

State University of New York at Buffalo

Department of Mechanical and Aerospace Engineering

MAE 405/505: MECHATRONICS

Project Report

**Distributed Sensing and
Control Framework for Mobile Robots**

(Intelligent Temperature & Distance Sensing Robot)

DATE: May 9, 2002.

Instructor: Dr. V. Krovi

SUBMITTED BY:

Lee Leng Feng
Rajankumar Bhatt
Krishnakumar Ramamoorthy

TABLE OF CONTENTS:

	Topics Covered	Pages
1.	Abstract.	1
2.	Objectives	1
3.	Goals	1
4.	Project Description.	
	<i>-Part 1. Requirements</i>	1
	<i>-Part 2. Our Implementation</i>	2
5.	Hardware used.	3
6.	Motivation.	
7.	Alternatives Discussed	16
8.	Circuit Diagrams	17
9.	Approach	19
	<i>-Part A. Master-Slave mode</i>	4
	<i>-Part B. Autonomous mode</i>	
10.	Overall Framework	
11.	Future Work	21
12.	Components used in this experiment and its price	21
13.	Conclusion	21
14.	Reference	22

LIST OF FIGURES

Figure	Description	Page
1	A Boe-Bot	3
2	SONY RM-V201 remote	3
3	The pulse train used in the pre-modified servomotor	5
4	Plot for calibration of Right wheel	7
5	Plot for Calibration of Right Wheel in Linear region	8
6	Plot for calibration of Left wheel	9
7	Plot for Calibration of Right Wheel in Linear region	10
8	Circuit diagram for interfacing the IR Detector	11
9	Flow Chart for the working of the program	16

LIST OF TABLES

Table	Description	Page
1	Calibration data of Right and Left servomotors	6
2	Components use in this lab and its Price	21
3	Contribution of each member	21

ABSTRACT:

In this project, we examine development and implementation of a distributed sensing and control framework for a system comprised of the mobile robot (with its onboard processor), the base station, which are coupled together by wireless IR and RF communication. We used the mobile robot which we built for previous lab with added features like IR and RF communication, temperature sensor, distance detector, etc. as our mobile robot and the INEX – 1000 board as the base (control) station. We also established our own communication protocol for communication between these two processors, which increased overhead of coordination and ultimately increased complexity. **We have also successfully** incorporated two different modes in which our mobile robot works.

1. Master – Slave mode in which user can fully control the mobile robot with the help of remote control.
2. Autonomous mode in which the mobile robot, with the help of its distance sensing capability, moves about in a maze

In both these modes of operation, the mobile robot sent the data of temperature and distance at discrete intervals of times to the base station.

OBJECTIVES:

The objectives of this project are:

1. Establishing our own communication protocol
2. Setup the programs with the capability of communicating/handshaking and distributing tasks.
3. Gain experience with using different sensors and chips not taught in class by looking at datasheets and manuals.

GOALS:

The goals of this experiment are:

1. Use the available hardware for distributed sensing application.
2. Gain knowledge of distributed control and experience the issues involved.
3. Learn how to successfully use new sensors and chips, which we never came across.

PROJECT DESCRIPTION:*1. Requirements**2. Our Implementation*

HARDWARE USED:


The basic hardware that we used in this lab are:

1. Stamp Works INEX-1000 lab board
2. BASIC Stamp II module
3. 12-Volt Wall transformer
4. 9 – Volt Wall transformer
5. Programming Cables
6. Resistors
7. LCD screen

Note: The details description can be found in our first, second and third lab report [8].


The other hardware used are as below:

1. BOE-Bot mobile robot Kits.

 <p><i>Figure 1. A BOE Bot.</i></p>	Power Supply	4 AA's for BASIC Stamp and motor control (not included)
	Assembly Tools	Angled cutters, small Phillips screwdriver, and 1/4" wrench.
	Time Required to Build and Program	2.0 hours - Boe-Bot uses the pre-built Board of Education to save time.
	I/O Components	LEDs, speaker, pushbutton, photoresistors, resistors and capacitors, infrared LEDs and receivers, '555 timer.
	Use of BASIC Stamp I/O pins	Components are placed on the breadboard and may be moved around. Flexible use of I/O pins.
	Object Detection	Infrared object detection, object detection with whiskers
	Expandability	Breadboard or through-hole AppMod will fit, but hang off the side of the Boe-Bot. Expandability is possible with the Compass Appmod or Line-Follower Module. (not included)

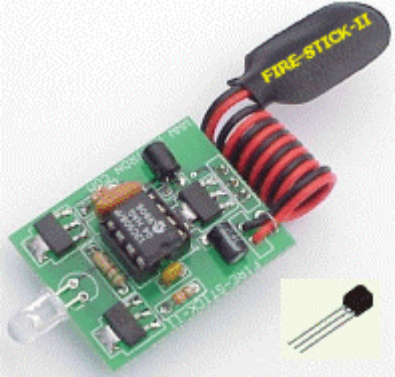
Note: For more information, [1],[4]

2. SONY universal remote control. Model: RM-V201.

 <p><i>Figure 2. SONY RM-V201 remote.</i></p>	<ul style="list-style-type: none"> • Operating distance: Approx. 23 feet (7m) • Power requirements Remote: AA x 2 Batteries (not supplied) • Battery life: Approx. 6 months (varies depending on frequency of use) • Weight: Approx. 3.5 oz. (100g), not including batteries • Dimension: 2 1/2" x 7 5/8" x 1 1/4" (61 x 191 x 30mm)
--	--


Note: For more information, [2]


3. IR Receivers and Fire-stick:

 <p><i>Figure 3. IR Receivers and Fire-stick</i></p>	<ul style="list-style-type: none"> • Carrier frequency: 38KHz. (40KHz available on request) • Maximum operating range: @ 9 VDC 40 to 60 foot (indoors). • Maximum operating voltage: 9 VDC. • Maximum recommended data-rate: 2400 baud • Minimum operating voltage: 7.5 VDC • Average current (continuous transmit): 30Ma @ 9VDC. • Average Idle current (Not Transmitting): 6mA @ 9VDC. • Peak pulsed current: 1A @ 9VDC. • Infrared LED beam angle: 60 degrees. • Infrared LED wavelength: 940nm.
--	---

Note: For more information, [3],[6]


4. RF Transmitter and Receiver:

 <p><i>Figure 4a. RF Transmitter</i></p>	<ol style="list-style-type: none"> RF Transmitter <ul style="list-style-type: none"> • Frequency: 300 – 433MHz • Modulation: AM (Code) • Supply Voltage: 1.5v – 12v dc • RF Output Power: 8mW. RF Receiver <ul style="list-style-type: none"> • Frequency: 300 – 433MHz • Modulation: AM • Supply Voltage: 4.5v – 5.5v dc • Sensitivity: 3uVrms • Output: Digital & Linear
---	--

 <p><i>Figure 4b. RF Receiver</i></p>	<ul style="list-style-type: none"> • Application: Radio Remote Control
--	--


Note: For more information, [8]

5. Stepper Motor:

 <p><i>Figure 5. IR Receivers and Fire-stick</i></p>	<ul style="list-style-type: none"> • Phase resistance (Ohms): 75 • Current (mA): 150 • Phase Inductance (mH): 39 • Detent torque (g-cm): 80 • Holding Torque (g-cm): 600 • Mounting hole space diagonal (in.): 1.73 • Mounting hole (in.) 0.11 • Shaft diameter (in.): 0.197 • Shaft length (in.): 0.43 • Motor Diameter (in.): 1.66 • Motor height (in.): 1.35 • Weight: 0.55 lbs.
---	---

Note: For more information, [9]

6. Sharp GP2D02 Infrared Ranger:

 <p><i>Figure 6. Sharp GP2D02 Infrared Ranger</i></p>	<ul style="list-style-type: none"> • Detection Range: 10 to 80 cm • Supply Voltage: -0.3 to 10 V • Input Terminal voltage: -0.3 to 3 V. • Output Terminal Voltage: -0.3 to 10 V • Operating Temperature: -10 to 60 °C • Storage Temperature: -40 to 70 °C
--	---

Note: For more information, [11] ,[12]

6. Digital Thermometer and Thermostat:



*Figure 6. Digital Thermometer and
Thermostat*

- **Measurement Range:** -55 to 125 °C
- **Supply Voltage:** 2.7 to 5.5 V
- **Resolution:** 0.5 °C
- **Conversion time (to digital):** 1 second

Note: For more information, [13]

MOTIVATION:

ALTERNATIVES DISCUSSED:

Circuit:

APPROACH

Part A. Master – Slave Mode

Introduction:

In this mode, the Control station controls the mobile agent. This mode is specially designed to allow full control of the mobile agent and its motion. The control station receives command from user and sends that to mobile agent. The agent sends the temperature and distance data to control station. Control station plots the distance data on stamp-plot lite and temperature data on thermometer.

