```
// Define Macros
#define F
_
CPU 8000000UL // CPU frequency to be considered by the delay macros
// Include Libraries
#include <avr/io.h> //AVR device-specific IO definitions
#include <util/delay.h> //delay library
#include <stdlib.h>
#include <stdint.h>
#include <avr/interrupt.h>
//Define subroutine prototypes
void Enable
_
Reverse(void);
void Enable
_
Forward(void);
int ADC
_
Read(char channel);
// Define variables
static int value
_
L; // global variable for left sensor result
static int value
_
R; // global variable for right sensor result
static int OCR1A
_
control; // control variable for servo PWM duty cycle
static int OCR0
OCR2
_
_
control; // speed motor PWM duty cycle
int Flag_
FW
movement=1;
_
float diff,sum;
float kp;
int main(void)
{
OSCCAL = 0xAD; // fine-tuned to achieve 50Hz servo PWM frequency from measurement
//Port PIN input/output assignment
DDRB = (1<<PB4)|(1<<PB3)|(1<<PB2)|(1<<PB1)|(1<<PB0); // PB0 to PB4 outputs,
PB3(OC0)
DDRD = (1<<PD5) | (1<<PD7); //PD5(OC1A) servo PWM output, PD7(OC2) motor reverse
PWM output
DDRC = 0xFF; //define PC1 and PC2 as extra output PINs (optional)
DDRA = 0x0; // Make ADC port as input for Sensors
// Timer counter settings
```

```
TCNT0 = 0; /* initialize timer0 count zero */
TCNT2 = 0; /* initialize timer2 count zero */
TCNT1 = 0; /* initialize timer1 count zero */
// Timer 1 settings
TCNT1 = 0; // initialize timer1 count zero
ICR1 = 19999; // Set TOP=ICR1 for 50 Hz PWM
TCCR1B =(1<<WGM13)|(1<<WGM12)|(0<<CS12)|(1<<CS11)|(0<<CS10);
TCCR1A =(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
// timer 0 settings
TCCR0
=(1<<WGM01)|(1<<WGM00)|(1<<COM01)|(0<<COM00)|(0<<CS02)|(1<<CS01)|(0<<CS00);
TCCR2 =
(1<<WGM21)|(1<<WGM20)|(1<<COM21)|(0<<COM20)|(0<<CS22)|(1<<CS21)|(0<<CS20);
56OCR1A
_
control = 1400; // servo motor at center position 7% duty cycle
OCR0
OCR2
_
_
control = 0 ; // speed motor speed = 0
// initialization of ADC (bitwise assignment)
OCR1A = OCR1A
control;
_
ADCSRA =(1<<ADEN)|(0<<ADPS2)|(0<<ADPS1)|(0<<ADPS0); // Enable ADC, fADC=f
CPU/2
_
ADMUX = (0<<REFS1)|(1<<REFS0)|(0<<MUX1)|(0<<MUX0); // Vref=Avcc, ADC channel: 0
sei(); // global interrupt enable
TIMSK=0; // disable all timer interrupts
Enable
_
Forward(); // change to forward direction
TIMSK = (1<<TOIE0); // enable forward movement interrupts
Flag_
FW
movement = 1;
_
_
delay_
ms(20);
while (1)
{
if (Flag_
FW
_
movement ==1)
{
if ((value
_
L <388) && (value
```

```
_
R < 388)) // 1.9
{
TIMSK=0; // disable all timer interrupts
Enable
_
Reverse(); // change to reverse direction
TIMSK = (1<<TOIE2); // enable Time 2 interrupt
Flag_
FW
movement = 0;
_
}
}
else
{
if (((value
_
L > 535 ll value
_
R > 535))) //when on track more than 1.9v - less than 2.5v
then move forward
{
TIMSK=0; // disable all timer interrupts
Enable
_
Forward(); // change to forward direction
