Position Control for Two Wheel Mobile Robot

Naveen Kumar[1] Joe Xavier Deepak.J[2] Vignesh.N[3] Praveen Kumar K[4] A.Elango[5]
[1,2,3,4]P.G. Scholar [5]Professor& Head
[1,2,3,4,5]Department of Mechanical Engineering
[1,2,3,4,5]Alagappa Chettiar College of Engineering & Technology Karaikudi, Tamil Nadu, 630004, India

Abstract— In two-wheel mobile robot various interrupt handling mechanisms such as timer overflow interrupts, timer compare interrupts, serial interrupts for doing specific tasks. In this research, we will have interrupt concept and will implement external hardware interrupts for position estimation of robots using position encoders. Interrupts that flow of the program and cause it to branch to ISR (Interrupt Service Routine). ISR does the task that needs to be done when interrupt occurs. Whenever position encoder moves by one tick it interrupts the microcontroller and ISR does the job of tracking position count. Each interrupt has a vector address assigned to it low in program memory. When the interrupt occurs, the program completes executing its current instruction and branches to the vector location associated with that interrupt. When an interrupt occurs, the return address is stored on the system stack. The RETI Assembly language instruction causes the return address the stack and continue program execution from the point where it was interrupted where position encoders which give position and velocity to the robot in the form of feedback. In this robot control position and velocity in a closed loop .In the position encoder which consists of optical encoder and slotted disc assembly. We get square wave signal where pulse count position and time period indicates velocity when this slotted disc moves in between the optical encoders.

Key words:

I. INTRODUCTION

In this research interrupts needs to be initialized before they become active. Initializing interrupt is a three step process. The first step is to select the trigger type for the interrupt. We are using falling edge trigger. This is selected by setting bits in EICRA (INT3 to INT0) and EICRB (INT7 to INT4) registers. Second step is to unmask the interrupt that we want to use in the EIMSK register. In the third step we globally enable all the unmasked interrupts. To enable unmasked interrupts we need to set global interrupt enable bit in the status register (SREG). This is done by instruction "sei();".

Interrupt switch is connected to the PORTE 7 pin of the microcontroller. It has 10Kohm external pull-up resistor. When switch is pressed PORTE 7 pin is connected with the ground. Interrupt switch is used as general purpose input device. Interrupt switch is configured as input with its internal pull-up resistor enabled. Each pin of the port can be addressed individually. Each pin individually can be configured as input or as output. While pin is input it can be kept floating or even pulled up by using internal pull-up. While pin is in the output mode it can be logic 0 or logic 1. To configure these ports as input or output each of the port has three associated I/O registers. These are Data Direction Register (DDRx), Port Drive Register (PORTx) and Port pins register (PINx) where 'x' is A to L (except I) indicating particular port name.

A. Data Direction Register (DDRx)

Data Direction Register (DDRx) This register is used for determine which port of the bit is used as input and which port of bit is used as output If the logic one is written in the DDRx, then corresponding port pin is configured as an output pin. If the logic zero is written in the DDRx, then corresponding port pin is configured as an input pin.
DDRA = 0xxF0; //sets the 8MSB bits of PORTB as output port and //8LSB bits as input port

B. Port Drive Register (PORTx)

If the port is worked as output port, then the PORTx register drives the value on output pins of the port.
DDRA = 0xFF0; //set all 8 bits of PORTB as input PORTB = 0xFF0;
//pull-up registers are connected on 4 MSB pins and 4 LSB pins which are floating.

C. Port pins register (PINx)

Reading from the input bits of port is done by reading port pin register x = PINB; //read all 8 pins of port B

DDRx	PORTx	I/O	Pull-up	Comments
0	0	Input Pin	No	floating input
0	1	Input Pin	Yes	Will source current if externally pulled low
1	0	Output Pin	No	Output Low (Sink)
1	1	Output Pin	No	Output High (source)

Table .1 Register pins

- X' represents port name – A, B, C, D, E, F, G, H, J, K, and L.
- Tri-State is the floating pin condition.

D. Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. It toggle one bit of port using signal bit instruction. Where 'x' is the port name and 'n' is the bit number.

II. ROBOT DIRECTION CONTROL & MOVEMENTS

The External Interrupts are activated by using the external pins INT2:0 for the SREG I-flag and interrupt mask in the EIMSK is set. Edges on INT2:0 is registered asynchronously. Pulses on INT2:0 pins wider than 30 nanoseconds which generate an interrupt. When Pulses are shorter which are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level will be there until the completion of the currently executing instruction to

generate an interrupt. While enabling a level triggered interrupt through which we generate an interrupt request as long as the pin is held low.

