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

1.0 Introduction ........................................................................................................................................ 1  
   1.1 Scope and Overview....................................................................................................................... 1  
   1.2 Basic Operation.............................................................................................................................. 1  
2.0 Overall system Design and operation................................................................................................. 2  
   2.1 Functional Block Diagram ............................................................................................................. 2  
   2.2 System Operation ........................................................................................................................... 3  
   2.3 Implementation of the potential field method ................................................................................ 4  
3.0 Components of the Mobile Robot ...................................................................................................... 6  
   3.1 The PIC 877A microcontroller....................................................................................................... 6  
   3.2 Servo Motor.................................................................................................................................... 6  
   3.3 Sonar Sensor................................................................................................................................... 7  
   3.4 Stepper motors............................................................................................................................... 7  
   3.5 The PIC 873A microcontroller – Sonar/ Servo controller ............................................................. 9  
   3.6 Mechanical Design .......................................................................................................................... 11  
   3.7 Circuit Diagram............................................................................................................................ 12  
4.0 Results .............................................................................................................................................. 12  
5.0 Limitations and Further Developments ............................................................................................ 13  

APPENDIX A : PIC Program : Main Controller ................................................................................... 14  
APPENDIX B : PIC Program : Servo and Sonar Controller ................................................................. 30
List of Figures

Figure 1 : The Mobile Robot .................................................................................................................... 1
Figure 2: Mobile Robot Navigating through Obstacles ........................................................................... 2
Figure 3: Block Diagram of the System ................................................................................................... 2
Figure 4 : Flow chart ................................................................................................................................ 3
Figure 5: Sonar Readings .......................................................................................................................... 4
Figure 6 : Repulsive Forces ..................................................................................................................... 5
Figure 7 : Local Goals .............................................................................................................................. 5
Figure 8: Hitec HS 300 Servo motor and the Sonar sensor .................................................................... 6
Figure 9 : SRF 05 Sonar Sensor ................................................................................................................ 7
Figure 10 : FDK Stepper motors – Unipolar mode operation .................................................................. 8
Figure 11 : PIC 873A microcontroller Pin Diagram ............................................................................... 9
Figure 12 : PIC 873A microcontroller - Used Pin layout ....................................................................... 9
Figure 13 : Robot Base ........................................................................................................................... 11
Figure 14 : Circuit Diagram ................................................................................................................... 12
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](image)

1.2 Basic Operation

When the robot is switched on it scans its surrounding environment by rotation the Ultrasonic sensor in 45° 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.

![Mobile Robot Navigating through Obstacles](image)

**Figure 2: Mobile Robot Navigating through Obstacles**

### 2.0 Overall system Design and operation

#### 2.1 Functional Block Diagram

![Block Diagram of the System](image)

**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.

**From sonar ranging module**

- Get sonar readings
- Mark the obstacles that are within the threshold distance

**All obstacles within threshold?**

- Yes
  - Reverse the robot
- No

**Calculate total repulsive forces’ x, y components**

Assign the goal and calculate the x, y components of the total resultant force on the robot

**Calculate the angle of the resultant force**

To motor controlling module

*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 angle 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.
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.

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.
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](image)

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

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 Diagram](image)

### Two way Excitation

<table>
<thead>
<tr>
<th></th>
<th>CW</th>
<th>CCW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Red</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Black</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Brown</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Yellow</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Orange</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Figure 10 : FDK Stepper motors – Unipolar mode operation**
3.5 The PIC 873A microcontroller – Sonar/ Servo controller

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
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.

