## Speech script for Stepper Interfacing with Firebird V (ATmega2560) video tutorial

|  |  |
| --- | --- |
| Title Page | Hello everyone! Welcome to the video tutorial on Firebird V robotics research platform. This platform is based on the ATmega2560 microcontroller. In this tutorial we will learn about stepper motors, ways to control them and how to interface a stepper motor with the Firebird V robot. |
| Agenda for Discussion | Let's see the agenda for discussion in this tutorial.  First we will have an introduction where we will discuss what is a stepper motor and its types.  Then we will move on to learn how to control a stepper motor in different stepping sequences like wave, full and half stepping modes followed by a comparison of the three stepping modes.  Then we will have a short demonstration of how to identify the wires of a usually unlabeled stepper motor.  This will be followed by a discussion of the circuitry required to drive a stepper motor and then finally we will jump on to actually programming the robot to control the stepper motor. |
| Prerequisite knowledge | Before we jump to learning how to interface stepper motors, make sure you understand input/output interfacing using ports in AVR and know about the timer features in AVR and how to use them. |
| What is a stepper motor? | So the first question that arises is, What is a stepper motor? A stepper motor is a special kind of motor,  **(Next)**  whose rotation is divided into discrete steps which allow precise control over its angle.  **(Next)**  It can be commanded to hold a step or move to the next step. |
| Types of stepper motor | Types  Steppers are mainly classified on the basis of their internal wiring. These classes are as follows.  **(Next)**  Bipolar Steppers usually have 4 wires to control them and their drivers are complex  **(Next)**  Unipolar Steppers on the other hand have 5 or 6 wires and are relatively easier to drive  **(Next)**  Since, Unipolar motors are commonly used, we will be using them in this tutorial |
| Stepping sequences | Moving on to how to control a stepper motor, a stepper motor can be controlled by sending specific sequences of signals to its wires. The different stepping sequences are  **(Next)**  Wave Stepping  **(Next)**  Full Stepping  **(Next)**  and Half Stepping  **(Next)**  We will now see and understand how to use each one of them. |
| Wave Stepping | Wave Stepping  It involves exciting each coil turn by turn in a circular fashion.  **(Next)**  The table shows which coils are to be excited for generating the wave stepping sequence. Here a 1 denotes ON whereas a 0 denotes OFF. To rotate the stepper motor in one direction, we send the sequence, 1-2-3-4-1 and so on while for the opposite direction, we send the sequence in reverse, i.e., 4-3-2-1-4 and so on. |
| Wave Stepping (contd.) | Shown here is an illustration that depicts the direction of the rotor inside a stepper motor in the wave stepping sequence. The rotor points to the excited coil at that instant. When coil A is excited, the rotor aligns itself to point to it. Next, when coil B is excited, the rotor turns to align itself again. This process repeats for the other coils. |
| Full Stepping | Full stepping  It involves exciting two adjacent coils turn by turn in a circular fashion.  **(Next)**  The table shows which coils are to be excited for generating the full stepping sequence. Notice that at each step exactly two coils are excited.  Now let's see what happens inside the motor when two coils are excited at once. |
| Full Stepping (contd.) | As you can see, when two coils, say A and B are excited, the rotor settles between them. This results in a higher torque than that found in wave stepping. |
| Half Stepping | Half Stepping  This stepping mode is a combination of wave and full stepping.  **(Next)**  If you look closely, half stepping uses full stepping between the step positions of wave stepping. Let's see what happens inside the motor during this sequence. |
| Half Stepping (contd.) | As you can see, there are 8 positions corresponding to the 8 steps shown in the previous table. It is called half stepping because it effectively halves the step angle and offers more resolution. |
| Comparison of stepping modes | We will now see the differences and similarities between the 3 different stepping modes  **(Next)**  Talking about torque, since only one coil at a time is energised in wave stepping, its torque is the lowest. In full stepping, 2 coils are excited at once creating a larger attractive force resulting in a larger torque. Since half stepping excites 1 or 2 coils per step, the torque output is intermediate.    Vibration in a stepper motor increases with torque and decreases when the step angle decreases. Since wave stepping has low torque, but a step angle larger than half stepping, it has an intermediate level of vibration. Full stepping has higher torque resulting in more vibration and half stepping, due to a smaller step angle exhibits the least vibration.    Next is speed. Speed is equal to the step angle times the step frequency, since the step angle is the same for wave and full stepping, they have the same speed. For half stepping, the stepping angle is halved and thus the speed is also halved.    Resolution is fineness of control over rotational angle and is equal to the step angle. The step angle is equal for wave and full stepping while it is halved for half stepping. |
