Mini Project

EN407 – Robotics

B.Sc. Engineering
University of Moratuwa

COLLISION AVOIDANCE MOBILE ROBOT

By

Dangampola D.L.	-	020059
De Abrew K.N.T.	-	020067
Kasthuriarachchi T.D.	-	020203
Malwatta K.A.	-	020241
Dahanayake J.K.	-	020057

Department of Electronic and Telecommunication Engineering

August 2006

Abstract

The objective of this project was to develop a Collision Avoidance Mobile robot with onboard sensors and a Microcontroller. The mobile robot designed is capable of moving in an environment which has obstacles avoiding collisions.

The Designed mobiles robot is a three wheeled Robot with differential steering. The Robot has an onboard rotating Ultrasonic Sensor and Bumper switches for improved safety. The Main Controller of the Robot is implemented in a PIC microcontroller.

The Mobile robot uses the Potential Field Method for Obstacle avoidance. The Algorithms runs on the PIC microcontroller based on the information received by the Ultrasonic Ranger.

Table of Contents

List of Figures

1.0 Introduction

1.1 Scope and Overview

The purpose of this project was to develop a mobile robot with the collisions avoidance capability in an obstructed environment. The mobile robot has been built as a fully autonomous vehicle with onboard sensors to get information about the surrounding environment.

The mobile robot is a three wheeled robot platform which employs the differential steering mechanism for motion in given angles. Two stepper motors have been used for the driving wheels. The robot has an onboard Ultrasonic sensor which is mounted on the standard servo motor. The Servo Motor and the Ultrasonic sensor are controlled by a dedicated microcontroller which sends the information collected to the main controller. To improve the reliability the bumper switches has been used as redundant sensors.

The Potential Field method has been used as the obstacle avoidance algorithm and the Algorithm is implemented in the main PIC microcontroller which is on the mobile robot. The Algorithm implemented is used to avoid the obstacle and to drive the robot to a locally generated goal.

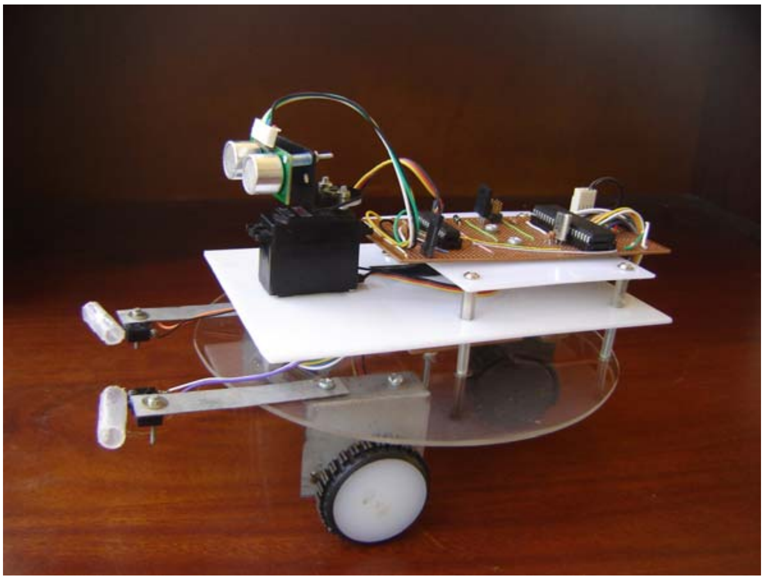


Figure 1 : The Mobile Robot

1.2 Basic Operation

When the robot is switched on it scans its surrounding environment by rotation the Ultrasonic sensor in 45^o steps. Then the distance of the nearest obstacle in each direction will be measured and the data is fed to the main controller. The main controller implements the Potential Field Algorithm and decides the direction which the mobile robots should move. According to that, the main controller sends the control signal sequence to each stepper motor to turn the robot to the specified angle using differential steering.

Thereafter the robot moves a predefined distance and the robot scans its environment again as mentioned earlier. This process continues when the mobile robot is switched on.

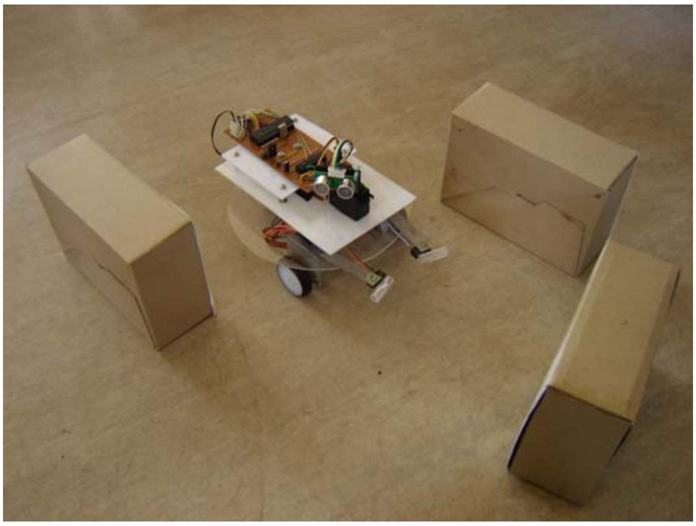


Figure 2: Mobile Robot Navigating through Obstacles

2.0 Overall system Design and operation

2.1 Functional Block Diagram

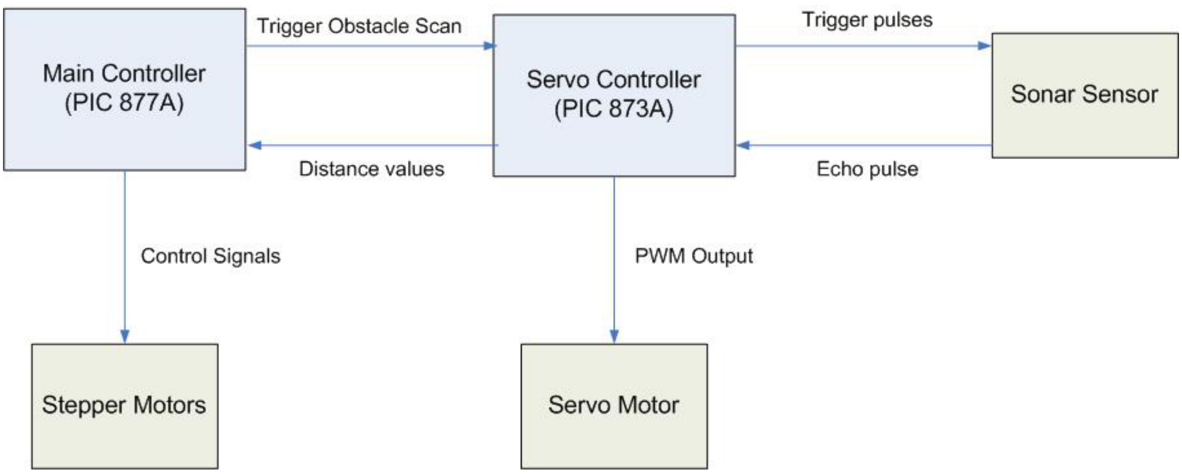


Figure 3: Block Diagram of the System

The above diagram is the functional block diagram of the entire system. The Main Controller will trigger the Servo Controller, receive the distance values, run the collision avoidance algorithm and control the Stepper motors. The Servo Controller controls the sonar sensor and servo motor while providing the readings from the sonar sensor to the Main Controller. A description of each of these functional blocks is given in next chapter.

2.2 System Operation

The operation of the whole system can be represented using the following flow chart.

