

MAE 576 Mechatronics

Final Project Spring 2003

Group F

Amit Kumar
Carlos Lollett
Garth Mathe
Sameer Patwardhan
Uma Sharma Shankar

State University of New York at Buffalo

Buffalo, NY

Table of Contents

Section	Page
1. Abstract	2
2. Introduction	3
3. Hardware and Equipment	4
4. Pathway Configuration	10
5. Base Station	10
5.1 RF Communication	11
5.2 Human Interface	12
5.3 Main Algorithm	13
6. Mobile Robot	14
6.1 Drive Subsystem	16
6.2 Sensor Subsystem	18
6.2.1 Ultrasonic Range Finder	18
6.2.2 Standard Servo	18
6.3 Calibration	19
6.4 Modes of Operation	20
6.4.1 Manual Mode	20
6.4.2 Automatic Mode	20
7. Combined System	21
8. Problems Faced and Comments	26
9. Conclusions	28
10. Possible Future Applications	28
11. Appendix	29
11.1 Base Station Code	29
11.2 Mobile Robot Code	32

1. Abstract

The number of autonomous mobile robots being used for specific tasks increases every year. Giving these robots certain capabilities to carry out various tasks is and will be a major area of research for many years to come.

For this project a mobile robot was designed which will autonomously find its way through a nonlinear, walled path. The robot uses an ultrasonic sensor to find the distance to a possible obstacle. Communicating between the robot and a base station is accomplished using an RF signal. The base station gives the mobile robot information it needs to complete its task of negotiating the pathway. Both the mobile robot and the base station use the Basic Stamp microprocessor as their “brains”.

Using simple computer programs and basic electronic circuit design, a fully functional and robust path following robot was designed and constructed. Several problems encountered during the design process were: dealing with the inconsistencies and difficulties of calibration of servo motors, overcoming the faults of ultrasonic sensing, and working around difficulties dealing with 2 way RF communication.

Possible areas of continuation of this project are: Maze solving (intelligent) abilities, location mapping by use of the microprocessor, and finding ways to make the 2 way RF communication process faster.

2. Introduction

The goal of this project was to examine the development and implementation of a distributed sensing and control framework for a system comprised of a mobile robot (BOE-BOT kit) and a base station (StampWorks kit). The two systems were to be coupled together by wired or wireless communication channels. Extensive freedom was given in the design of the mobile robot and base station system.

We in group F decided to design a system where a mobile robot autonomously would find its way through a walled path of arbitrary configuration. It was decided that the robot and the base station would use RF communication due to its practicality. An ultrasonic sensor was used to give the robot information about the distance between the robot and a potential obstacle.

The purpose of the base station was primarily to give an observer information on what the robot was experiencing. A LCD was used to display messages sent between the base station and the mobile robot. This allowed an observer to completely understand what the robot was doing and why it was doing it. The base station was also used to give several instructions to the robot while the robot traversed its course.

Two modes of operation were implemented for the robot. One mode was the autonomous mode previously mentioned and the other was a “dumb” user controlled mode. In the “dumb” mode the user controlled the motions of the robot through a 16 button keypad. The user control is very similar in concept to that of a toy RC car.

3. Hardware and Equipment

Parts/Equipment	Quantity	Source
Basic Stamp – StampWorks Kit	1	Parallax
Basic Stamp – Boe-Bot Kit	1	Parallax
8-Bit Remote Control Combo Package	1	Reynolds Electronics

SRF04 Ultrasonic Range Finder	1	Reynolds Electronics
Keypad	1	Jameco

StampWorks Kit

The Stamp Works Kit, using mainly the NX1000 board, 2x16 LCD display and the Parallax standard servo.



Figure 1 StampWorks kit

Boe-Bot Kit

An assembled robot was provided using the parts of the kit.

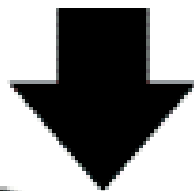
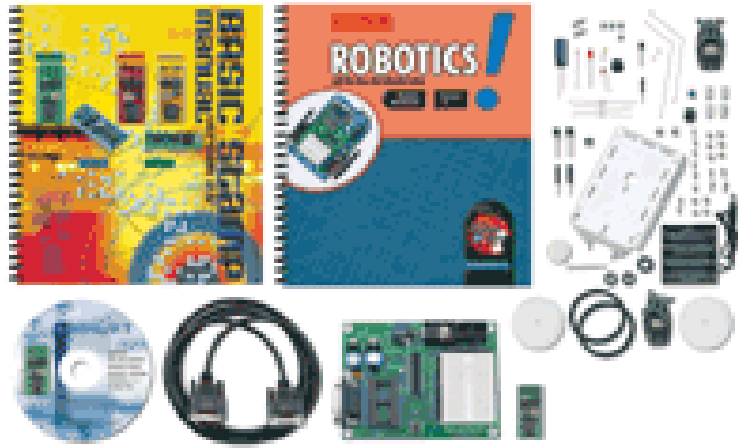


Figure 2 Boe-Bot kit

8-Bit Remote Control Combo Package

Two 8-Bit Remote Control Combo Packages were used, one from the Boe-Bot Kit and one from Reynolds electronics. From each kit these parts were used:

- [2] TWS-ANT 433MHz stud-mount, whip style antennas.



Figure 3 Whip style antenna

- [1] TWS-434A transmitter module. The transmitter sends a 433.92MHz signal using AM modulation. The signal power at 5V is 14dBm. Its pinout:
 1. Ground
 2. Data input
 3. Vcc
 4. Antenna

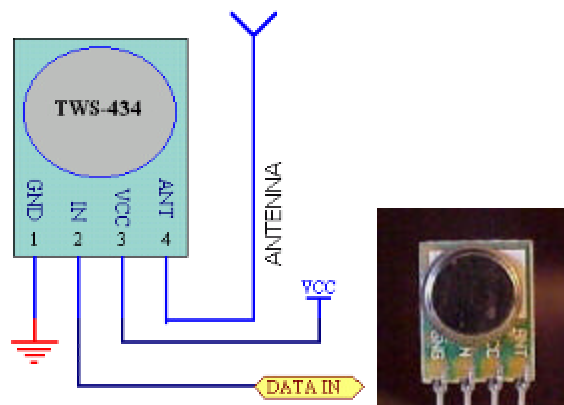


Figure 4 Transmitter schematic and picture

- [1] RWS-434 receiver module. The receiver is an AM demodulator with 1MHz of Bandwidth. Its pinout:
 1. Ground
 2. Digital data output
 3. Linear output
 4. V_{cc}
 5. V_{cc}
 6. Ground
 7. Ground
 8. Antenna

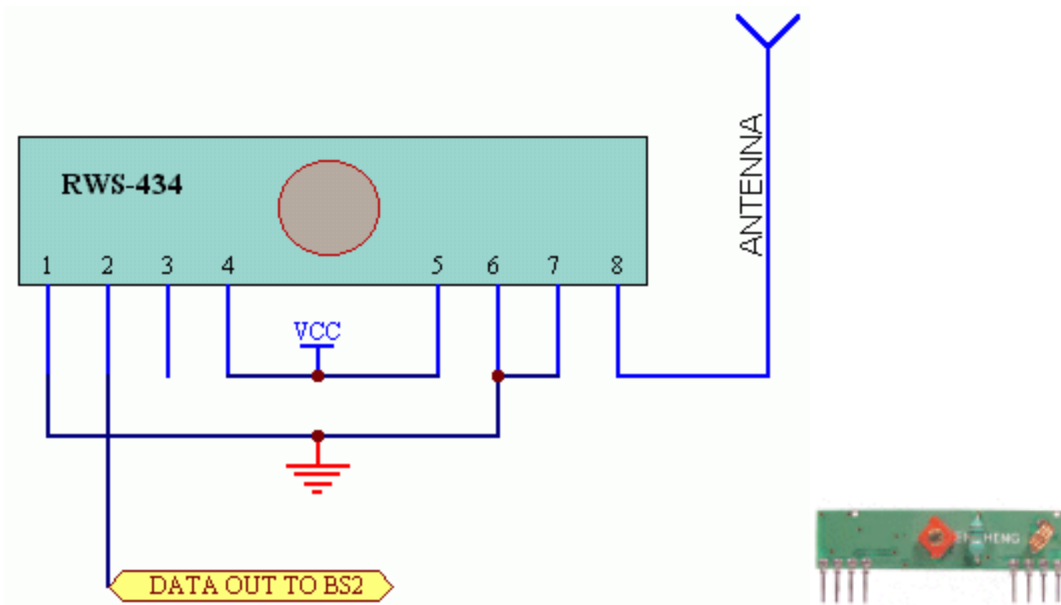


Figure 5 Receiver schematic and picture

The transmitter and receiver were used to create a bi-directional RF communication link between the base station and the mobile robot.

SRF04 Ultrasonic Range Finder

The Devantech SRF04 Ultrasonic Range Finder transmits a cone shaped ultrasonic pulse from the ranger. The reflected sound wave returns to the ranger from any object in the path of this sonic wave. The range of the sensor is 3cm-3m.

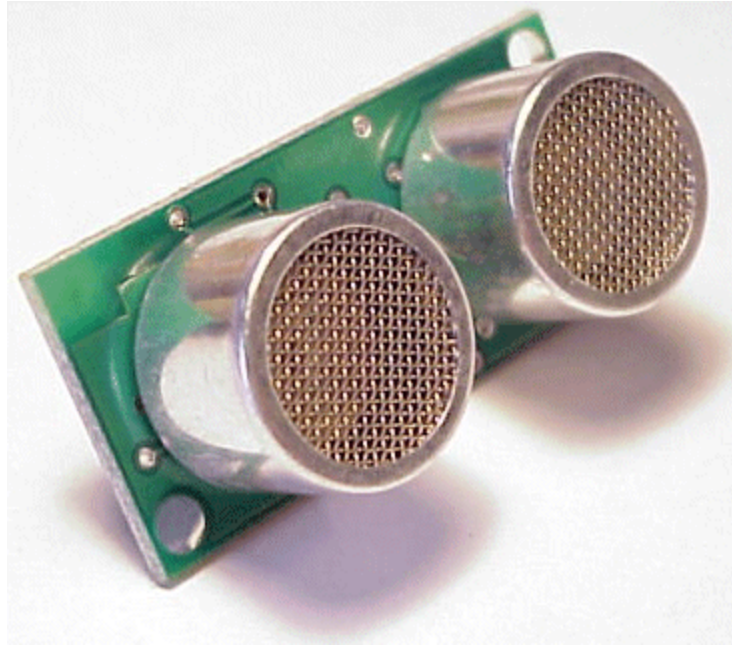


Figure 6 SRF04 ultrasonic range finder

4x4 Keypad

Standard 16 button keypad of 4 rows and 4 columns, easy to interface with any microcontroller.



