Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC Approved by AICTE &
Permanently Affiliated to JNTUH
Estd. 2001

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

SKETCH WITH ARDUINO

STARTS ON January 2, 2022

In association with TLC

Registration : Free

Course Duration : 4 Week

Weekend Course (Saturday)

Invited Participants: Third Year ECE, EEE, CSE

Restricted to 30 Participants/Slot

Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Mr. E.Parasuramu	Prof.k.Ashok Babu	Dr.G.Suresh
9989575859		

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

Sketch with Arduino

Date: From 02.01.2022 (4 Week Course, Only on Saturday)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4
1	18D41A0447	Chilkur Teja Vardhan Reddy	Teja Vardhan	Teja Vardhan	A	Teja Vardhan
2	18D45A0403	Etikyala Arunkumar				
3	18D41A0490	Kamsetty Vinay				
4	17D41A0402	Jatavath Subhash Naik				
5	18D41A04J7	Saluvala Bharath Kumar				
6	19D41A0427	Bheemreddy Sanjana	B. Sanjana	B. Sanjana	B. Sanjana	B. Sanjana
7	19D41A04C9	Namani Sindhu				
8	18D41A0429	Sowmya Boddu		A		
9	17D41A0498	Kalavakuri Mahesh Babu				
10	17D41A04A4	Kathi Kartheek Reddy				
11	17D41A04L6	T.Sanjana				
12	17D41A04E9	Muppidi Pramod				
13	18D45A0432	Jangili Raghavendra				
14	17D41A04H9	R Sai Abhinav Goud				
15	17D41A04H7	Quazi Mohammad Abdul Raheem Siddique				
16	17D41A04N2	Vemuri Moses Abhishek				A
17	19D41A0412	Anemoni Shirisha Mudhiraj				

18	17D41A04L7	T Kavya	T. Kavya	T. Kavya	T. Kavya	T. Kavya	
19	17D41A04J5	Rangaraju Rahul	Rangaraju	Rangaraju	Rangaraju	Rangaraju.	
20	19D41A0441	Dachepally Sasri	Sisi	Sisi	Sisi	(A)	
21	19D41A04A0	Lunavath Rahul Naik					
22	19D41A0435	B.Sainath	Sainath	Sainath	Sainath	Sainath	
23	17D41A04G9	Pasunuri Shiva	shiva	shiva	shiva	shiva	
24	19D41A0436	Buduru Gayatri	Gayatri	Gayatri	Gayathri	Gayatri.	
25	19D41A0417	Ashwala Madhuri	madhuri				
26	19D41A04A3	Mahesh Bhukya	Mahesh	Mahesh	Mahesh	Mahesh.	
27	18D41A0451	Rudhinika	Rudhinika	Rudhinika	Rudhinika	Rudhinika.	
28	17D41A04G1	Nimisha Reddy	Nimisha Reddy	Nimisha Reddy	Nimisha Reddy	Nimish Reddy.	
29	19D41A0424	Beemagani Dedeepya	Dedeepya	Dedeepya	Dedeepya	Dedeepya	
30	17D41A04C5	L.Pruthvi Kiran	Pruthvi Kiran	Pruthvi Kiran	Pruthvi Kiran	Pruthvi Kiran	
31	17D41A04B2	Kontham Lasya Reddy	Lasya Reddy	Lasya Reddy	Lasya Reddy	Lasya Reddy	
			(31)	(30)	(30)	(29)	

Coordinator

Convener

HOD/ECE

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE
ON

SKETCH WITH ARDUINO

STARTS ON January 2, 2022

In association with TLC

Registration : Free
Course Duration : 4 Week
Weekend Course (Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 30 Participants/Slot
Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Mr. E.Parasuramu	Prof.k.Ashok Babu	Dr.G.Suresh
9989575859		

The easiest way for beginners to get started with Arduino is by creating circuits using a solderless breadboard. These simple projects will teach you the basics of Arduino Uno, electronics and programming. In this tutorial, you will be creating circuits using the following electronic components:

- LED
- RGB LED
- Temp Sensor
- Pushbutton
- Potentiometer
- Photoresistor
- Servo
- Motor
- Buzzer
- LCD screen

This tutorial is going to allow you to jump right in and start building circuits. If you need some background on the Arduino Uno board or the tools that are needed, please check out post – Arduino Uno For Beginners.

