## Speech script for Servo 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 Servo motors, their working, how to control them and how to interface a Servo motor with the Firebird V robot. |
| Agenda for Discussion | Let's see the agenda for discussion in this tutorial.  The presentation starts with how a servo motor works, its internal circuitry, what makes it different from dc motor and how can we rotate it. We will learn selecting a servo for specific use and how to interface servo with Firebird-V.  Then finally we will jump on the programming for the same. |
| Prerequisite knowledge | Before we proceed with the tutorial, make sure you understand input/output interfacing using ports in AVR and know about the timer features in AVR and how to use them. |
| Introduction | So, let us see what a servo motor is. It is basically a dc motor that can rotate with precise angle, speed and acceleration.  Servos can put up about 42 oz/inch of torque.  They are easily found in toys like model car, airplanes, etc. controlling steering or adjusting wing surfaces.  They are relatively inexpensive. Hence widely used for educational purpose in mechatronics as they can be controlled via controller. |
| Principle of Working (Under section: Servo Motor) | A servo mainly contains a dc motor coupled with gears having proper gear ratio and a sensor for position feedback. Generally, this work is done by coupling it with a potentiometer. And a control circuitry to form a closed loop control system.  **(Next)**  Looking into the control system, we find that the output port is connected with one of the input terminals of the error detector amplifier and electrical signal is given to rotate servo is given at another terminal of error detector and amplifier.  **(Next)**  The difference between the signals to the error detector are amplified and applied to dc motor to drive it.  **(Next)**  This in turn rotates the potentiometer which is coupled to the shaft of dc motor using gears. As the angular position of the potentiometer knob progresses the output or feedback signal increases.  **(Next)**  When the potentiometer reaches the desired position, the error in the signal from output and the applied control signal is negligible. Hence, there would be no input signal to the dc motor to rotate it.  Continuously applying the same control signal makes the motor to stay at that position. This is how a simple conceptual servo motor works. |
| Operating Servo Motor | Now let us see how to operate a servo motor.  Servo motor is operated using the wires provided in it.  It is driven to a particular angle by control signal sent to it via a controller. These signals are basically Pulse Width Modulated (PWM) waves and that angular position is determined by their ‘on-time’.  **(Next)**  This on-time period for various angles depends on the servo model and the manufacturer while not on the frequency of PWM.  **(Next)**  It is interesting to note that this angular position is independent of the duty cycle of the PWM signal.  **(Next)**  Generally this time period is around 1ms for 0o and around 2ms for 180o but depends on model and the manufacturer as said above. Graph of on-time period vs. respective angle is linear. Therefore, the values of time period for angles other than these can be easily calculated by getting the eq. of line from the 2 values given. |
| Selection of Servo motor | The typical specifications of servo motors are torque, speed, weight and its dimensions.  A manufacturer may compromise torque over speed or speed over torque in different models. The weight and dimensions are directly proportional to the torque. The selection of a servo should be made as per the requirement of these specifications. |
| Controlling servo using Firebird V  (Under section: AVR ATmega2560) | Now let’s move on to the interfacing part.  This figure shows the servo connector. The ground wire marked as no. 1 in the fig. is generally brown or black in colour. The no. 2 pin, i.e. power pin is generally red in colour. While, the signal pin is typically yellow or orange in colour which is marked as no.3  **(Next)**  This figure shows were to put these pins in the firebird V robot. |