Figure 7 Keypad

Parallax Continuous Rotation Servo

The servo is capable of 360 degrees of rotation, used as electromechanical device to move wheels. It has standard Futaba configuration



Figure 8 Parallax continuous rotation servo

Parallax Standard Servo

The servo's movement range is 180 degrees, used as electromechanical device for basic movements (Used in this project to create a distance scanner). It has the standard Futaba configuration.



Figure 9 Parallax standard servo

4. Pathway Configuration

The path walls were constructed of polystyrene blocks measuring 7 inches in height and 24 inches in length. The walls were faced with corrugated paper in order to allow the ultrasonic sensor to “see” them better. Multiple wall blocks were used so many different path configurations could be tested. For simplicity the turns were constrained to 90 degrees either left or right. Figure 10 shows one of the pathway configurations tested. Note the corrugated “anti-stealth” wall covering.

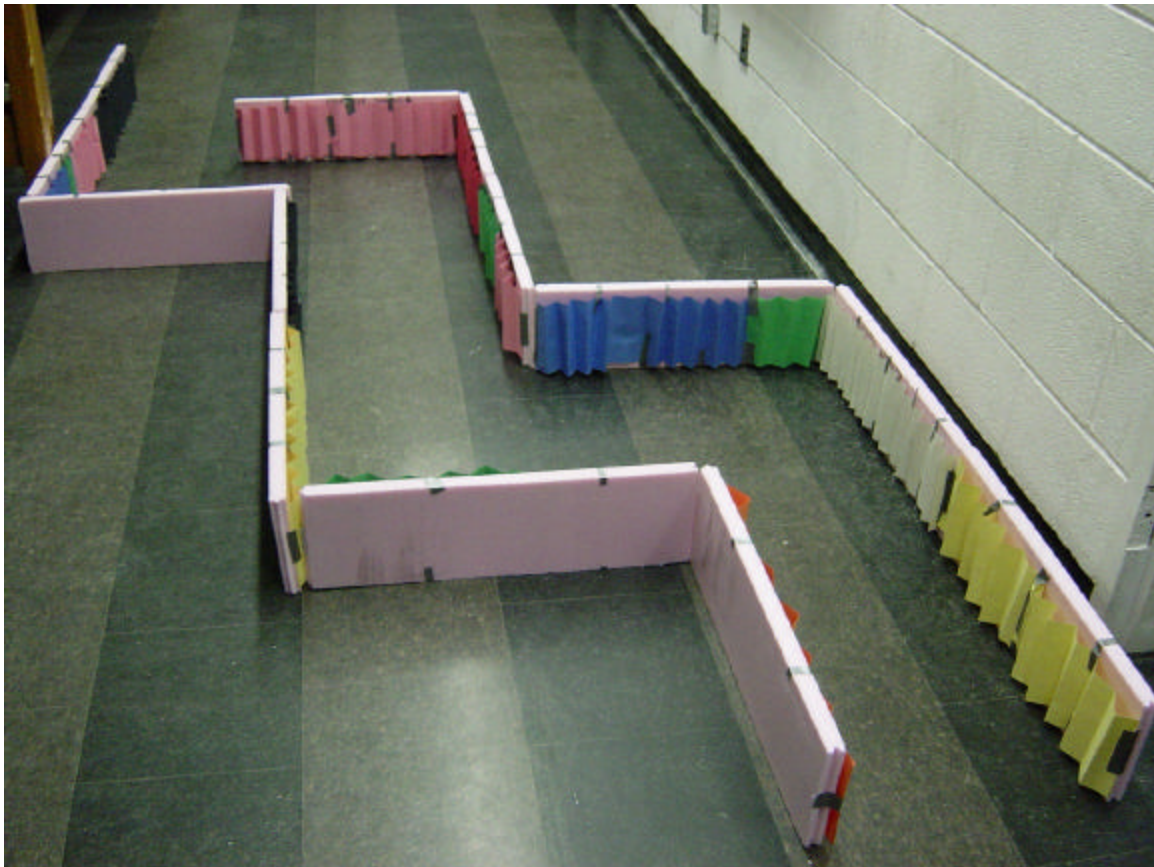


Figure 10 Pathway with 2 right turns and 2 left turns

5. Base Station

The StampWorks board NX1000 is main component of the Base Station. The keypad is used as an interface to introduce commands for the mobile robot. The LCD display is used to show the communication with the Robot. The main subsystems of the Base Station are:

1. Main Processor: The basic stamp and its I/O ports. They are used to run the algorithm which drive the base station
2. RF Communication Subsystem. Its use an 8-Bit Remote Control Combo Package to let the Basic Station send and receive information to and from the Robot.
3. Human interface Subsystem. Includes:
 - a. Input. Keypad to receive order from a user.
 - b. Output. LCD To show messages related to the process, specifically the communication between the Base Station and the Robot

The purpose of the Base Station is guide the robot by human entered command to the entrance of the tunnel. After entering the tunnel the Base Station will alert the user to the obstacles the robot sees and actions that the robot is doing. Figure 11 shows a photo of the base station with its various appendages.

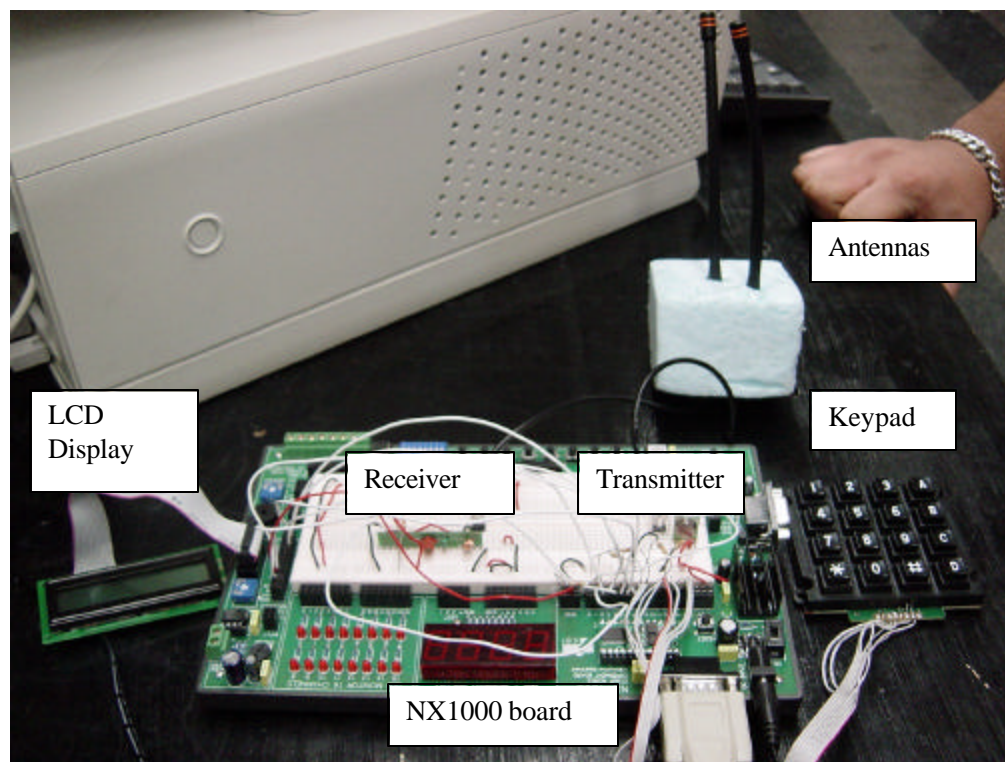


Figure 11 Base station

5.1 RF Communication

The Base Station RF communication section uses an asynchronous communication between the Base Station and the Robot. Since Transmitter and Receiver

channels are at the same frequency, a different code start string was added to the messages of Base Station (“A”) and the Robot (“B”). One leading byte was added to the messages to make easier its catching sequence by the receiver. The transmission is executed by SEROUT command:

Serout RF outport, baudrate, [leading byte, Station Code, Message1, Message2,...]

One or more messages can be sent with each transmission String. The RF input is synchronized to receive messages from specific code:

Serin RF inport, baudrate, [WAIT (Station code), Message1, Message2,...]

For both communications the baud rate was 9600 bauds.

5.2 Human Interface

The keypad is used to introduce commands into the Basic Stamp. The keypad has four columns and four rows, the algorithm scan:

```

scan:
for i=12 to 15          ' For very row assigned to ports
12-15
  high i                ' set the row to 5V
  keypad=inC            ' Check all the columns(ports 8-11)
  low i                 ' reset the row to 0V
  if keypad> 0 then gotkey ' If any column is set means
some button was pressed so go to decoder routine
  next
  return
gotkey:
lookdown keypad,[0,1,2,4,8],keypad ' assign the number 0-4 and then
code the keypad accordingly to get a integer number
  keypad=((i-12)*4)+keypad ' for each button is an unique
integer 1-16 that is returned when that button is pressed
  RETURN

```

A keypad is a network of switches that connect each column to each row. The keypad used has 16 buttons, or switches. When a button is pressed a switch is closed making a direct path between row i and column j. The scan algorithm Set 5V to a specific row i and check if there is a 5V at any column j, if there is 5V at column j means there is a direct electric path between i and j, therefore the button in row i and column j was pressed.