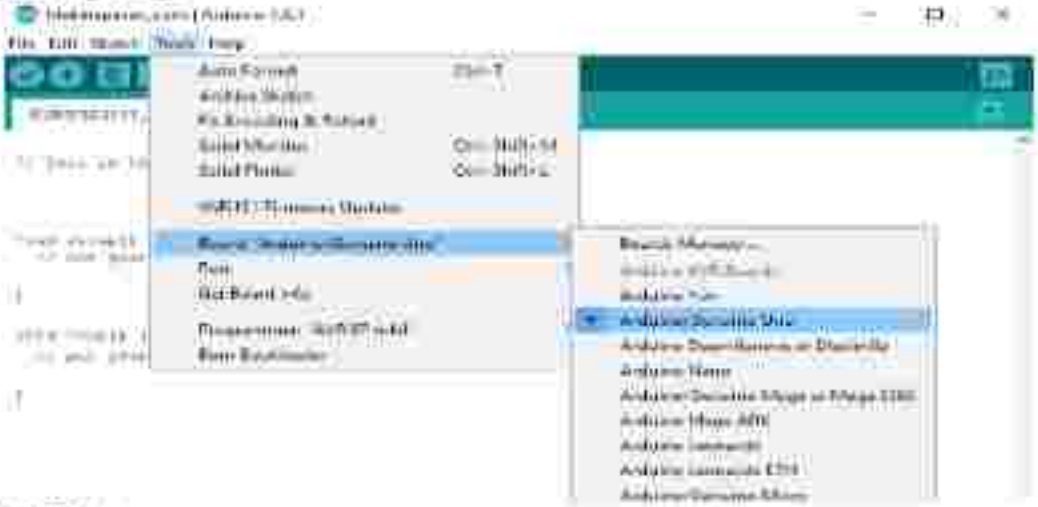
Getting Started

Before you can start working with Arduino, you need to make sure you have the IDE software installed on your computer. This program allows you to write, view and upload the code to your Arduino Uno board. You can download the IDE for free on Arduino's website.

Once the IDE is installed, you will need to connect your Arduino to your computer. To do this, plug one end of the USB cable to the Arduino Uno and then the other end of the USB to your computer's USB port.

Select The Board

Once the board is plugged in, you will need to open the IDE and click on **Tools > Board > Arduino Uno** to select the board.



Select Serial Port

Next, you have to tell the Arduino which port you are using on your computer. To select the port, go to **Tools > Port** and then select the port that says **Arduino.**

Project Code

To complete the projects in this tutorial, you will need to download the project code which are known as sketches. A sketch is simply a set of instructions that tells the board what functions it needs to perform. For some of these projects, we are using open-source code that was released by the good people at Sparkfun and Arduino. Use the link below to download the zip folder containing the code.

Download Project Code – (ZIP File)

Once the file has been downloaded, you will need to unzip/extract the folder in order to use it.

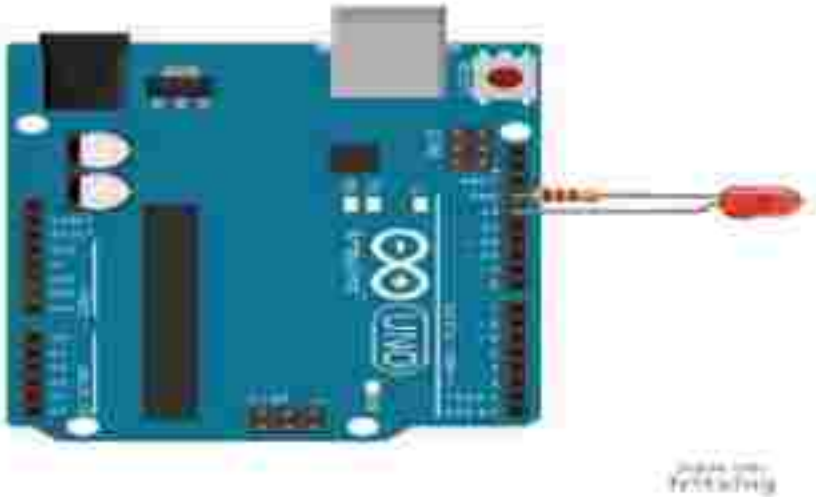
#1 – Test Arduino

The first project is one of the most basic and simple circuits you can create with Arduino. This project will test your Arduino by blinking an LED that is connected directly to the board.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) LED 5mm
- (1) 220 Ω Resistor

Project Diagram



Project Steps

1. Twist a 220 Ω resistor to the long leg (+) of the LED.
2. Push the short leg of the LED into the ground (GND) pin on the board.
3. Push the resistor leg that's connected to the LED into the #13 pin.

Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_01_TestArduino**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

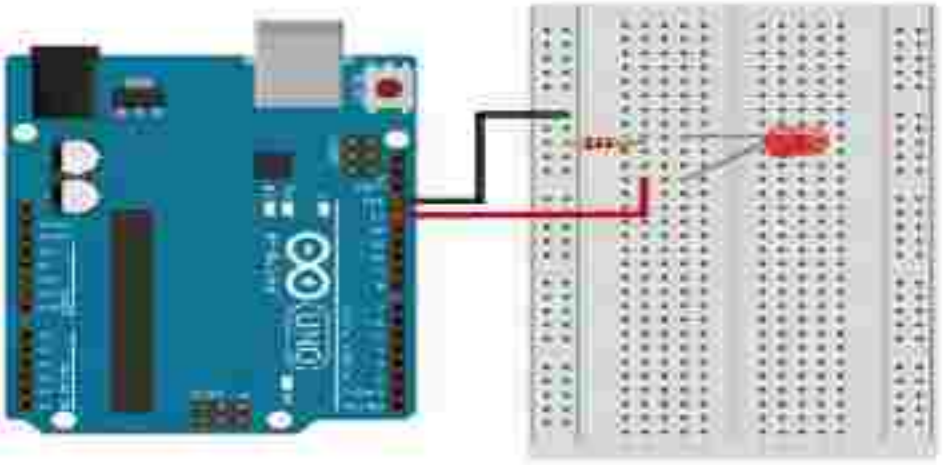
#2 – Blink an LED

This project is identical to project #1 except that we will be building it on a breadboard. Once complete, the LED should turn on for a second and then off for a second in a loop.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (2) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_02_Blink**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

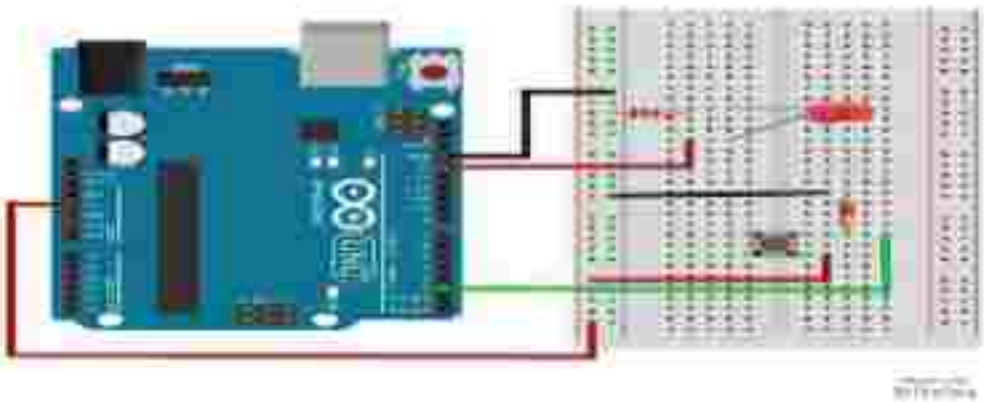
#3 – Push Button

Using a push button switch, you will be able to turn on and off an LED.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (1) 10K Ω Resistor
- (1) Push Button Switch
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_03_Pushbutton**

3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

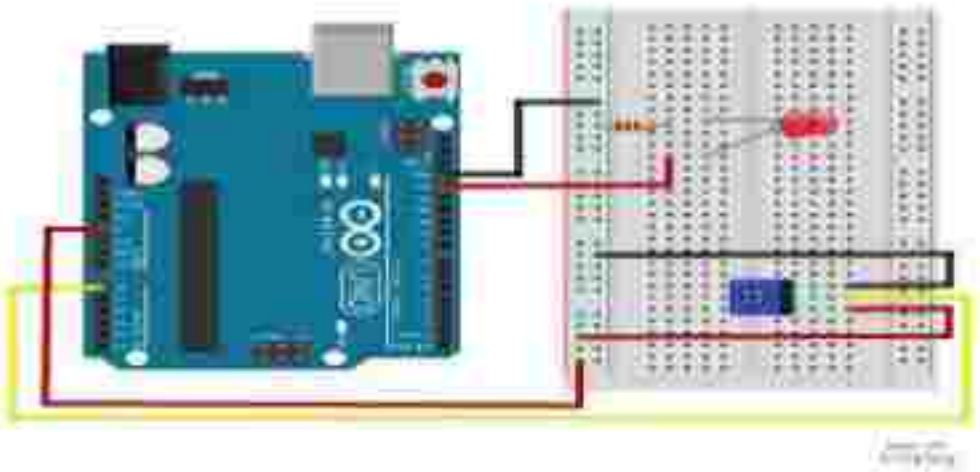
#4 – Potentiometer

Using a potentiometer, you will be able to control the resistance of an LED. Turning the knob will increase and decrease the frequency the LED blinks.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (1) Potentiometer (10k Trimpot)
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_04_Potentiometer**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