![Circuit Diagram](image)

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)

```assembly
#include "p16f877A.INC" ; processor specific variable definitions

;CONFIG1H = 0x00 ; configuration word

;Variables

;*** Variable Definitions

temp equ 20
status_temp equ 21
plush_temp equ 22

time_count_left equ 23
time_count_right equ 24

left equ 25
right equ 26

left_cue_st equ 27 ; memory tag used to store the motion conditions of the left and right motors

left_cue equ 29
right_cue equ 29
```

END
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 program
data_save_1 equ 2E
data_save_2 equ 2F  ;used for bumper switches
time_reg equ 30

vehicle_param_1 equ 31
vehicle_param_2 equ 32
vehicle_param_3 equ 33
temp r equ 34
y equ 35
a equ 36
cnt equ 37

result equ 38
;0_temp equ 2E, 2A
;status_temp equ 2B, 21
;peelset_temp equ 8D, 22
count equ 59, 23

min equ 5A  ; 27
mini equ 5B  ; 28
objects equ 5C  ; 26
object equ 5D  ; 2b
left equ 5E  ; 2c
right equ 5F, 2d
front equ 60  ; 2e
back equ 61, 2f
mapped equ 62, 30
xcomp equ 63, 31

ycomp equ 45, 32
radius equ 46, 33

;CONSTANTS
CONST_left equ .24
CONST_right equ .24

threshold equ .29
goalpct1 equ .80, 100
goalpct2 equ .60
;val equ .100

;************************** Program Start ****************************
cry 0 ;Reset Vector
goto INIT
cry 4 ;Interrupt Vector
goto INIT

org 5
include table.asm
include rotatable.asm

;************************** Program Start ****************************

INIT

entry

clrf TRISA
clrf TRISB
clrf TRISD
clrf TRISC
EN407 : Robotics

Collision Avoidance Mobile robot

; INITIALISER USAR7

; DEFINE CONSTANTS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS

; INITIALISER USAR7

; DEFINE CONSTANTS

; CALLS
```assembly
incf test_reg.i
bclf cu_ccv_stop.1
bclf cu_ccv_stop.0
bclf cu_ccv_stop.2

bclf INTCON,TH900
call scan
jte next_end
; clrf test_reg

next_1
movf test_reg.o
movf vehicle_speed.w
btfsc STATUS,2
jte next_2

incf test_reg.i
btfsc objects,7
jte next_1
bclf cu_ccv_stop.0
bclf cu_ccv_stop.2
jte next_end

next_1.1
bclf cu_ccv_stop.0
bclf cu_ccv_stop.2
jte next_end

next_2

movf test_reg.o
movf vehicle_speed.w
btfsc STATUS,2
jte work

incf test_reg.i
bclf cu_ccv_stop.0
bclf cu_ccv_stop.2
clrf test_reg

next_end
movf h'64'
movf test_reg.3
movf CONST_left
movf time_count_left
movf speed_left_reg
movf CONST_right
movf time_count_right
movf speed_right_reg

jte work

;******************************************************************************
timer_int
bclf INTCON,TH900
movf h'00'
movf TMRO0
decfsz test_reg.1,f
jte next_1.4

movf speed_left_reg.5
movl .12
btfsc STATUS.2
movl .12
decf speed_left_reg.f

movf speed_right_reg.5
movl .12
btfsc STATUS.2
decf speed_right_reg.f
```
;*******************************************************************************
; TEST
;  incl test_reg_1
;  incl test_reg_1, f
;
;*******************************************************************************

    lw     hi, 0'
addi    lo, 0

    bhfix cu_qew_stop_0
    goto start_right
    decf xax_see_count_left_4, f
    goto start_right

;*******************************************************************************
; LEFT MOTOR START
;*******************************************************************************
start_left
    lw     hi, 0'
addi    lo, 0
move   lo, xax_left
    bhfix cu_qew_stop_1
    goto stop_1_left

;*******************************************************************************
; LEFT MOTOR FORWARD MOTION
;*******************************************************************************
step_1_left
    bcft PORTE, 0
    bcft PORTE, 1
    bcft xax_left, 0
    bcft xax_left, 1
    goto end_set_left

    bhfix xax_left, 1
    goto step_2_left
    bcft PORTE, 0
    bcft PORTE, 3
    bcft xax_left, 1
    bcft xax_left, 2
    goto end_set_left

    bhfix xax_left, 2
    goto step_3_left
    bcft PORTE, 0
    bcft PORTE, 3
    bcft xax_left, 2
    bcft xax_left, 3
    goto end_set_left

    bhfix xax_left, 3
    goto step_4_left
    bcft PORTE, 1
    bcft PORTE, 2
    bcft xax_left, 3
    bcft xax_left, 0

end_set_left
    movf speed_left_reg_w
movf xax_see_count_left
    goto start_right
; LEFT MOTOR BACKWARD MOTION