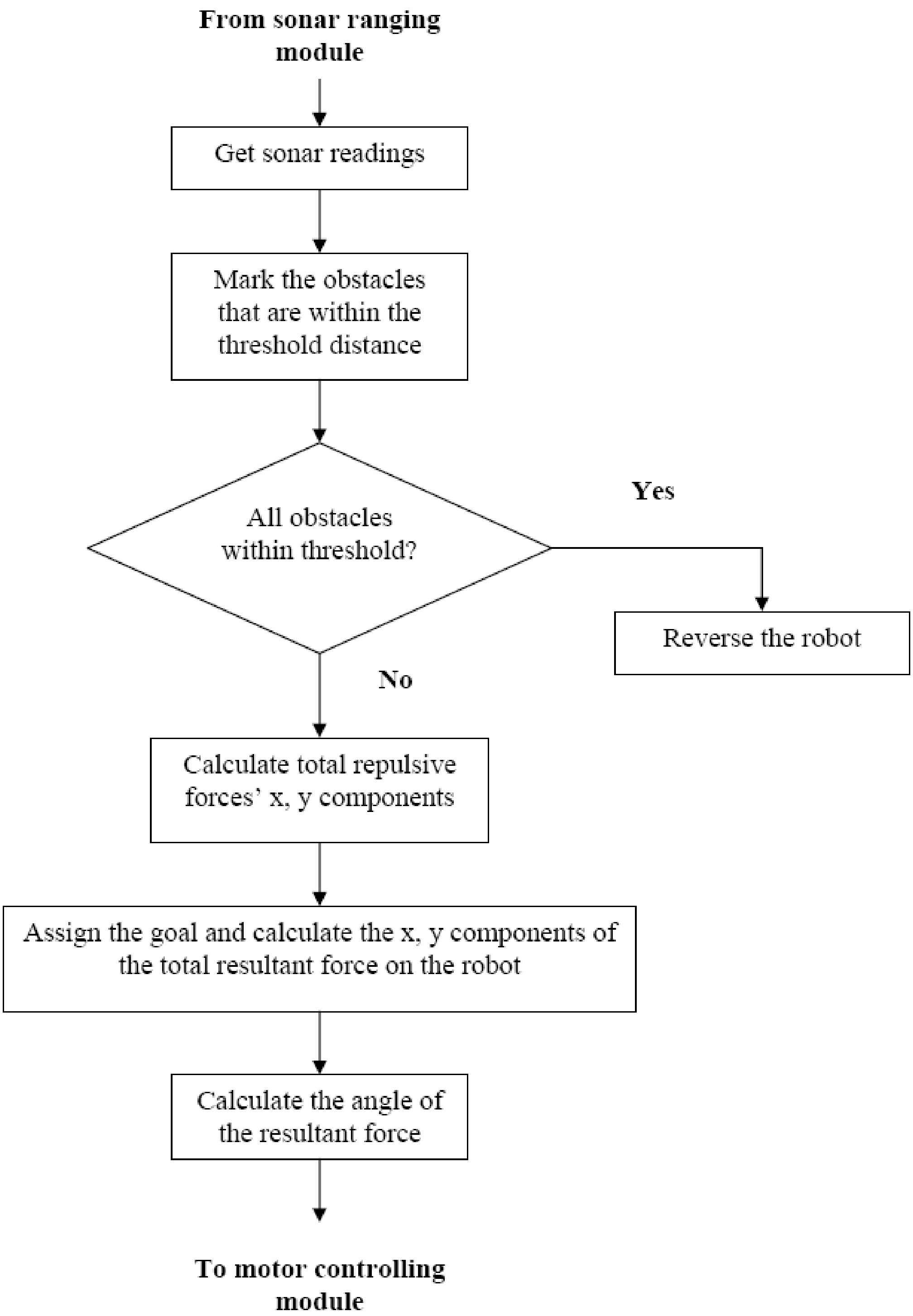


Figure 4 : Flow chart

2.3 Implementation of the potential field method

Here we implement the conventional potential field method with some modifications so that it is not computationally expensive in the PIC microcontroller. Here we do not use the inverse square law to calculate the repulsive force exerted by the obstacles because it involves very expensive calculations that takes lot of time in a microcontroller. Instead we use weights so that obstacles that are nearer to the robot exert high weights while obstacles that are far from the robot exert low weights. These weights are equivalent to the repulsive forces.

The distances to the obstacles are given by the sonar sensor readings. These readings come from angles 0, 45, 90, 135 and 180 degrees as shown in the figure (a). There's an assumption that we make at this point. When we get a reading from a certain direction we assume that the obstacle is aligned with that axis. For example the obstacle **A** will give a reading in 0 degrees direction so we assume that **A** is aligned with 0 degrees axis in calculating the repulsive force though it is not the real case. We make this assumption because it is hard to calculate the exact angel at which the obstacle is located using a PIC microcontroller. For the sonar sensor the probability of finding an object is distributed as shown in figure (b). This is a bell-shaped probability distribution having the highest probability along the main axis of the sonar beam. Therefore we think that our assumption is reasonable enough to apply.

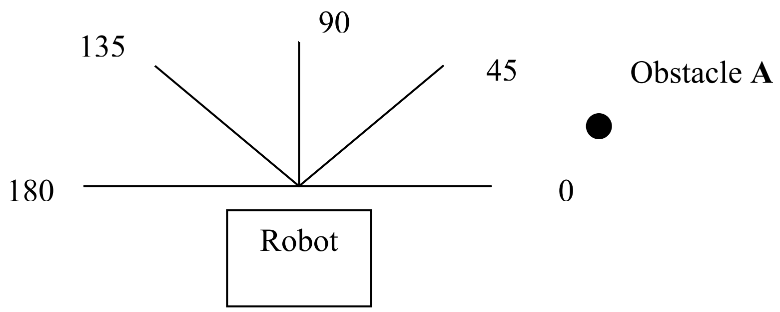


Figure (a)

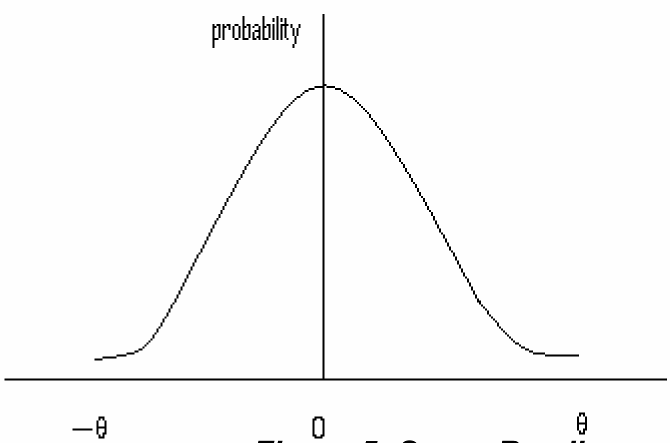


Figure (b)

Figure 5: Sonar Readings

When we calculate the resultant force we ignore the obstacles that are beyond a certain threshold distance D_{Th} to avoid unnecessary calculations. For example obstacle B is taken into account because it is within the threshold while obstacle A is discarded from the process of calculating the resultant force as it is away from the threshold distance.

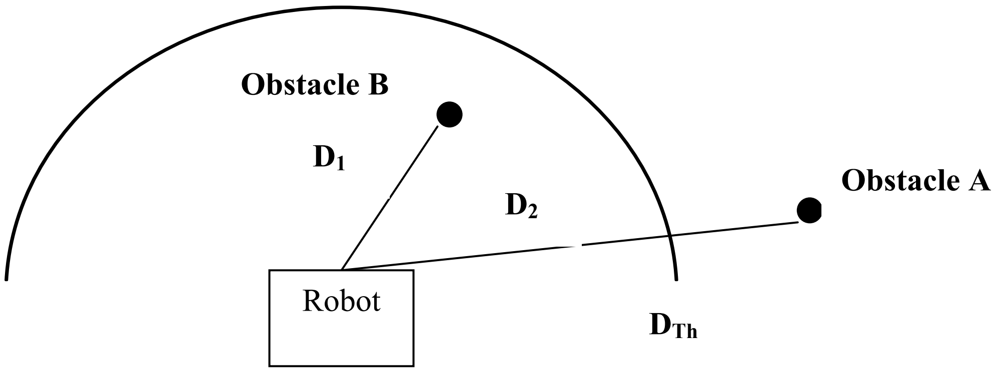


Figure 6 : Repulsive Forces

The repulsive force (or weight) for obstacle B is calculated as

$$F = D_{Th} - D_1$$

If all five sonar readings come from within the threshold distance, the repulsive force is not calculated and the robot is instructed to reverse.

All the repulsive forces are resolved and the total repulsive forces' x and y components are calculated. Resolving for 45 and 135 degrees components is done using a table lookup. The advantage of these angles is that we don't have to use two separate tables for sin and cos.

Selecting the goal is done on a priority basis. This priority scheme is shown in the following figure.

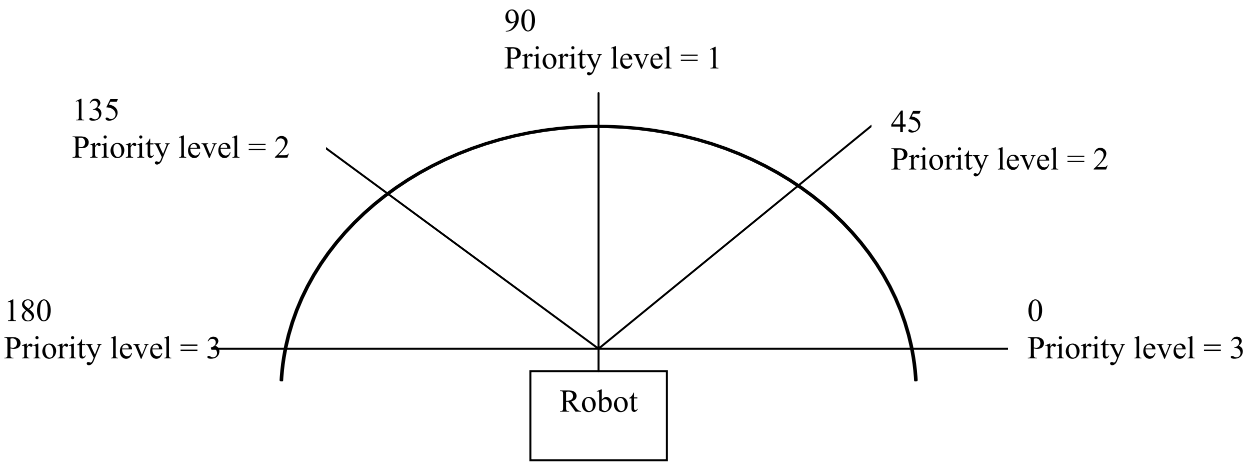


Figure 7 : Local Goals

For example if the sonar reading indicates that the 90 direction is clear (that is there's no obstacle within the threshold distance in that direction) it is the direction that is chosen as the goal direction even if any other directions are clear.

After assigning the goal, its attractive force which is a constant is resolved if necessary and is subtracted or added accordingly to the resultant repulsive forces' x and y components to calculate the x and y components of the total force acting on the robot. Then the angle of the resultant force is calculated from these x and y components and it is fed to the motor controlling unit.

3.0 Components of the Mobile Robot

3.1 The PIC 877A microcontroller

This is the main controller of the mobile robot. When the robot is turned on, the main controller PIC sends a trigger to the servo control PIC to perform an obstacle scan. The results of the obstacle scan are then obtained by the main controller PIC through USART reception. Once the data is received, it is inputted into the conventional potential field algorithm as described earlier. This algorithm will decide the angle to which the mobile robot should turn. Then the appropriate control signals are sent to the two stepper motors in order to obtain this angle. Afterwards the control signals required for the movement (forward or backward) is give to the stepper motors which will be followed by stopping the mobile robot and sending the trigger signal to the servo control PIC. This will continue as a cycle.

3.2 Servo Motor

The Servo motor is used to rotate the sonar sensor to 5 predetermined positions. At these positions the reading of the sonar sensor is obtained 4 times and averaged.

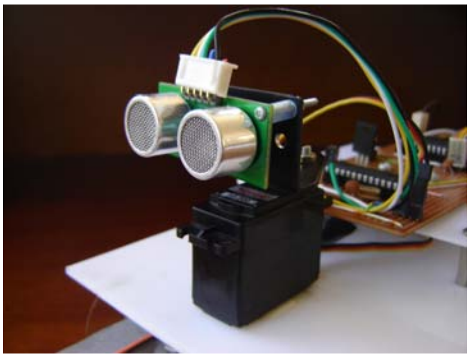


Figure 8: hitec HS 300 Servo motor and the Sonar sensor

The servo motor consists of three wires.