Additionally, a command protocol was assigned to the Base Station and interpreted in the Robot.

1= Turn 45 degrees to the left

2= Go Forward

3= Turn 45 degrees to the right

4=Turn 90 degrees to the left

6= Turn 90 degrees to the right

8= Go Backwards

9= Go to automatic mode

The LCD display is used to show the information about the operation mode (manual or automatic) as well as the communication between the Base Station and the Robot during the automatic mode.

5.3 Main Algorithm

There are two modes of operation: the manual mode and the automatic mode.

- In the manual mode the base station keeps uses a loop to check if the user presses any keypad button. When a button is pressed the base station encapsulates the command in a message and sends this message to the robot to be executed. This routine keeps running until the user sends the command to change to automatic mode.
- In the automatic mode the base station waits until the robot sends a message reporting an obstacle. If that message arrives then the base station asks the robot to scan the environment and send it back the turning direction. The turning direction is encoded as an integer 0-10. The integer value represents an angle from -90 degrees to 90 degrees in increments of 18 degrees.

The circuit diagram for the base station can be seen in figure 12.

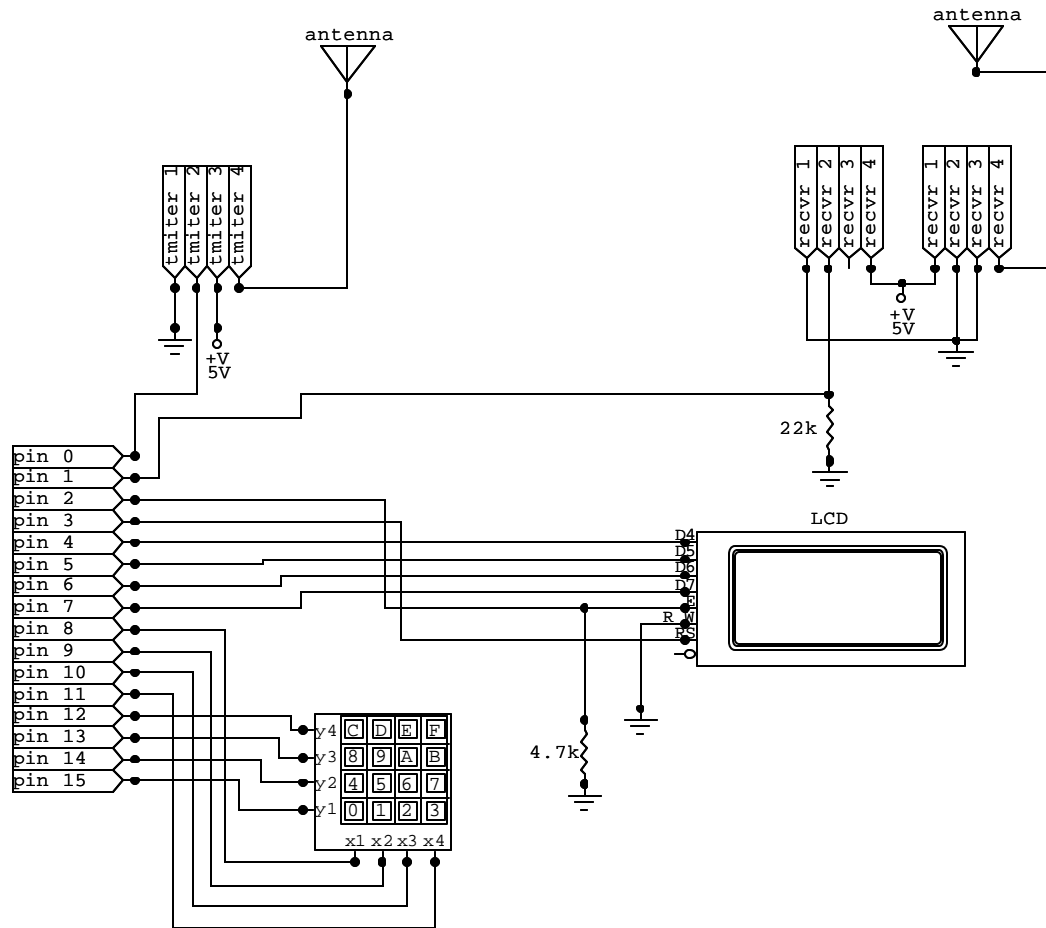


Figure 12 Base station circuit diagram

6. Mobile Robot

The Boe-Bot Kit (figures 13 and 14) with some additional components had the robot role. The robot, a tricycle model, has 3 wheels, one powerless at the rear (white ball-wheel, see figure 13) and two servo-powered disk-type wheels at the front. The main subsystems in the mobile robot are:

1. Main Processor: The basic stamp and its I/O ports. They are used to run the algorithm which drive the robot
2. RF Communication Subsystem. Its use an 8-bit remote control combo package to let the base station send and receive information to and from the robot.

3. Drive Subsystem: The two Parallax continuous modification servos are connected to the disk wheels and they drive the wheels according the pulses sent by the Basic Stamp.
4. Sensor Subsystem. The Devantech SRF04 ultrasonic range finder senses the distance between the robot and the closest object in the direction that the sensor is pointing. Additionally, the sensor is placed on the Parallax standard sensor that allows the sensor to be move from -90 to 90 degrees with 0 degrees directly forward.

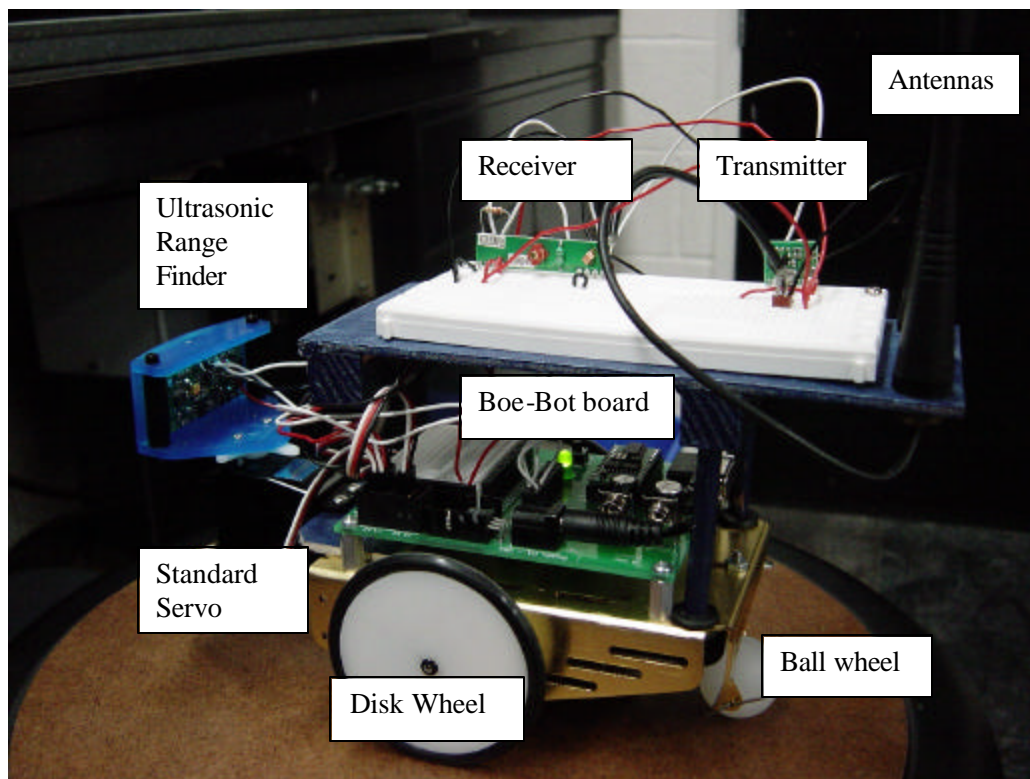


Figure 13 Mobile robot left side view

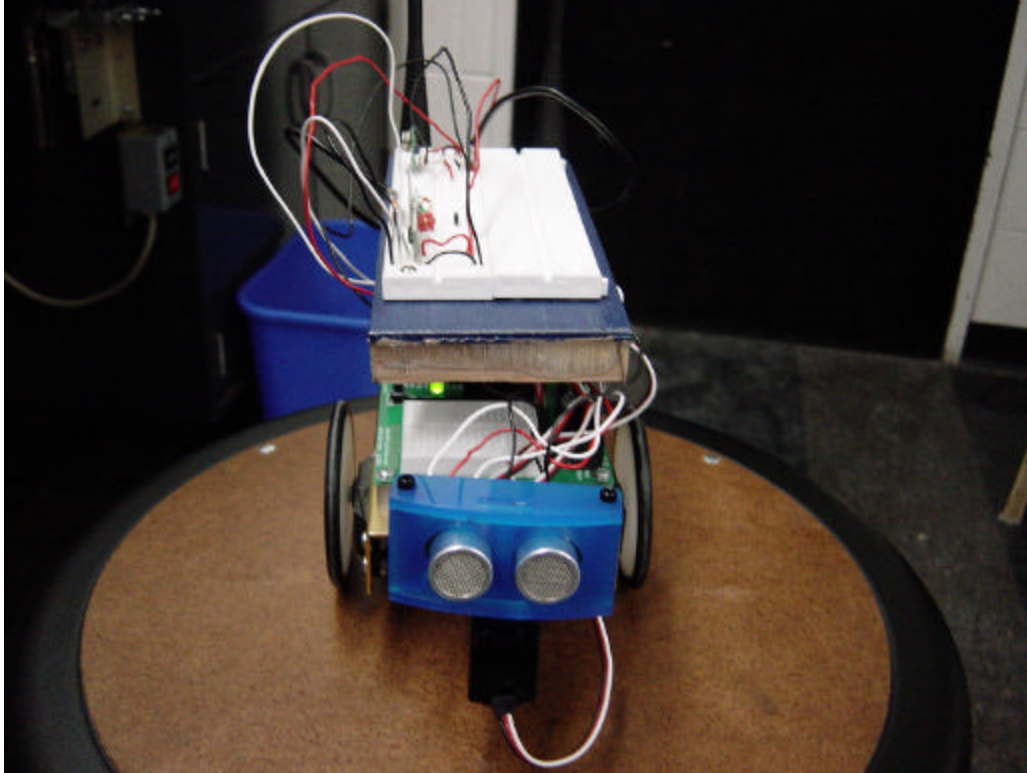


Figure 14 Mobile robot front view with SRF04 ultrasonic range finder

6.1 Drive Subsystem

The drive system has two 360 degrees rotational servos. The servos are driven by the Basic Stamp activity board from the Boe-Bot kit. The servos are driven by a V_{dd} source of 6 volts (from the batteries of the Boe-Bot), and controlled by a pulse width modulation command (PWM), from an I/O pin. The Boe-Bot board contains servo-port connections (figure 15) for ports 12-15. For the wheel servo, ports 12 (left) and 13 (right) were used.