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
    bcf 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,0
    bcf mem_left,1
    bcf mem_left,0
    goto end_set_left_back

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

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

end_set_left_back

    movf speed_left_reg,0
    movwf time_count_left
    goto start_right

; RIGHT MOTOR START

start_right
    btfss syﺻور,0
    goto INT_END
    ; goto INT_END
    decfsz time_count_right,0
    goto INT_END

    movlw 10'
    addwf PORTD,0
    movlw 10'
    addwf PORTD,0
EN407 : Robotics

Collision Avoidance Mobile robot

; LEFT MOTOR FORWARD MOTION

step_1_right
bcf PORTD, 4
bcf PORTD, 6
bcf aon_right, 0
bcf aon_right, 1
goto end_set_right

step_2_right
bcf aon_right, 1
goto step_3_right
bcf PORTD, 4
bcf PORTD, 7
bcf aon_right, 1
bcf aon_right, 2
goto end_set_right

step_3_right
bcf aon_right, 2
goto step_4_right
bcf PORTD, 5
bcf PORTD, 7
bcf aon_right, 2
bcf aon_right, 3
goto end_set_right

step_4_right
bcf PORTD, 6
bcf PORTD, 6
bcf aon_right, 0
bcf aon_right, 0
 goto INT_END

end_set_right

movl speed_right_segn, v
movl time_gonna_right, j
goto INT_END

; LEFT MOTOR BACKWARD MOTION

start_right_backward
bcf aon_right, 0
goto step_2_right_back

step_1_right_back
bcf PORTD, 6
bcf PORTD, 6
bcf aon_right, 0
bcf aon_right, 1
goto end_set_right_back

step_2_right_back
bcf aon_right, 1
goto step_3_right_back
bcf PORTD, 6
bcf PORTD, 7
bcf aon_right, 1
bcf aon_right, 2
goto end_set_right_back
step_3_right_back
    btfsc aux_right,2
    goto step_4_right_back
    bcf PORTD,4
    bcf PORTD,7
    bcf aux_right,3
    goto end_set_right_back
step_4_right_back
    bcf PORTD,4
    bcf PORTD,6
    bcf aux_right,3
    bcf aux_right,0
    goto end_set_right_back
end_set_right_back

movf speed_right_reg,0
    movf time_count_right
    goto INT_END

;******************************************************************************
scan
    bank_0
    bcf BRESTA,SEKH
    bcf BRESTA,LEKH
    movf REGV,Y
    movf BREGV,Y
    movf DREGV,Y ; flush rx register
    bcf STATUS,EP0 ; bank1
    bcf SREG,ESIX ; receive enable
    bcf STATUS,EOO ; bank0
    bcf IN7C0E,PEIX
    clr count ; set $ of bytes read to zero
    movlw reduce
    movf PTR ;initialize the data pointer
    bcf PORTC,1 ;SHOULD BE PORTC
    ;movf PORTB,1 ;-------------- indicate that scan started
    nop
    nop
    nop
    bcf PORTC,0 ;trigger the scan
    return

;******************************************************************************
uartInit
    ;bcf FREQB,0 ;------------------------
    ;bcf FRB,0 ;------------------------
    ;bfr PORTB,3 ;--------- indicate uart reception

;movlw .254
;movf val;
    movf BOCR0.U
    btfsc flag_reg,7
    goto xx
    goto xx

xx
    bcf flag_reg,7
    goto INIT_END

xx
    movf INDF
    incf count,Y ; increase the byte count
    incf YS,Y ; increase the pointer
    movlw S
    movf count,U
    btfsc STATUS,2
    goto thcheck
    goto INIT_END

;----- thcheck : check the region for clearance -------
thcheck
    clr objects
    clr flag
    movlw r7data
    movf YS
    k1
    ; movlw threshold
movf INY,Y
movlw threshold
rlf objects,Y
movf count,F
movf $1
btfsc STATUS,Z
goto potCal ; do potential field calculations
movf PSR,F
goto AI