Yellow wire - Control

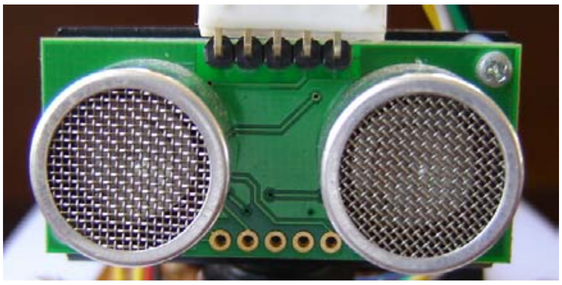
Red wire - +5 V

Black wire - Ground

The position of the servo motor is determined by the width of the (3-5V) pk-pk square pulse sent to its Control wire. The pulses should be given every 20 ms. Given below are the pulse durations required for the servo motor positions are given below.

- Center Position – 1.7 ms
- 90 degrees counter clockwise from center position – 0.9 ms
- 45 degrees counter clockwise from center position – 1.3 ms
- 90 degrees clockwise from center position – 2.1 ms
- 45 degrees clockwise from center position - 2.5 ms

3.3 Sonar Sensor



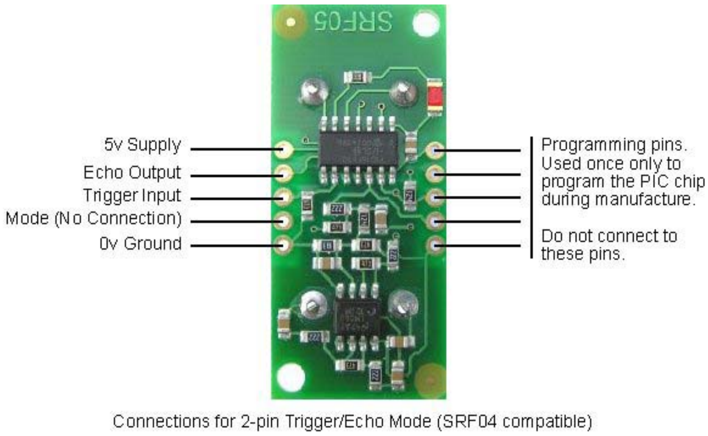


Figure 9 : SRF 05 Sonar Sensor

The mode pin is not used , thus we have used separate pins for echo output and trigger input. The sonar sensor is triggered by sending a 10us pulse. The echo pulse width determines the distance to the object. The range of the echo pulse is from 100us to 25 ms.

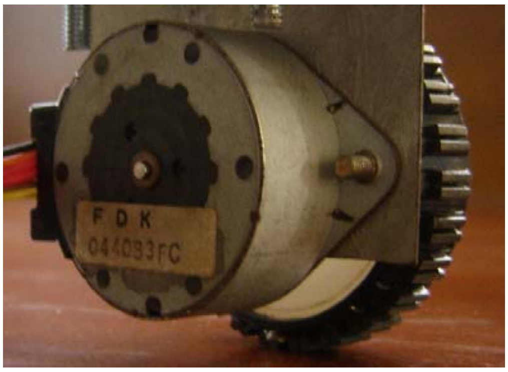
The 10 us is sent to the trigger input and the echo output is received by the PIC 873A microcontroller.

3.4 Stepper motors

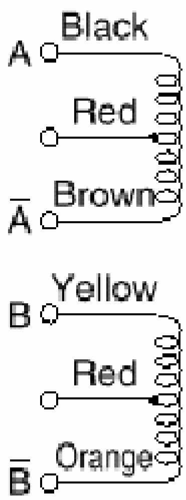
Two FDK stepper motors are used and are controlled independently of each other to obtain the differential steering of the mobile robot. The step angle of the stepper motor is 7.5 degrees. 2 wires (red) are present to give power and the other 4 wires (black, brown, yellow and orange) are for controlling the motor.

Drive mode – Unipolar mode

Excitation - Two way excitation



Unipolar Mode



	CW → / ← CCW			
Red	+	+	+	+
Red	+	+	+	+
Black	−	−		
Brown			−	−
Yellow	−			−
Orange		−	−	

Two way Excitation

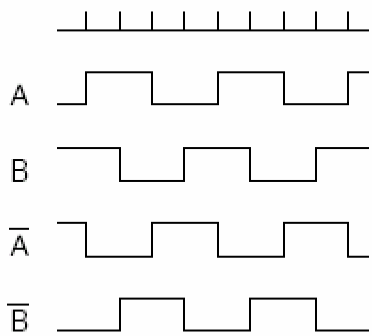


Figure 10 : FDK Stepper motors – Unipolar mode operation

3.5 The PIC 873A microcontroller – Sonar/ Servo controller

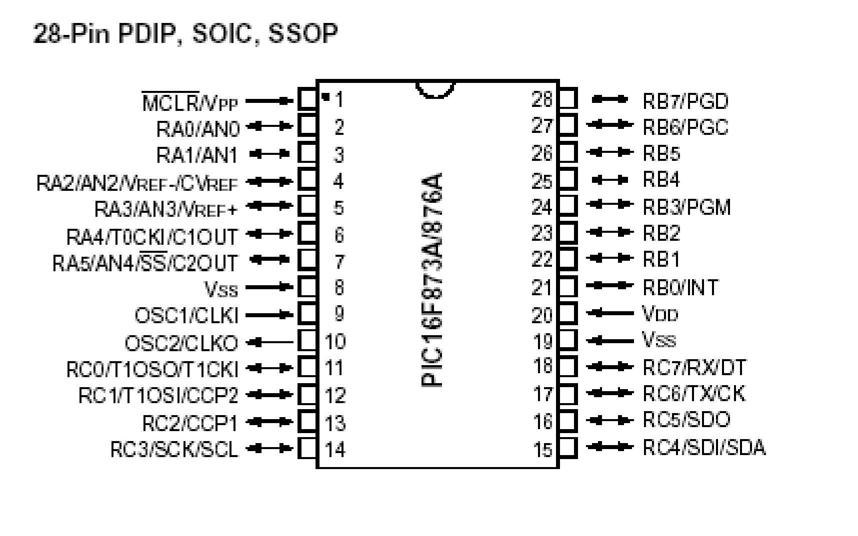


Figure 11 : PIC 873A microcontroller Pin Diagram

In the above PIC microcontroller the following pins are configured as inputs and outputs

Inputs

RA1 - Echo output from sonar

RB0 – Interrupt from Master PIC

Outputs

RA3 - Trigger input to the Sonar

RC0 -PWM output to the Servo Motor

RC6 -USART TX to the Master PIC

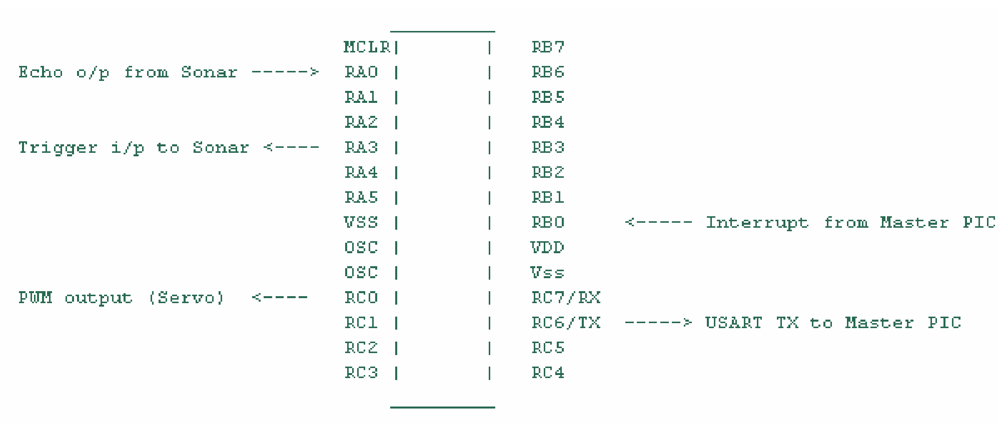
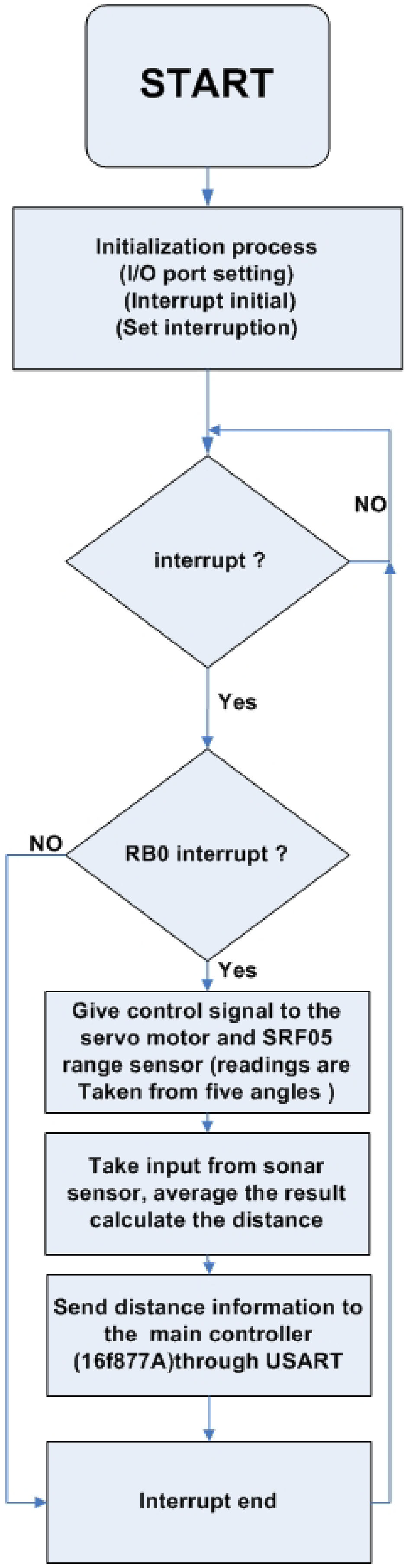


Figure 12 : PIC 873A microcontroller - Used Pin layout



When an interrupt arrives from the Master PIC to perform obstacle surveillance, the PWM output will be given such that the servo motor will move the sonar sensor to *Position 1*. At this position the sonar will be triggered and four echo outputs will be obtained and averaged. Then the servo motor will move the sonar to *Position 2* and as before the readings from the sonar sensor will be obtained and averaged. This procedure is performed in 5 positions. Finally the averaged distance readings from these 5 positions will be sent through USART transmission to the master PIC. The software flow chart of the PIC program is given below. The program is given in Appendix B .

