/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Written by: Vinod Desai, NEX Robotics Pvt. Ltd.

Edited by: Sachitanand Malewar, NEX Robotics Pvt. Ltd.

AVR Studio Version 4.17, Build 666

Date: 26th December 2010

This experiment demonstrates the application of a simple line follower robot. The

robot follows a white line over a black background, if any obstacle comes in front of

the robot, robot stops and buzzer beeps.

Application example: Adaptive Cruise Control (ACC)

Concepts covered: ADC, LCD interfacing, motion control based on sensor data

LCD Connections:

LCD Microcontroller Pins

RS --> PC0

RW --> PC1

EN --> PC2

DB7 --> PC7

DB6 --> PC6

DB5 --> PC5

DB4 --> PC4

ADC Connection:

ACD CH. PORT Sensor

0 PF0 Battery Voltage

1 PF1 White line sensor 3

2 PF2 White line sensor 2

3 PF3 White line sensor 1

4 PF4 IR Proximity analog sensor 1\*\*\*\*\*

5 PF5 IR Proximity analog sensor 2\*\*\*\*\*

6 PF6 IR Proximity analog sensor 3\*\*\*\*\*

7 PF7 IR Proximity analog sensor 4\*\*\*\*\*

8 PK0 IR Proximity analog sensor 5

9 PK1 Sharp IR range sensor 1

10 PK2 Sharp IR range sensor 2

11 PK3 Sharp IR range sensor 3

12 PK4 Sharp IR range sensor 4

13 PK5 Sharp IR range sensor 5

14 PK6 Servo Pod 1

15 PK7 Servo Pod 2

\*\*\*\*\* For using Analog IR proximity (1, 2, 3 and 4) sensors short the jumper J2.

To use JTAG via expansion slot of the microcontroller socket remove these jumpers.

Motion control Connection:

L-1---->PA0; L-2---->PA1;

R-1---->PA2; R-2---->PA3;

PL3 (OC5A) ----> PWM left; PL4 (OC5B) ----> PWM right;

LCD Display interpretation:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*LEFT WL SENSOR CENTER WL SENSOR RIGHT WL SENSOR BLANK \*

\*BLANK CENTER SHAPRP SENSOR CENTER IR SENSOR BLANK BLANK \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Note:

1. Make sure that in the configuration options following settings are

done for proper operation of the code

Microcontroller: atmega2560

Frequency: 147456

Optimization: -O0 (For more information read section: Selecting proper optimization

options below figure 2.22 in the Software Manual)

2. Make sure that you copy the lcd.c file in your folder

3. Distance calculation is for Sharp GP2D12 (10cm-80cm) IR Range sensor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Copyright (c) 2010, NEX Robotics Pvt. Ltd. -\*- c -\*-

All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions are met:

\* Redistributions of source code must retain the above copyright

notice, this list of conditions and the following disclaimer.

\* Redistributions in binary form must reproduce the above copyright

notice, this list of conditions and the following disclaimer in

the documentation and/or other materials provided with the

distribution.

\* Neither the name of the copyright holders nor the names of

contributors may be used to endorse or promote products derived

from this software without specific prior written permission.

\* Source code can be used for academic purpose.

For commercial use permission form the author needs to be taken.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE

LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.

Software released under Creative Commence cc by-nc-sa licence.

For legal information refer to:

<http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <math.h> //included to support power function

#include "lcd.c"

void port\_init();

void timer5\_init();

void velocity(unsigned char, unsigned char);

void motors\_delay();

unsigned char ADC\_Conversion(unsigned char);

unsigned char ADC\_Value;

unsigned char flag1 = 0;

unsigned char flag2 = 0;

unsigned char Left\_white\_line = 0;

unsigned char Center\_white\_line = 0;

unsigned char Right\_white\_line = 0;

unsigned char Front\_Sharp\_Sensor=0;

unsigned char Front\_IR\_Sensor=0;

//Function to configure LCD port

void lcd\_port\_config (void)

{

DDRC = DDRC | 0xF7; //all the LCD pin's direction set as output

PORTC = PORTC & 0x80; // all the LCD pins are set to logic 0 except PORTC 7

}

//ADC pin configuration

void adc\_pin\_config (void)

{

DDRF = 0x00;

PORTF = 0x00;

DDRK = 0x00;

PORTK = 0x00;

}

//Function to configure ports to enable robot's motion

void motion\_pin\_config (void)

{

DDRA = DDRA | 0x0F;

PORTA = PORTA & 0xF0;

DDRL = DDRL | 0x18; //Setting PL3 and PL4 pins as output for PWM generation

PORTL = PORTL | 0x18; //PL3 and PL4 pins are for velocity control using PWM.