TIMSK = (1<<TOIE0); // enable forward movement interrupts
Flag_
FW
movement = 1;
_
}
}
OCR1A = OCR1A
_
control; // update the servo control command
if (Flag_
FW
_
movement == 1)
{
}
else
OCR0 = OCR0
OCR2
_
_
control; // update speed duty cycle command
57{
OCR2 = 70; // update speed duty cycle command
}
```

```
}
}
{
ISR(TIMER0
OVF
_
_
vect)
kp = 1.35; //1.3
value
value
L = ADC
_
_
Read(0); // Read ADC0
R = ADC
_
_
Read(1); // Read ADC1
diff = value
L - value
R;
_
_
OCR1A
_
control = 1580 - (kp*diff);
if (OCR1A
_
control > 1960)//1960
{
}
{
}
{
}
{
}
else
{
}
OCR1A
control = 1960;
_
else if (OCR1A
_
control < 1180)
OCR1A
control = 1180;
_
if (((diff >= 0) && (diff < 110 )) ll ((diff <= 0) && (diff > -110 )) ) // diff smaller than 0.5V
OCR0
```

```
OCR2
control= 67;//65
_
_
else if (((diff > 0) && (diff < 221 )) ll ((diff <= 0) && (diff > -221 )) ) // diff smaller than 1 V
OCR0
OCR2
control=63;//60
_
_
OCR0
OCR2
control=63;//50
_
_
}
ISR(TIMER2
OVF
_
_
vect) //reverse
{
value
_
value
_
OCR1A
L = ADC
_
Read(0); // Read ADC0
R = ADC
_
Read(1); // Read ADC1
_
control =1580; //(7.4% straight)
58OCR0
OCR2
_
_
control = 55; //original 55
void Enable
}
{
_
Reverse(void)
OCR0 = 0; // forward duty cycle= 0 (narrow PWM pulse exists in PB3)
TCCR0 &= ~(1<<COM01); // OC0 disconnected from PB3 to eliminate short circuit risk
PORTB &= ~(1<<PB3); // PB3(OC0) PIN set to “0”
PORTC &= ~(1<<PC0); //PC0 (SCL) =0 Disable S1 Motor Forward Digital Output // setup
other digital I/O
outputs for reverse movement
_
```

```
delay_
us(4); // add a short delay (adjustable)
PORTC |= (1<<PC1); //PC1 =1 Enable S3 Motor Reverse Digital Output
TCCR2 |= (1<<COM21); // OC2 connected to PD7
OCR2 = 0; // initialize reverse duty cycle to zero
void Enable
}
{
_
Forward(void)
OCR2 = 0; // reverse duty cycle = 0 (narrow PWM pulse exists in PD7)
TCCR2 &= ~(1<<COM21); // OC2 disconnected from PD7 to eliminate short circuit risk
PORTD &= ~(1<<PD7); // PD7(OC2) PIN set to “0”
PORTC &= ~(1<<PC1); //PC1 (SCL) =0 Disable S3 Motor Forward Digital Output // setup
other digital I/O
outputs for reverse movement

_
delay_
us(4); // add a short delay (adjustable)
PORTC |= (1<<PC0); //PC0 =1 Enable S1 Motor Reverse Digital Output
//PORTB |= (1<<PB3); //PD7 =1 Enable S4 Motor Reverse PWM Output
TCCR0 |= (1<<COM01); // OC2 connected to PB3
OCR0 = 0; // initialize forward duty cycle to zero
}
{
int ADC

_
Read(char channel)
int ADC
value; // local variable for ADC result

_
ADMUX = (0x40) | (channel & 0b0001111); // set input channel to read
ADCSRA |= (1<<ADSC); // start conversion
while((ADCSRA &(1<<ADIF))== 0); // wait ADC completion interrupt ADIF=1
ADCSRA |= (1<<ADIF); // clear interrupt flag (ADIF=0)
ADC

_
value = (int)ADCL; // read lower byte
ADC
value = ADC

_
return ADC

_
value + (int)(ADCH<<8); // add high byte

_
value; // return digital value
}
```