The 5 Position that readings from the sonar sensor are taken

Position 1 – 90 degrees counter clockwise from the center position

Position 2 - 45 degrees counter clockwise from center position

Position 3 – Center position

Position 4 - 45 degrees clockwise from center position

Position 5 - 90 degrees clockwise from center position

3.6 Mechanical Design

The mobile robot has a circular shape of 10cm radius. The circular shape chosen to suit the differential steering mechanism and to minimize the collisions when steering. The robot consists of two driving wheels and one passive wheel.

The robot base is basically constructed using plastics. The two motors are mounted to the base using metal sheets. The wheels are directly coupled to the motor shafts and the wheels were cut from Nylon.

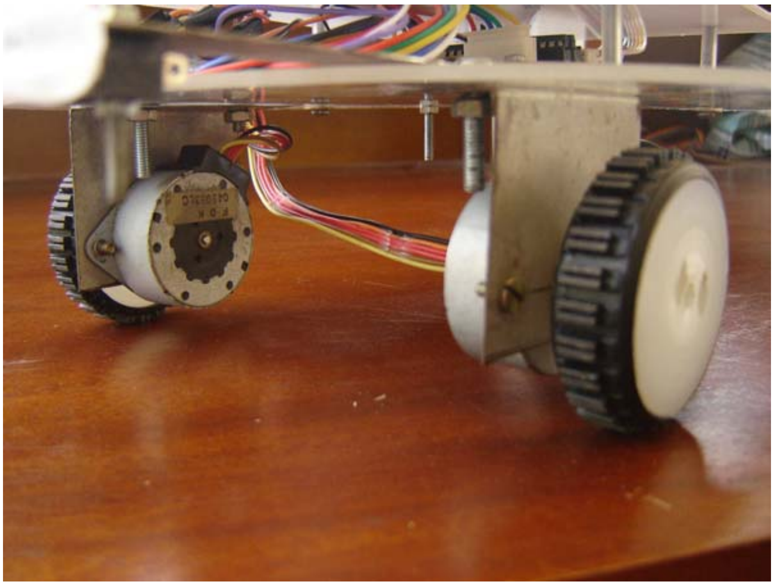


Figure 13 : Robot Base

3.7 Circuit Diagram

The Circuit Diagram of the Electronic circuit of the robot is shown below.

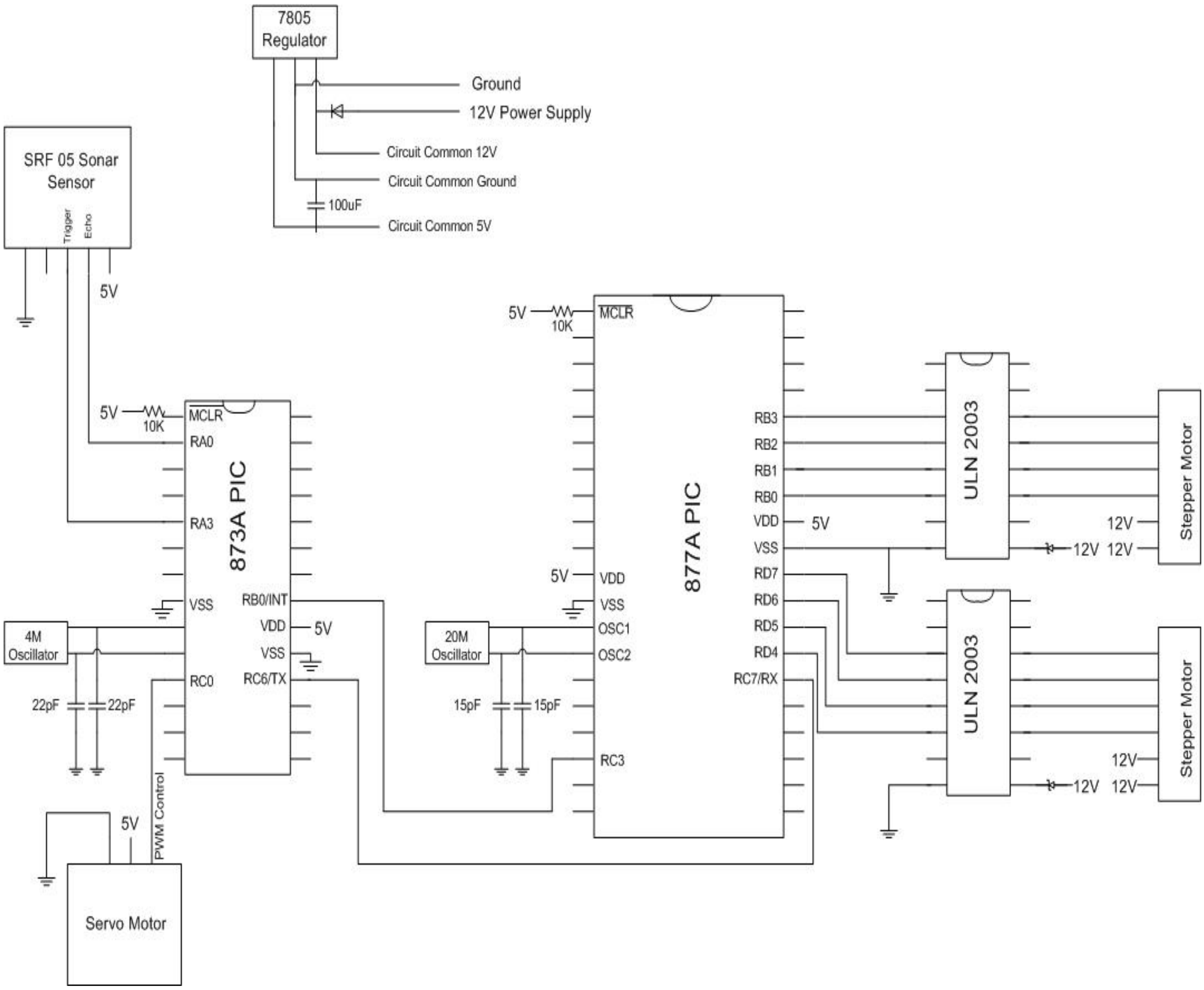


Figure 14 : Circuit Diagram

4.0 Results

After creating the mobile robot, implementing the collision avoidance algorithm on the microcontroller, testing and with modifications we were able to achieve our project goal. That is to design a collision avoidance robot. Our final version of the mobile robot was able to avoid collisions

90% of the time (according to test results). Since it is quite difficult to develop a 100% collision avoidance system, we believe that the achieved collision avoidance rate is satisfactory.

5.0 Limitations and Further Developments

There are several limitations that exist in the current system which should be addressed in further developments.

The mobile robot has information only about its local environment and does not localize itself in a global environment. Thus it is impossible to introduce a define goal to the mobile robot to reach in global environment.

The robot scans through the sonar sensor only in five predefined directions. Thus it is assumed that any obstacle detected lies in those directions only. This effect can be minimized by incorporating probabilistic models to the system which is somewhat difficult in a microcontroller.

Also sometimes some obstacles are not detected when the obstacle surface isn’t in an angle to sufficiently reflect the waves sent by the sonar sensor.

A proper attention should be paid to the above matters in a further development of this project.