Individual Tasks:

The project was divided in many individual tasks. Each task was completed and tested independently before integrating all tasks together. The main emphasis for the Master – Slave mode was on communication. Since, the aim was to give control commands from the control station and guide the mobile robot, it was extremely necessary that the information sent by the base station be conveyed to the mobile robot without any loss. Hence, the main two individual tasks for this mode were to achieve asynchronous serial IR and RF communications. As discussed above, we also plot the temperature data on analog thermometer, which is basically a temperature-scale with pointer mounted on shaft of stepper motor. This introduces one more task of controlling stepper motor. Also, we used stamp-plot lite to display a continuous graph of obstacle distance. Hence, learning how to plot data using stamp-plot lite became equally necessary.

IR Communication

The asynchronous serial IR communication was achieved by using Fire-Stick II. The application note[3] and the datasheet [6] made our task easy. We decided to put Fire stick on the base station and the IR receiver on the mobile agent. We connected the circuit required for the functioning of IR communication only as shown in circuit diagram below.

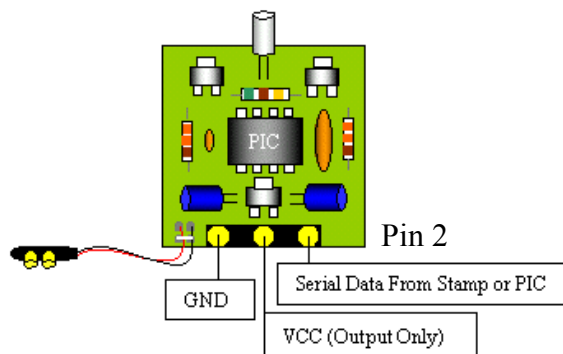


Fig 9a. Circuit on Base station

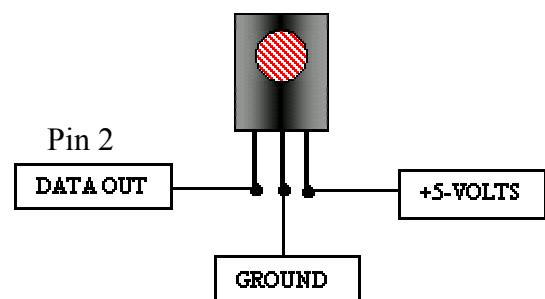


Fig 9b. Circuit on mobile agent

After connecting the circuit, we tested the program whose code is as follows:

```

CODE ON MOBILE AGENT
baud   con      813                ' 1200 Baud, (NON-INVERTED)
start:
  serin 0,baud, [ wait ("A"), b0 ] ' Wait for ASCII letter A
  debug ? b0                       ' Show value received in B0
  debug cr                          ' Carriage return
  goto start                        ' Loop

CODE ON BASE STATION
time   var byte
time   =       250
baud   con      17197             '1200 Baud, (INVERTED)
number var      byte
begin:
  for number = 1 to 100
    serout 15,baud,["A",number] ' N1200 baud
  pause time
  next
  goto begin

```

At the time of testing the IR communication, the mobile agent was connected with a personal computer using a serial cable. The data received by the agent was displayed in the debug window and this confirmed that the asynchronous serial IR communication was working.

RF Communication

The asynchronous serial RF communication was achieved by using RF transmitter with antenna and RF receiver with antenna. This task was again divided into subtasks. Each subtask was tested individually and then integrated to achieve RF Communication.

The first subtask was to achieve wired asynchronous serial communication. This was achieved by following the instructions given at the end of SERIN and SEROUT commands in the stampworks manual.[14] The two basic stamps were connected via wires with the help of circuit as shown in figure below.

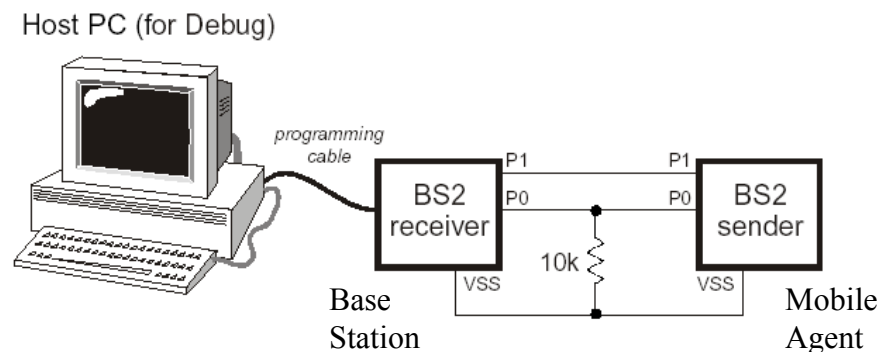


fig 10. circuit for wired asynchronous serial communication

```

' we connected the circuit shown above and ran
' this program on the mobile agent. This program demonstrates the
' use of Flow Control (FPin). Without flow control, the sender would transmit the whole

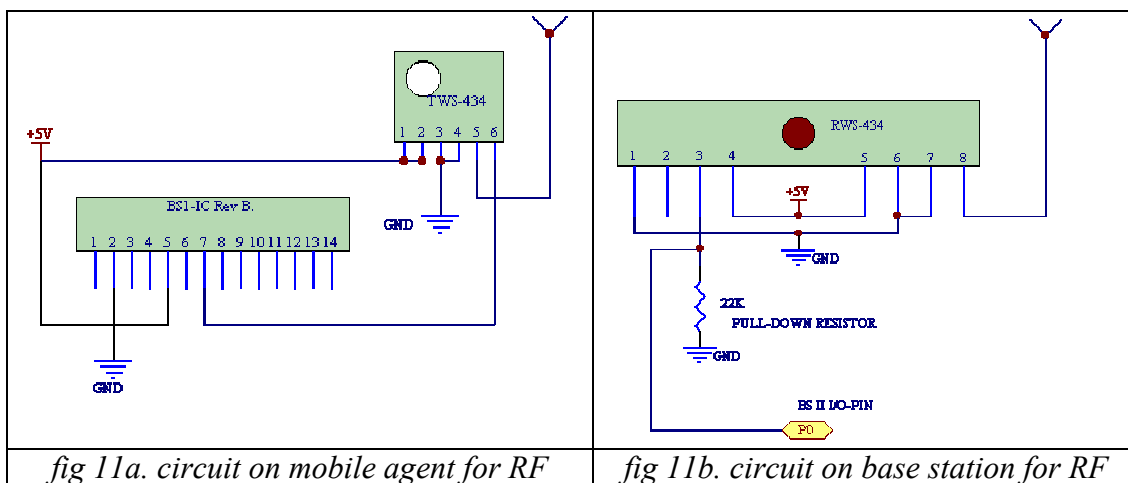
```

```

' word "HELLO!" in about 6 ms. The base station would catch the first byte at most; by
' the time it got back from the first 1-second PAUSE, the rest of the data would be long
' gone. With flow control, communication is flawless since the sender waits for the
' receiver to catch up.
'{$STAMP BS2} 'STAMP directive (specifies a BS2)
  Loop:
    SEROUT 1\0, 16468, ["HELLO!"] ' Send the greeting.
    PAUSE 2500
    GOTO Loop
' we connected the circuit shown above and ran
' this program on the base station. This program demonstrates the
' use of Flow Control (FPin). Without flow control, the sender would transmit the whole
' word "HELLO!" in about 6 ms. The base station would catch the first byte at most; by
' the time it got back from the first 1-second PAUSE, the rest of the data would be long
' gone. With flow control, communication is flawless since the sender waits for the
' receiver to catch up.
  Letter VAR BYTE
  Again:
    SERIN 1\0, 16468, [Letter] ' Get 1 byte.
    DEBUG Letter ' Display on screen.
    PAUSE 1000 ' Wait a second.
    GOTO Again

```

Once this communication was tested, we then used the RF Transmitter on mobile agent and connected the circuit as shown in fig 11a. and the RF receiver on base station and connected the circuit as shown in fig 11b. The base station and mobile agent were connected with different computers via a serial cable.



Now, we made a little change in code, the flow control line was removed, thus the SERIN line in code would now be read as

```
SERIN 1, 16468, [Letter] ' Get 1 byte
```

And similarly the SEROUT line in code would now be read as

```
SEROUT 1, 16468, ["HELLO!"] ' Send the greeting.
```

With this change we could read the letter 'H' on the base station, which confirmed that this communication was working.