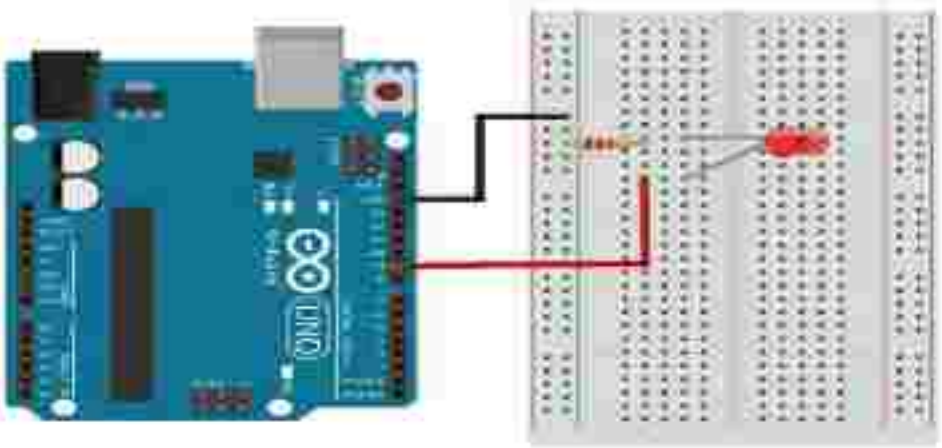
#5 – Fade an LED

By using a PWM pin on the Arduino, you will be able to increase and decrease the intensity of brightness of an LED.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 220 Ω Resistor
- (2) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_05_Fade**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

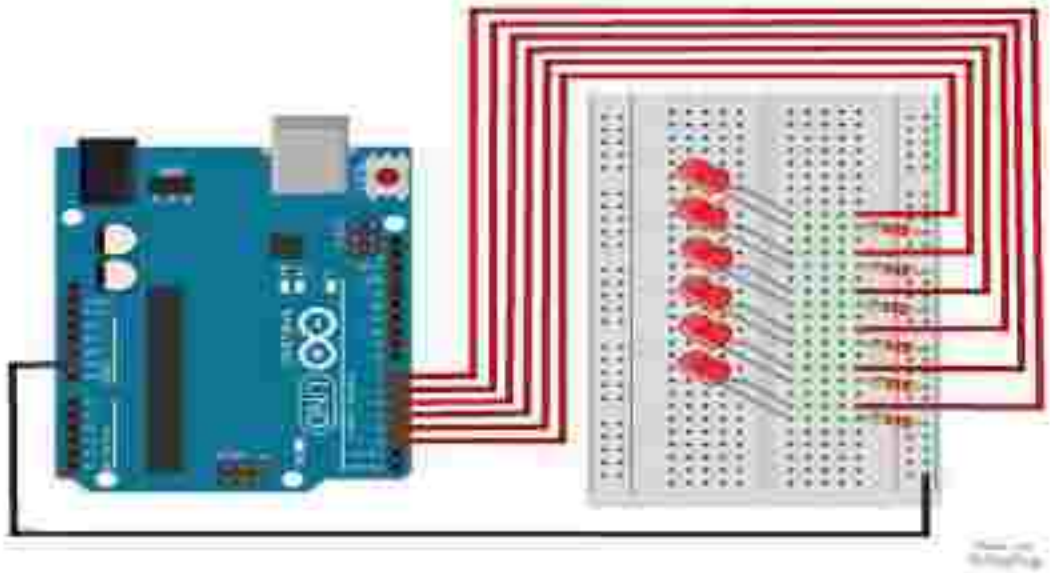
#6 – Scrolling LED

This project will blink 6 LEDs, one at a time, in a back and forth formation. This type of circuit was made famous by the show Knight Rider which featured a car with looping LEDs.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (6) LED 5mm
- (6) 220 Ω Resistor
- (7) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_06_Scrolling**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

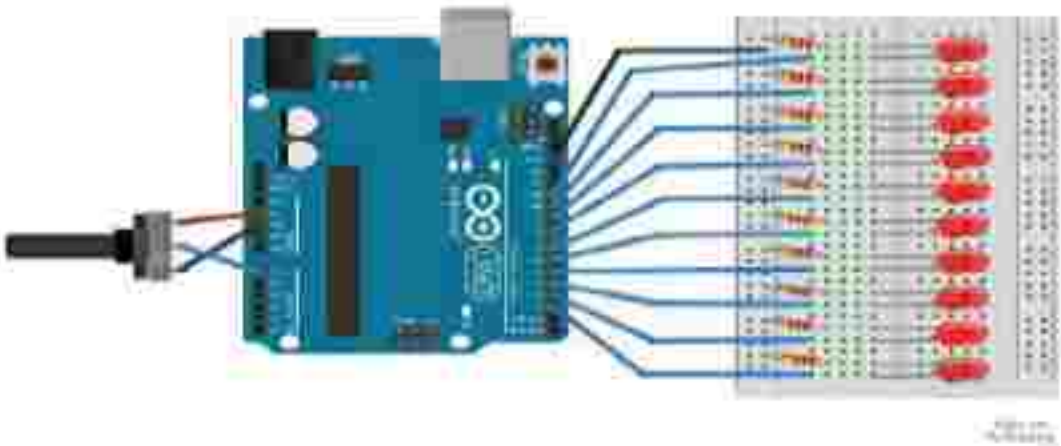
#7 – Bar Graph

Using a potentiometer, you can control a series of LEDs in a row. Turning the potentiometer knob will turn on or off more of the LEDs.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Potentiometer – Rotary
- (10) LED 5mm
- (10) 220 Ω Resistor
- (11) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_07_BarGraph**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