APPENDIX A : PIC Program : Main Controller

(The Editable .asm files are given in the CD)

```
;****************************************************************************
;                                                                           *
;    Filename:maincontroller.asm                                            *
;                                                                           *
;    File Version:01                                                        *
;                                                                           *
;   ROBOTICS GROUP PROJECT                                                  *
;                                                                           *
;   Date            : 2006/06/25                                            *
;   Group members   : Malwatta K.A.                                         *
;                     Dahanayaka J.K                                        *
;                     Dangampola D.L.                                       *
;                     Abrew K.N.T                                           *
;                     Kasturiarachi T.D.                                    *
;****************************************************************************
;                                                                           *
;    Files required:                                                        *
;                                                                           *
;                                                                           *
;                                                                           *
;****************************************************************************
;                                                                           *
;    Notes:                                                                 *
;                                                                           *
;                                                                           *
;                                                                           *
;                                                                           *
;****************************************************************************


    list        p=P16f877A.INC,c=140    ; list directive to define processor
    errorlevel 1, -(305)
    #include     "p16f877A.INC"  ; processor specific variable definitions
    __config _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF & _BODEN_OFF & _WRT_OFF & _LVP_OFF & _CPD_OFF
    ;__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _RC_OSC & _WRT_OFF & _LVP_ON & _CPD_OFF

; '__CONFIG' directive is used to embed configuration data within .asm file.
; The labels following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.


;MACRO**********************************************************************
;***************************************************************************
bank_0 macro
    bcf STATUS,RP0
    bcf STATUS,RP1
    endm
bank_1 macro
    bSf STATUS,RP0
    bcf STATUS,RP1
    endm
bank_2 macro
    bcf STATUS,RP0
    bSf STATUS,RP1
    endm
bank_3 macro
    bSf STATUS,RP0
    bSf STATUS,RP1
    endm

;***** VARIABLE DEFINITIONS**********************************************
w_temp          equ 20
status_temp     equ 21
pclath_temp     equ 22
time_count_left equ 23
time_count_right    equ 24
mem_left        equ 25
mem_right       equ 26
cw_ccw_stop     equ 27  ;memory taht used to store the motion conditions of the left and right motors
                        ;|*|*|*|*|right_cw or right_ccw|right_start|left_cw or left_ccw|left_start|
speed_left_reg  equ 28
speed_right_reg equ 29
```

```
temp_1 equ  2A
temp_2 equ  2B


test_reg    equ 2C
test_reg_1  equ 2D  ;used store 8 conditions at the end of progrsm

data_save_1 equ 2E
data_save_2 equ 2F  ;used for bumper swithes
flag_reg equ 30

vehicle_para_1  equ 31
vehicle_para_2  equ 32
vehicle_para_3  equ 33
temp
x equ 34
y equ 35
a equ 36
cnt equ 37
result equ 38
;w_temp     equ 2E; 20
;status_temp    equ 2F; 21
;pclath_temp    equ 30; 22
count   equ 39; 23
min     equ 3A  ; 27
mini    equ 3B      ; 28
objects equ 3C  ; 2a
objcopy equ 3D; 2b
left    equ 3E; 2c
right   equ 3F; 2d
front   equ 40; 2e
back    equ 41; 2f
mapped  equ 42; 30
xcomp   equ 43; 31

ycomp   equ 45; 32
rxdata  equ 46; 33

;CONSTANTS
CONST_left      equ .24
CONST_right     equ .24

threshold   equ .59
goalpot1    equ .80 ;100
goalpot2    equ .60
;val            equ .100



;**************** Program Start *******************************************
    org 0 ;Reset Vector
    goto INIT
    org 4 ;Interrupt Vector
    goto INT

    org 5
    #include table.asm
    #include arctantable.asm

;***********************************************************************
INIT

    bank_1
    clrf TRISA
    clrf TRISB
    clrf TRISD
    clrf TRISC
```

```
    bsf      TRISC,7      ;usart rx
    bcf      TRISC,6      ;tx

    bsf TRISD,2
;   bsf TRISD,3
;OPTION REGISTOR RBPU/INTEDG/TOCS/PSA/PS2/PS1/PS0
;RBPU=1 pull ups are disabled,PSA=0 prescaler assigned to TMR0 module ,PS2:PS1 prescaller rate selection bits
    movlw    b'10000100'
    movwf    OPTION_REG
    bank_0
    clrf PORTA
    clrf PORTB
    clrf PORTD
    bsf  PORTC,3

    clrf     cw_ccw_stop
    bsf      cw_ccw_stop,0
    bsf      cw_ccw_stop,2
    bcf      cw_ccw_stop,1
    bsf      cw_ccw_stop,3
    clrf     mem_left
    bsf      mem_left,0
    clrf     mem_right
    bsf      mem_right,0
    movlw    h'00'
    movwf    TMR0
    movlw    CONST_left
    movwf    time_count_left
    movlw    CONST_right
    movwf    time_count_right

    movlw    CONST_left
    movwf    speed_left_reg
    movlw    CONST_right
    movwf    speed_right_reg


    clrf test_reg
    clrf test_reg_1

    movlw    h'84'
    movwf    temp_1


    bsf      flag_reg,7
;   clrf     data_save_2 ;used to bumper


    movlw    .1

    movwf vehicle_para_1
    movlw    .30
    movwf vehicle_para_2
    movlw    .42
    movwf vehicle_para_3

;***************************************************
;INITIALIZE USART
    bank_1
    bcf      TXSTA,SYNC
    bcf      TXSTA,BRGH
    movlw    .31
    movwf    SPBRG
    bank_0

    movlw    h'A0'
    movwf    INTCON
;   bsf      INTCON,INTE
;***************************************************

;***************************************************
work

    btfsc    PORTD,2
    call     reverseRobot
    movf     test_reg,w
    xorwf    vehicle_para_1,w
    btfss    STATUS,Z
    goto     next_1

    clrf data_save_2;bumper
```

```
    incf test_reg,f
    bcf      cw_ccw_stop,1
    bsf      cw_ccw_stop,3
    bcf cw_ccw_stop,0
    bcf cw_ccw_stop,2



    bcf INTCON,TMR0IE
    call scan

    goto next_end
;   clrf test_reg



next_1
    movf    test_reg,w
    xorwf   vehicle_para_2,w
    btfss   STATUS,Z
    goto    next_2

    incf test_reg,f
    btfss objects,7
    goto    next_1_1
    bsf cw_ccw_stop,0
    bcf cw_ccw_stop,2
    goto next_end
next_1_1
    bcf cw_ccw_stop,0
    bsf cw_ccw_stop,2
    goto next_end

next_2


    movf    test_reg,w
    xorwf   vehicle_para_3,w
    btfss   STATUS,Z
    goto    work

    incf test_reg,f

    bsf     cw_ccw_stop,0
    bsf     cw_ccw_stop,2
    clrf    test_reg

next_end
    movlw   h'84'
    movwf   temp_1
    movlw   CONST_left
    movwf   time_count_left
    movwf   speed_left_reg
    movlw   CONST_right
    movwf   time_count_right
    movwf   speed_right_reg


    goto work

;****************************************************
timer_int
    bcf INTCON,TMR0IF
    movlw h'00'
    movwf TMR0
    decfsz temp_1,f
    goto    count_84

    movf    speed_left_reg,w
    xorlw   .12
    btfss   STATUS,Z
    decf    speed_left_reg,f

    movf    speed_right_reg,w
    xorlw   .12
    btfss   STATUS,Z
    decf    speed_right_reg,f
```

```
;********************************
;TEST
    incf test_reg,f
;   incf test_reg_1,f

;**********************************



    movlw   h'84'
    movwf   temp_1
count_84

    btfss cw_ccw_stop,0
    goto  start_right
    decfsz time_count_left,f
    goto start_right
;************************************************************************************
;LEFT MOTOR START
;************************************************************************************
start_left
    movlw h'f0'
    andwf PORTB,f
    movlw h'0f'
    addwf PORTB,f
    btfss cw_ccw_stop,1

    goto  start_left_backward

    btfss mem_left,0
    goto step_2_left
;************************************************************************************

;***********************************************************************************
;LEFT MOTOR FORWARD MOTIOM
