6. A stepper motor

The use of a stepper motor is demonstrated. A standard brushed or brushless motor can rotate; the bigger the applied voltage the faster it rotates, its use is simple and the torque can be high. However, it is very complex to move brushed or brushless motor for a given angle or to make it stay in a selected position; there is no torque to hold the rotor in a stationary position. Retaining the position can easily be achieved by a stepper motor, and they are primarily used where positioning is a main goal.

6.1. Hardware – a stepper motor basics

A stepper motor basically consists of four stationary electromagnets (A, B, C, D) and a permanent magnet (R) which serves as a rotor, and has two magnetic poles N and S. Consider a current flowing through only one of the electromagnets (A, B, C or D) at a time. Figure 6.1 gives corresponding orientations of the rotor made of permanent magnet in the middle. It is to be emphasized 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.

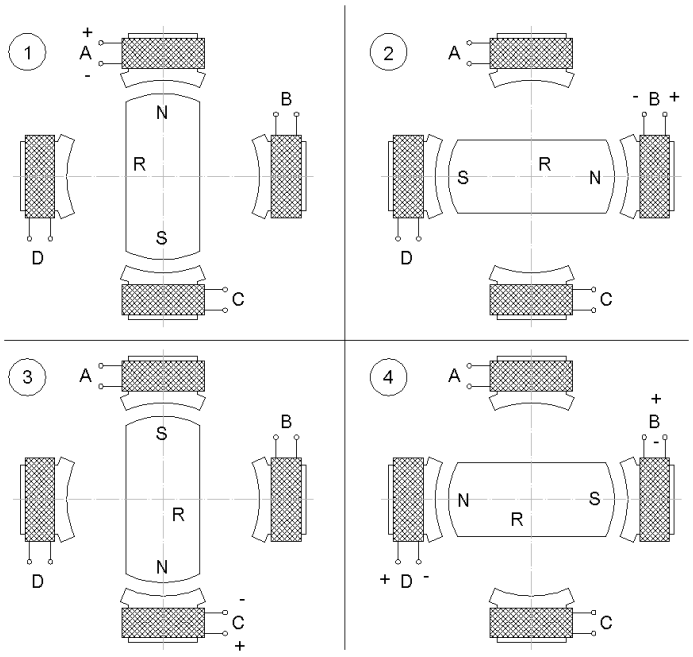

Figure 6.1: Four (1, 2, 3, and 4) possible orientations of the rotor versus the active electromagnet (A, B, C, D)

There are two kinds of stepper motors as far as the wiring of the electromagnets is concerned.

- The first, weaker version, has one side of each electromagnet connected together and exposed to the user. The other side of each electromagnet is also exposed to the user. Such stepper motor has **five wires** (sometimes **six**, since the common side of all electromagnets is available

through two separate wires). The common wire is supposed to be tied to a power supply, and other four wires are to be connected to the ground using four switches as in Fig. 6.2. 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 0001b, 0010b, 0100b, 1000b, 0001b … will rotate the rotor in one direction, while the reverse sequence will rotate the rotor in the other direction. The transition from one number to the other number must be delayed 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 with this the torque of the motor, two adjoining electromagnets can be powered simultaneously. This roughly doubles the torque (and current consumption), and is implemented by OR-ing the adjoining numbers from the sequence above to get: 0011, 0110, 1100, 1001, 0011 … for rotation in the same direction.

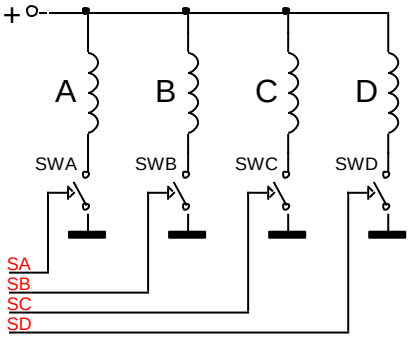

Figure 6.2: The connection of the five-wire stepper motor

- There is also the second, stronger version of a motor which has two transversely positioned electromagnets connected in series, giving altogether **four wires** to drive two pairs of electromagnets. The connection of such motor is shown in Fig. 6.3. It 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 pairs of electromagnets. The sequence of signals required to drive the switches in any version of a motor is the same and we will not elaborate here anymore on this. 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.

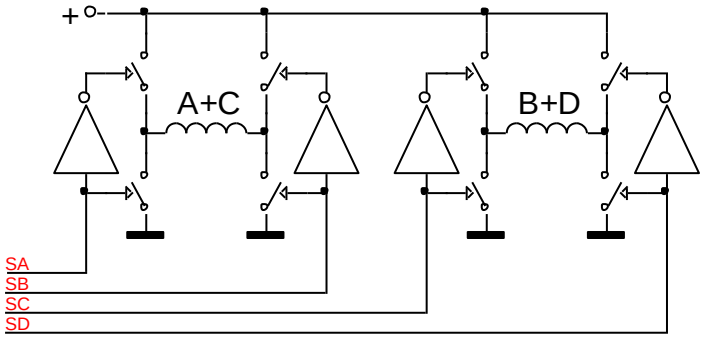

Figure 6.3: The connection of a four-wire stepper motor

A chip capable of driving a stepper motor is included on the BaseBoard. It is the L293D, 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 almost 1A and the chip can work with a power supply from 5V to 36V. The outputs from the chip are initially disabled, but can be enabled using two enable lines. All inputs lines to the chip can be driven directly from the microcontroller and protective diodes against voltage spikes caused by a motor are integrated on the chip. The stepper motor requires more current than available from the STM32F4_Discovery board, so an external power supply is

mandatory. The external power supply can be connected to the BaseBoard only to supply the L293D, the voltage should be set to conform to the requirements of the stepper motor, but remain in the safe range for the L293D.