| Using Timer1 for PWM generation | A prescaler is specifically used to slow down the counting process by dividing the crystal frequency by a power of 2.  The prescaler is very helpful when the crystal frequency changes by a power of 2, i.e. changing the clock speed from 1Mhz to 8Mhz can easily be compensated by changing the prescaler from 'x' to '8x' without changing other code.  Let’s choose 256 as prescaler  Here we would use Timer1 of ATmega2650 for generating PWM signals. Being 16-bit wide, it provides high resolution. Let’s choose the most commonly used mode i.e. Fast PWM in mode 14. The table shows "WGM bit description" from the software manual, table no. 5.3 of firebird V. We would use the internal crystal of the controller for delay purpose in timer.  **(Next)**  The main logic behind the PWM wave generation is a counter is started to count from 0 to specified assigned value i.e. TOP value. And the o/p where servo is connected is set high simultaneously. When the value in the counter becomes same as that stored in OCR1 register, the o/p at the same port is cleared. Hence, the value in OCR1 register can be used to generate waveform of desired on time period which can be used to set angular positions of the servo motor.  **(Next)**  The TOP value can be calculated from the desired PWM frequency as.  **(Next)**  Here, the TOP value for 50 Hz PWM frequency is calculated.  **(Next)**  The value of the OCR register can be calculated from the on time period for particular angle into clock frequency by prescaler. Dividing the prescaler by clock frequency, we get the time required by the counter to increment by one count. Hence, multiplying it by total time period required for certain angle we can get the required count value to be provided to OCR1 for rotating servo to that angle.  **(Next)**  Values of register like TCCR1 and TCNT1 can be found using ATmega2560 datasheet and the value of prescaler. |
| Code | Now, let’s jump to code.  This block shows the header file required to be included in the program.  **(Next)**  This block shows the port initialization. The 1st command will make PORTB 5 pin as output port and set it high using 2nd command.  **(Next)**  Now let’s see how to get counter value for the angles of which t-on time period is not given i.e. for angles other than 0o and 180o.  As we know the relation between the on time period and the corresponding degree is linear, and also on other hand, we find that for a given fixed prescaler, the relation between the counter value and the on time period is linear. Hence, knowing the count value for 0o and 180o from the eq. of OCR1 discussed earlier, we can get the plot of count vs. degree and hence getting the general eq. of the curve which would be a line, we get counter value for any degree required as.  **(Show code in Atmel Studio)**  Now, we will write a code in Atmel Studio and burn it on robot.  I have already configured Atmel Studio for ATmega2560 and written a code to drive a servo. Looking at the code, we find clock frequency and header files defined at the top of the code and are essential for any program. Then comes the port initialization. Port at which servo is connected is initialized as o/p port and set high. Than timer is initialized. Here, the values of TCCR1A and TCCR1B are obtained from datasheet of ATmega2560 keeping in mind fast PWM mode 14 and prescaler which is equal to 256 in our case. Hence, we get the values of TCCR1A and TCCR1B as shown. These values of TCCR1A/B can be obtained from WGM bit description, COM bit description and CS bit description tables of software manual.  For rotating servo to an angle, it takes on-time period for that angle. So we need to convert angles in terms of on-time period which than is converted to certain count value based on clock frequency and prescaler.  This block converts angle in terms of count. This formula we already derived in previous section. This is basically eq. of line obtained by plotting count vs. respective degrees. Here, we note that slope of the line obtained is 0.512 and the y-intercept is 34.56 which is count value for 0o rotation. This value is stored in OCR1A register. Similar task is done for servo2 and servo3.  Now let’s move on to main program. So, we start from initializing peripherals and devices which initializes ports and timer. Than command is given to rotate the servo to 0o. This 0o is passed as argument to the function servo\_2 which converts degrees in terms of count and saves it in OCR register. Hence, in this program, we rotate servo from 0o-90o-180o-0o infinite times by keeping tis in infinite loop.  Now we shall build this code and burn it on robot. We can see the process succeeded wit 0 error. Now we would see the o/p by burning tis code on the robot.  **(Show working video of servo)** |
| Thank you | Going through the tutorial and knowing how we can drive servo motor and the how to program a controller for that, you can try different things like velocity control, 1 degree precision, etc. You can also challenge the operating frequency range of PWM.  With this we end this video tutorial here. Thank you for listening! For any doubts or suggestions feel free to mail them at helpdesk@e(hyphen)yantra(dot)org  This is Vishal H. Rajai signing off! |