;----------- potential field calculations -----------
potCal:
call count
call last
call right
call front
call back ; clear registers
movf objects,Y
movwf objectCopy ; make a copy of objects
movlw half
btfsc STATUS,Z
goto reveseRobot ; if the region is not clear so it can't find a goal direction, then reverse the robot
ripf objectCopy,Yipf objectCopy,Yipf objectCopy,Y
ripf objectCopy,Y

rjmp
ripf objectCopy,Y
btfsc STATUS,Z
goto endRepForce
movlw rdata ; map the value if it is within threshold range
addwf count,Y
movf PSR
movf INY,Y
movw threshold
movf mapped ; mapped=threshold-INP -- end fo mapping ; switch statements
movf count,X
movlw 0
btfsc STATUS,Z
goto switch0
movf count,X
movlw 1
btfsc STATUS,Z
goto switch1
movf count,X
movlw 2
btfsc STATUS,Z
goto switch2
movf count,X
movlw 3
btfsc STATUS,Z
goto switch3
movf count,X
movlw 4
btfsc STATUS,Z
goto switch4
goto endRepForce

switch0
movf mapped,Y
addwf right,F ; first reading comes from left, so the repulsive force is to the right
goto endRepForce

switch1
call resolve ; Y-resolved component of 'mapped'
addwf right,F
addwf back,F
goto endRepForce

switch2
movf mapped,Y
addwf back,F
goto endRepForce

switch3
call resolve
addwf left,F
addwf back,F
goto endRepForce

switch4
movf mapped,Y
addif left, F
endSpForce
load count, F
movlw 5
movf count, W
btfsc STATUS, Z
goto repul

; set the goal
btfsc objects, Z
goto setg1
movlw goalpoint1
movf front
goto unhit2

setg1
btfsc objects, L
movf right, F
; ------- check for overflow
movlw endhit3

setg2
btfsc objects, B
movlw goalpoint2
movf front
addif right, F ; ------- check for overflow
goto unhit4

setg3
btfsc objects, 4
movf right, F
; ------- check for overflow
goto end6

setg4
btfsc objects, 0
movf default
movlw goalpoint1
addif right, F ; ------- check for overflow

end6

default
; code for reverse -- this is not necessary because this condition is never reached
goto reverseRobot

endSet6
; calculate the resultant force
movf left, W
subwf right, W
btfsc STATUS, C
movf reverseOrderX
movf xcamp
b cif objects, 7 ; set objects.7 if the x-component is in 'right' direction
goto findVComp
reverseOrderX
movf right, W
subwf left, W
movf xcamp
b cif objects, 7 ; set objects.7 if the x-component is in 'left' direction
findVComp
movf back, W
subwf front, W
btfsc STATUS, C
movf reverseOrderY
movf ycomp
b cif objects, 6 ; set y-component is in 'front' direction
goto endPos
reverseOrderY
movf front, W
subwf back, W
movf ycomp
b cif objects, 6
goto endHas
reverseRobot
b cif cu_eccu_stop, 0
b cif cu_eccu_stop, 2
b cif cu_eccu_stop, 1
b cif cu_eccu_stop, 3
EN407 : Robotics

Collision Avoidance Mobile robot

```
htes STATUS_2
goto region_1
movw '11100100'
movw text_reg_1,w
htes STATUS_2
goto region_3

region_1
movw '0090'
subw result,v
htes STATUS_3
goto region_1_1
movw .96
movw vehicle_para_3
goto end_region
region_1_1
movw .31
movw vehicle_para_3
goto end_region

region_2
movw '0090'
subw result,v
htes STATUS_3
goto region_2_1
movw .96
movw vehicle_para_3
goto end_region
region_2_1
movw .31
movw vehicle_para_3
goto end_region

region_3
movw '0090'
subw result,v
htes STATUS_3
goto region_3_1
movw .96
movw vehicle_para_3
goto end_region
region_3_1
movw .42
movw vehicle_para_3
goto end_region

region_4
movw '0060'
subw result,v
htes STATUS_3
goto region_4_1
movw .47
movw vehicle_para_3
goto end_region
region_4_1
movw .42
movw vehicle_para_3
goto end_region

region_5
movw '0080'
subw result,v
htes STATUS_3
goto region_5_1
movw .47
movw vehicle_para_3
goto end_region
region_5_1
movw .54
movw vehicle_para_3
goto end_region

region_6
movw '0060'
subw result,v
```
EN407 : Robotics
Collision Avoidance Mobile robot