NOTE: Because of pause of 1000 milliseconds in the code for base station, it could read and display only the first letter of greeting. We changed the greeting to test whether we were getting appropriate values.

Stepper Motor

As discussed above, we used stepper motor to display analog temperature. The temperature data was received from the agent at regular intervals of time. This temperature was updated on the Thermometer shown in figure 12. Hence, it was extremely necessary to test this subsystem before incorporating it in the larger system. The following approach was followed to test this subsystem.

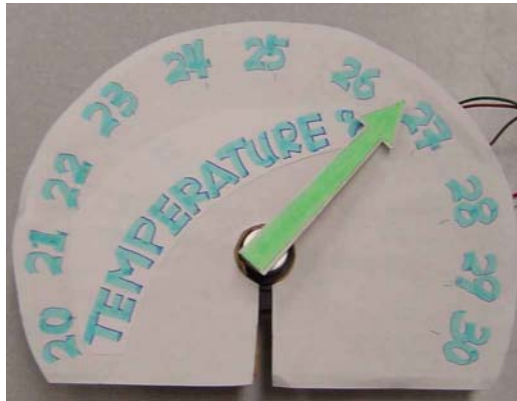


fig 12. Thermometer

- The first thing that we tested was to connect the proper circuit and make the motors rotate clockwise and anticlockwise.
- Then we measured the number of rotations required to finish one rotation.
- The next step was to decide the range of temperature to be displayed and the total angle to be used to display that range. The range of temperature was decided from 20⁰C to 30⁰C and the angle was decided to be 120 degrees.
- Then, we marked the readings and tested the working with the help of the program written below.

```
'{$STAMP BS2}

coils      var      outd      'stepper was connected to nib4
saddr      var      byte      'address used to give sequence to stepper
counter var  byte      'used in for loop
prevtemp   var      byte      'stores previous temp value
noofloop   var      byte      'stores the value of number of loops
temperature var  nib      'stores value of current temperature

dird = %1111s      'initialization of pins

step1 DATA  %1100      'data stored in memory
step2 DATA  %0110      'which defines the sequence
step3 DATA  %0011      'for rotation of stepper motor
step4 DATA  %1001

saddr = 0      'initialization of variables
temperature = 10
prevtemp = 0

startloop      'get, set, go
for temperature = 0 to 10
  pause 2000
  if temperature > prevtemp then rotateforward
    noofloop = (prevtemp-temperature)*6      'rotate anticlockwise if temp decreases
    for counter = 0 to noofloop
      saddr = saddr + 3//4
```

```

        read (stepl+saddr),coils
        pause 5
    next
    goto donetemp
rotateforward:
    noofloop = (temperature-prevtemp)*5      'rotate clockwise if temp increases
    for counter = 0 to noofloop
        saddr = saddr + 1//4
        read (stepl+saddr),coils
        pause 5
    next
donetemp:                                     'if no change in temp, do nothing
    debug ? temperature
    prevtemp=temperature
next
pause 5000
goto startloop                               'keep doing

```

- Once this was achieved, we changed the code a bit and used Random command to generate a random number and display that temperature. Proper working of this ensured its place in the system.

Logging and Plotting data using Stamp-plot Lite

Stamp-plot Lite was used to display and log the distance of obstacle sensed by IR ranging system. Hence, this was also one of the crucial parts of our project. We downloaded the file from parallaxinc website.[15] We followed the tutorial step by step and could use the software to plot the data.

Combining Tasks:

After testing each subtask individually, we started with integrating them one by one. We started with the IR and RF Communications. The idea was to use IR communication for sending command to mobile agent and then use RF communication as Flow Control line. Along with its use as a Flow Control line, it was also used for transferring data of temperature and distance of obstacle to the base station. The working of communication protocol can be explained as follows:

The base station will be in a constant loop waiting for the input from user. User has to send a command and tell the base station, where he wants the mobile agent to move. As soon as base station receives command from user, it sends the command to mobile agent and waits for confirmation from the agent. Mobile agent keeps sending error code, temperature and distance to base station until it receives some information from the base station. As soon as it receives the command from the base station, it sends confirmation to the base station along with temperature and distance of obstacle. On the other hand, after sending the command to the mobile agent, the control station waits for 1 second for the confirmation. If it doesn't receive the confirmation, it gives a beep to user and again sends the command to the agent and waits for one more second. If after one more second, it does not receive confirmation, it increases the frequency of the beep and alerts user. It keeps on doing this for 10 times. Even after 10 tries, it cannot reach the agent, base station gives up and rings a continuous beep to bring immediate attention of user. This was achieved by combining the codes written for individual IR and RF testing and making necessary changes as can be seen in the code.

The, the code for displaying temperature and distance was incorporated as a subroutine and the call to this subroutine was made at proper places.

COMMUNICATION PROTOCOL FORMAT:

The communication protocol that was used had following format.

*From Mobile agent to Base station***status,distance, temperature**

status: this was used to send information to the base station whether the mobile agent received data or not

distance: the distance of obstacle (reading from ir ranging system)

temperature: the temperature measured in degree centigrade

*From Base station to Mobile agent***mode, motion,noofloops**

mode: 1 means Master-slave mode and 2 means autonomous mode

motion: determines the motion for mobile agent for instance, forward, backward, spin left, etc.

noofloops: Determines by what amount it should move in particular fashion. User can change it with the help of Volume up and Volume down buttons on remote control.

Flow-Chart:

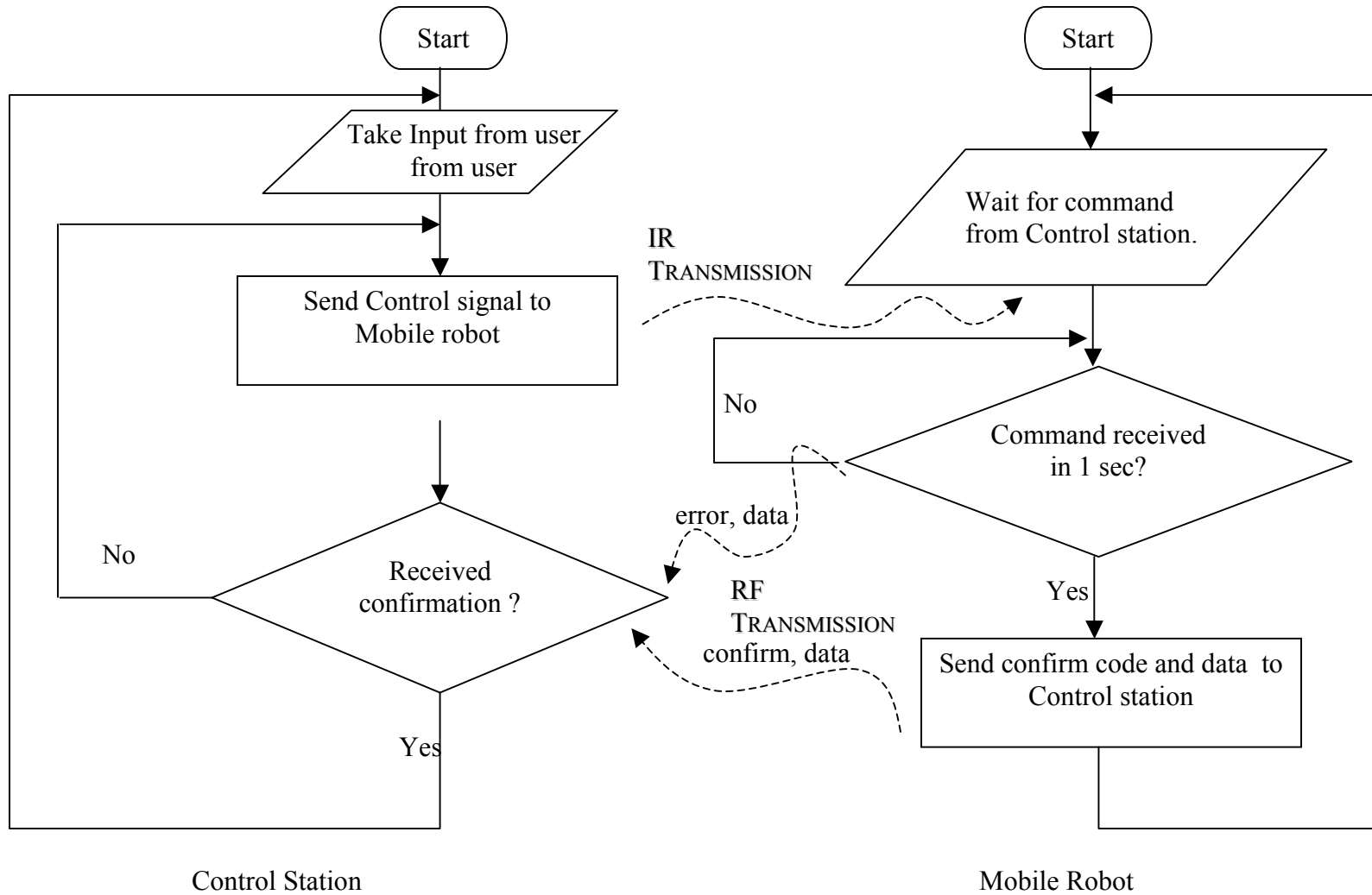


fig 13. Flow chart for Master-Slave mode

Coding:Code for Base station for Master-Slave Mode

```

'{$STAMP BS2}
'-----Declaration-----
baud      con      17197      '1200 baud, (INVERTED)
confirm var   byte           'store confirm byte
frequency  var    word       'frequency for buzzer
distance  var    byte       'var to store distance
temperature var   byte       'var to store temperature
IR        con      6         'ir transmitter on pin 6
ERROR    con      222       'indicates communication error1
p1       var    word       'var for remote control
p2       var    word       'var for remote control
p3       var    word       'var for remote control
p4       var    word       'var for remote control
p5       var    word       'var for remote control
p6       var    word       'var for remote control
noofloops var   byte       'control of no. of rotations
remote var   byte       'var to store command
coils      var    outd      'pins connected to stepper
saddr     var    nib       'used as counter for stepper
counter var   byte       'used as counter for rotations
prevtemp  var    nib       'var that stores previous temp
noofloop  var    byte       'var for stepper moter
E         con      0         'LCD Enable pin (1 =enabled)
RS        con      3         'Register Select (1 = char)
LCDout   VAR    OutC      '4-bit LCD data
char     VAR    Byte      'character sent to LCD
index    VAR    Byte      'loop counter

step1  DATA  %1100      ' data for stepper
step2  DATA  %0110      ' data for stepper
step3  DATA  %0011      ' data for stepper
step4  DATA  %1001      ' data for stepper
Msg    DATA  "GROUP E ",0 ' preload EEPROM with message
Msgcm  DATA  "Communicating",0
Msgip  DATA  "Expecting Input",0
Msgmd  DATA  "Independent mode",0
Msgger DATA  "ERROR...",0

ClrLCD  CON    $01      ' clear the LCD
CrsrHm  CON    $02      ' move cursor to home position
CrsrLf  CON    $10      ' move cursor left
CrsrRt  CON    $14      ' move cursor right
DispLf  CON    $18      ' shift displayed chars left
DispRt  CON    $1C      ' shift displayed chars right
DDRam   CON    $80      ' Display Data RAM control

'-----

Initialize:
  Dirs = %1111111111011000 ' setup pins for peripherals
  GOSUB LCDinit           ' initialize LCD for 4-bit mode

remote=0                 ' Initialize variables
noofloops = 50           ' Initialize variables
frequency = 900          ' Initialize variables
saddr = 0                ' Initialize variables
temperature=20           ' Initialize variables
prevtemp = 0             ' Initialize variables
distance = 0             ' Initialize variables
DEBUG "!USRS GROUP E - Rover Control",13 'show on stamp-plot lite

  index=msg              ' Write message to LCD
  GOSUB WriteLCD
  PAUSE 2000            ' Display msg for sometime

```

```

'-----start loop-----
begin:
  DEBUG dec distance,cr          ' plot in stamp-plot lite
  GOSUB getRemote              ' in a loop
giveoutput:
  GOSUB WriteOutput            ' Command mobile robot
  PAUSE 250                    ' PAUSE for sometime
retryinput:
  GOSUB ReadInput              ' Read data from mobile robot
goto begin

'-----READ input through rf-----

ReadInput:
  SERIN 1,baud,1000,retryinput,[wait("DEF"),confirm,distance,temperature]

'-----SERIN-----
' Read input on pin 1 at baud rate of 17197
' wait for 1 second for input with first letter as DEF
' if do not recieve input, go to retryinput, else
' store successive data in confirm, distance and temperature
'-----
  GOSUB Display_temperature    'goto subroutine to display temp
  if confirm=ERROR THEN errorcomm ' if there is error, goto errorcomm
  frequency=900                'if proper communication, reset frequency
  errchkdone:
RETURN

'-----write output via IR-----

WriteOutput
DEBUG "!USRS Communicating with Rover",13 'Display on stamp-plot lite
  index = Msgcm                 'Show on LCD
  GOSUB WriteLCD
  serout 2,baud,["ABC",1,remote,noofloops]
RETURN

'-----take action on ERROR in communication-----

errorcomm:
DEBUG "!USRS ERROR in communication",13 'Display on stamp-plot lite
  index = Msgcr                 'show on LCD
  GOSUB WriteLCD
  freqout 4,500,frequency       'produce an error beep
  frequency=frequency+10        'increase beep frequency
  IF frequency > 1000 THEN infiloop 'if error for 10 times, hang
GOTO giveoutput

infiloop:
  frequency=1000
goto errchkdone                'errorcomm means infiloop

getremote:
  DEBUG "!USRS Waiting for Input",13 'Display on stamp-plot lite
  index = Msgip                 'Show on LCD
  GOSUB WriteLCD
  HIGH 7                        'Glow LED
  PULSIN IR,0,p1                'Get input from remote
  IF (p1 < 1000) THEN Begin      'Check is pulse is great THEN 1000
  PAUSE 25

  PULSIN IR,0,p1 'Read the head
  PULSIN IR,0,p1 'Destroy the head and use p1 to store first pulse
  PULSIN IR,0,p2 'Read second pulse
  PULSIN IR,0,p3 'Read third pulse
  PULSIN IR,0,p4 'Read fourth pulse
  PULSIN IR,0,p5 'Read fifth pulse
  PULSIN IR,0,p6 'Read sixth pulse and forget about the rest.

```

```

'*****
' * This subroutine will convert the pulse from the remote and convert *
' * it to a decimal number and THEN call the subroutine according to *
' * the signal READ in but the 38k IR receiver. *
'*****

    remote = 1 + (p1/450)          'Convert the first pulse to binary and add 1
    remote = 2*(p2/450) + remote  'Con P2 to bin THEN shift left 1 and add to remote
    remote = 4*(p3/450) + remote  'Con P3 to bin THEN shift left 2 and add to remote
    remote = 8*(p4/450) + remote  'Con P4 to bin THEN shift left 3 and add to remote
    remote = 16*(p5/450) + remote 'Con P5 to bin THEN shift left 4 and add to remote
    remote = 32*(p5/450) + remote 'Con P6 to bin THEN shift left 5 and add to remote

LOW 7

    IF remote = 51 THEN incrnoofloops      'if volume up, goto incrnoofloops
    IF remote = 52 THEN decrnoofloops      'if volume down, goto decrnoofloops
    IF remote > 15 THEN begin              'if greater than 15, neglect

RETURN

incrnoofloops:                          'increment no. of rotations
    IF noofloops>230 THEN begin
        noofloops=noofloops+20
        DEBUG ? noofloops
    goto begin

decrnoofloops:                          'decrement no. of rotations
    IF noofloops<30 THEN begin
        noofloops=noofloops-20
        DEBUG ? noofloops
    goto begin

Display_temperature:                    'Display the temperature on analog meter
    temperature=temperature-20          'reset temperature to minimum value
displayed
    IF temperature < 0 THEN donetemp     'neglect error data
    IF temperature > 11 THEN donetemp    'neglect error data
    IF temperature = prevtemp THEN donetemp 'if no change in temp, do nothing
    IF temperature > prevtemp THEN rotateforward 'if temp increased, rotateforward
    noofloop = (prevtemp-temperature)*6  'rotatebackward
    for counter = 0 to noofloop
        saddr = saddr + 3//4
        READ (stepl+saddr),coils
        PAUSE 5
    next
    goto donetemp
rotateforward:                          'rotateforward
    noofloop = (temperature-prevtemp)*5
    for counter = 0 to noofloop
        saddr = saddr + 1//4
        READ (stepl+saddr),coils
        PAUSE 5
    next
donetemp:
    DEBUG ? temperature                  'display temperature
    DEBUG ? prevtemp                     'dispaly previous temperature
    prevtemp=temperature                 'store temp as previous temp
temperature=temperature+20              'restore original value of temp
RETURN

WriteLCD:
    char = ClrLCD                        ' clear the LCD
    GOSUB LCDcommand

ReadChar:
    READ index,char                      ' get character from EEPROM
    IF char = 0 THEN MsgDone             ' IF 0, message is
complete