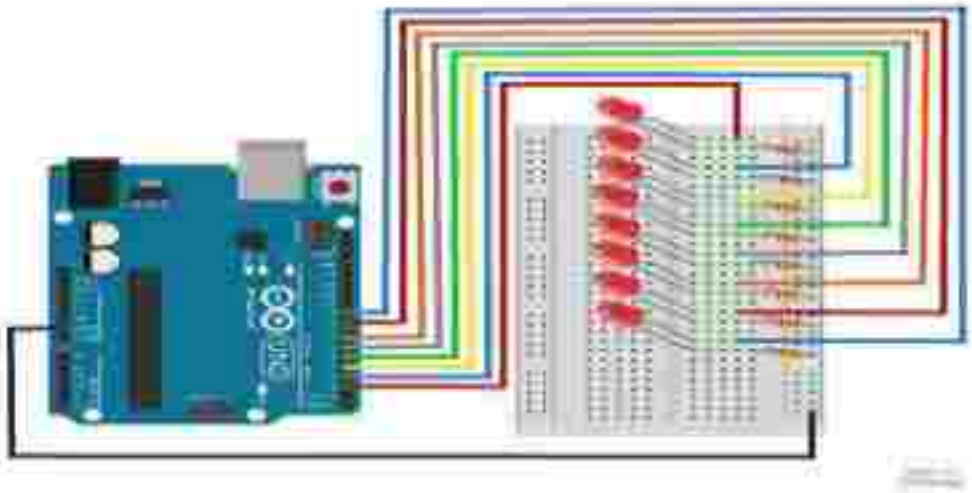
#8 – Multiple LEDs

This project will use 8 pins on the Arduino board to blink 8 LEDs at the same time.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (8) LED 5mm
- (8) 330 Ω Resistor
- (9) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_08_MultipleLEDs**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

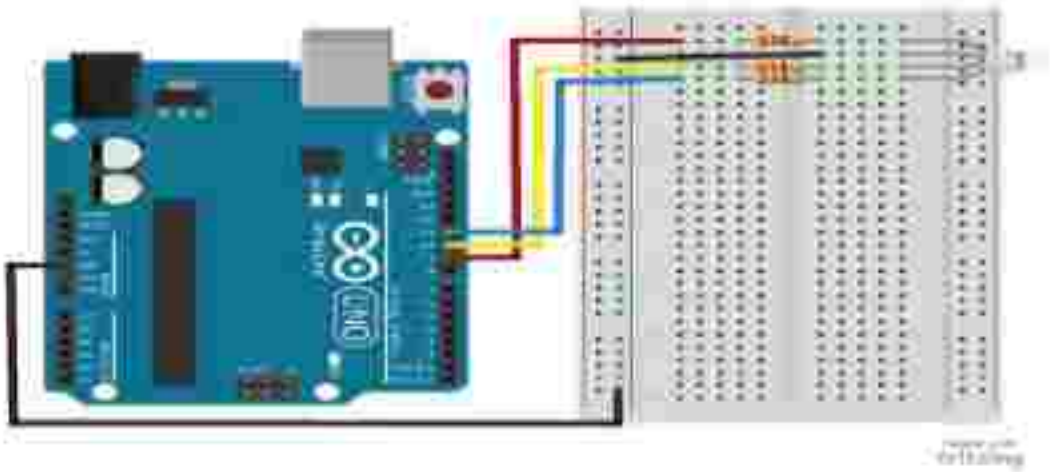
#9 – RGB LED

This project will be using an RGB LED to scroll through a variety of colors. RGB stands for Red, Green and Blue and this LED has the ability to create nearly unlimited color combinations.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) RGB LED
- (3) 330 Ω Resistor
- (5) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_09_RGBLED**