BITSET STATUS, C
GOTO region 5_5
MOV r14, #47
MOV A, vehicle_para_9
GOTO end_region

region 5_5
MOV r14, #47
MOV A, vehicle_para_9
GOTO end_region

region_7
MOV r14, #49
MOV A, result_w
BITSET STATUS, C
GOTO region_7_7
MOV r14, #47
MOV A, vehicle_para_9
GOTO end_region

region_7_7
MOV r14, #42
MOV A, vehicle_para_9
GOTO end_region

region_8
MOV r14, #49
MOV A, result_v
BITSET STATUS, C
GOTO region_8_8
MOV r14, #47
MOV A, vehicle_para_9
GOTO end_region

region_8_8
MOV r14, #42
MOV A, vehicle_para_9
GOTO end_region

; issue motor commands

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

end_region
MOV r14, #10
MOV A, vehicle_para_1

BIF INT/CON, INR018
DClF vehicle_para_8.w
MOV A, text_seg
GOTO INT_END

;-------------------------
; Fractional division ;
; Given x,y this routine finds:
; a = x F / y
;

MOV r14, #number of bits in the result
MOV A, cnt
Clr A ; the result
MOV r14, #w, w

Clr
Clr y, f ; if abs of y is too small we know x/y
Clr x, f ; and that the abs of 'x' should be set

DClF y, f ;But we still need to subtract the sigmoid from the dividend just in case y is less than 256.
sync
Clr y, f ; If yes, but y is too
BIF a, #0 ; we still need to set a=0

Bitset a, #0 ; If you then we shouldn't have
Add w, y, f ; since the subtraction

26
SNAP   Y,9

BRIEF  0.xf
BBRIEF 1

MOVWF	temp ;Temporarily store the index
CALL	arc_can_code ;Get arc_can_code (n>4) + l]
MOVWF	result ;Store temporarily in result

BRIEF
temp,Y ;Get the saved index
CALL	arc_can_code ;Get arc_can_code (n>4) 

SUBWF
temp,Y ;n-al-al, This is always positive.
SUBWF	result,Y ;al = al - |al-Y| - Y

CLRF
temp ;Clear the product

RETSC   %,0

BRIEF
temp,Y
CLRC

RETSC   %,1
BBRIEF	temp,Y
CLRC

RETSC   %,2
BBRIEF	temp,Y
CLRC

RETSC   %,3
BBRIEF
temp,Y
CLRF
temp,Y
BBRIEF	result,Y
RETURN

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

INST

MOVWF	v_temp ; Save off current V register contents
MOVWF	status_temp ; Save off contents of STATUS register
MOVWF	PCLATH,v ; Move PCLATH register into v register
MOVWF	PCLATH_temp ; Save off contents of PCLATH register

bitset: INDF0H,TH00H
goto
timer_int
bitesc: FISL,MO1
goto
userLib
goto
ZB

; ISR code can go here or be located at a call subroutine elsewhere

ISR_END:

MOVWF	PCLATH_TEMP ; Restore pre-int PCLATH register contents
MOVWF	status_temp ; Restore pre-int STATUS register contents
MOVWF	STATUS ; Restore pre-int STATUS register contents
swapf
temp,f
swapf
temp,f ; Restore pre-int Y register contents
retfie ; Return from interrupt