```

```

GOSUB LCDwrite                                ' write the character
index = index + 1                             ' point to next character
GOTO ReadChar                                 ' go get it
MsgDone:                                       ' the message is complete

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 = %0000110                             ' 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

```

Code for Mobile Robot for Master-Slave Mode

```

'({$STAMP BS2})
value      var      byte
baudnoninv con      813    ' 1200 Baud, (NON-INVERTED)
baud       con      17197  ' 1200 Baud, (INVERTED)
confirm con  111      ' con for confirm
error      con      222    ' con for error
temp       var      byte    ' var for temperature
motion var  nib      ' var to store kind of motion
looptimes  var      byte    ' var to store loop times
mode       var      nib     ' autonomous or controlled
counter var byte     ' var for counting
T          var      Byte    ' var to store temp
dist       var      byte    ' var to store distance

DQ         CON      10     ' Data pin for 1620
Clock      CON      8     ' Clock pin for 1620
Reset      CON      9     ' Reset pin for 1620

tempIn VAR  Word          ' raw temperature
sign       VAR  tempIn.Bit8 ' 1 = negative temperature
tSign      VAR  Bit
tempC      VAR  Word      ' Celsius
tempF      VAR  Word      ' Fahrenheit

RdTmp      CON  $AA          ' read temperature
WrHi       CON  $01          ' write TH (high temp)
WrLo       CON  $02          ' write TL (low temp)
RdHi       CON  $A1          ' read TH
RdLo       CON  $A2          ' read TL
StartC     CON  $EE          ' start conversion
StopC      CON  $22          ' stop conversion
WrCfg      CON  $0C          ' write config register
RdCfg      CON  $AC          ' read config register

```

```

'-----
'INITIALIZE

dir3 =%1
dir4 =%0
temp=0
GOSUB DS1620init           ' initialize the DS1620
dist = 0

start:

  serin 2,baudnoninv,1000,resendrf, [ wait ("ABC"), mode,motion,looptimes ] ' Wait for
  ASCII letter A

callattended:

  GOSUB measuredistance
  GOSUB gettemperature

  serout 1,baud,["DEF",confirm,dist,tempC]
  GOTO start ' Loop

dependent:
  if motion=2 then Forward
  if motion=8 then Backward
  if motion=4 then spinRight
  if motion=6 then spinLeft
  PAUSE 1000
goto callattended

spinLeft:                ' Spin left
  for counter = 0 to looptimes
    pulsout 15,612
    pulsout 13,622
    pause 20
  next
goto callattended

spinRight:               ' Spin right
  for counter = 0 to looptimes
    pulsout 15,858
    pulsout 13,820
    pause 20
  next
goto callattended

backward:                'move backward in straight line
  for counter = 0 to looptimes
    pulsout 15,860
    pulsout 13,612
    pause 20
  next
goto callattended

forward:                 'move forward in straight line
  for counter = 0 to looptimes
    pulsout 15,622
    pulsout 13,858
    pause 20
  next
goto callattended

resendrf:
  serout 1,baud,["DEF",error,dist,tempC]
  debug "came to resendrf",cr
goto start

DS1620init:
  HIGH Reset           ' alert the DS1620
  SHIFTOUT DQ,Clock,LSBFirst,[WrCfg,%10] ' use with CPU; free-run

```

```

LOW Reset
PAUSE 10
HIGH Reset
SHIFTOUT DQ,Clock,LSBFirst,[StartC] ' start conversions
LOW Reset
RETURN

GetTemperature:
HIGH Reset ' alert the DS1620
SHIFTOUT DQ,Clock,LSBFIRST,[RdTmp] ' give command to read temp
SHIFTIN DQ,Clock,LSBPRE,[tempIn\9] ' read it in
LOW Reset ' release the DS1620

tSign = sign ' save sign bit
tempIn = tempIn/2 ' round to whole degrees
IF tSign = 0 THEN NoNeg1
tempIn = tempIn | $FF00 ' extend sign bits for negative

NoNeg1:
tempC = tempIn ' save Celsius value
tempIn = tempIn */ $01CC ' multiply by 1.8
IF tSign = 0 THEN NoNeg2
tempIn = tempIn | $FF00 ' if negative, extend sign bits

NoNeg2:
tempIn = tempIn + 32 ' finish C -> F conversion
tempF = tempIn ' save Fahrenheit value
RETURN

measureDistance: ' subroutine to measure distance
high 3
pause 3
low 3
FOR t = 1 TO 70
  PAUSE 1
  IF In4=1 THEN Jump
NEXT

Jump:

shiftin 4,3,2,[value\8]
debug ? value
IF value <= 67 THEN calib1 ' decide the calibration curve
IF value <=109 THEN calib2
IF value <=147 THEN calib3
IF value <=167 THEN calib4
IF value <=200 THEN calib5

'-----
'Different calibration lines for different ranges
'-----

calib1:
dist = 1974 - 5375/100*value + (value*value*375/1000)
RETURN

calib2:
dist = 275 - (447*value/100) + (value*value*2/100)
RETURN

calib3:
dist = 98 - value + (value*value*3/1000)
RETURN

calib4:
dist =12
RETURN

calib5:
dist=10
RETURN

```

Discussion and Testing:

The most difficult part to achieve was proper communication. When we sent the command from the base station to the mobile agent, the agent, in many cases was not receiving the command. We then exploited the “time out” facility provided in SERIN command. With the help of this facility, we could send a signal indicating that the agent has not received the signal and thus the control station will again send the signal.

Because of our communication protocol, it was not necessary for us to use the encoder and decoder chips with the transmitter and receiver modules. We also tried testing the communication protocol by putting some obstacle between the Fire-Stick and IR sensor. Because of this, the mobile agent will not get the signal and send error message to base station.

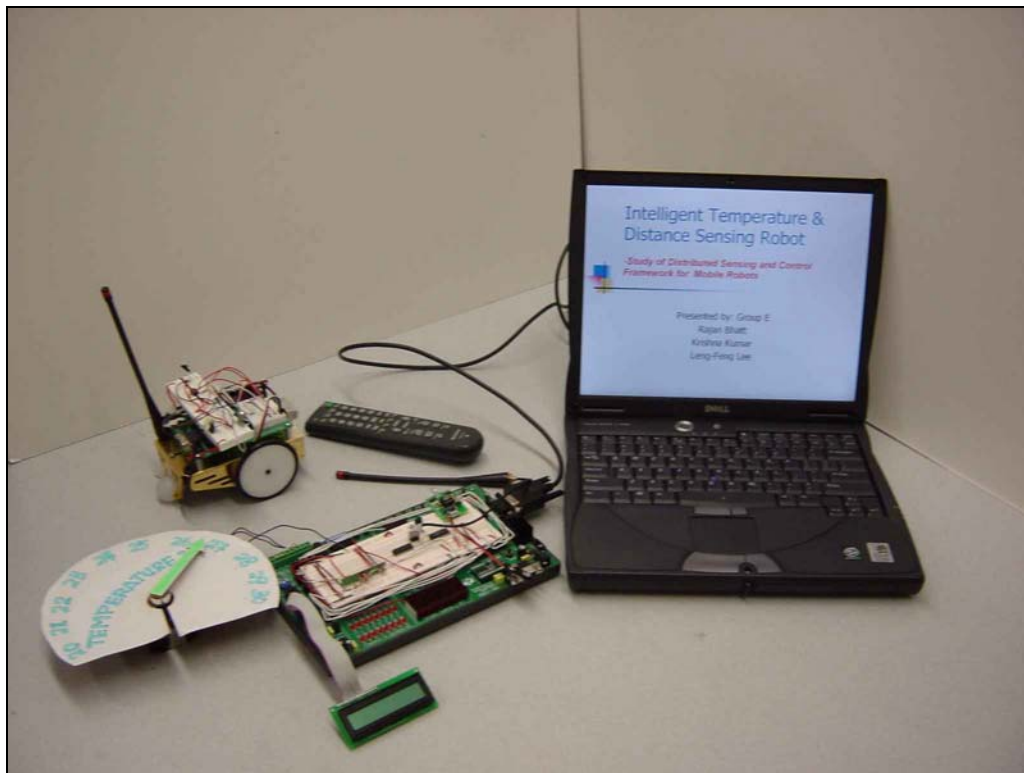


fig 14. complete system

Part B. Autonomous Mode

Introduction:

In this mode the mobile robot is set to navigate autonomously with the help of the infrared distance-measuring sensor. This makes the robot suitable to be employed for surveillance in the foreign environment.