3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

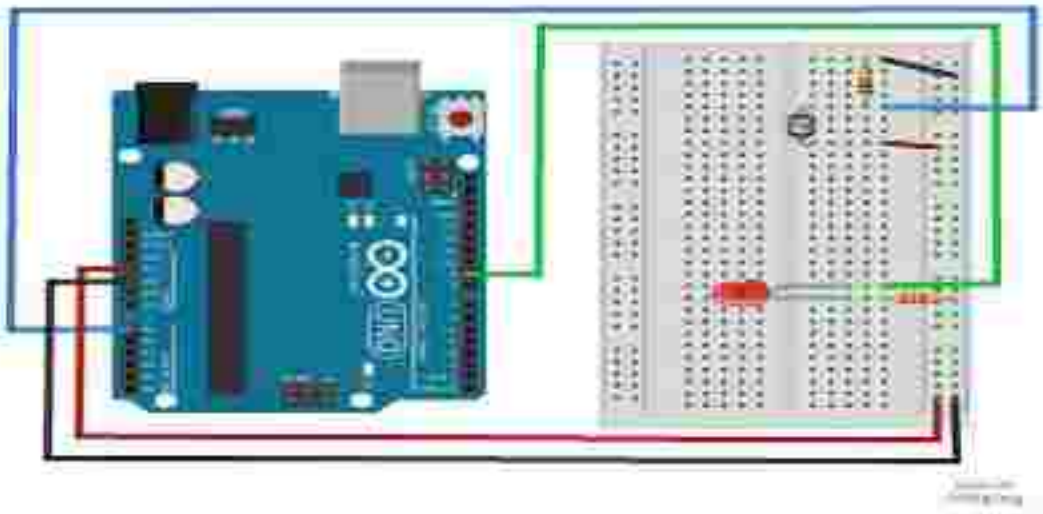
#10 – Photoresistor

A photoresistor changes the resistance a circuit gets based on the amount of light that hits the sensor. In this project, the brightness of the LED will increase and decrease based on the amount of light present.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LED 5mm
- (1) 330 Ω Resistor
- (1) 10K Ω Resistor
- (1) Photoresistor
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_10_Photoresistor**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

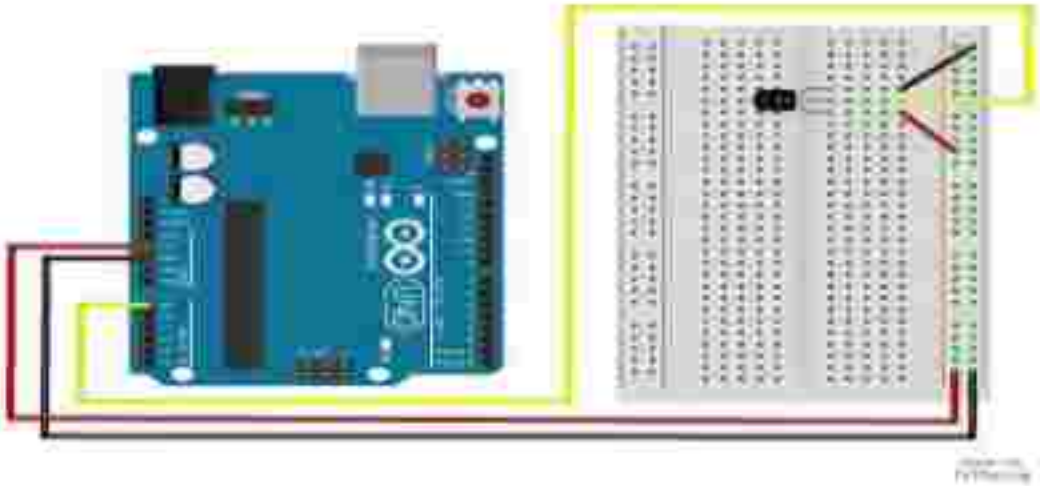
#11 – Temp. Sensor

A temperature sensor measures ambient temperatures of the world around it. In this project, we will be displaying the temperature in the serial monitor of the Arduino IDE.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Temperature Sensor – TMP36
- (5) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_11_TempSensor**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

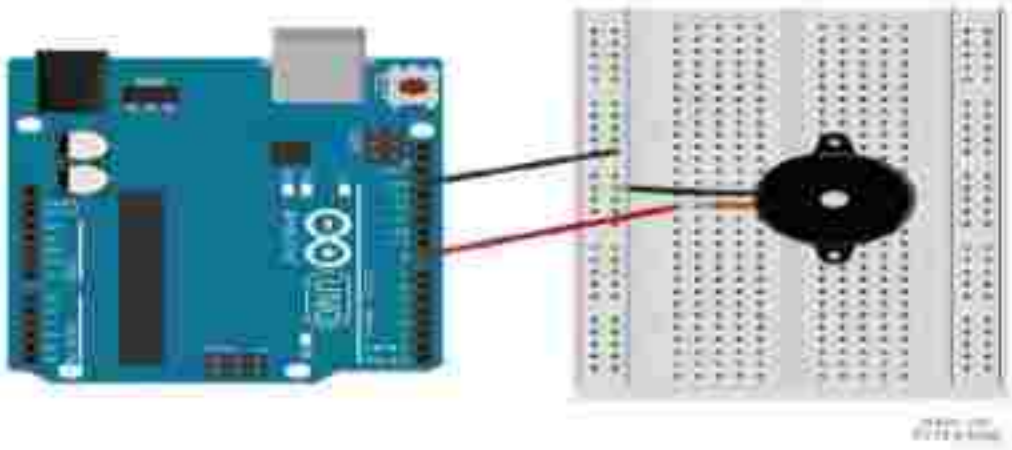
#12 – Tone Melody

The project will use a piezo buzzer speaker to play a little melody.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Piezo Buzzer Speaker
- (2) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_12_ToneMelody**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