END ; Directive 'end of program'
```
:table end

; end of resolve table
```
<table>
<thead>
<tr>
<th>arc_tan_table</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MBW</td>
<td>0</td>
</tr>
<tr>
<td>RHTLW .20</td>
<td>\text{atan}[1/16] = 3.57669^\circ \approx 206/10</td>
</tr>
<tr>
<td>RHTLW .41</td>
<td></td>
</tr>
<tr>
<td>RHTLW .60</td>
<td></td>
</tr>
<tr>
<td>RHTLW .80</td>
<td></td>
</tr>
<tr>
<td>RHTLW .99</td>
<td></td>
</tr>
<tr>
<td>RHTLW .117</td>
<td></td>
</tr>
<tr>
<td>RHTLW .154</td>
<td></td>
</tr>
<tr>
<td>RHTLW .181</td>
<td></td>
</tr>
<tr>
<td>RHTLW .182</td>
<td></td>
</tr>
<tr>
<td>RHTLW .216</td>
<td></td>
</tr>
<tr>
<td>RHTLW .210</td>
<td></td>
</tr>
<tr>
<td>RHTLW .222</td>
<td></td>
</tr>
<tr>
<td>RHTLW .234</td>
<td></td>
</tr>
<tr>
<td>RHTLW .245</td>
<td></td>
</tr>
<tr>
<td>RHTLW .0</td>
<td>\text{atan}(12/32) = 46.25^\circ \approx 256/10</td>
</tr>
</tbody>
</table>

EN407 : Robotics                                       Collision Avoidance Mobile robot
APPENDIX B : PIC Program : Servo and Sonar Controller

;=========================================================================================================
;
; Servo and Sonar Controller
;
; PROJECT : PIC Program
;
; DESCRIPTION : This program controls a servo and sonar sensor for a mobile robot.
;
; INPUTS :
; 1. IR: Interrupt to signal the PIC (from the master PIC) to start distance measurement
; 2. PA0: Echo output from Sonar
; 3. P1 (P1.0 to P1.7): Signal to the servo motor
;
; OUTPUTS :
; 1. P26: USART TX pin used to transmit data to the master PIC (UART 0, not used)
; 2. PA3: Trigger input to the sonar
; 3. PA7: LED to indicate the sonar operation
;
; CRYSTAL FREQUENCY: 20 MHz
;
;=========================================================================================================

; INPUTS:

; Echo output from Sonar ---->
; P1.0: IR
; P1.1: PA0
; P1.2: PA1
; P1.3: PA2
; P1.4: Trigger input to Sonar ---->
; P1.5: IR
; P1.6: PA4
; P1.7: PA5
;
; COMM output (Servo) ---->
; P1.13: DC1
; P1.14: DC2
; P1.15: DC3
; P1.16: DC4

;=========================================================================================================

; list
list p:\16f873A.INC,0,L40 ; list directive to define processor
errlevel 1,-(305)
include 'p:\16f873A.INC' ; processor specific variable definitions
config _XT_osc = _BD7_OFF = _BD6_ON = _BD5_OFF = _BD4_OFF = _BD3_OFF = _BD2_OFF = _BD1_OFF = _BD0_OFF

; 'CONFIG' directive is used to embed configuration data within .s files.
; The 16F873A is designed to be a low-cost, high-performance microcontroller.
; See the data sheet for more information.

status_reg equ h'20'
ir_reg equ h'21'
pin11_reg equ h'22'
macro_reg equ h'23'
macro_reg2 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'
l equ h'2C'
j equ h'2D'
b equ h'2E'
result_1 equ h'2F'
result_2 equ h'30'
result_3 equ h'31'
result_4 equ h'32'
result_5 equ h'33'
result_6 equ h'34'

; 30 to 33 addresses used for averaging
; 34 to 37 addresses used for averaging
; 38 to 41 addresses used for averaging
; 42 to 45 addresses used for averaging
; 46 to 49 addresses used for averaging

30
count_4_times equ h'46'
value equ h'46'
average equ h'47'