Circuit:

Individual Tasks:

The following tasks were done in the assembly of the robot and mounting accessories.

1. The mobile robot is assembled according to the manual and the wheels are mounted over the servomotors.
2. The wall mounted power source was used to power the robot as it provides constant supply of voltage.

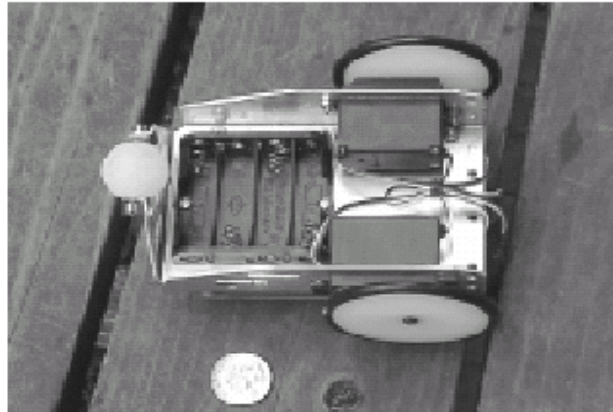


fig 15. Robot Chassis with Wheels and Servos Mounted

3. The BOE (Board of Education), which holds the voltage source and sink, and the Basic Stamp II was mounted.

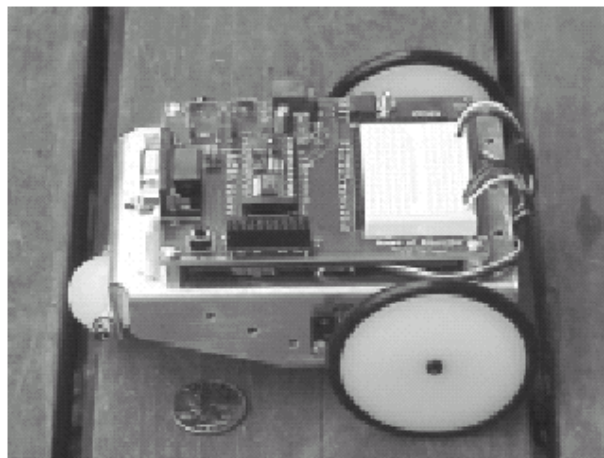


fig 16. Robot with BOE mounted

4. The Basic Stamp II on the BOE was checked for proper functioning by running the test program.
5. The servos were connected to the BOE and a constant voltage is provided through the BOE. The pin15 and pin13 were used in the project.

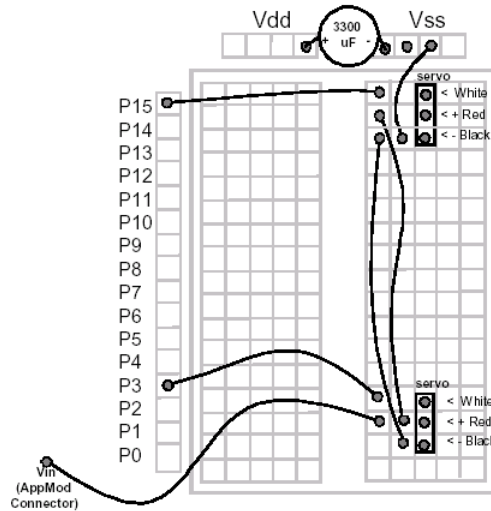


fig 17. BOE with servos connected and powered

6. The temperature measuring chip, DS1620, was mounted on the board according to the recommendation. Pin 10,8,9 were used in the project correspondingly for pin 0,1,2 in the figure.

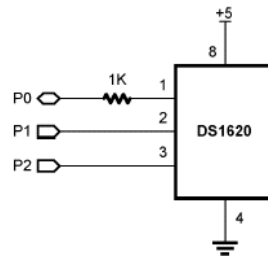


fig 18 Recommended DS1620 circuit

7. The Infrared Distance Measuring Sensor was mounted on the Robot as per the directions in the front end of the chassis and connections made as per recommendation. Pin4 and Pin3 were used in the project.



Fig 19. Recommended Connections for IR Distance Sensor

8. The RF transmitter was mounted on the BOE and fastened mechanically to the chassis. The connections were given as per recommendation. Pin1 was used in the project.

Combining Tasks:

Once the assembling of the robot was done as individual task, the testing of every component to ensure its proper working was performed. After testing the individual components were combined to perform designated task. The testing procedure was as follows:

1. The servos were tested for proper working and correct calibration. Sample program was run on both the servos. The values in the pin 15 and 13 were varied and the effect were observed to ensure the proper working

```
for counter = 0 to 100
pulsout 15,622
pulsout 13,858
next
```

2. The digital thermometer DS1620 was checked for proper functioning by measuring the temperature and checking it in debug window. The temperature measured is stored in tempin variable and which is converted to obtain the temperature reading. The chip is slightly heated and increases in temperature noted to ensure proper functioning.

```
GetTemperature:
HIGH Reset                                ' alert the DS1620
SHIFTOUT DQ,Clock,LSBFIRST,[RdTmp]        ' give command to read temp
SHIFTIN DQ,Clock,LSBPRE,[tempIn\9]        ' read it in
LOW Reset                                  ' release the DS1620
tSign = sign                               ' save sign bit
tempIn = tempIn/2                          ' round to whole degrees
IF tSign = 0 THEN NoNeg1
tempIn = tempIn | $FF00                    ' extend sign bits for negative

NoNeg1:
tempC = tempIn                             ' save Celsius value
tempIn = tempIn * / $01CC                  ' multiply by 1.8
IF tSign = 0 THEN NoNeg2                  'if negative, extend sign bits
tempIn = tempIn | $FF00

NoNeg2:
tempIn = tempIn + 32                       ' finish C -> F conversion
tempF = tempIn                             ' save Fahrenheit value
RETURN
Debug ? tempin                             ' Shows the temp in debug window
```

3. The Infrared distance measurement sensor is checked for proper functioning by varying its distance from an obstacle and measuring the distance of the obstacle. The calibration chart from the manufacturer is used as reference and values are converted into actual distance. The non linearity of the calibration curve was dealt by breaking the curve into linear regions. The calibration formulas obtained were

```
' Calibration formula selected based on the value obtained
if value <= 67 then calib1
if value <=109 then calib2
if value <=147 then calib3
if value <=167 then calib4
if value <=200 then calib5
```

```

return
' Calibration formula
calib1:
dist = 1974 - 5375/100*value + (value*value*375/1000)
return
calib2:
dist = 275 - (447*value/100) + (value*value*2/100)
return
calib3:
dist = 98 - value + (value*value*3/1000)
return
calib4:
dist =12
return
calib5:
dist=10
return
Debug ? dist ' Displays the distance

```

4. The RF transmitter is checked for proper functioning by giving out the data from the mobile robot and displaying the received data using the debug window through computer connected to the base station.

Mobile Segment Part

```

serout 1,baud,["GHI",111,dist,tempC] ' Giving out data
debug ? dist ' display send dist data
debug ? tempC ' display send temp data