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 to make 200 steps to make one complete circle; therefore one step rotates the 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. 6.4. This time the rotor is not a permanent magnet, but a solid iron. 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 groves with electromagnet B; the rotor rotates for ¼ of a groove. Next electromagnet C is turned on and electromagnet B off. The result is an additional rotation for ¼ 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.

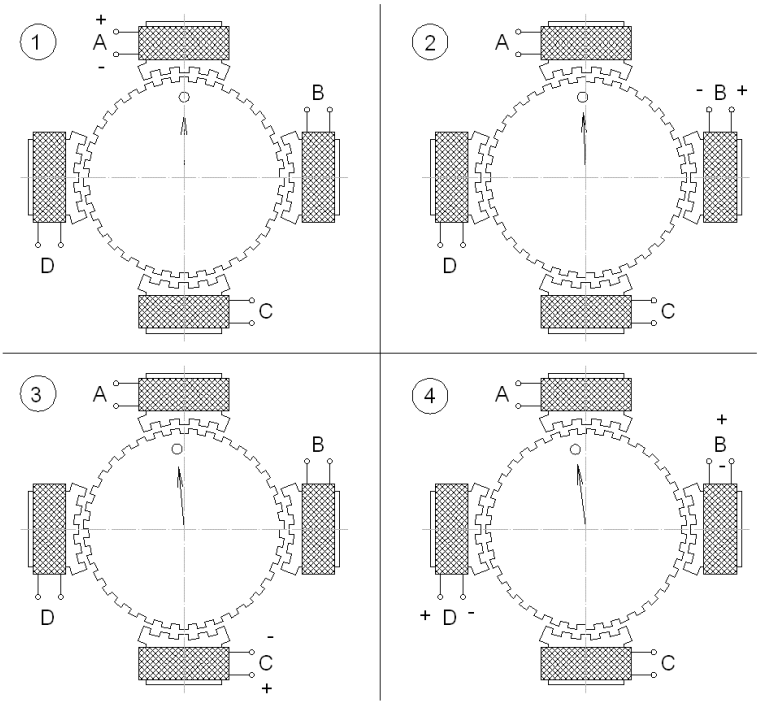

Figure 6.4: Stepper motor with more steps per revolution

6.2. Software to drive the stepper motor

In order to use a stepper motor it must first be connected following either of the figures 6.2 or 6.3. Since port C controls the input to the driver L293D, corresponding pins PC6 to PC9 and PC11 must be configured as push-pull outputs. Here pin PC11 serves as an enable signal for the driver L293D, and must be set to high. Other pins serve as control lines for the driver chip, their values follow the pattern stated above.

The function for the initialization of the port C to handle the stepper motor driver is given below.

```
void STEPPERinit (void)   {
GPIO_InitTypeDef        GPIO_InitStructure;

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);                                    // 4
```

```
  GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_6 | GPIO_Pin_7;                    // 5
  GPIO_InitStructure.GPIO_Pin  |= GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_11;     // 6
  GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_OUT;                             // 7
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;                             // 8
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;                           // 9
  GPIO_Init(GPIOC, &GPIO_InitStructure);                                     // 10
}
```

The function follows the standard form presented in chapter 4. Line four enables the clock for port C, and line 10 calls the function to configure the port C, selected pins, with the desired options. The options are written into the initializing structure GPIO_InitStructure in lines 5 to 9.

A demonstration program is prepared. The program configures necessary pins of the microprocessor, and then enters the infinite while loop where it periodically reads the state of pushbuttons. If pushbutton S370 is pressed then the motor is rotated clockwise. If pushbutton S371 is pressed then the motor is rotated counterclockwise. The loop is terminated by an empty loop to slow-down the execution. The current position of the motor is written to the LCD screen.

The pattern to drive the motor is initialized in an array at the global declare section of the program. The pattern in this array might need to be adjusted to the order of wires connected to the driver L293D.

The complete listing of the program is given below. The function for the initialization of the port C “STEPPERinit()” was given above, and the function for the initialization of pins needed for pushbuttons “SWITCHinit()” was given in chapter 4.

```
#include "stm32f4xx.h"
#include "stm32f4xx_rcc.c"
#include "stm32f4xx_gpio.c"
#include "LCD2x16.c"
#include "dd.h"

int steps[4] = {BIT_6, BIT_7, BIT_8, BIT_9};
//int steps[4] = {BIT_6 | BIT_7, BIT_7 | BIT_8, BIT_8 | BIT_9, BIT_9 | BIT_6};
int ptr_steps = 0;

void main (void)  {              //
char switches = 0;

  STEPPERinit();
  SWITCHinit();

  LCD_init();                           // init LCD
  LCD_string("Stepper motor", 0x01);    // display title string
  GPIOC->ODR |= GPIO_Pin_11;            // enable motor driver

  while (1) {                   //
    switches = GPIOE->IDR;
    if (switches & S370)  ptr_steps++;
    if (switches & S371)  ptr_steps--;
    GPIOC->ODR &= ~0x3c0;
    GPIOC->ODR |= steps[ptr_steps & 0x03];

    LCD_sInt16(ptr_steps,0x40,0x05);    // write to LCD, signed, 16 bits
    for (int i = 0; i<1000000; i++) {}; // waste some time

  };
}
```

Note that due to the pushbutton pressed a variable ‘ptr_steps’ is modified. The last two bits of this variable are used as a pointer to the array with predefined patterns to rotate the motor in the correct position. This trick simplifies the rotation and simultaneous following of the position. It also allows negative positions to be interpreted correctly.