}

//Function to initialize Buzzer

void buzzer\_pin\_config (void)

{

DDRC = DDRC | 0x08; //Setting PORTC 3 as outpt

PORTC = PORTC & 0xF7; //Setting PORTC 3 logic low to turnoff buzzer

}

//Function to Initialize PORTS

void port\_init()

{

lcd\_port\_config();

adc\_pin\_config();

motion\_pin\_config();

buzzer\_pin\_config();

}

// Timer 5 initialized in PWM mode for velocity control

// Prescale:256

// PWM 8bit fast, TOP=0x00FF

// Timer Frequency:225.000Hz

void timer5\_init()

{

TCCR5B = 0x00; //Stop

TCNT5H = 0xFF; //Counter higher 8-bit value to which OCR5xH value is compared with

TCNT5L = 0x01; //Counter lower 8-bit value to which OCR5xH value is compared with

OCR5AH = 0x00; //Output compare register high value for Left Motor

OCR5AL = 0xFF; //Output compare register low value for Left Motor

OCR5BH = 0x00; //Output compare register high value for Right Motor

OCR5BL = 0xFF; //Output compare register low value for Right Motor

OCR5CH = 0x00; //Output compare register high value for Motor C1

OCR5CL = 0xFF; //Output compare register low value for Motor C1

TCCR5A = 0xA9; /\*{COM5A1=1, COM5A0=0; COM5B1=1, COM5B0=0; COM5C1=1 COM5C0=0}

For Overriding normal port functionality to OCRnA outputs.

{WGM51=0, WGM50=1} Along With WGM52 in TCCR5B for Selecting FAST PWM 8-bit Mode\*/

TCCR5B = 0x0B; //WGM12=1; CS12=0, CS11=1, CS10=1 (Prescaler=64)

}

void buzzer\_on (void)

{

unsigned char port\_restore = 0;

port\_restore = PINC;

port\_restore = port\_restore | 0x08;

PORTC = port\_restore;

}

void buzzer\_off (void)

{

unsigned char port\_restore = 0;

port\_restore = PINC;

port\_restore = port\_restore & 0xF7;

PORTC = port\_restore;

}

void adc\_init()

{

ADCSRA = 0x00;

ADCSRB = 0x00; //MUX5 = 0

ADMUX = 0x20; //Vref=5V external --- ADLAR=1 --- MUX4:0 = 0000

ACSR = 0x80;

ADCSRA = 0x86; //ADEN=1 --- ADIE=1 --- ADPS2:0 = 1 1 0

}

//Function For ADC Conversion

unsigned char ADC\_Conversion(unsigned char Ch)

{

unsigned char a;

if(Ch>7)

{

ADCSRB = 0x08;

}

Ch = Ch & 0x07;

ADMUX= 0x20| Ch;

ADCSRA = ADCSRA | 0x40; //Set start conversion bit

while((ADCSRA&0x10)==0); //Wait for conversion to complete

a=ADCH;

ADCSRA = ADCSRA|0x10; //clear ADIF (ADC Interrupt Flag) by writing 1 to it

ADCSRB = 0x00;

return a;

}

//Function To Print Sesor Values At Desired Row And Coloumn Location on LCD

void print\_sensor(char row, char coloumn,unsigned char channel)

{

ADC\_Value = ADC\_Conversion(channel);

lcd\_print(row, coloumn, ADC\_Value, 3);

}

//Function for velocity control

void velocity (unsigned char left\_motor, unsigned char right\_motor)

{

OCR5AL = (unsigned char)left\_motor;

OCR5BL = (unsigned char)right\_motor;

}

//Function used for setting motor's direction

void motion\_set (unsigned char Direction)

{

unsigned char PortARestore = 0;

Direction &= 0x0F; // removing upper nibbel for the protection

PortARestore = PORTA; // reading the PORTA original status

PortARestore &= 0xF0; // making lower direction nibbel to 0

PortARestore |= Direction; // adding lower nibbel for forward command and restoring the PORTA status

PORTA = PortARestore; // executing the command

}

void forward (void)

{

motion\_set (0x06);

}

void stop (void)

{

motion\_set (0x00);

}

void init\_devices (void)

{

cli(); //Clears the global interrupts

port\_init();

adc\_init();

timer5\_init();

sei(); //Enables the global interrupts

}

//Main Function

int main()

{

init\_devices();

lcd\_set\_4bit();

lcd\_init();

while(1)

{

Left\_white\_line = ADC\_Conversion(3); //Getting data of Left WL Sensor

Center\_white\_line = ADC\_Conversion(2); //Getting data of Center WL Sensor

Right\_white\_line = ADC\_Conversion(1); //Getting data of Right WL Sensor

Front\_Sharp\_Sensor = ADC\_Conversion(11);

Front\_IR\_Sensor = ADC\_Conversion(6);

flag1=0;

flag2=0;

print\_sensor(1,1,3); //Prints value of White Line Sensor1

print\_sensor(1,5,2); //Prints Value of White Line Sensor2

print\_sensor(1,9,1); //Prints Value of White Line Sensor3

print\_sensor(2,4,11); //Prints Value of Front Sharp Sensor

print\_sensor(2,8,6); //Prints Value of Front IR Sensor

if(Front\_Sharp\_Sensor>0x45 || Front\_IR\_Sensor<0xF0)

{

flag2=1;

stop();

buzzer\_on();

}

if(Front\_Sharp\_Sensor<0x45 || Front\_IR\_Sensor>0xF0)

{

flag2=1;

forward();

buzzer\_off();

}

}

}