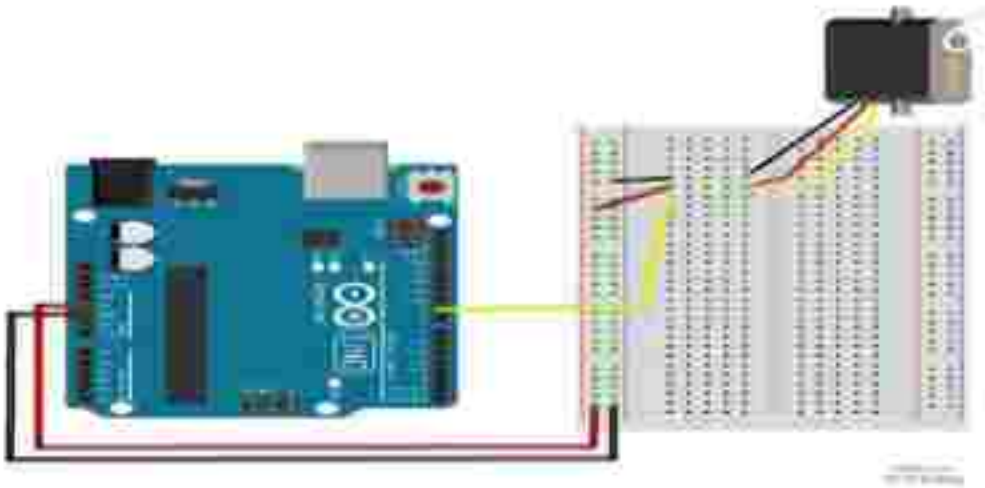
#13 – Servo

In this project, you will be able to sweep a servo back and forth through its full range of motion.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) Servo
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_13_Servo**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

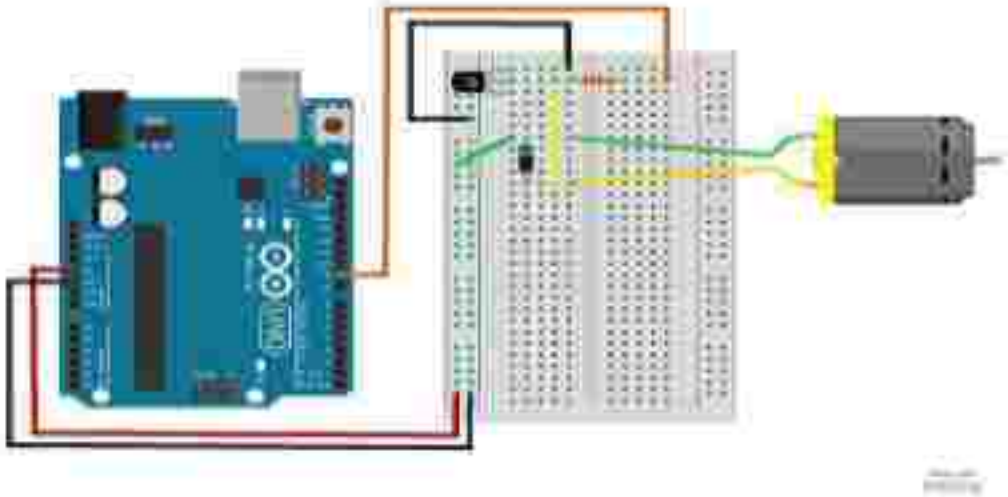
#14 – Motor

Using a switching transistor, we will be able to control a DC motor. If everything is connected correctly, you should see the motor spinning.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) DC Motor
- (1) 330 Ω Resistor
- (1) Diode 1N4148
- (1) NPN Transistor
- (6) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_14_Motor**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