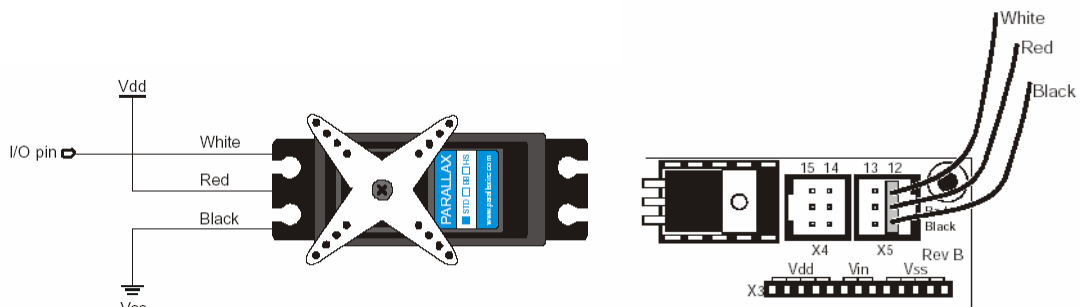


Figure 15 Rotational servo and its Boe-Bot board connections

By controlling the movement of both servos basic movements can be made:

- Forward
- Backward
- Turns

Since the wheel's rotations are not sensed, a feedback control system is not used for positioning. The servo movements were characterized and through proper calibration, various movements could be executed. The servos required pulses every 2ms (1000 units of 2μsec), to be driven. This value drives the servo to move fast clockwise or counterclockwise. The middle point ('dead point') of the servo range was used to calibrate the servos. The PWM command was used with various amplitudes to obtain the necessary motions. The PWM signal sent to a servo is the sum of its dead point plus its amplitude for the chosen movement. Both servos have different dead points and their linear constants differ. After a process of calibration, the following values were obtained:

Movement	Left Dead Point	Left Amplitude	Right Dead Point	Right Amplitude
Forward	620	190	735	-200
Backward	620	-190	735	200
Turn Left	620	-190	735	-200
Turn Right	620	190	735	200

It was also necessary to determine the amount of pulses necessary to make a certain motion. This feature is critical in the turns where the number of PWM pulses determines the turn angle. To obtain a turn angle of 18 degrees, 11 and 7 pulses were assigned to turn left and right respectively. Carrying out a basic movement turns the robot at an angle $\Theta \in \{0,36,54,72,90\}$ to the left or right. For example to turn right:

```
TURNRIGHT:
  for j=0 to anglesteps
    for i=1 to 7
      pulsout 13, RightDeadpoint+200
      pulsout 12, LeftDeadpoint+190
    next
  pause 100
next
RETURN
```

The basic movement of turning to the right 18 degrees is repeated according the angle step required.

6.2 Sensor Subsystem

The sensor subsystem uses two main parts:

6.2.1 Ultrasonic Range Finder

The sensor pulse trigger input, connected to the pin 0 of the BasicStamp, receives a ping request from the BasicStamp. An ultrasonic pulse is sent out by the range finder, which then waits for its pulse to return. The RCTIME command is used to measure the time taken for the pulse to return. The distance to an obstacle is then estimated based on the RCTIME response time. The routine used:

```
sr_sonar2:
    pulsout INIT, 5
    output INIT
    RCTIME ECHO,1,wDist
    wDist=wDist/convfac
    pause 10
    return
```

6.2.2 Standard Servo

In order to explore more than one direction the sensor was attached to a standard servo (BasicStamp servo-port 14) which can move in range of 180 degrees, enough to explore the front and the sides of the robot's path. Different from rotational servos, the PWM determines the angle of the servo instead of the strength or velocity of the movement. This servo was characterized to rotate at the angle $\Theta \in \{0,36,54,72,90\}$ to the left or right. Using this routine:

```
for j=0 to 10
  for i=1 to 200
    pulsout 14,240+(j*90)
  next
next
```

The exploration range includes 11 angles (from 90 right to 90 left in 18 degree-steps). Each angle is kept for 200 iterations to give stability to the servo, and the PWM is dependent of the desired angle step j . In the robot this servo has two sensor scanning routines: one complete scanning routine to detect the best path to continue (after the robot has detected an obstacle and stopped), and a smaller scanning routine to detect obstacles while in motion.

6.3 Calibration

There are basically two servos and they run in opposite direction if we want the Robot to go in one direction. The two servos need separate calibration and their sense of rotation in any particular direction is totally opposite.

For the left wheel servo, the center lies at the point 620 Pulse width for pulseout command. Whether for the right servo the center lies at the point 735. Center means the point of at which the servo stops rotating and then changes the turn direction. Certain issues in the calibration were to get the exact movement of the two servos in order to move in any direction like forward or backward. For this kind of calibration we measured the revolutions of the servo for particular time. Then depending on the revolutions we actually got the idea how fast the servos are moving for a particular pulse width.

Then the real test was moving this robot on the actual ground and then tests the calibration. By testing both the servo movements on the ground and with minor corrections finally the servo moments were calibrated.

The most difficult calibration was the servo from parallax used for the Ultrasonic sensor. This issue is well described in the 'problems faced and comments' section. In the calibration of this servo also the basic aim was to find the center point and then with respect to it find the accurate movement of the servo. As this servo was the basis of the movement of the Ultrasonic sensor, the calibration of it was an important task. We found the center of this servo at 652 and the extremes were 0 and 1000. Also we calibrated it in order to scan 180° divided in 18° parts. The 18° was selected because at these points the servo was really stable. To have the accurate positioning of the servo and to avoid the averaging error we provided the pulse to the servo 150 to 200 times to achieve the exact position. These points at the intervals of 18° were found out by trial and error.

The next issue was synchronization of the directions of the robot and the sensor servo. We find the direction based on the Ultrasonic sensor and that is mounted on the Parallax servo. Let's assume after scanning, the robot needs to turn in 72° to left from the current position. The angle is provided with respect to the sensor servo. Now to turn in exactly that direction there needs to be synchronization in between the two calibrations. We numbered the directions scanned by the sensor from 0 to 11 with 0 on the extreme right. We passed the direction number to the turning subroutines and depending on that

number we turned the robot in the direction of interest and the synchronization of the directions really need great amount of calibration and also actual testing on the ground. We used angular scales and rulers to calibrate the synchronization.

6.4 Modes of Operation

6.4.1 Manual Mode

The robot acts as an R/C car waiting for the commands from the base station. Basic movements include: forward, backward, and turns (90 degrees and 36 degrees, to the left or right).

6.4.2 Automatic Mode

When the Robot is in manual mode and receives a '9' command from the keypad, it changes to automatic mode, then start to explore forward in movement steps followed by small scanning of 36 degrees(-18 degrees to 18degrees , with 0 degrees directly forward). If an obstacle is found:

1. Obstacle detected report is sent to the base station
2. Large scanning (180 degrees) is made to determine the direction with greatest distance to an obstacle.
3. This direction is sent the base station
4. The robot advances forward to the sensor rotation axis. This is to compensate the difference between the robot wheel base position and sensor rotation axis.
5. The proper turn to the selected direction is executed.
6. Start to move forward and repeat the cycle.

The circuit diagram for the mobile robot can be seen in figure 16.

