

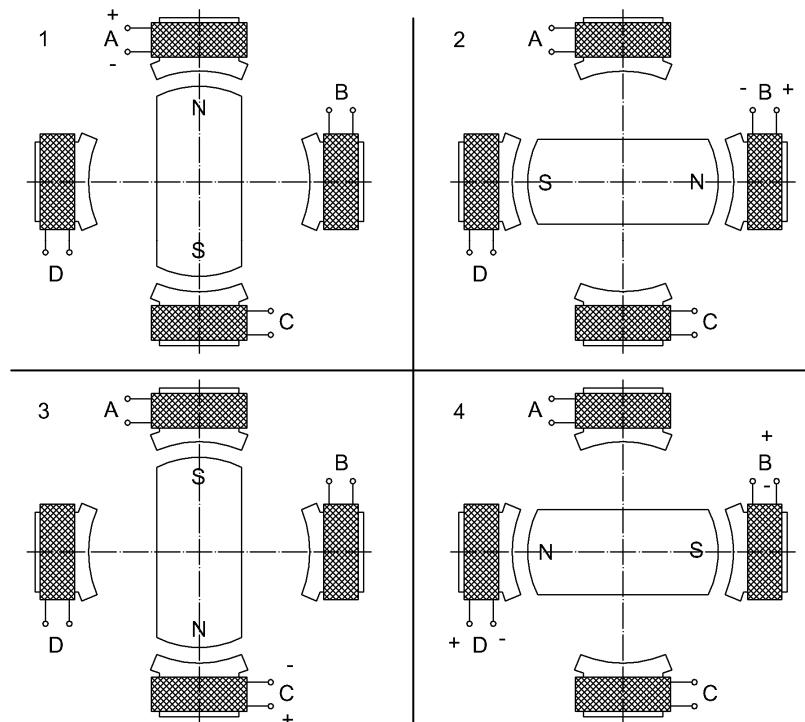


Figure 1: Four (1, 2, 3, 4) possible orientations of the rotor versus the active electromagnet (A, B, C, D)

There are two kinds of stepper motors as far as the connections to the electromagnets are concerned.

- The first, weaker version has one side of each electromagnet connected together and available to the user as a common connection. The other side is made available to the user.

Such stepper motor therefore has **five wires** (sometimes **six**, since the common side of electromagnets is available through two separate wires). The common wire is supposed to be tied to a ground, and other four wires are to be connected to the power supply using four switches as in Fig. 2a. If switches are controlled by four digital signals in a way that an individual bit of a four-bit binary number controls individual switch (1 turns it ON, 0 turns it OFF), then a sequence of numbers 0001, 0010, 0100, 1000, 0001 ... will rotate the rotor in one direction, while the opposite sequence will rotate the rotor in the other direction. The transition from one number to the other number must be slow enough to allow the movement of the rotor.

The current flows through one electromagnet at a time therefore 3 out of 4 of the electromagnets are wasting space inside the motor. In order to boost the use of space and effectiveness (torque), two adjacent electromagnets can be powered simultaneously. This roughly doubles the torque (and current consumption), and is implemented by OR-ing the adjacent numbers from the sequence above to get: 0011, 0110, 1100, 1001, 0011 ... for rotation in the same direction.

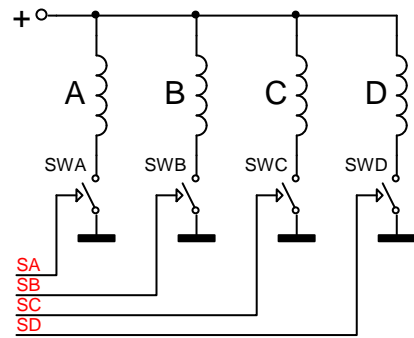


Figure 2: The connection of the five-wire stepper motor

- There is also the second, stronger version which has two diagonally positioned electromagnets connected together, giving altogether **four wires** to drive two pairs of electromagnets. The connection of such motor is shown in Fig. 3. Such connection requires more elaborate combination of switches called “a bridge connection” and the current can flow in one of two possible directions through each of the two electromagnets. The chip included on the test board can drive any version of a motor and since the sequence of numbers needed to rotate this version is the same as above we will not elaborate here anymore. The only thing to emphasize is that both electromagnets are conducting current permanently in this motor; only the direction of current changes to move the rotor. This results in the best utilization of the space inside the motor and maximum torque.

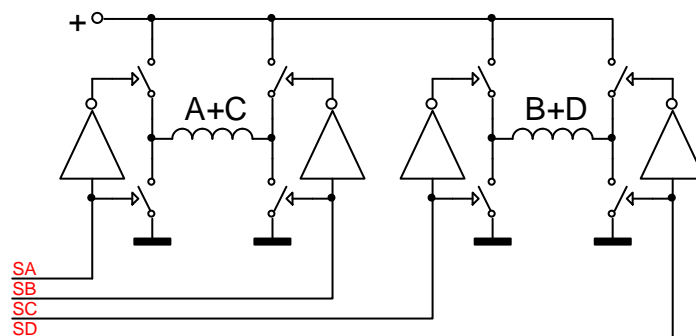


Figure 3: The connection of a four-wire stepper motor

A chip (actually two chips for two motors, but we will use only one in this experiment) capable of driving a stepper motor is included on the test board. It is the L293, an industrial standard driver for small power motors which includes four individually controllable halves of a bridge connection. Maximum current for one half of a bridge is about 0.6A (1A with a version which does not include output clamp diodes) and the chip can work with a power supply from 5V to 36V. Two enable signals are available. All input signals are TTL compatible and protective diodes against voltage spikes caused

by a motor are integrated on the chip. The correct power supply for the motor can be adjusted by adjusting the power supply to the complete board, which can be anywhere between 7V and 24V. If a power supply out of this range is required, the jumper J220 should be extracted and external power supply connected to the remaining header.