;***********************************************************************************
step_1_left
    bcf PORTB,0
    bcf PORTB,2
    bcf mem_left,0
    bsf mem_left,1
    goto end_set_left


step_2_left
    btfss mem_left,1
    goto  step_3_left
    bcf PORTB,0
    bcf PORTB,3
    bcf mem_left,1
    bsf mem_left,2
    goto end_set_left

step_3_left
    btfss mem_left,2
    goto  step_4_left
    bcf PORTB,1
    bcf PORTB,3
    bcf mem_left,2
    bsf mem_left,3
    goto end_set_left

step_4_left
    bcf PORTB,1
    bcf PORTB,2
    bcf mem_left,3
    bsf mem_left,0


end_set_left

    movf speed_left_reg,w
    movwf time_count_left
    goto start_right
```

```
;*************************************************************************************
;LEFT MOTOR BACKWARD MOTIOM
;*************************************************************************************

start_left_backward
    btfss mem_left,0
    goto step_2_left_back
step_1_left_back
    bcf PORTB,1
    bcf PORTB,2

    bcf mem_left,0
    bsf mem_left,1
    goto end_set_left_back


step_2_left_back
    btfss mem_left,1
    goto  step_3_left_back
    bcf PORTB,1
    bcf PORTB,3

    bcf mem_left,1
    bsf mem_left,2
    goto end_set_left_back

step_3_left_back
    btfss mem_left,2
    goto  step_4_left_back
    bcf PORTB,0
    bcf PORTB,3
    bcf mem_left,2

    bsf mem_left,3
    goto end_set_left_back

step_4_left_back
    bcf PORTB,0
    bcf PORTB,2
    bcf mem_left,3
    bsf mem_left,0
    goto end_set_left_back

end_set_left_back

    movf speed_left_reg,w
    movwf time_count_left
    goto start_right


;*************************************************************************************
;RIGHT MOTOR START
;*************************************************************************************
start_right
    btfss cw_ccw_stop,2
    goto  INT_END
;   goto  INT_END

    decfsz time_count_right,f
    goto INT_END

    movlw h'0f'
    andwf PORTD,f
    movlw h'f0'
    addwf PORTD,f
```

```
    btfss cw_ccw_stop,3
    goto  start_right_backward

    btfss mem_right,0
    goto step_2_right

;*****************************************************************************************
;RIGHT MOTOR FORWARD MOTION
;*****************************************************************************************
step_1_right
    bcf PORTD,4
    bcf PORTD,6
    bcf mem_right,0
    bsf mem_right,1
    goto end_set_right

step_2_right
    btfss mem_right,1
    goto  step_3_right
    bcf PORTD,4
    bcf PORTD,7
    bcf mem_right,1
    bsf mem_right,2
    goto end_set_right

step_3_right
    btfss mem_right,2
    goto  step_4_right
    bcf PORTD,5
    bcf PORTD,7
    bcf mem_right,2
    bsf mem_right,3
    goto end_set_right

step_4_right
    bcf PORTD,5
    bcf PORTD,6
    bcf mem_right,3
    bsf mem_right,0
    goto end_set_right

end_set_right

    movf speed_right_reg,w
    movwf time_count_right
    goto INT_END

;*****************************************************************************************
;RIGHT MOTOR BACKWARD MOTION
;*****************************************************************************************



start_right_backward
    btfss mem_right,0
    goto step_2_right_back
step_1_right_back
    bcf PORTD,5
    bcf PORTD,6
    bcf mem_right,0
    bsf mem_right,1
    goto end_set_right_back


step_2_right_back
    btfss mem_right,1
    goto  step_3_right_back
    bcf PORTD,5
    bcf PORTD,7
    bcf mem_right,1
    bsf mem_right,2
    goto end_set_right_back
```

```
step_3_right_back
    btfss mem_right,2
    goto  step_4_right_back
    bcf PORTD,4
    bcf PORTD,7
    bcf mem_right,2
    bsf mem_right,3
    goto end_set_right_back

step_4_right_back
    bcf PORTD,4
    bcf PORTD,6
    bcf mem_right,3
    bsf mem_right,0
    goto end_set_right_back

end_set_right_back

    movf speed_right_reg,w
    movwf time_count_right
    goto INT_END
;***********************************************************************
scan
    bank_0
    bsf      RCSTA,SPEN
    bsf      RCSTA,CREN
    movf     RCREG,W
    movf     RCREG,W
    movf     RCREG,W         ; flush rx register
    bsf      STATUS,RP0      ; bank1
    bsf      PIE1,RCIE       ; receive enable
    bcf      STATUS,RP0      ; bank0
    bsf      INTCON,PEIE
    clrf     count       ;set # of bytes rxed to zero
    movlw    rxdata
    movwf    FSR         ;initialize the data pointer

    bcf      PORTC,3     ;SHOULD BE PORTC
    ;bcf         PORTB,1     ;---------******** indicate that scan started

    nop
    nop
    nop
    nop
    bsf      PORTC,3     ;trigger the scan
    return
;*********************************************************************
usartInt
;bcf         PORTB,0     ;-----------*********
;bcf         PORTB,1     ;---------------***********
;bsf         PORTB,2     ;------- indicate uart reception

;movlw  .254
;movwf  val2
    movf     RCREG,W
    btfsc    flag_reg,7
    goto     zz
    goto     xx

zz
    bcf      flag_reg,7
    goto     INT_END
xx
    movwf    INDF
    incf     count,F     ;increment the byte count
    incf     FSR,F       ;increment the pointer
    movlw    5
    xorwf    count,W
    btfsc    STATUS,Z
    goto     thcheck
    goto     INT_END

;----- thcheck  : check the region for clearence ---------
thcheck
    clrf     objects
    clrf     count
    movlw    rxdata
    movwf    FSR

A1
;    movlw    threshold
```

```
    movf    INDF,W
    sublw   threshold
    rlf     objects,F
    incf    count,F
    movlw   5
    xorwf   count,W
    btfsc   STATUS,Z
    goto    potCalc     ; do potential field calculations
    incf    FSR,F
    goto    A1

;-------------- potential field calculations -----------------
potCalc
    clrf    count
    clrf    left
    clrf    right
    clrf    front
    clrf    back        ; clear registers
    movf    objects,W
    movwf   objcopy     ; make a copy of objects
    xorlw   h'1f'
    btfsc   STATUS,Z
    goto    reverseRobot    ; if the region is not clear so it can't find a goal direction, then reverse the robot
    rlf     objcopy,F
    rlf     objcopy,F
    rlf     objcopy,F

repul
    rlf     objcopy,F
    btfss   STATUS,C
    goto    endRepForce
    movlw   rxdata      ; map the value if it is within threshold range
    addwf   count,W
    movwf   FSR
    movf    INDF,W
    sublw   threshold
    movwf   mapped      ; mapped=threshold-INDF -- end fo mapping
    ; switch statements
    movf    count,W
    xorlw   0
    btfsc   STATUS,Z

    goto    switch0
    movf    count,W
    xorlw   1
    btfsc   STATUS,Z
    goto    switch1
    movf    count,W
    xorlw   2
    btfsc   STATUS,Z
    goto    switch2
    movf    count,W
    xorlw   3
    btfsc   STATUS,Z
    goto    switch3
    movf    count,W
    xorlw   4
    btfsc   STATUS,Z
    goto    switch4
    goto    endRepForce

switch0
    movf    mapped,W
    addwf   right,F         ; first reading comes from left, so the repulsive force is to the right
    goto    endRepForce
switch1
    call    resolve         ; W=resolved component of 'mapped'
    addwf   right,F
    addwf   back,F
    goto    endRepForce
switch2
    movf    mapped,W
    addwf   back,F
    goto    endRepForce
switch3
    call    resolve
    addwf   left,F
    addwf   back,F
    goto    endRepForce
switch4
    movf    mapped,W
```

```
    addwf   left,F
endRepForce
    incf    count,F
    movlw   5
    xorwf   count,W
    btfss   STATUS,Z
    goto    repul

; set the goal
    btfsc   objects,2
    goto    setg1
    movlw   goalpot1
    movwf   front
    goto    endSetG
setg1
    btfsc   objects,1
    goto    setg2
    movlw   goalpot2
    movwf   front
    addwf   right,F     ; ****** check for overflow
    goto    endSetG
setg2
    btfsc   objects,3
    goto    setg3
    movlw   goalpot2
    movwf   front
    addwf   left,F      ; ****** check for overflow
    goto    endSetG
setg3
    btfsc   objects,4
    goto    setg4
    movlw   goalpot1
    addwf   left,F      ; ****** check for overflow
    goto    endSetG
setg4
    btfsc   objects,0
    goto    default
    movlw   goalpot1
    addwf   right,F     ; ****** check for overflow

    goto    endSetG
default
    ; code for reverse  -- this is not necessaty because this condition is never reached
    goto    reverseRobot
endSetG
    ; calculate the resultant force
    movf    left,W
    subwf   right,W
    btfss   STATUS,C
    goto    reverseOrderX
    movwf   xcomp
    bsf     objects,7       ; set objects.7 if the x-component is in 'right' direction
    goto    findYComp
reverseOrderX
    movf    right,W
    subwf   left,W
    movwf   xcomp
    bcf     objects,7       ; clear objects.7 if the x-component is in 'left' direction

findYComp
    movf    back,W
    subwf   front,W