| Identifying the wires of a stepper motor | We will now learn how to identify the wires of a stepper motor.  **(Video to show how to identify wires of stepper motor – to be inserted using video editing tool)** |
| Stepper Motor Driver Circuit | This is the circuit diagram for the stepper motor driver  The ULN2003 is a transistor array package of 7 transistors that are capable of switching 500 milli Amperes of current per transistor. The pins 1-7 are the inputs to the transistors, while the pins 16-10 are the corresponding open-collector outputs. All transistors have a commmon emitter that is connected to the ground at pin 8. The COM pin at pin 9 is connected externally to the supply voltage. The circuit is basically used to drive the stepper's high current windings by a microcontroller's General Purpose IO pins. |
| Interfacing with ATmega2560 – GPIO pins | We will now discuss how to interface the Firebird V robot with the stepper motor using this circuit  **(Next)**  These are the Firebird V robot's expansion slot pins which we are going to use to connect to the driver circuit shown previously.  Expansion slot pin 17 i.e., PL7 is connected to the ULN2003 driver IC's pin 1  Expansion slot pin 18 i.e., PL6 is connected to the ULN2003 driver IC's pin 2  Expansion slot pin 19 i.e., PD1 is connected to the ULN2003 driver IC's pin 3  Expansion slot pin 20 i.e., PD0 is connected to the ULN2003 driver IC's pin 4  Expansion slot pin 23 i.e., Ground is connected to the ULN2003 driver IC's Ground on pin 8  **(Next)**  The pins are numbered on the slot as shown in the figure. Note the unusual snake-like numbering of the expansion slot. Pins 5-16 and 25-52 are not shown for clarity. We will be using the pins 17, 18, 19, 20 and 23. |
| Interfacing with ATmega2560 - Timer Configuration | Now for an example, we will be rotating the stepper motor in one direction at a fixed speed for a complete revolution and then change its direction and repeat the process in the opposite direction. For this, we can either use delays between steps or we can use one of the AVR timers for the delays. For a challenge, let's talk about achieving this with the second method.  **(Next)**  Moving the stepper a single step every 3.333 ms means running the motor at a step frequency of 300Hz  **(Next)**  To achieve this, we select the 16-bit Timer1 and run it in CTC mode i.e., Clear on Timer Compare Match by setting the Waveform Generation Mode bits to 0100.  **(Next)**  We will set the prescaler to 1 as we do not need the timer clock to be divided by any factor.  **(Next)**  Since the required frequency is 300Hz, we set the TOP value to 49151 calculated by this formula. The timer's OCR1A register value is used as the TOP value in CTC mode.  **(Next)**  Finally we enable the compare match interrupt to step the motor in the Interrupt Service Routine every time a compare match occurs. |
| Interfacing with ATmega2560 - Code | Now that we have understood how to use the timer for our purpose, let's have a look at the code.  **(Next)**  This is a list of headers to be included, which are: avr/io.h, avr/interrupt.h and a custom header called stepper.h that contains the functions for controlling the GPIO pins of the stepper in different step modes.  **(Next)**  The interrupt service routine is called whenever a compare match occurs and we will use this to make a single step. direction here is a global variable that is initially set to +1 and is changed to -1 after every 200 steps by the following code which increments stepcount in each call of the ISR and changes direction from +1 to -1 and vice versa if the stepcount exceeds 200. It then resets the stepcount to 0.  The stepper motor we use, completes a revolution in 200 steps. Modify this value if your stepper motor's steps per revolution value is different. |
| Interfacing with ATmega2560 – Code (contd.) | Moving on to the main function, first we initialize the IO ports for the stepper motor.  **(Next)**  Then, we setup Timer1 with the configuration we decided upon. To do this, we first temporarily disable all interrupts while setting up and set Timer1 to work in CTC mode by setting WGM bits to 0100 in the TCCR1B register. Only WGM12 is set as the other WGM bits are by default zero. Then the Output Compare Match Interrupt for Timer1 is enabled by setting bit OCIE1A in the TIMSK1 register. Finally we re-enable all the interrupts after initialization  **(Next)**  We then set the TOP value in the OCR1A register using the given formula. Here F\_CPU and SPEED are previously defined constant whose values are 14745600 and 300 respectively.  **(Next)**  Then we finally start the timer by setting the prescaler to 1 through the CS12, CS11 and CS10 bits in the TCCR1B register.  **(Next)**  Then we handover the job to the interrupts and thus let the processor go into an infinite loop. |
|  | **(Show code in Atmel Studio)**  **(Show working video of stepper)**  So by now we have successfully understood the control of stepper motors and how to interface a stepper motor with the ATmega2560 based Firebird V robotics platform. You may modify the code to experiment with different stepping modes and speeds or by setting it to specific angles, etc. |
| Thank you | With this we have reached the end of this video tutorial. Thank you for listening! For any doubts or suggestions feel free to mail them at helpdesk@e(hyphen)yantra(dot)org  This is Joel Pinto signing off! |