ISCn1	ISCn2	Description
0	0	It generates interrupt request.
0	1	INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

Table -2Robot interrupt

While enabling a level triggered interrupt through which we generate an interrupt request as long as the pin is held low. When we change the ISCn bit, an interrupt can occur. Robot's motors are controlled by L293D motor controller from ST Microelectronics. Using L293D, microcontroller can control direction and velocity of both of the motors. FOR change the direction logic levels (High/Low) for IC L293D's direction pins. Controlling the velocity using pulse width modulation (PWM) for Enable pins of L293D IC.

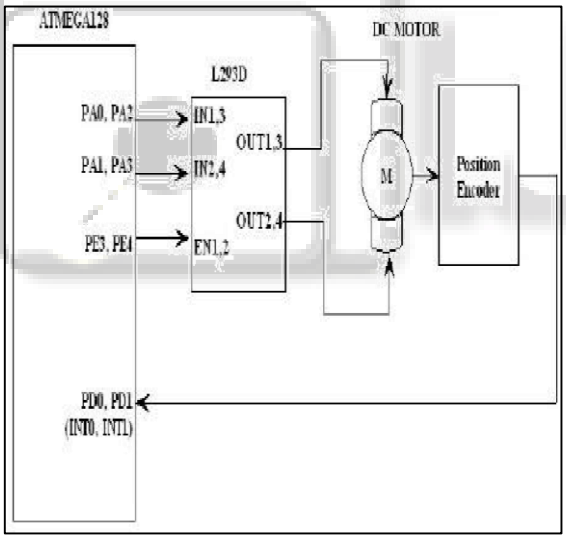

Fig. 1: Robot direction control

DIRECTION	LEFT BWD (LB) PA0 (L1)	LEFT FWD(LF) PA1 (L2)	RIGHT FWD(RF) PA2 (R1)	RIGHT BWD(RB) PA3 (R2)	PWM PL3 (PWML) for left motor PL4 (PWMR) for right motor
FORWARD	0	1	1	0	As per velocity requirement
REVERSE	1	0	0	1	As per velocity requirement
RIGHT *(Left wheel forward, Right wheel backward)*	0	1	0	1	As per velocity requirement
LEFT*(Left wheel backward, Right wheel forward,)*	1	0	1	0	As per velocity requirement
SOFT RIGHT*(Left wheel forward,, Right wheel stop)*	0	1	0	0	As per velocity requirement
SOFT LEFT*(Left wheel stop, Right wheel forward,)*	0	0	1	0	As per velocity requirement
SOFT RIGHT 2 *(Left wheel stop, Right wheel backward)*	0	0	0	1	As per velocity requirement

Table 2 Logic table for motor direction control

– All the soft turns should be used when you need more accuracy during turning
– Soft left 2 and Soft right 2 motions are very useful in grid navigation.

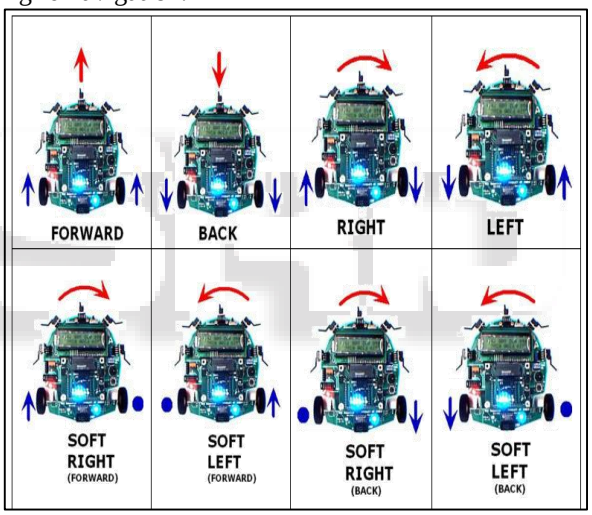

Fig. 2: Robot Movements

III. CALCULATIONS

Calculation for position encoder resolution

– Phase 1:

Robot can move forward or backward (position encoder resolution is in mm)
Wheel diameter: 10cm
Wheel circumference: 10cm * 3.14 = 31.4cm = 314mm
Number slots on the encoder disc: 60
Position encoder resolution: 314 mm / 60 = 5.23mm / pulse.