    btfss   STATUS,C
    goto    reverseOrderY
    movwf   ycomp
    bsf     objects,6       ; if y component is in 'front' direction
    goto    endRes
reverseOrderY
    movf    front,W
    subwf   back,W
    movwf   ycomp
    bcf     objects,6
    goto endRes

reverseRobot
    bsf     cw_ccw_stop,0
    bsf     cw_ccw_stop,2
    bsf     cw_ccw_stop,1
    bcf     cw_ccw_stop,3
```

```
    clrf    test_reg
    movlw   h'84'
    movwf   temp_1
    movlw   CONST_left
    movwf   time_count_left
    movwf   speed_left_reg
    movlw   CONST_right
    movwf   time_count_right
    movwf   speed_right_reg

    movlw   .15
    movwf    vehicle_para_1

;   bsf data_save_2,0;bumper

    bsf     INTCON,TMR0IE
    goto    INT_END
endRes
    CLRF    x
    movf  xcomp,w
    subwf ycomp,w
    btfss STATUS,C
    goto  normalxy
    bcf   objects,5
    movf xcomp,w
    movwf y
    movf ycomp,w
    movwf x
    goto angle_calc

normalxy
    bsf   objects,5   ;x>y
    movf xcomp,w
    movwf x
    movf ycomp,w
    movwf y
angle_calc
    CALL    FRAC_DIV
    movf    a,w

    movwf   x
    CALL    arctan
    movf    objects,w
    andlw   b'11100000'
    movwf   test_reg_1


    movlw   b'00000000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_5

    movlw   b'00100000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_4

    movlw   b'01000000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_2

    movlw   b'01100000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_3

    movlw   b'10000000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_6

    movlw   b'10100000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_7

    movlw   b'11000000'
    xorwf   test_reg_1,w
```

```
    btfsc   STATUS,Z
    goto    region_1

    movlw   b'11100000'
    xorwf   test_reg_1,w
    btfsc   STATUS,Z
    goto    region_8

region_1
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_1_1
    movlw   .36
    movwf   vehicle_para_3
    goto    end_region
region_1_1
    movlw   .31
    movwf   vehicle_para_3
    goto    end_region

region_2
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_2_2
    movlw   .36
    movwf   vehicle_para_3
    goto    end_region
region_2_2
    movlw   .31
    movwf   vehicle_para_3
    goto    end_region

region_3
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_3_3

    movlw   .36
    movwf   vehicle_para_3
    goto    end_region
region_3_3
    movlw   .42
    movwf   vehicle_para_3
    goto    end_region

region_4
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_4_4
    movlw   .47
    movwf   vehicle_para_3
    goto    end_region
region_4_4
    movlw   .42
    movwf   vehicle_para_3
    goto    end_region


region_5
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_5_5
    movlw   .47
    movwf   vehicle_para_3
    goto    end_region
region_5_5
    movlw   .54
    movwf   vehicle_para_3
    goto    end_region


region_6
    movlw   h'80'
    subwf   result,w
```

```
    btfss   STATUS,C
    goto    region_6_6
    movlw   .47
    movwf   vehicle_para_3
    goto    end_region
region_6_6
    movlw   .54
    movwf   vehicle_para_3
    goto    end_region

region_7
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_7_7
    movlw   .47
    movwf   vehicle_para_3
    goto    end_region
region_7_7
    movlw   .42
    movwf   vehicle_para_3
    goto    end_region


region_8
    movlw   h'80'
    subwf   result,w
    btfss   STATUS,C
    goto    region_8_8
    movlw   .36
    movwf   vehicle_para_3
    goto    end_region
region_8_8
    movlw   .42
    movwf   vehicle_para_3
    goto    end_region

    ; issue motor commands

;--------------- end of potential field calculation --------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end_region
    movlw   .10
    movwf vehicle_para_1

    bsf     INTCON,TMR0IE
    decf    vehicle_para_2,w
    movwf   test_reg
    goto    INT_END
;----------------------------------------------------------
FRAC_DIV:
;-------------------
;Fractional division
;
; Given x,y this routine finds:
;  a = 256 * y / x
;

    movlw  8    ;number of bits in the result
    movwf  cnt
    clrf   a    ; the result
    movf   x,w

L1:

    clrc
    rlf     y,f   ;if msb of y is set we know x<y
    rlf     a,f   ;and that the lsb of 'a' should be set
    subwf   y,f   ;But we still need to subtract the
                  ;divisor from the dividend just in
                  ;case y is less than 256.
    skpnc         ;If y>x, but y<256
    bsf    a,0   ; we still need to set a:0


    btfss   a,0   ;If y<x then we shouldn't have
    addwf y,f   ;done the subtraction
```

```
    decfsz cnt,f
    goto   L1

    return

;***************************************************************************
arctan

    SWAPF    x,W
    ANDLW    0xf
    ADDLW    1
    MOVWF    temp                    ;Temporarily store the index
    CALL     arc_tan_table           ;Get a2=atan( (x>>4) + 1)
    MOVWF    result                  ;Store temporarily in result

    DECF     temp,W                  ;Get the saved index
    CALL     arc_tan_table           ;Get a1=atan( (x>>4) )

    SUBWF    result,W                ;W=a2-a1, This is always positive.
    SUBWF    result,F                ;a1 = a1 - (a1-W) = W

    CLRF     temp                    ;Clear the product
    CLRC

    BTFSC    x,0
    ADDWF   temp,F
    RRF      temp,F
    CLRC

    BTFSC    x,1
    ADDWF   temp,F
    RRF      temp,F
    CLRC

    BTFSC    x,2
    ADDWF   temp,F
    RRF      temp,F
    CLRC

    BTFSC    x,3
    ADDWF   temp,F

    RRF      temp,W

    ADDWF    result,F

    RETURN


;****************************************************************************
INT
    movwf    w_temp           ; save off current W register contents
    movf     STATUS,w         ; move status register into W register
    movwf    status_temp      ; save off contents of STATUS register
    movf     PCLATH,w         ; move pclath register into w register
    movwf    pclath_temp      ; save off contents of PCLATH register
    btfsc    INTCON,TMR0IF
    goto     timer_int
    btfsc    PIR1,RCIF
    goto     usartInt
    goto     INIT


; isr code can go here or be located as a call subroutine elsewhere
INT_END
    movf     pclath_temp,w    ; retrieve copy of PCLATH register
    movwf    PCLATH           ; restore pre-isr PCLATH register contents
    movf     status_temp,w    ; retrieve copy of STATUS register
    movwf    STATUS           ; restore pre-isr STATUS register contents
    swapf    w_temp,f
    swapf    w_temp,w         ; restore pre-isr W register contents
    retfie                    ; return from interrupt


    END                         ; directive 'end of program'
```

```
;table,asm*****************************************
;**************************************************
resolve
    decf    mapped,W
    ;movf   mapped,W
    addwf   PCL,F   ; ***********
    retlw   .41 ;58
    retlw   .40 ;57
    retlw   .40 ;56
    retlw   .39 ;55
    retlw   .38 ;54
    retlw   .37 ;53
    retlw   .37 ;52
    retlw   .36 ;51
    retlw   .35 ;50
    retlw   .35 ;49
    retlw   .34 ;48
    retlw   .33 ;47
    retlw   .33 ;46
    retlw   .32 ;45
    retlw   .31 ;44
    retlw   .30 ;43
    retlw   .29 ;42
    retlw   .29 ;41
    retlw   .28 ;40
    retlw   .27 ;39
    retlw   .27 ;38
    retlw   .26 ;37
    retlw   .25 ;36
    retlw   .25 ;35
    retlw   .24 ;34
    retlw   .23 ;33
    retlw   .23 ;32
    retlw   .22 ;32
    retlw   .22 ;31
    retlw   .21 ;30
    retlw   .21 ;29
    retlw   .20 ;28
    retlw   .19 ;27
    retlw   .18 ;26
    retlw   .18 ;25

    retlw   .17 ;24
    retlw   .16 ;23
    retlw   .16 ;22
    retlw   .15 ;21
    retlw   .14 ;20
    retlw   .13 ;19
    retlw   .13 ;18
    retlw   .12 ;17
    retlw   .11 ;16
    retlw   .11 ;15
    retlw   .10 ;14
    retlw   .9  ;13
    retlw   .8  ;12
    retlw   .8  ;11
    retlw   .7  ;10
    retlw   .6  ;9
    retlw   .6  ;8
    retlw   .5  ;7
    retlw   .4  ;6
    retlw   .4  ;5
    retlw   .3  ;4
    retlw   .2  ;3
    retlw   .1  ;2
    retlw   .1  ;1
    retlw   .0  ;0
; end of resolve table
```

```
;arctantable.asm******************************************
;*********************************************************
arc_tan_table
        ADDWF   PCL,F
        RETLW   .0
        RETLW   .20     ;atan(1/16) = 3.576deg * 256/45
        RETLW   .41
        RETLW   .60
        RETLW   .80
        RETLW   .99
        RETLW   .117
        RETLW   .134
        RETLW   .151
        RETLW   .167
        RETLW   .182
        RETLW   .196
        RETLW   .210
        RETLW   .222
        RETLW   .234
        RETLW   .245
        RETLW   .0  ;atan(32/32) = 45deg * 256/45
```