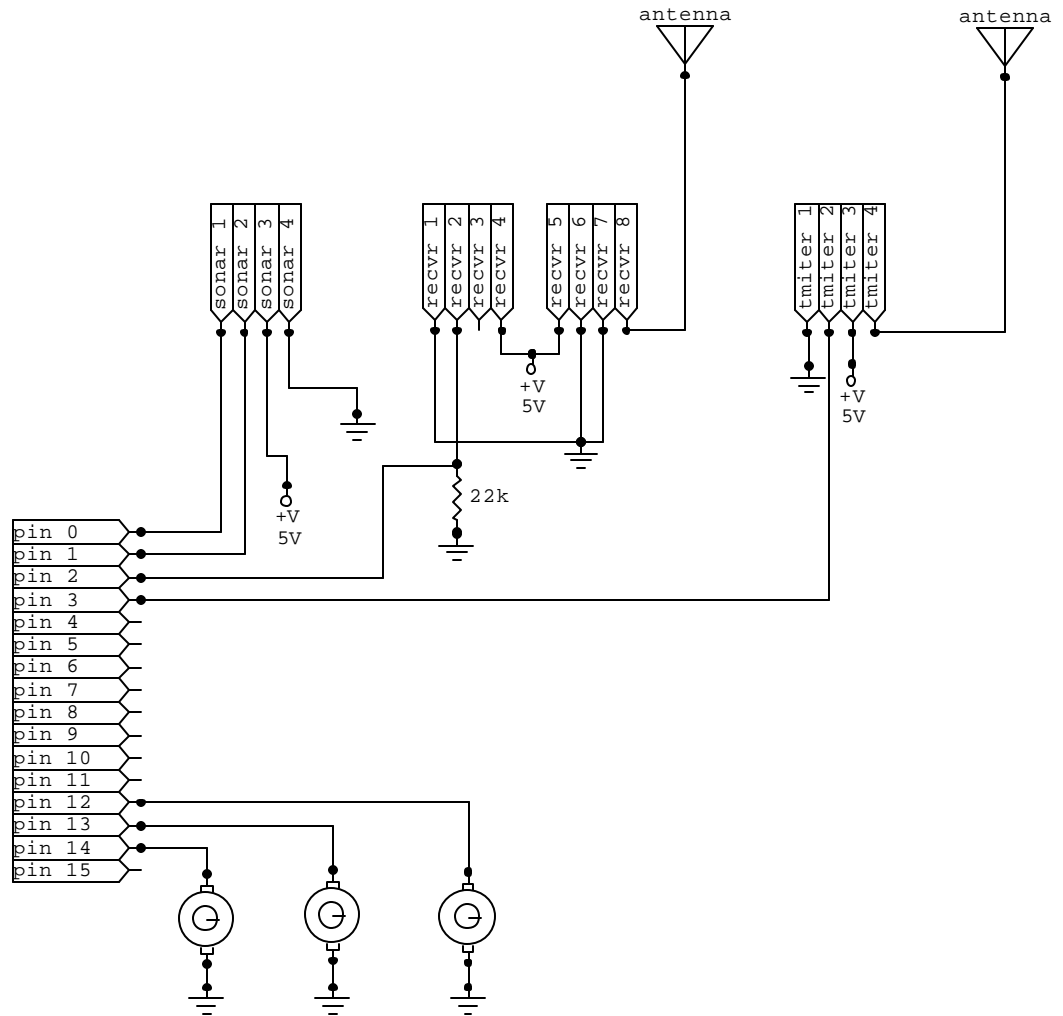


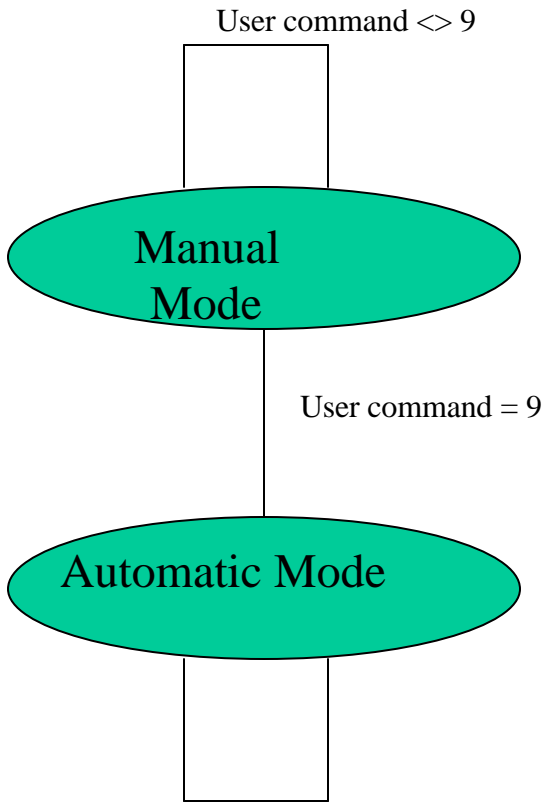
Figure 16 Mobile robot circuit diagram

7. Combined System

In this section the algorithm of the combined base station/mobile robot system will be examined in more detail.

A state diagram of the combined base station/mobile robot system can be seen in figure 17.

Base Station



Robot

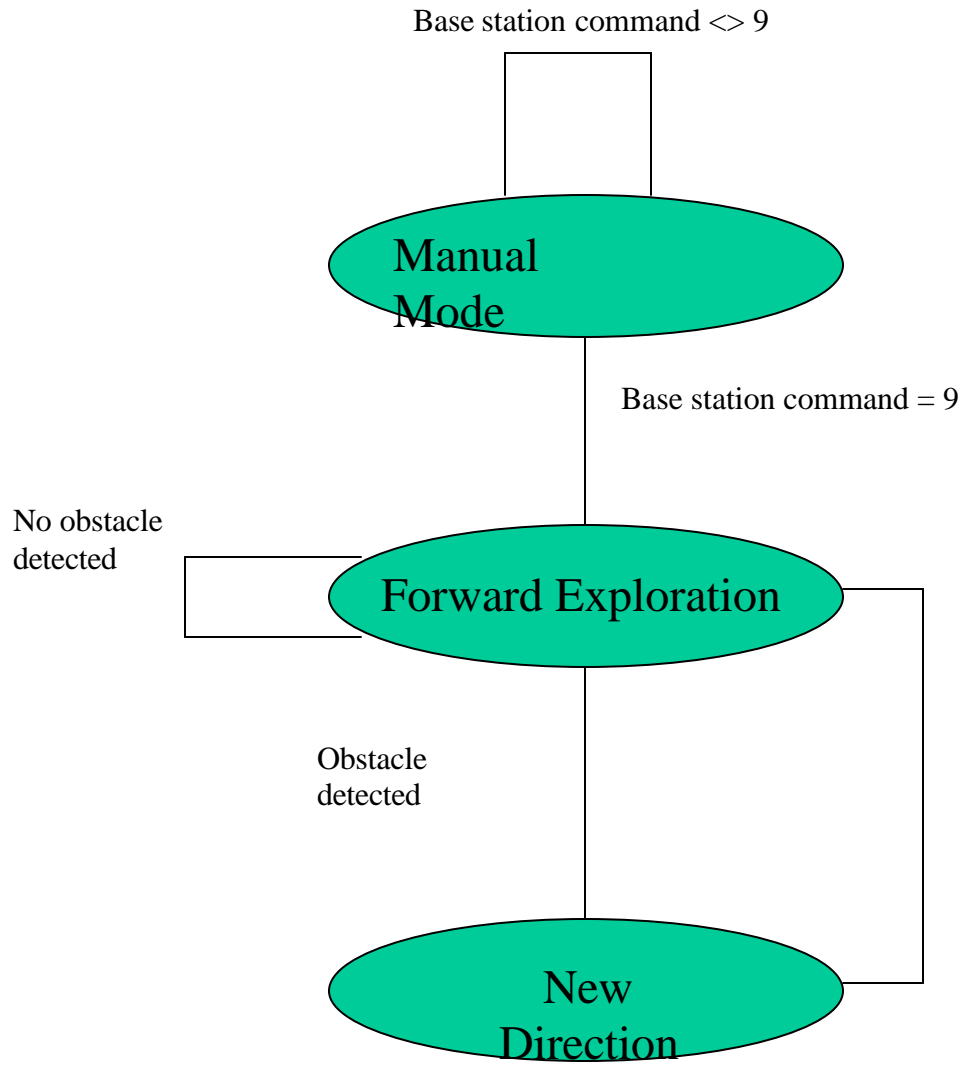


Figure 17 State diagram for the combined system

A flowchart of the combined base station/mobile robot system can be seen in figure 18.

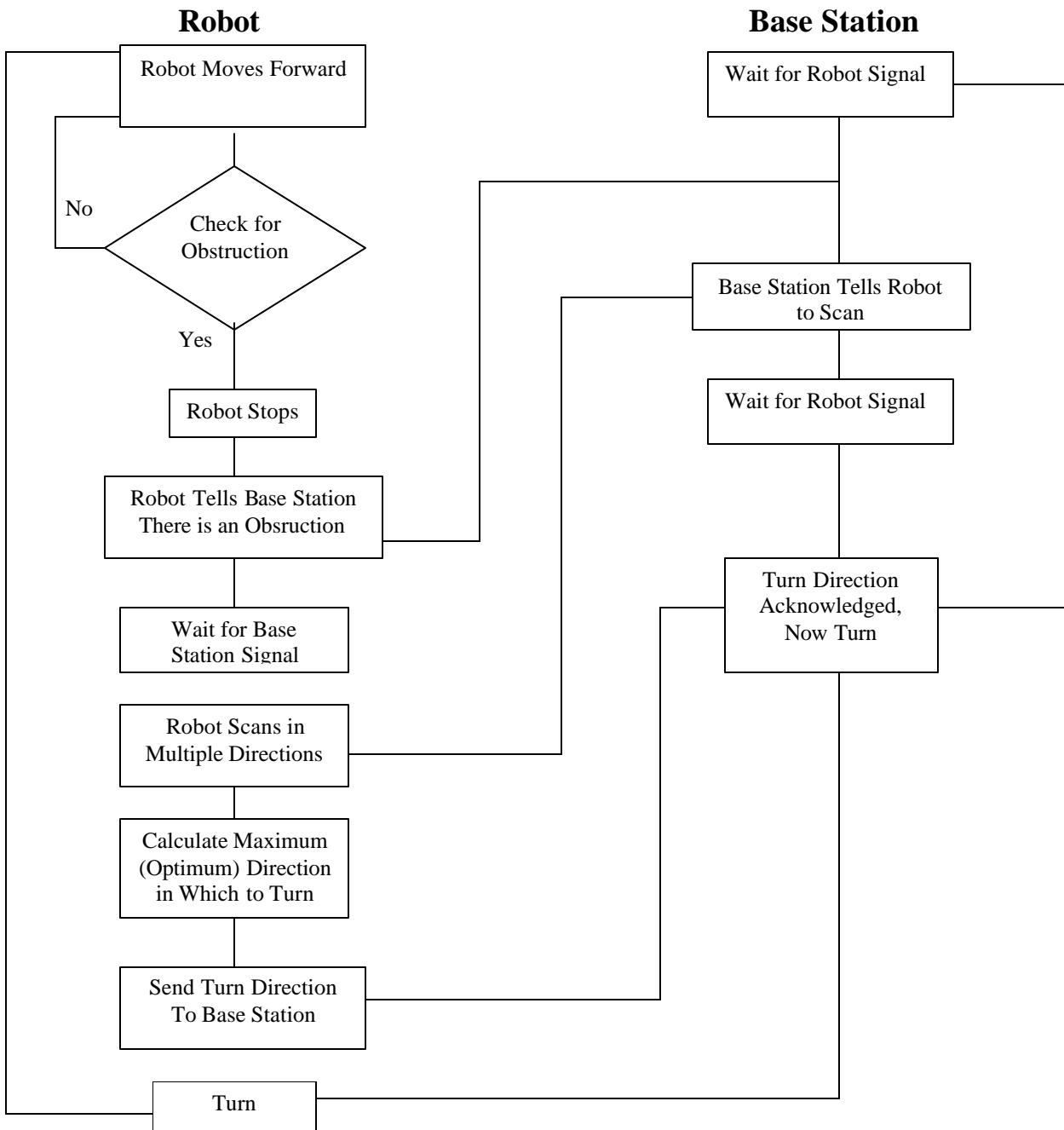


Figure 18 System flowchart

The mobile station here acts just like a Human body and the base station acts like a brain of the body. The mobile robot always asks base station for solutions and base station solves its problem. The flow chart above shows how the process of the communication works and also how the data transfer is taking place between the two stations.