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;CONSTANTS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

bank_0 macro
    bcf STATUS,0
    bcf STATUS,1
endm
bank_1 macro
    bsf STATUS,0
    bcf STATUS,1
endm
bank_2 macro
    bcf STATUS,0
    bcf STATUS,1
endm
bank_3 macro
    bsf STATUS,0
    bsf STATUS,1
endm

mx_delay macro macro_var1,macro_var2 ;this macro is used to give desired delay -for the required delay impact should be 
    ;given to the macro appropriately

local x_1
local x_2
movl macro_var1
movf macro_var1
movw macro_var1
movb macro_var1

x_1
decfsz x_2+1,f
goto x_1

x_2
decfsz x_1+1,f
goto x_2

endm

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
org h'00'
    ;When power is given to the pic program starts from this point
org h'04'
    ;When interrupts are occurred program automatically jumps to this point

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

intialize
    ; "intialize"

bank_0
    cli
    cli
    cli
    cli

bank_1
    movlw b'00000111'
    movwf ADCON1

    movlw b'00000001'
    movwf TETS
    movlw b'00000011'
    movwf TEC2
    movlw b'00000000'
    movwf TELSC

    bcf OPTION_REG,7
    cli
    movlw h'90'
    movwf INTCON

    cli

bank_0
    cli

; cli
mainloop

begin
    INTCON.0IE ; check for RB0 interrupt
    mov
    bcf
    INTCON.0IE ; If so, break to ISR routine
    goto mainloop

; ; ...........................................................................
; ; Interrupts
; ; Interrupts Used in this ISR routine [unpoll.]: RB0(IE0) External Interrupt
; ; ...........................................................................

interrupt
    movf w_save ; save current W register contents
    movf STATUS,w ; move status register into W register
    movf status_save ; save off contents of STATUS register
    movf PCLATH.w ; move pclath register into W register
    movf pclkch_save ; save off contents of PCLATH register
    btfsc INTCON.INTB ;External interrupt(RB0)?
    goto initialize
    goto rb0_int

rb0_int

begin
    INTCON. INTB ; flag cleared

    clrif value
    clrif average

; ; .............................................................................
; ; first_reading_initialization
; ; movlw b'30' ;
; ; movwf count_a_times

first_reading

begin
    movlw .88
    movwf serva_position

first_reading_delay

call serva_0_degree
    call ten_ms_delay
decfsz serva_position,f
goto first_reading_delay

call sonar_start
    movwf distance,w
    movlw value

    bcf STATUS,C
    rrf value,w
    bcf STATUS,C
    rrf value,w
    movf value,w
    addwf average,w
    incf count_a_times
    movlw b'34'
    subwf count_a_times,w
    btfsc STATUS,1
    goto first_reading
    mov
    movlw average,w
    movlw result
    clrif average
    goto second_reading

; ; .............................................................................

second_reading

movlw .90
movwf serva_position

second_reading_delay

call serva_0_degree /
    call ten_ms_delay
decfsz serva_position,f
goto second_reading_delay
Collision Avoidance Mobile robot

```
call sonar_start
movf distance,v
movf value
bset STATUS,C
rref value,3
bset STATUS,C
rref value,3
movf value,v
addwf average,v

incf count_4_times
movlw 1'h34'
subwf count_4_times,W
btfsc STATUS,Z
goto second_reading
nop
movf average,v
movwf result_2
clef average
goto third_reading

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

third_reading
movlw 1'0a'
movwf servo_position

third_reading_delay

call servo_90_degree
call ten_us_delay
decfsz servo_position,f
goto third_reading_delay

call sonar_start
movf distance,v
movf value
bset STATUS,C
rref value,3
bset STATUS,C
rref value,3
movf value,v
addwf average,v

incf count_4_times
movlw 1'h0c'
subwf count_4_times,W
btfsc STATUS,Z
goto third_reading
nop
movf average,v
movwf result_3
clef average
goto fourth_reading

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

fourth_reading
movlw 1'0a'
movwf servo_position

fourth_reading_delay

call servo_135_degree
call ten_us_delay
decfsz servo_position,f
goto fourth_reading_delay

call sonar_start
movf distance,v
movf value
bset STATUS,C
rref value,3
bset STATUS,C
rref value,3
movf value,v
addwf average,v

incf count_4_times
```
EN407: Robotics

Collision Avoidance Mobile Robot

```
;--------------------------------------------------------------------------;
; fifth_reading
;--------------------------------------------------------------------------;
;--------------------------------------------------------------------------;
```
EN407: Robotics