– Phase 2

We turned robot with one wheel rotates clockwise while other wheel is rotates anti clockwise and rotation Center is in the centre of line passing from the wheel axel then both wheels will rotates in opposite direction (Position encoder resolution is in degrees)
Distance between Wheels = 20cm
Radius of Circle formed in 3600 rotation of Robot = Distance between Wheels / 2
20/2 =10 cm

Distance Covered by Robot in 3600 Rotation = Circumference of Circle traced
= 2 x 10 x 3.14
= 62.8 cm or 628mm
Number of wheel rotations of in 3600 rotation of robot
= Circumference of Traced Circle / Circumference of Wheel
= 628 / 314
= 2
Total pulses in 3600 Rotation of Robot
= Number of slots on the encoder disc / Number of wheel rotations of in 3600 rotation of robot = 60 x 2
=130
Position Encoder Resolution in Degrees = 360 /130
= 2.769 degrees per count

– Phase 3

Robot is turning with one wheel fixed with other wheel is rotating clockwise or anti clockwise. Centre of rotation is centre of the fixed wheel (Position encoder resolution is in degrees)
In this phase only one wheel is rotating and other wheel is stationary so robot will complete its 3600 rotation with stationary wheel as its centre.
Radius of Circle formed in 3600 rotation of Robot = Distance between Wheels = 20 cm
Distance Covered by Robot in 3600 Rotation = Circumference of Circle traced
= 2 x 2 x 3.14
= 125.6cm or 1256mm
Number of wheel rotations of in 3600 rotation of robot
= Circumference of Traced Circle / Circumference of Wheel
= 1256/314
= 4
Total pulses in 3600 Rotation of Robot = Number of slots on the encoder disc / Number of wheel rotations of in 3600 rotation of robot = 60 x4
= 240
Position Encoder Resolution in Degrees = 360 /240
= 1.5 degrees per count

IV. PROGRAM FOR CONFIGURING POSITION ENCODER

```
left_encoder_pin_config()
//Function to configure INT4 (PORT 4) pin as input for the left position encoder
Void ()
left_encoder_pin_config (void)
{
 DDRE = DDRE & 0xxEF; //Set the direction for the PORTE 3 pin as input PORT = PORT | 0xx20; //Enable internal pull-up in PORT 4 pin
}
 right_encoder_pin_config ( )
//Function to configure INT5 (PORT 5) pin take input for the right position encoder
void right_encoder_pin_config
 Void ()
right_encoder_pin_config (void)
{
DDRE = DDRE & 0xxDF;
//Set the direction for the PORT 5 pin as input PORT = PORT | 0xx30; //Enable internal pull up for PORT 4 pin.
}
```

V. CONCLUSION

In this project robot movement are controlled by System Programming (ISP), external programmer or using boot loader benefits with the boot loader is that we don't required any external hardware to load .hex files on the micro-controller and it protects robot hardware from possible damage from static electricity and it prevents accidental changes in the fuse settings of the micro-controller. If you load .hex file on the robot using any ISP programming then this boot loader will get erased and it needs to be reloaded again using ISP programmed. If Boot loader is loaded on the micro-controller, it allows system programming used through serial port with no need for the external ISP programmer. Code responsible for In System Programming via serial port resides in the configurable boot memory section of the micro-controller. When signal using outside switch for resetting the micro-controller it gets active and waits for communication from the PC for copying .hex file on the micro-controller's flash memory. The position encoder which consists of optical encoder and slotted disc assembly. We get pulse count position and time period indicates velocity when this slotted disc moves in between the optical encoders.

ACKNOWLEDGMENT

I express my very profound gratitude to my parents and to my partner Swati Agarwal for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching. This accomplishment would not been possible without them. Thank you.

REFERENCES

[1] Roy J, Whitcomb LL. Adaptive Force Control of Position/Velocity Controlled Robots: Theory and Experiment. IEEE Transactions on Robotics and Automation 2008;18:121–37.
[2] Goldsmith PB, Francis BA, Goldenberg AA: Stability of hybrid position/force control applied to manipulators with flexible joints. International Journal of Robotics and Automation 2011;14:146–60.
[3] Huang CQ, Shi SJ, Wang XG, Chung WK. Parallel Force/Position Controls for Robot Manipulators with Uncertain Kinematics. International Journal of Robotics and Automation 2008;20:158–68.
[4] Siciliano B, Villani L. Parallel force and position control of flexible manipulators. IEE Proceedings – Control Theory and Application 2009;147:605–12.
[5] Ziliani G, Visioli A, Legnani G. Gain Scheduling for Hybrid Force/Velocity Control in Contour Tracking Task. International Journal of Advanced Robotic Systems 2013;3:367–74.
[6] Cho HT, Jeon P, Jung S. Implementation and control of a roadway crack tracking mobile robot with force regulation. Proceedings of the IEEE International Conference on Robotics and Automation 2012;3:2444–9.
[7] Pholsiri C, Rabindran D, Pryor M, Kapoor C. Extended generalized impedance control for redundant manipulators. Proceedings of the 42nd IEEE Conference on Decision and Control 2014;4:3331–6.