Base station always waits for the signal from the mobile station. Mobile station or moving robot always tries to move forward till its path is obstructed by some obstacle. When the mobile station sense any obstacle using the Ultrasonic sensor it stops and then sends signal to the base station that it has the problem and it sees an obstacle ahead and it waits for the signal from the base station for the action to be taken. Then base station asks it scan the area ahead and decide the optimum turn direction based on the scan. Then Robot scans the area ahead and then depending on the distances scanned it decides the optimum direction to move forward. Then it sends back the direction to the base station and base station asks robot to move forward when it gets the optimum direction. Then again Robot moves forward in that direction till it gets the any obstacle again.

Here, the base station is directing the mobile robot. All the required abilities of movement like turning, moving forward-backward are in the mobile robot. Robot also has the ability to scan the area using the Ultrasonic sensor. Thus, the from the communication point of view it is very useful that Base station keeps track of the movements of the robot and also we can store the positioning of the robot in the EEPROM provided it is that big enough.

Following situation of figure 19 is kind of unsolved in our case but we have thought of the solutions for the same.

The situation is represented as follows:

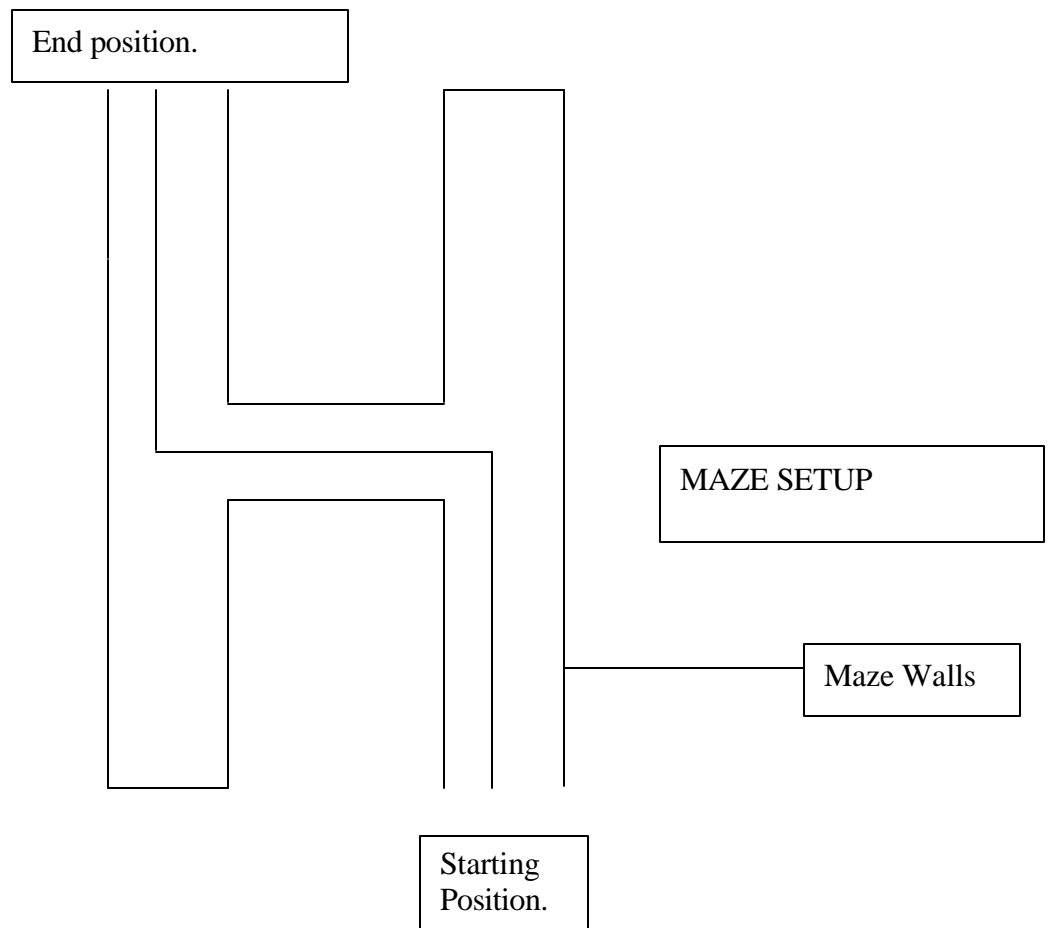


Figure 19 Maze solving situation

In the above situation of the maze, it is evident that from our algorithm our robot first will go to the end of the first straight pathway, and then will turn and then it will again follow the same path back to where it started. To avoid this and to make our robot to go in the correct path to the end position, it is necessary to check every time when it sense the obstacle ahead, whether it is dead end or else situation. In case of dead end

ahead it will reverse the path and again follow the same path again but in this time it will retrace the path with continuous scanning ahead of it in 180 degrees.

A solution for this problem is to keep track of the mobile robot with respect to the base station or some reference and so we can always check whether it is following the same coordinates and then we can check whether it is following the correct path by scanning the area in more depth.

Both these approaches are memory dependent and hence we have to take help of the third processor for these two solutions like a PC and serial communication with it.

Another solution is to use another Ultrasonic sensor for more accurate sensing and also in this case the third processor will help to deal with the fractions that arise in the in depth scanning. All these issues are relating to the future expansions in the project but these are mentioned here because they deal with the algorithm and its validity in case of many situations.

8. Problems Faced and Comments

In the project we faced many problems and we come up solutions also. Some of them were implemented and some of them were left for the future implementation or expansion. They are outlined as follows.

1. The first difficult task was the calibration issue. The calibration of the ultrasonic sensor servo was crucial, and with many difficulties we found out a way to avoid the averaging error by providing a pulse of the same duration for large number of times.
2. The calibration of the wheel servos was done first without actually putting the robot on the ground. But during the testing on the ground we found out that the same calibration does not work very well and so we made many changes to the pulseout values. Also the calibration related to the ground was a little bit unreliable.
3. The calibration of the sensor servo was toughest task because of the unreliability and very poor repeatability of the servo. The calibration also produced problems when we have to synchronize the calibration of this servo with the calibration of the servos used for the wheels.

4. The tire behavior was not least important. From the tire point of view we saw that the tires have some wear and because of that the distance traveled was changing a little bit.
5. The battery decay was also a problem and the solution to that can be the change in calibration of the servo depending on the time elapsed by the battery.
6. As far as the sensor was concerned, it was very good and also reliable. There was very good repeatability too. But the real problem was the surface from which the sound waves are getting reflected. First we just used foam as our reflecting surface and found that for acute angles it was working very poor and the results were just abnormal. We decided to change the surface of the foam by covering it with crumbled A4 size paper and it produced good results but still it was giving problems with acute angles and we wanted to have 0% error because the optimum direction was dependent on the scanned distances and we were choosing the maximum scanned distance as the optimum direction. So, we put zigzag edged surface on the foam and it worked really nice and almost perfect.
7. The next issue was the maze setup. At some points, we were getting some abnormal distances and we found out the solution to it as the proper distance between two parallel walls of the maze so that the scanned distances were with real meaning and we get perfect results and good image of the surfaces ahead.
8. We planned for the bidirectional RF communication using the 2 sets of receiver and transmitters. But the problem arises while synchronizing the two communications. We tried to put many subroutines, signals for various actions and protocols initially. We also planned for sending the array to the base station by the robot and then analyze it and send back the optimum direction back to the robot. But it was taking lot of time and also some times it was problematic that robot send signal prior or late when the base station actually waiting for the signal. The timing of both the communications was critical and due to these problems we changed the design and the optimum direction finding procedure was made to be the task of the robot and not the base station.

9. We also planned for storing the position coordinates of the robot with respect to the position of the base station and we were successful in storing them in the EEPROM partially as we ran out of the variable space and also the EEPROM. But using the third processor we can easily deal with this problem and then we can even retrace the path followed by the robot and also we can manually put the coordinates that robot should follow and this memory problem can be easily solved.

9. Conclusions

Initially, in designing an objective for this project, a more complex and intelligent system was imagined. The goal was to traverse a complex maze while mapping a database of potential turning options and past movements into the Basic Stamp's memory. The system was also to have some intelligence so the robot could choose an appropriate turning direction at a multiple path intersection. This proved to be too lofty of a goal for the timeframe given and the hardware available.

Our design deals with the initial steps needed to implement a more complex system such as the one described above. The robot was given the ability to identify walls and open pathways, which is a large and integral part of complex maze solving. In terms of the goals presented in the introduction (communication between the base station and the mobile robot, and the ability to autonomously follow a path) this project was a success. Building on the methods implemented in this project one could indeed solve the maze problem given more time and equipment, specifically memory for the maze mapping.

10. Possible Future Applications

- 1) Completion of the maze solving objective.
- 2) Addition of sensors so robot does not need to stop and scan every time it encounters an obstacle.
- 3) Addition of memory so the robot can remember where it has been and have an idea of where it needs to go.
- 4) Coordination of multiple mobile robots to solve the maze faster.

- 5) Different, smaller mechanical platform so the robot could deal with smaller pathways.
- 6) Implement a better ranging sensor so the walls won't need a corrugated wall covering.