Collision Avoidance Mobile robot

call delay ; delay for the count assigned above
bcf STATUS,R
movf echo_length,W
sblw .255 ; (L-W)->W
; check whether if echo length is for the maximum echo time
; (case as no obstacle in the 4 s range)
btocc STATUS,R
goto sonar_end ; echo_length is for maximum range(52)
neg
btocc PORTA,0 ; echo_length is in between dead zone(16) and max range(52)
goto sonar_echo ; YES: Then repeat the above statements till it is 00
goto sonar_end ; NO: Echo o/p is 100
sonar_end
movf echo_length,W ; THIS PORTION HAS TO CHANGE
movf distance,
return

;************************************************************************************
delay
movf count,W
movf temp,
as
decfsz temp,O
goto xx
return

;************************************************************************************
// SERVO MOTOR control
;************************************************************************************

//servo_0_degree ;LEFTMOST
bcf PORTC,0
as_delay .01..255 ;0.2ms delay
bcf PORTC,0
return

//servo_45_degree ;45 FROM LEFT
bcf PORTC,0
as_delay .17..255 ;1.3ms delay
bcf PORTC,0
return

//servo_90_degree ;MIDDLE
bcf PORTC,0
as_delay .51..201 ;1ms delay
bcf PORTC,0
return

//servo_135_degree ;45 FROM RIGHT
bcf PORTC,0
as_delay .77..255 ;1ms delay
as_delay .77..255 ;1ms delay
as_delay .77..255 ;0.1ms delay
bcf PORTC,0
return

//servo_180_degree ;RIGHTMOST
bcf PORTC,0
as_delay .77..255 ;1ms delay
as_delay .77..255 ;1ms delay
as_delay .77..255 ;0.1ms delay
as_delay .77..255 ;0.4ms delay
bcf PORTC,0
return

;************************************************************************************
tem as_delay
movf .12
movf count_high
movlw .250
movf count_low
tecn
decfsz count_low,f
goto ten ;Inner loop
decfsz count_high,f
goto ten ;Outer loop
return
EN407: Robotics

Collision Avoidance Mobile robot

;*******************************************************************************;
;  USART7 Transmission to the Master PIC inside ISR routine;
;*******************************************************************************;

data_transmit

begin

movlw .0
; 9600 Baud (from tables for 16 MHz Crystal with BRG=0(LOWFREQ))

movwf SPSR8

movlw 'b'01000000'
; X/8-bit/TX Enabled/8-nos/0-Low Speed/1-Read only/Parity

movf 7C08A,WF
;Serial Port Enabled

call delay_TX
;Give some time

movf result_1,0
; distance related to 0 degree (LEFT)

call delay_TX
;Give some time to transmit the byte

movf result_2,0
; distance related to 45 degree (45 FROM LEFT)

call delay_TX
;Give some time to transmit the byte

movf result_3,0
; distance related to 90 degree (RIGHT)

call delay_TX
;Give some time to transmit the byte

movf result_4,0
; distance related to 135 degree (45 FROM RIGHT)

call delay_TX
;Give some time to transmit the byte

movf result_5,0
; distance related to 180 degree (RIGHT)

call delay_TX
;Give some time to transmit the byte

data_transmit_over

goto interrupt_end

;*******************************************************************************;
;
; interrupt_end
;*******************************************************************************;

result

begin

decfsz i,1

goto begin

decfsz j,1

goto begin

decfsz k,1

goto begin

return

;*******************************************************************************;
;
;  INTERRUPT.EXIT
;*******************************************************************************;

interrupt_end

begin

bf INCOM_INTF
;Flag cleared

bf COM7,? ; LED OFF to indicate exit of Sonar mode of operation

nop

movf pclamb_save,m
; retrieve copy of PCLAMB register

movf PCLAM
; retrieve pre- isr PCLAM register contents

movf status_save,m
; retrieve copy of STATUS register

movf STATUS
; retrieve pre- isr STATUS register contents

movf w_save,i

movf w_save,o
; restore pre- isr Y register contents

retfie ; return from interrupt

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

end