APPENDIX B : PIC Program : Servo and Sonar Controller

```
;*************************************************************************************;
;
;   SONAR RANGE SENSOR CONTROLLER BOARD
;   VERSION 01
;
;   ROBOTICS GROUP PROJECT
;
;   Date             : 2006/06/25
;   Group members    : Malwatta K.A.
;                      Dahanayaka J.K
;                      Dangampola D.L.
;                      Abrew K.N.T
;                      Kasturiarachi T.D.
;
;*************************************************************************************;
;
;   INPUTS       :   RB0: Interrupt to signal this PIC (from the Master PIC) to start distance measurement
;                    RA0: Echo o/p from Sonar
;                    RC2: PWM(CCP1) signal to the Servo motor
;   OUTPUTS      :   RC6: USART TX pin used to transmit map data to Master PIC (USART RX not used)
;                    RA3: Trigger i/p to the Sonar
;                    RB7: LED to indicate the Sonar operation
;
;   CRYSTAL FREQUENCY: 20 MHz
;
;*************************************************************************************;
;                                    _________
;                               MCLR|         |   RB7
;   Echo o/p from Sonar ----->  RA0 |         |   RB6
;                               RA1 |         |   RB5
;                               RA2 |         |   RB4
;   Trigger i/p to Sonar <----  RA3 |         |   RB3
;                               RA4 |         |   RB2
;                               RA5 |         |   RB1
;                               VSS |         |   RB0      <----- Interrupt from Master PIC
;                               OSC |         |   VDD
;                               OSC |         |   Vss
;   PWM output (Servo)  <----   RC0 |         |   RC7/RX
;                               RC1 |         |   RC6/TX   -----> USART TX to Master PIC
;                               RC2 |         |   RC5
;                               RC3 |         |   RC4
;                                    _________
;
;*************************************************************************************;
;*************************************************************************************;


    list        p=P16f873A.INC,c=140    ; list directive to define processor
    errorlevel 1, -(305)
    #include    "p16f873A.INC"   ; processor specific variable definitions
    __config _XT_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF & _BODEN_OFF & _WRT_OFF & _LVP_OFF & _CPD_OFF


; '__CONFIG' directive is used to embed configuration data within .asm file.
; The labels following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.

;*****************************************************************************
status_save equ     h'20'
w_save      equ     h'21'
pclath_save equ     h'22'
macro_mem1  equ     h'23'
macro_mem2  equ     h'24'
count_high  equ     h'25'
count_low   equ     h'26'
servo_position equ  h'27'
echo_length equ     h'28'
count       equ     h'29'
distance    equ     h'2A'
temp        equ     h'2B'
i           equ     h'2C'
j           equ     h'2D'
k           equ     h'2E'
result_1    equ     h'30'    ;(30 to 33 addresses used for averaging)
result_2    equ     h'34'    ;(34 to 37 addresses used for averaging)
result_3    equ     h'38'    ;(38 to 3B addresses used for averaging)
result_4    equ     h'3C'    ;(3C to 3F addresses used for averaging)
result_5    equ     h'40'    ;(40 to 43 addresses used for averaging)
```

```
count_4_times   equ h'45'
value           equ h'46'
average         equ h'47'

;*************************************************************************************
;CONSTANTS

;MACRO********************************************************************************
bank_0 macro
    bcf STATUS,RP0
    bcf STATUS,RP1
    endm
bank_1 macro
    bSf STATUS,RP0
    bcf STATUS,RP1
    endm
bank_2 macro
    bcf STATUS,RP0
    bSf STATUS,RP1
    endm
bank_3 macro
    bSf STATUS,RP0
    bSf STATUS,RP1
    endm

ms_delay macro  macro_var1,macro_var2 ;this macro is used to give disired delay -for the required delay inputs should be
                                      ; given to the macro appropriatly
    local   ms_1
    local   ms_2
    movlw   macro_var1
    movwf   macro_mem1  ;.77
    movlw   macro_var2
    movwf   macro_mem2  ;.255
    nop
    nop
ms_1
    decfsz  macro_mem2,f
    goto    ms_1

ms_2
    decfsz  macro_mem1,f
    goto    ms_2

    endm

;****************************************************************************************
    org     h'00'           ;When power is given to the pic program starts from this point
    goto    initialize
    org     h'04'           ;When inturupts are occured program automaticaly jumps to this point
    goto    interrupt

;****************************************************************************************;
;               Initialize & Mainloop                                                    ;
;****************************************************************************************;

initialize                          ; "MainLine"

    bank_0
    clrf        PORTA
    clrf        PORTB
    clrf        PORTC               ; Clear all ports initially
    bank_1
    movlw       b'00000111'         ; Port A is Digital inputs
    movwf       ADCON1

    movlw       b'00000001'
    movwf       TRISA
    movlw       b'00000001'
    movwf       TRISB
    movlw       b'00000000'
    movwf       TRISC               ; PORT A,B,C Input-Output configuration

    bcf         OPTION_REG,7        ; PORTB Internal Pullup Resistors are ENABLED
    clrf        INTCON              ; GIE and all interrupts disabled
    movlw       h'90'
    movwf       INTCON
    bsf         INTCON, INTE        ; RB0(Int) Interrupt is Enabled
    bank_0
;   clrf        indirect
```

```
mainloop
    bsf         INTCON, GIE         ; check for RB0 interrupt
    nop
    bcf         INTCON,GIE          ; If so, break to ISR routine
    goto        mainloop


;*************************************************************************************;
;               Interrupts                                                           ;
; Interrupts Used in this ISR routine (unpolled) : RB0(INT) External Interrupt       ;
;*************************************************************************************;
interrupt
    movwf   w_save          ; save off current W register contents
    movf    STATUS,w        ; move status register into W register
    movwf   status_save     ; save off contents of STATUS register
    movf    PCLATH,w        ; move pclath register into w register
    movwf   pclath_save     ; save off contents of PCLATH register
    btfss   INTCON,INTF     ;External interrupt(RB0)?
    goto    initialize
    goto    rb0_int

rb0_int
    bcf     INTCON,INTF     ;Flag cleared


    clrf    value
    clrf    average

;*************************************************************************************;

first_reading_initialization
        movlw   h'30'
        movwf   count_4_times

first_reading
        movlw   .80
        movwf   servo_position

first_reading_delay

        call    servo_0_degree
        call    ten_ms_delay
        decfsz  servo_position,f
        goto    first_reading_delay

        call    sonar_start
        movf    distance,w
        movwf   value

        bcf     STATUS,C
        rrf     value,F
        bcf     STATUS,C
        rrf     value,F
        movf    value,w
        addwf   average,F

        incf    count_4_times
        movlw   h'34'
        subwf   count_4_times,W
        btfss   STATUS,Z
        goto    first_reading
        nop
        movf    average,w
        movwf   result_1
        clrf    average
        goto    second_reading

;*************************************************************************************;

second_reading
        movlw   .40
        movwf   servo_position

second_reading_delay
        call    servo_45_degree
        call    ten_ms_delay
        decfsz  servo_position,f
        goto    second_reading_delay
```

```
        call    sonar_start
        movf    distance,w
        movwf   value

        bcf     STATUS,C
        rrf     value,F
        bcf     STATUS,C
        rrf     value,F
        movf    value,w
        addwf   average,F

        incf    count_4_times
        movlw   h'38'
        subwf   count_4_times,W
        btfss   STATUS,Z
        goto    second_reading
        nop
        movf    average,w
        movwf   result_2
        clrf    average
        goto    third_reading
;*************************************************************************************;

third_reading
        movlw   .40
        movwf   servo_position

third_reading_delay
        call    servo_90_degree
        call    ten_ms_delay
        decfsz  servo_position,f
        goto    third_reading_delay

        call    sonar_start
        movf    distance,w
        movwf   value

        bcf     STATUS,C
        rrf     value,F

        bcf     STATUS,C
        rrf     value,F
        movf    value,w
        addwf   average,F

        incf    count_4_times
        movlw   h'3C'
        subwf   count_4_times,W
        btfss   STATUS,Z
        goto    third_reading
        nop
        movf    average,w
        movwf   result_3
        clrf    average
        goto    fourth_reading
;*************************************************************************************;

fourth_reading
        movlw   .40
        movwf   servo_position

fourth_reading_delay
        call    servo_135_degree
        call    ten_ms_delay
        decfsz  servo_position,f
        goto    fourth_reading_delay

        call    sonar_start
        movf    distance,w
        movwf   value

        bcf     STATUS,C
        rrf     value,F
        bcf     STATUS,C
        rrf     value,F
        movf    value,w
        addwf   average,F

        incf    count_4_times
```

```
        movlw   h'40'
        subwf   count_4_times,W
        btfss   STATUS,Z
        goto    fourth_reading
        nop
        movf    average,w
        movwf   result_4
        clrf    average
        goto    fifth_reading
;*************************************************************************************;

fifth_reading
        movlw   .40
        movwf   servo_position

fifth_reading_delay
        call    servo_180_degree
        call    ten_ms_delay
        decfsz  servo_position,f
        goto    fifth_reading_delay

        call    sonar_start
        movf    distance,w
        movwf   value

        bcf     STATUS,C
        rrf     value,F
        bcf     STATUS,C
        rrf     value,F
        movf    value,w
        addwf   average,F

        incf    count_4_times
        movlw   h'44'
        subwf   count_4_times,W
        btfss   STATUS,Z
        goto    fifth_reading
        nop
        movf    average,w

        movwf   result_5
        clrf    average
six_reading
        movlw   .20
        movwf   servo_position

six_reading_delay
        call    servo_90_degree
        call    ten_ms_delay
        decfsz  servo_position,f
        goto    six_reading_delay

        goto    data_transmit



;*************************************************************************************;
;               Sonar Data Retrieval to register 'distance'                           ;
;*************************************************************************************;

sonar_start
        clrf    echo_length
        clrf    count
        clrf    distance
sonar_trigger
        bsf     PORTA,3                 ; Sonar Trigger i/p is ENABLED
        movlw   .4                      ;  delay
        movwf   count
        call    delay                   ; Dealy
        bcf     PORTA,3                 ; Sonar Trigger i/p is DISABLED
sonar_burst
        movlw   .28                     ; set the delay for measuring the output
        movwf   count
A0low   btfss   PORTA,0                 ; Has the Echo o/p gone HIGH?
        goto    A0low                   ; NO: check again
        nop                             ; YES: find the distance

sonar_echo
        incf    echo_length,F           ; increment the length of the ECHO signal
```

```
        call    delay                   ; delay for the count assigned above
        bcf     STATUS,Z
        movf    echo_length,W
        sublw   .255                    ; (L-W)-->W
                                        ; check whether if echo length is for the maximum echo time
                                        ; (same as no obstacle in the 4 m range)
        btfsc   STATUS,Z
        goto    sonar_end               ; echo_length is for maximum range(d2)
        nop                             ; echo_length is in between dead zone(d1) and max range(d2)
        btfsc   PORTA,0                 ; check to see if the output is still high
        goto    sonar_echo              ; YES: Then repeat the above statements till it is LOW
        goto    sonar_end               ; NO: Echo o/p is LOW
sonar_end
        movf    echo_length,W           ; THIS PORTION HAS TO CHANGE
        movwf   distance                ;
        return
;*************************************************************************************;

delay
        movf    count,w
        movwf   temp
xx      decfsz  temp,F
        goto    xx
        return
;*************************************************************************************;
;           SERVO MOTOR control                                                        ;
;*************************************************************************************;

servo_0_degree              ;LEFTMOST
        bsf     PORTC,0
        ms_delay .44,.255   ;0.9ms delay
        bcf     PORTC,0
        return

servo_45_degree             ;45 FROM LEFT
        bsf     PORTC,0
        ms_delay .177,.255  ;1.3ms delay
        bcf     PORTC,0
        return

servo_90_degree             ;MIDDLE
        bsf     PORTC,0
        ms_delay .77,.255   ;1ms delay
        ms_delay .31,.201   ;0.7ms delay
        bcf     PORTC,0
        return

servo_135_degree            ;45 FROM RIGHT
        bsf     PORTC,0
        ms_delay .77,.255   ;1ms delay
        ms_delay .77,.255   ;1ms delay
        ms_delay .11,.21    ;0.1ms delay
        bcf     PORTC,0
        return

servo_180_degree            ;RIGHTMOST
        bsf     PORTC,0
        ms_delay .77,.255   ;1ms delay
        ms_delay .77,.255   ;1ms delay
        ms_delay .11,.21    ;0.1ms delay
        ms_delay .31,.101   ;0.4ms delay
        bcf     PORTC,0
        return


;*************************************************************************************;
ten_ms_delay
    nop
    movlw   .13
    movwf   count_high
    movlw   .250
    movwf   count_low
ten
    decfsz  count_low,f
    goto    ten                 ;Inner loop
    decfsz  count_high,f
    goto    ten                 ;Outer loop
    return
```

```
;*******************************************************************************;
;           USART Transmission to the Master PIC (inside ISR routine)          ;
;*******************************************************************************;

data_transmit

    bank_1
    movlw       .6                      ; 9600 Baud (From tables for 4 MHz Crystal with BRGH=0[LOWSPEED])
    movwf       SPBRG
    movlw       b'00100000'             ; X/8-bit/TX Enabled/Async/-/Low Speed/-(Read only)/Parity
    movwf       TXSTA
    bank_0
    bsf         RCSTA,SPEN              ;Serial Port Enabled
    call        delay_TX                ;Give some time

    movf        result_1,W              ; distance related to 0 degree (LEFTMOST)
    movwf       TXREG                   ;As soon as Data fed to TXREG, Transmission starts
    call        delay_TX                ;Give some time to transmit the byte
    nop

    movf        result_2,W              ; distance related to 45 degree (45 FROM LEFT)
    movwf       TXREG                   ;As soon as Data fed to TXREG, Transmission starts
    call        delay_TX                ;Give some time to transmit the byte
    nop

    movf        result_3,W              ; distance related to 90 degree (MIDDLE)
    movwf       TXREG                   ;As soon as Data fed to TXREG, Transmission starts
    call        delay_TX                ;Give some time to transmit the byte
    nop

    movf        result_4,W              ; distance related to 135 degree (45 FROM RIGHT)
    movwf       TXREG                   ;As soon as Data fed to TXREG, Transmission starts
    call        delay_TX                ;Give some time to transmit the byte
    nop

    movf        result_5,W              ; distance related to 180 degree (RIGHT)
    movwf       TXREG                   ;As soon as Data fed to TXREG, Transmission starts
    call        delay_TX                ;Give some time to transmit the byte
    nop

data_transmit_over
    goto        interrupt_end

;*******************************************************************************;
delay_TX
    movlw   h'02'
    movwf   i
    movlw   h'02'
    movwf   j
    movlw   h'02'
    movwf   k
begin   decfsz  i,F
        goto    begin
        decfsz  j,F
        goto    begin
        decfsz  k,F
        goto    begin
    return


;*******************************************************************************;
;                   INTERRUPT EXIT                                              ;
;*******************************************************************************;
interrupt_end
    bank_0
    bcf     INTCON,INTF     ;Flag cleared
;   bcf     PORTB,7         ; LED OFF to indicate exit of Sonar mode of operation
    nop
    movf    pclath_save,w   ; retrieve copy of PCLATH register
    movwf   PCLATH          ; restore pre-isr PCLATH register contents
    movf    status_save,w   ; retrieve copy of STATUS register
    movwf   STATUS          ; restore pre-isr STATUS register contents
    swapf   w_save,f
    swapf   w_save,w        ; restore pre-isr W register contents
    retfie                  ; return from interrupt

;*******************************************************************************;

    end
```