11. Appendix

11.1 Base Station Code With Comments

```

'{$STAMP BS2}
E          CON      2          ' LCD Enable pin (1 = enabled)
RS         CON      3          ' Register Select (1 = char)
LCDout    VAR      OutB      ' 4-bit LCD data

ClrLCD     CON      $01      ' clear the LCD

char       VAR      Byte      ' character sent to LCD
index     VAR      Byte      ' loop counter
number    var byte          ' keypad variable to store the column
keypad    var byte
inserial  con 1          'RF input port
outserial con 0          'RF output port
synchA    con "A"      'Robot ID
synchB    con "B"      'Base Station ID
junk      con 126      'leading bits for RF
LETTERS   CON "S"
NUMBER1   CON 1
NUMBER2   CON 2
UV        var byte(11)  ' Ultrasound sensor measurement array
DAT1     VAR BYTE      ' Communication buffer
DirG     var Byte      ' Robot turn angle
DirG1    var Byte      ' Robot turn anglefirst digit
DirG2    var Byte      ' Robot turn anglesecond digit
pos1     var Byte      ' character "R" for right and "L" for left
J        var byte      ' index
i        var byte      ' index
DLY      con 1000      ' delay
Dirrecl  var Byte
'NUMBER1=%00000001
msg1     DATA "BASE STATION:",0,"WAITING",1 ' preload EEPROM with message
msg2     DATA "MOBILE STATION:",0," OBSTACLE DETECTED",1 ' preload EEPROM with message
msg3     DATA "BASE STATION:",0,"LOOK FOR DIRECTION",1 ' preload EEPROM with message
msg4     DATA "MOBILE STATION:",0,"DIRECTION ",1 ' preload EEPROM with message
msg5     DATA "BASE STATION:",0,"GO FORWARD",1 ' preload EEPROM with message
msg6     DATA "BASE STATION:",0,"MANUAL MODE",1 ' preload EEPROM with message

DirL = %11111001      ' setup pins for LCD
GOSUB LCDinit
' initialize LCD for 4-bit mode

index=msg6
GOSUB WriteLCD

' First go to Manual Mode

GOTO Manual

' Automatic Mode
Main:
index=msg1
GOSUB WriteLCD
'Waiting for Robot message about obstacle, its marked as P

```

```

FIRSTCOMIN:
    Serin inserial,16780,[WAIT(synchA),DAT1]
    IF DAT1="P" THEN FIRSTCOMOUT
    GOTO FIRSTCOMIN
FIRSTCOMOUT:

index=msg2
GOSUB WriteLCD

PAUSE DLY
index=msg3
GOSUB WriteLCD

' Command to initiate large scanning in Robot
Serout outserial,16780,[junk,synchB,LETTERS,1]
'-----
' Waiting for turn angle from the Robot
SECONDCOMIN:
    Serin inserial,16780,[WAIT(synchA),DAT1]
index=msg4
GOSUB WriteLCD
dirG=dat1

' Mapping 0-10 integers to 90R,72R,54R,36R,18R,0,18L,36L,54L,72L,90L

if dat1>=5 then posi
dirG=(5-dat1)*18
'char="R"
dirG1=dirG/10
dirG2=dirG-(10*dirG1)
pos1="R"
goto shownumber
posi:
dirG=(dat1-5)*18
pos1="L"
dirG1=dirG/10
dirG2=dirG-(10*dirG1)
shownumber:
debug ? dat1
char=pos1
    GOSUB LCDwrite
char=dirG1+48
    GOSUB LCDwrite
char=dirG2+48
    GOSUB LCDwrite

SECONDCOMOUT:
    PAUSE DLY
index=msg5
GOSUB WriteLCD
' Confirming order to go forward
    Serout outserial,16780,[junk,synchB,LETTERS,2]
'-----

GOTO Main
'-----

WriteLCD:
char = ClrLCD                ' clear the LCD
GOSUB LCDcommand
PAUSE 500
'index = Msg                ' get EE address of message

ReadChar:
READ index,char                ' get character from EEPROM
IF char = 0 THEN FirstDone    ' if 0, message is complete
IF char = 1 THEN MsgDone
GOSUB LCDwrite                ' write the character
index = index + 1            ' point to next character

```

```

    GOTO ReadChar                                ' go get it
FirstDone:                                     ' the message is complete
    char=128+64                                  ' go to next line
    GOSUB LCDcommand
        index=index+1
    GOTO ReadChar
MsgDone
    PAUSE 2000                                  ' wait 2 seconds

RETURN
LCDinit:
    PAUSE 500                                    ' let the LCD settle
    LCDout = %0011                              ' 8-bit mode
    PULSOUT E,1
    PAUSE 5
    PULSOUT E,1
    PULSOUT E,1
    LCDout = %0010                              ' 4-bit mode
    PULSOUT E,1
    char = %00001100                            ' disp on, crsr off, blink off
    GOSUB LCDcommand
    char = %00000110                            ' inc crsr, no disp shift
    GOSUB LCDcommand
    RETURN

LCDcommand:
    LOW RS                                       ' enter command mode

LCDwrite:
    LCDout = char.HighNib                       ' output high nibble
    PULSOUT E,1                                 ' strobe the Enable line
    LCDout = char.LowNib                       ' output low nibble
    PULSOUT E,1
    HIGH RS                                     ' return to character mode
    RETURN

-----
Manual:
    low outD                                    ' set rows to 0
    gosub scan
    number=0
    if keypad=0 then Manual                    ' if not button
pressed goto Manual
    lookdown keypad,[14,1,2,3,5,6,7,9,10,11],number ' Map the keypad output to
the list of buttons
    PAUSE 500
    serout outserial,16780,[junk,synchB,number] ' Send the information to the Robot
    if number=9 then Main                      ' If command is 9 goto
Automatic mode
    pause 50
    goto Manual

scan:
for i=12 to 15                                ' For very row assigned to ports 12-15
    high i                                      ' set the the row to 5V
    keypad=inC                                  ' Check all the columns(ports 8-11)
    low i                                       ' reset the row to 0V
        if keypad> 0 then gotkey              ' If any column is set means some button was
pressed so go to decoder routine
    next
    return

gotkey:
    lookdown keypad,[0,1,2,4,8],keypad         ' assign the number 0-4 and then code the keypad
    accordingly to get a integer number
    keypad=((i-12)*4)+keypad                   ' for each button is an unique integer 1-16 that is
returned when that button is pressed

```


RETURN

11.2 Mobile Robot Code With Comments

The program starts with the Manual mode and one can always switch from Manual to the automatic mode. Manual mode is for manual maneuvering of the robot using the keypad provided at the base station. In program for the Mobile station we have provided the code such that it directly shifts itself to the manual mode.

```
Initialization:
'Align the servo to the center.
  GOSUB CENTERALIGN
'Go to the manual mode of operations of the mobile robot.
  GOTO Manual
```

In the manual mode, the Robot waits for the input from the base station key pad. According to the input it switchovers to the corresponding movement subroutine and moves accordingly.

```
SERIN 2,16780,[WAIT(synchB),DAT1]      'Takein the input from the
                                         keypad at the base station.

  IF DAT1=2 THEN GOFORWARD
  IF DAT1=4 THEN LEFT
  IF DAT1=6 THEN RIGHT
  IF DAT1=8 THEN GOBACKWARD
  IF DAT1=1 THEN LEFTDIAG
  IF DAT1=3 THEN RIGHTDIAG
  IF DAT1=9 THEN preMain                'Goto the premain
                                         automatic mode.

  GOTO ENDORDER
```

Here, the key pad input is DAT1 variable and our robot follows the value of DAT1.

When it is 2, it will go forward and by calibration we have set the forward move to be 3 inches.

```
'Subroutine to move forward.
GOFORWARD:
  GOSUB forward2
  GOTO ENDORDER
```

Actual movement routine is:

```
'Subroutine to move forward.
```

```

Forward2:
for i=1 to forwardtime
pulsout 13, RightC-200
pulsout 12, LeftC+190
next
return

```

When it is 8, it will go backward and by calibration we have set the backward move to be 3 inches.

```

'Subroutine to move backward.
GOBACKWARD:
    GOSUB backward2
    GOTO ENDORDER
Actual movement subroutine is
'Subroutine to move backward.
Backward2:
for i=1 to forwardtime
pulsout 13, RightC+200
pulsout 12, LeftC-190
next
return

```

When it is 4, it will go forward to match the centers of rotation and by calibration we have set the left turning to be 90 degree.

```

'Subroutine to turn left 90 degree.
LEFT:
    Dir=10
    GOSUB TURNLEFT2
    GOTO ENDORDER
Actual turning subroutine is
'Subroutine to move left.
TURNLEFT2:
    for i=1 to forwardtime-30
    pulsout 13, RightC-200
    pulsout 12, LeftC+190
    next

    for j=0 to Dir-5
    for i=1 to 10
    pulsout 13, RightC-200
    pulsout 12, LeftC-190
    next
    pause 100
    next
RETURN

```

In this routine we have first moved our robot a little forward and then turned in 90 degree. This is basically required for the automatic mode. We get the optimum turn direction with respect to the Ultrasonic sensor turning center and so we have to move first

our robot turning center which we assume at the center of the robot to the turning center position of the robot and then make the actual turn. This issue is also elaborated in the section problem faced and comments.

When it is 6, it will go forward to match the centers of rotation and by calibration we have set the right turning to be 90 degree.

```
'Subroutine to turn right 90 degree.
RIGHT:
  Dir=0
  GOSUB TURNRIGHT2
  GOTO ENDORDER
```

Actual turning subroutine is

```
'Subroutine to move right.
TURNRIGHT2:
  for i=1 to forwardtime-30
  pulsout 13, RightC-200
  pulsout 12, LeftC+190
  next
  for j=0 to 5-Dir
  for i=1 to 7
  pulsout 13, RightC+200
  pulsout 12, LeftC+190
  next
  pause 100
  next
RETURN
```

When it is 1, it will go left diagonal direction and by calibration we have set the left turning to be 45 degree.

```
'Subroutine to turn left 45 degree.
LEFTDIAG:
  Dir=8
  GOSUB TURNLEFT2
  GOTO ENDORDER
```

Here we provide the direction of turning and then the robot makes corresponding turning movement. When it is 3, it will go right diagonal direction and by calibration we have set the right turning to be 45 degree.

```
'Subroutine to turn right 45 degree.
RIGHTDIAG:
  Dir=2
  GOSUB TURNRIGHT2
  GOTO ENDORDER
```

Here also we provide the direction of turning and then the robot makes corresponding turning movement.