```

Base Station Part

*****Refer to the topic on Master-Slave mode on this report *****

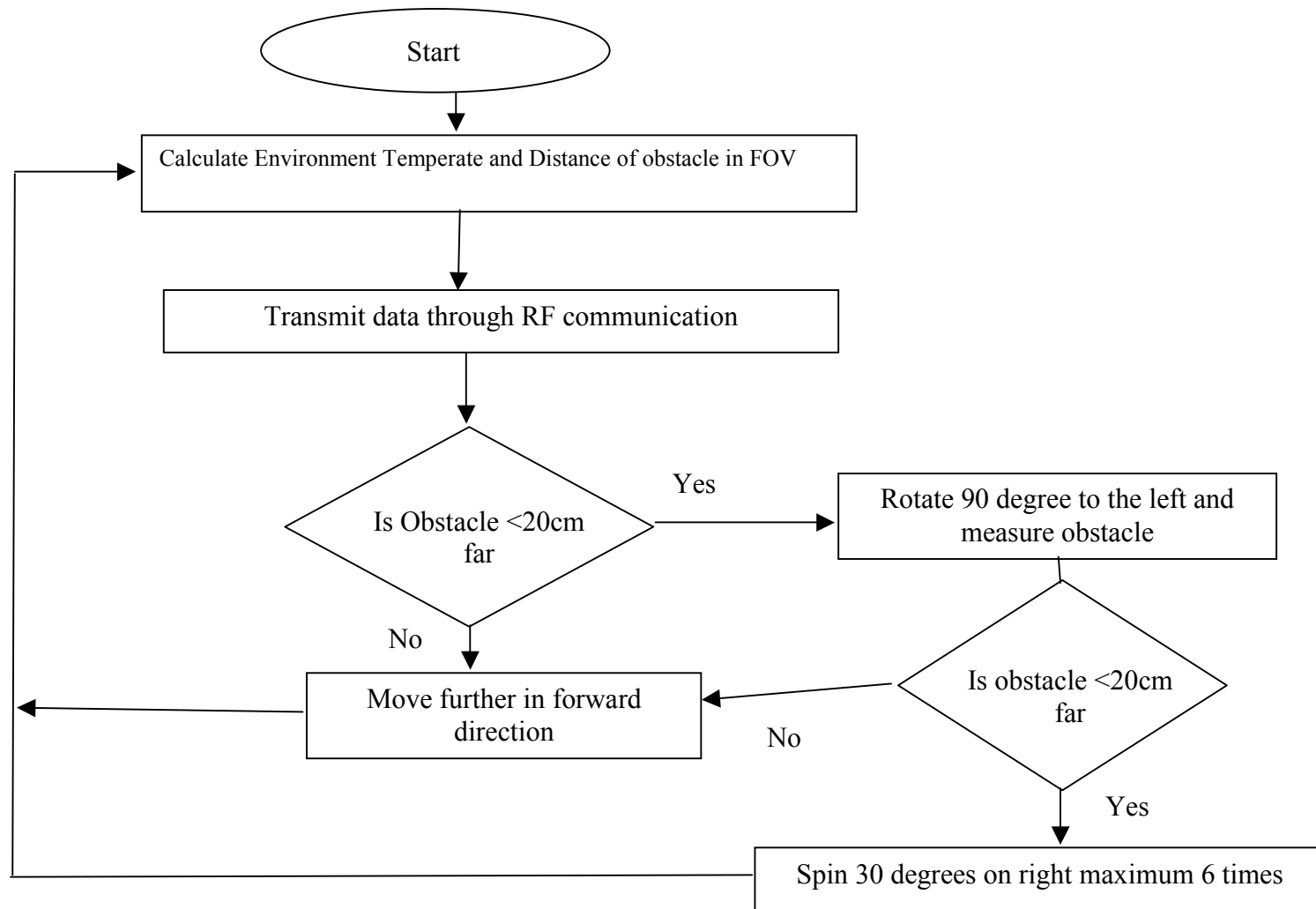
Discussion:

After combining the different tasks the mobile robot does the function of data collection by navigating itself autonomously through a foreign environment without ending in collision. The working is explained in steps below:

1. The robot is powered through a wall mounted power supply
2. The robot measures the temperature of the environment it is in
3. The robot measures its distance from the first obstacle in the line of sight of the infrared distance measurement sensor
4. The temperature and the distance of the obstacle data was send to the base station through the RF transmitter
5. The distance of the first obstacle in the line of sight is measured again
6. If the precision of the value obtained is less than 5 cm different then proceed else keep checking until consistent value obtained. This is used to check the error in measurement
7. If the distance of the first obstacle is more than 25cm the robot moves forward for a distance of 10 cm
8. The steps 2 to 6 are repeated and if the distance of the first obstacle is less than 25cm then the robot has to decide which way to move as it will hit the obstacle in next few steps

9. The safe distance is given as 25cm as the Infrared distance-measuring sensor is effective in the range of 15 to 60 cm and the step size for straight motion is 10cm
10. Once the obstacle is found in the vicinity of 25cm or less, the robot is designed to look for the farthest obstacle around it.
11. The robot is programmed to look 200 degrees around it and find the farthest obstacle.
12. The 200 degrees field of view is obtained by the robot spinning 90 degrees to the right and then spinning 30 degrees in steps
13. For every spin it makes it measures the distance and temperature and transmits to the base station.
14. Again at every location in the 200 degree FOV the robot measures the distance of the first obstacle
15. If the distance measured is more than 25cm the robot continues in that direction
16. Thus the robot avoids obstacles and transmit the temperature and distance data after sensing at every location

Flow-Chart: The flowchart explaining the operation of the mobile robot is depicted below:



Coding:

```

'{$STAMP BS2}
=====
' This Program sets the Mobile Robot in Autonomous Mode
=====
'Declaration of Variable and Constants
-----
counter      var      byte      'determines the step size
value var      byte      'stores the output from the IR distant
                                ranging sensor
dist         var      byte      'Value converted to actual distance
                                stored in this
t            var      byte      'specifies the max time for distance
                                ranging
i            var      byte      'loop counter
j            var      byte      'loop counter
tempdis var      byte      'temporary variable to check consistency
                                of measurement
baud         con      17197
DQ           con      10      'Data pin of DS 1620
Clock CON     8          'Clock pin of DS 1620
Reset CON     9          'Reset pin of DS 1620

tempIn VAR    Word      ' raw temperature
sign VAR     tempIn.Bit8 ' 1 = negative temperature
tSign VAR    Bit
tempC VAR    Word      ' Celsius
tempF VAR    Word      ' Fahrenheit

RdTmp CON    $AA      ' read temperature
WrHi CON     $01      ' write TH (high temp)
WrLo CON     $02      ' write TL (low temp)
RdHi CON     $A1      ' read TH
RdLo CON     $A2      ' read TL
StartC CON   $EE      ' start conversion
StopC CON    $22      ' stop conversion
WrCfg CON    $0C      ' write config register
RdCfg CON    $AC      ' read config register

-----
'initialize
-----
gosub boot_subroutine 'Subroutine to use reset button on
                        Mobile as on/off switch

dir3 =%1
dir4 =%0
GOSUB DS1620init      ' initialize the DS1620
dist = 0
larger = 10

Autonomous:          'Main loop for autonomous mode

GOSUB measuredistance 'Measure first obstacle in the line of
                        sight
GOSUB Gettemperature  'Measure temperature of the current
                        location
serout 1,baud,["GHI",111,dist,tempC] 'Transmit through RF to
                        the base station the dist and temp data
tempdis=dist         'Store dist and check again to eliminate
                        error
GOSUB measuredistance
if not (tempdis-dist)<5 then independent 'Error margin set
                                        as 5 cm
if dist > 20 then straight 'If obstacle >20cm away move straight
if dist <=20 then decide  'If obstacle <20cm away decide the
                        direction to move

debug ? dist,cr
goto independent

Decide:              'Subroutine that decides the direction to move
gosub spind          'Spin 90 degrees to the right

```

```

pause 100                                'Wait for no reason
gosub measuredistance 'Measure the first obstacle
gosub direction          'Goto direction subroutine to find if the
                        direction safe
for j = 1 to 6          'in the loop to determine direction
  gosub direction      'goto direction subroutine again
  gosub spin1          'spin 30 degrees to the left
  gosub measuredistance 'measure distance of first obstacle
pause 500              'Wait for no reason
next
goto independent

Measuredistance:      'Subroutine to measure distance of first obstacle

high 3
pause 3
low 3
for t = 1 to 30      'Time for the IR to send and receive modulated
                    signals
pause 1
if In4=1 then Jump  'If Obstacle reflects then find the distance
next

Jump:                'Subroutine to receive the reflected signal and
                    convert to distance
shiftn 4,3,2,[value\8] 'Receive the reflected signal
pause 500
if value <= 67 then calib1 'Conversion of the value to the
                           actual distance

if value <=109 then calib2
if value <=147 then calib3
if value <=167 then calib4
if value <=200 then calib5
return

calib1:
dist = 1974 - 5375/100*value + (value*value*375/1000)
return
calib2:
dist = 275 - (447*value/100) + (value*value*2/100)
return
calib3:
dist = 98 - value + (value*value*3/1000)
return
calib4:
dist =12
return
calib5:
dist=10
return

Direction:          'Determines if the direction is safe to move forward

if dist>20 then straight
return

Straight:           'move forward in straight line
debug "straight",cr
  for counter = 0 to 100
    pulsout 15,622
    pulsout 13,858
  next
goto independent

Spind:              'Spin about 90 degrees to the left
debug "spind",cr
  for counter = 0 to 130
    pulsout 15,612
    pulsout 13,622
  next
return

spin1:              'Spin about 30 degrees to the right
debug "spin1",cr