From the discussion above a stepper motor will make one turn in four consecutive steps. However, such division is much too coarse for positioning. A typical stepper motor has 200 steps to make one complete circle; therefore one step rotates a rotor for 1.8 degrees. This is done by additional shaping of the core of the electromagnet and the rotor. Consider the drawing in Fig. 4. This time the rotor is not a permanent magnet. When electromagnet A is energized, the grooves on the rotor align with the grooves on the core of the electromagnet A. When electromagnet A is turned off and electromagnet B is turned on, the rotor changes its position to align the grooves with electromagnet B; the rotor rotates for $\frac{1}{4}$ of a groove. Next electromagnet C is turned on and electromagnet B off. The result is an additional rotation for $\frac{1}{4}$ of a groove. The same rotation takes place for the last electromagnet D, therefore in four consecutive steps the rotor turns for one groove. The number of steps for one full turn is defined by 4 times the number of grooves.

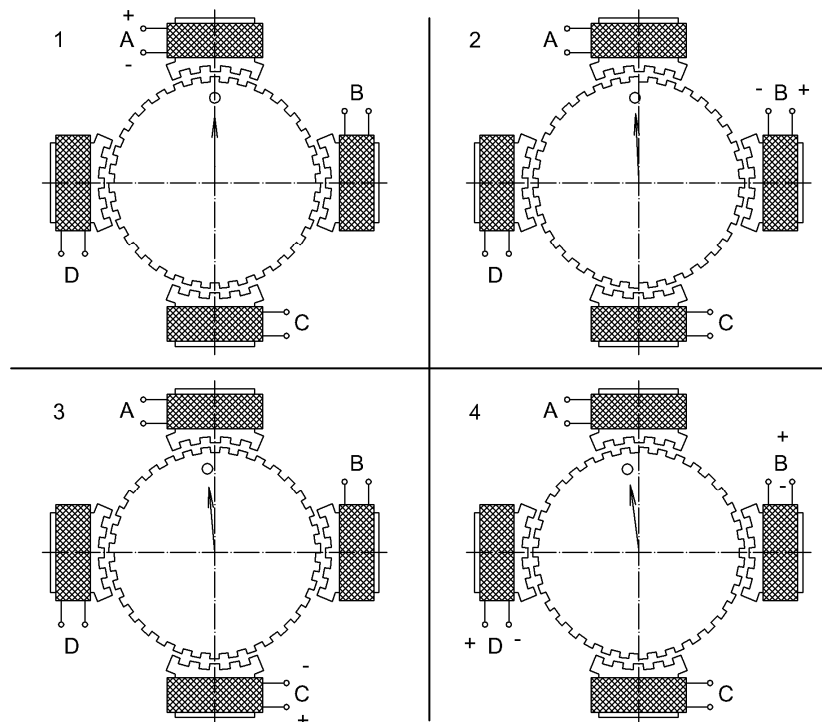


Figure 4: Stepper motor with more steps per revolution

In order to use the driver a stepper motor must first be connected and then port C must be programmed as push-pull outputs at bits PC06 to PC09 and PC12. Here bit PC12 serves as an enable signal for the driver chip, and must be set to high. Other bits serve as control lines for the driver chip, their values follow the pattern stated above. The complete demo program is given in Fig. 4. The program consists of initialization part and the infinite loop. The pattern to be sent out is initialized in an array, and should be adjusted to the connection of the stepper motor to the driver chip.

- The initialization. First line declares and initializes an array with bit patterns for successive steps of a stepper motor. Bear in mind that the bits are located at PC06 to PC09 (therefore bits marked with x in a port of 16 bits 0000 00xx xx00 0000 are used). Next a pointer to the

array is declared and initialized, last two bits of this pointer always point to one of four elements of the array, while the complete pointer gives the position of the motor. This position will be sent to LCD and the LCD needs to be initialized, clock for ports C and E enabled and corresponding bits of port C set-up as push-pull outputs.

- The infinite loop. The program checks the status of switches S370 and S371. If either is pressed then the requested position for a stepper motor is increased or decreased. In any case the requested position is written to the LCD and the element of the array which corresponds to the position selected is sent to port C, then some time is wasted to prevent the instantaneous repeat of the loop.

The complete listing of the demo program is given in Fig. 5.

```
#include "stm32f4xx.h"
#include "LCD2x16.c"

void main (void){
int Steps[4] = {0x0c0, 0x180, 0x300, 0x240};
int PtrSteps = 0, Del = 1000000;

LCD_init(); // init LCD
RCC->AHB1ENR |= 16 + 4; // Enable clock for GPIO C, E: motor, switches
GPIOC->MODER |= 0x001055000; // MODE Register

LCD_string("Stepper PC:K500", 0x00); // display title string
LCD_string("pos=", 0x42); // display initial text

while (1) {
if (GPIOE->IDR & 0x01) PtrSteps++; // if button S370 -> move right
if (GPIOE->IDR & 0x02) PtrSteps--; // if button S371 -> move left
LCD_sInt16(PtrSteps,0x46,1); // write position to LCD
for (int i = 0; i<Del; i++) {}; // waste some time

GPIOC->ODR = (GPIOC->ODR & ~0x3c0) | Steps[PtrSteps & 0x3] | 0x1000; // move
};
}
```

Figure 5: A listing of the program to rotate a stepper motor and display its position