If we do not get any input from the keypad then our Robot will always keep checking the input from the keypad for the movement using the subroutine 'ENORDER'.

If keypad input is 9 then we go to the Automatic mode and then the robot follows the Automatic subroutine from the Main.

In the automatic mode, robot always moves forward and checks for the obstacle ahead. Before every forward move it checks for the obstacle ahead in 3 directions: center and almost 15^0 either direction of the center. Following subroutines are made for these tasks:

1. Subroutine for the constant checking of the obstacle.

```
IF maxmeas>7 THEN Forward
    GOSUB Sense                'Go for scanning the area and
                              action plan for osstacle
                              avoidance.
    GOTO Submain              'Continue submain subroutine
```

2. Subroutine for scanning the front area in 3 directions before making any forward move.

```
SCANNINGFRONT:
  for j=4 to 6
    for i=1 to 200
      pulsout 14,240+(j*90)
    next
    GOSUB sr_sonar2
    pause 100
    UV(j)=wDist
  next
RETURN
```

When robot see any obstacle then immediately it stops as the program diverts its run from the forward subroutine and it goes to the sense subroutine where it senses the scanning area and goes to the other subroutines also.

In the subroutine sense, it goes to the analyze subroutine where it goes to subroutines communicate and also action. First the Robot sends a protocol for the obstacle as letter "P" which stands for the problem faced and help from the basic station is needed. Communicate subroutine is as follows:

```

Communicate:                                'Communication subroutine.
      PAUSE 500
      SEROUT 3,16780,100,[junk,synchA,DAT3]
      SERIN 2,16780,[WAIT(synchB),DAT1,DAT2]
      RETURN

```

When base station receives “P” from the robot it sends back the number 1 which stands for the action subroutine. In this subroutine basically the robot scans the front area in 180⁰ and then analyzes the result of the scanned distances and calculates the optimum direction.

```

ANALIZE:
      DAT3 = "P"                                'Signal to Base station
                                              signal telling it that
                                              'robot has seen obstacle
                                              ahead.
      GOSUB Communicate                        'Send the signal.

'Subanalyze always try to analyze what signal base station sending to
the mobile robot.
SubAnalyze:  IF DAT2 = 1 THEN ACTION

Terminate:  RETURN

```

The action subroutine is given as follows:

```

ACTION:
      GOSUB SCANNING                            'Scanning the area.
      GOSUB getmax                              'Get the optimal direction.
      Dir=maxsensor                            'Getting the optimal
                                              direction.
      DAT3 = Dir                                'Storing the direction in
                                              the communication variable
      GOSUB Communicate                        'Send the direction back to
                                              the base station.
      GOSUB TURNWINDIRI                        'Turn in the optimum
                                              direction.
      GOSUB CENTERALIGN                        'Center align the ultrasonic
                                              sensor.
      GOTO Terminate                            'Terminate the action plan
                                              for obstacle avoidance.

```

When the robot finds the optimum direction it sends back that direction to the base station and then again through the communicate subroutine base station sends the acknowledgement that it has received the direction and then the robot moves forward in the optimum direction.

The subroutine for the Ultrasonic sensor is as follows:

```
'Subroutine for measuring the distance using the Ultrasonic sensor.
sr_sonar2:
    pulsout INIT, 5
    output INIT
    RCTIME ECHO,1,wDist
    wDist=wDist/convfac
    pause 10
    return
```

The subroutine for finding the scanning the front area is as follows:

```
' Scanning the front area in 11 directions and storing them.
SCANNING:
    for j=0 to 10
        for i=1 to 200
            pulsout 14,240+(j*90)
        next
        GOSUB sr_sonar2
        pause 100
        UV(j)=wDist
    next
    RETURN
```

Here, we scan the front area in 5 directions on either side of the center and the center accounts for the 11th direction. Thus the scan angle is equal to the 18⁰ and in the calibration section it has been explained why the choice for 18⁰ angle was made.

The subroutine to get the optimal direction is as follows:

```
'Subroutine to calculate the optimum direction.
getmax:
    maxsensor=0
    maxmeas=0
    for i=0 to 10
        if UV(i)>maxmeas then changemax
        goto finish1
changemax:
    maxsensor=i
    maxmeas=UV(i)
finish1:
    next
return
```

Here simple Reverse bubble sort algorithm is used to find the optimal direction. We check every directional distance and sweep the values in the variable maxsensor when we get the more directional distance.

Care is also taken that after every turning movement of the robot in the optimal direction the Ultrasonic sensor is again aligned at the center direction or it is made to

head in the optimal direction in which the robot has turned and thus every time it will point to the direction of heading. The subroutine for that is as follows:

```
'Subroutine to center align the ultrasonic sensor.
CENTERALIGN:
  for i=1 to 255
    pulsout 14,652
  next
RETURN
```

The following subroutine is to turn the robot in right or left direction depending upon the value of the variable Dir.

```
'Subroutine to change the direction of the mobile robot in the optimum
direction.
TURNWINDIRI:
```

```
  IF Dir<5 THEN TURNRIGHT
  IF Dir>=5 THEN TURNLEFT
RETURN
```

```
'Turning Mobile robot in left direction.
TURNLEFT:
```

```
  for i=1 to forwardtime-30
    pulsout 13, RightC-200
    pulsout 12, LeftC+190
  next
  for j=0 to Dir-5
    for i=1 to 11
      pulsout 13, RightC-200
      pulsout 12, LeftC-190
    next
    pause 100
  next
RETURN
```

```
'Turning Mobile robot in right direction.
TURNRIGHT:
```

```
  for i=1 to forwardtime-30
    pulsout 13, RightC-200
    pulsout 12, LeftC+190
  next
  for j=0 to 5-Dir
    for i=1 to 7
      pulsout 13, RightC+200
      pulsout 12, LeftC+190
    next
    pause 100
  next
RETURN
```

From the Base Station point of view, it is always waiting for the Robot to send signal and then accordingly it takes the steps for the automatic mode and in case of the

manual mode in which the program for the Robot starts it gives the instructions from the keypad.

In the manual mode, there are basically 2 subroutines. The manual subroutine is basically for taking the account of the key pressed from the keypad. The scan subroutine is for checking which key is pressed in the keypad and then accordingly store the value in the variable 'number'.

The manual subroutine is as follows:

```
'Manual mode of the basic stamp.
Manual:
  low outD
  gosub scan

  number=0

  if keypad=0 then Manual

  lookdown keypad,[14,1,2,3,5,6,7,9,10,11],number

  PAUSE 500
  serout outserial,16780,[junk,synchB,number]

  if number=9 then Main

  pause 50
  goto Manual
```

'Go to the scan for checking which key is pressed on the key pad.
'Initialization of the number to 0.
'Actually means that when 0 is pressed do nothing or if the robot is doing some movement, stop it when 0 is pressed.
'Lookdown table for keypad.
'Send the data to the robot.
'Number 9 for the automatic mode.

The manual subroutine does very important job of sending back the value of the key pressed to the robot. The subroutine scan which keeps track of the key pressed on the keypad is as follows:

```
'Subroutine for the scanning the keypad for getting the number of the key pressed.
scan:
  for i=12 to 15
    high i
```



```

        keypad=inC
        low i
        if keypad> 0 then gotkey
    next
    return
gotkey:
lookdown keypad,[0,1,2,4,8],keypad
    keypad=((i-12)*4)+keypad
    RETURN

```

In the automatic mode, the base station always waits for the signal from the robot. Here we assume that base station always waits for the signal “P” from the robot which tells the base station that robot needs help. The subroutine that always checks the signal from the robot is as follows:

```

FIRSTCOMIN:
    Serin inserial,16780,[WAIT(synchA),DAT1]
    IF DAT1="P" THEN FIRSTCOMOUT
    GOTO FIRSTCOMIN

```

When the robot sends the base station signal “P” then the base station sends signal “1” back to the robot and this is been implemented in the following subroutine. The LCD messages are also incorporated to show the status of the robot and also the base station.

```

FIRSTCOMOUT:
index=msg2
GOSUB WriteLCD
PAUSE DLY
index=msg3
GOSUB WriteLCD
Serout outserial,16780,[junk,synchB,LETTERS,1]

```

'send back the
signal to the
robot to 'scan
the front area.

When the robot sends back the optimum direction in which it is going to turn, it displays that direction on the LCD and also it keeps track of what the robot is doing by flashing the messages on the LCD.

```

SECONDCOMIN:
Serin inserial,16780,[WAIT(synchA),DAT1]

```

'Getting the
optimum
direction in
which
'the robot is
going to

```

turn.
index=msg4
GOSUB WriteLCD

```

```

dirG=dat1
if dat1>=5 then posi
' code to
display the
optimum
direction
of turn

dirG=(5-dat1)*18
decided by the robot.
dirG1=dirG/10
dirG2=dirG-(10*dirG1)
pos1="R"
goto shownumber
posi:
dirG=(dat1-5)*18
pos1="L"
dirG1=dirG/10
dirG2=dirG-(10*dirG1)
shownumber:
debug ? dat1
char=pos1
GOSUB LCDwrite
char=dirG1+48
GOSUB LCDwrite
char=dirG2+48
GOSUB LCDwrite

```

Also the acknowledgement subroutine is also provided in the base station as follows:

```

SECONDCOMOUT:
PAUSE DLY
index=msg5
GOSUB WriteLCD
Serout outserial,16780,[junk,synchB,LETTERS,2]
'Acknowledgement
by the base
station of
the direction
of turn of the
robot.

```

The subroutines regarding the LCD display are not written in the report because they are redundant and also explained in previous lab reports.