```

```

    for counter = 0 to 33
      pulsout 15,858
      pulsout 13,828
    next
  return

DS1620init:
  HIGH Reset          ' alert the DS1620
  SHIFTOUT DQ,Clock,LSBFirst,[WrCfg,%10] ' use with CPU;
                                          free-run

  LOW Reset
  PAUSE 10
  HIGH Reset
  SHIFTOUT DQ,Clock,LSBFirst,[StartC] ' start conversions
  LOW Reset
  RETURN

GetTemperature:
  HIGH Reset          ' alert the DS1620
  SHIFTOUT DQ,Clock,LSBFIRST,[RdTmp]' give command to read temp
  SHIFTOUT DQ,Clock,LSBPRE,[tempIn\9]' read it in
  LOW Reset          ' release the DS1620
  tSign = sign       ' save sign bit
  tempIn = tempIn/2  ' round to whole degrees
  IF tSign = 0 THEN NoNeg1
  tempIn = tempIn | $FF00 ' extend sign bits for negative

NoNeg1:
  tempC = tempIn     ' save Celsius value
  tempIn = tempIn */ $01CC ' multiply by 1.8
  IF tSign = 0 THEN NoNeg2 ' if negative, extend sign bits
  tempIn = tempIn | $FF00

NoNeg2:
  tempIn = tempIn + 32 ' finish C -> F conversion
  tempF = tempIn      ' save Fahrenheit value
  RETURN

boot_subroutine:      ' Subroutine label.

  eeprom_val var byte ' Declare a byte to store eeprom values.

  read 500, eeprom_val ' Read from EEPROM address 500 and store
                      in eeprom_val variable.

  eeprom_val = eeprom_val + 1 ' Increment eeprom_val.

  write 500, eeprom_val      ' Write this value back to EEPROM
                              address 500.

  if eeprom_val > 1 then go_to_sleep ' eeprom_val will be
                                      greater than one every-
                                      other time you press
                                      the BOE's reset button.

  return                    ' Return and continue the program if eeprom_val > 1.

go_to_sleep: ' Label used to for suspending Boe-Bot operation when
             eeprom_val > 1

  eeprom_val = 0          ' Set eeprom_val to zero.
  write 500, eeprom_val  ' Write to memory so next press of
                          reset button will toggle the Boe-
                          Bot "on" and cause it to execute
                          its main routine.

  end                    ' Stops program execution and
                          places BASIC Stamp 2 in low power
                          mode.

```

Discussion:

The design of the autonomous navigation of the robot was achieved after fine calibration and overcoming problems. In this section we would like to discuss the development of this mode and the problems faced.

One important criterion is the logic behind avoiding the obstacles. Two ways were discussed to be implemented, one that is used now and the other were to store the distance of the obstacles around and move in the direction of the farthest obstacle. The storing of the distance of the obstacle around and choosing the best direction was implemented and required storing of the values. The largest value among those stored was chosen and the robot was programmed to retract to that appropriate location.

This method required very fine calibration to retract to the original position. There were errors in the calibration of the servo which hindered the exact 30 degree spin of the mobile robot and slip of the wheel contributed to the error. This method doesn't work well when circular boundaries are encountered.

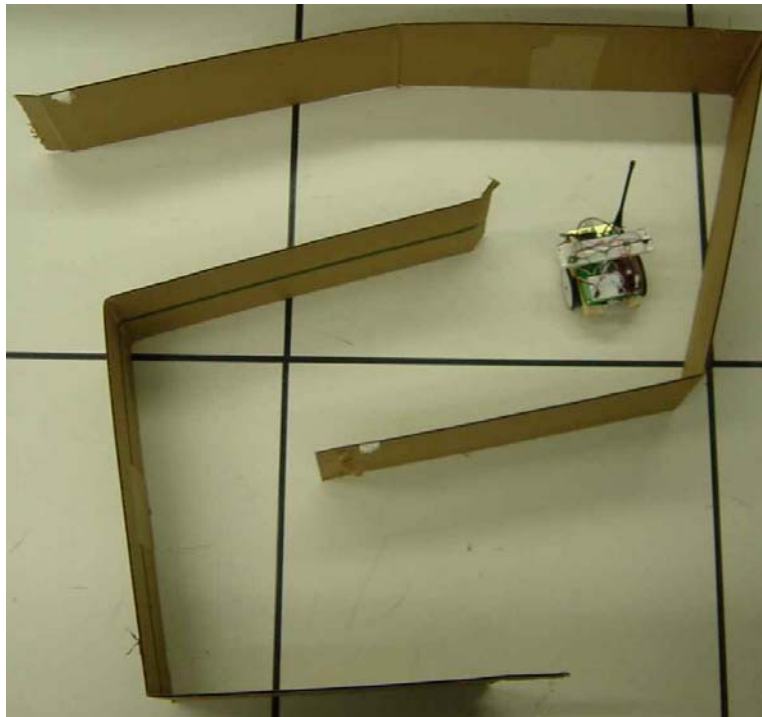
The next method and the method that was implemented will take the first available safe direction. The working of the autonomous mode clearly explains the sequence of operation. In this mode the calibration was made fine and errors weren't magnified, as the robot wasn't required to retract to the best direction.

This autonomous navigation mode had some limitation too due to the short field of view of the infrared distance measuring sensor. The sensor itself had field of view width of 5 to 10cm. This caused problem when the robot encounters slant boundaries where the sensor didn't detect the obstruction. The infrared distance measurement sensor worked well for the region of 15 to 60cm. This limited the robot to stay at least 20cm away from the obstacle to properly detect.

Testing & Evaluation of Autonomous Mode

The robot was tested after combining and checked if the designated task is performed. The Robot performed the following tasks:

1. The distance of the first obstacle was measured and the temperature measured.
2. The temperature and distance data was transmitted
3. The distance of the obstacle was assured and decision was made to move straight.
4. After moving a step size of 10cm the robot again measured the temperature and the distance and transmitted to the base.
5. The distance was now less than 20cm and so the safe direction was selected and moved straight.
6. The temperature and the direction data were transmitted during the direction selection too.



The Mobile Robot in Action (autonomous mode)

OVERALL FRAMEWORK:

FUTURE WORK:

COMPONENTS USE IN THIS LAB AND ITS PRICE:

	Components	Price	Source
1	BOE bot Kits	\$186.15	www.parrallaxinc.com
2	SONY RM-V201	\$9.90	www.sel.sony.com

3	Computers	Free	System laboratory, ARM lab
4	IS1U60 IR receiver	\$3.25	www.hvwtech.com/sensors.htm
5	4 AA size batteries	Free	Provided by Group B.

CONTRIBUTION OF EACH MEMBER:

1	Lee Leng Feng	Task 1 & 3, Programming, report writing.
2	Rajankumar Bhatt	Task 1 & 2 & 3, programming, report writing, assembly.
3	Krishnakumar A Ramamoorthy	Task 1 & 2, programming, report writing.
Calibration and testing was done by all as a group		

CONCLUSION:

REFERENCES:

- [1] Boe Bot Kits.
http://www.parallaxinc.com/html_files/products/StampsIC/boe-bot_brief.asp
- [2] Sony Universal remote control. RM-V201.
<http://www.sel.sony.com/SEL/consumer/ss5/home/accessories/universalremote/rm-v201.shtml>
- [3] Fire-Stick serial IR Communications
<http://www.rentron.com/FS-2.htm>
- [4] Boe bot Kits reference.
http://www.stampsinclass.com/html_files/sic_curr/curriculum_robo.asp
- [5] Robotics! Student workbook. Version 1.5. Parallaxinc educational materials.
<http://www.stampsinclass.com/downloads/Robo/rob.pdf>
- [6] Sharp IS1U60 Data Sheet.
<http://www.hvwtech.com/dnload/datasheets/is1u60.pdf>
- [7] IR code for Sony remote
<http://cgl.bu.edu/GC/shammi/ir/>
- [8] RF Remote Control
http://www.rentron.com/Stamp_RF.htm
- [9] Stepper motor
http://www.parallaxinc.com/downloads/Documentation/Unipolar Stepper Motor/Unipolar stepper Motor 1_1.PDF
- [10] First, Second and Third Lab reports
http://www.eng.buffalo.edu/Courses/mae505/LAB_REPORTS
- [11] Sharp GP2D02 Infrared Ranger
http://www.hvwtech.com/dnload/DIRRS_1_4.pdf
- [12] Sharp GP2D02 Interface to a Basic Stamp II
<http://www.acroname.com/robotics/info/examples/GP2D02-4/GP2D02-4.html>
- [13] Digital Thermometer and Thermostat – DS1620
http://www.physics.brandeis.edu/phys32b_2002/DS1620.pdf
- [14] Basic Stamp Manual
<http://www.parallaxinc.com/downloads/Documentation/Basic Stamps/BASIC Stamp Manual v2.0.pdf>
- [15] Stamp-Plot Lite software and its Tutorial
http://www.parallaxinc.com/html_files/products/StampsIC/stamp_plot.asp