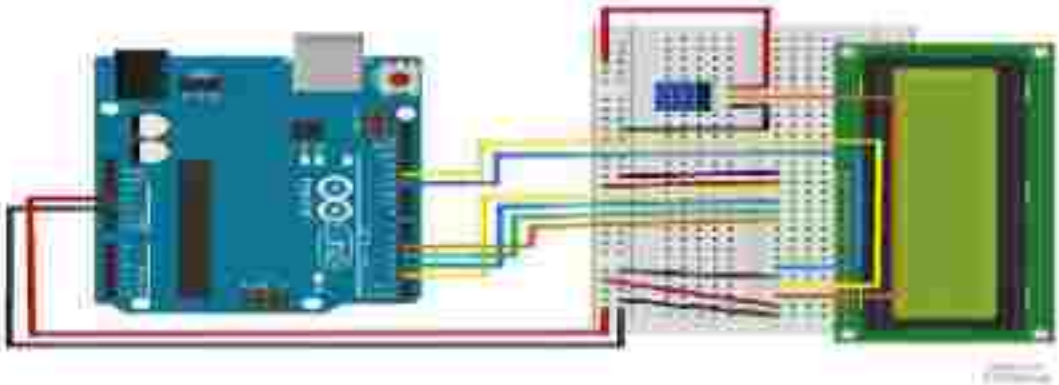
#15 – LCD Screen

An LCD is a liquid crystal display that is able to display text on its screen. In this project, you should see the words "hello,world!" displayed on the screen. The potentiometer is used to adjust the contrast of the display.

Parts Needed

- (1) Arduino Uno
- (1) USB A-to-B Cable
- (1) Breadboard – Half Size
- (1) LCD Screen
- (1) Potentiometer
- (16) Jumper Wires

Project Diagram



Project Code

1. Connect the Arduino board to your computer using the USB cable.
2. Open project code – **Circuit_15_LCD**
3. Select the board and serial port as outlined in earlier section.
4. Click upload button to send sketch to the Arduino.

Troubleshooting

- Make sure your board and serial port is selected in the IDE. To do this, plug your board in and go to **Tools > Board >Arduino** to select your board. Next, go to **Tools > Port >Com (Arduino)** to select your serial port.
- The long leg of the LED is the (+) positive and the short leg is the (-) negative. Make sure the correct leg of the LED is in the proper pin of the Arduino or breadboard as directed.
- It can be easy to put a component or jumper into the wrong pin on the Arduino or the breadboard. Double check the correct pin is being used.

Experiment 1: Turn an LED

Turn an LED on for one second, off for one second, and repeat forever.

```
void setup()
{
 pinMode(13, OUTPUT);
}

void loop()
{
 digitalWrite(13, HIGH);   // Turn on the LED
 delay(1000);              // Wait for one second
 digitalWrite(13, LOW);    // Turn off the LED
 delay(1000);              // Wait for one second
}
*
```

Experiment 2: Turns on and off LED

Turns on and off a light emitting diode(LED) connected to digital
pin 13, when pressing a pushbutton attached to pin 2.

The circuit:
* LED attached from pin 13 to ground
* pushbutton attached to pin 2 from +5V
* 10K resistor attached to pin 2 from ground

* Note: on most Arduinos there is already an LED on the board
attached to pin 13.

```
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
```

```
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Experiment 3: Display RGB LED

```
const int RED_PIN = 9;
const int GREEN_PIN = 10;
const int BLUE_PIN = 11;

const int DISPLAY_TIME = 1000;  // used in mainColors() to determine the
// length of time each color is displayed.

void setup()       // Configure the Arduino pins to be outputs to drive the LEDs
{
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
}

void loop()
{
  mainColors();      // Red, Green, Blue, Yellow, Cyan, Purple, White
  //  showSpectrum();   // Gradual fade from Red to Green to Blue to Red
}

/**************************************************************************
 * void mainColors()
 * This function displays the eight "main" colors that the RGB LED
 * can produce. If you'd like to use one of these colors in your
 * own sketch, you can copy and paste that section into your code.
 **************************************************************************/
void mainColors()
{
  // all LEDs off
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
```

```
  digitalWrite(BLUE_PIN, LOW);
  delay(DISPLAY_TIME);

  // Red
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
  delay(DISPLAY_TIME);

  // Green
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, LOW);
  delay(DISPLAY_TIME);

  // Blue
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, HIGH);
  delay(DISPLAY_TIME);

  // Yellow (Red and Green)
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, LOW);
  delay(DISPLAY_TIME);

  // Cyan (Green and Blue)
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, HIGH);
  delay(DISPLAY_TIME);

  // Purple (Red and Blue)
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, HIGH);
  delay(DISPLAY_TIME);

  // White (turn all the LEDs on)
  digitalWrite(RED_PIN, HIGH);
  digitalWrite(GREEN_PIN, HIGH);
  digitalWrite(BLUE_PIN, HIGH);
  delay(DISPLAY_TIME);
}

/*****************************************************************
* void showSpectrum()
*
* Steps through all the colors of the RGB LED, displaying a rainbow.
* showSpectrum() calls a function RGB(int color) that translates a number
* from 0 to 767 where 0 = all RED, 767 = all RED
*
```

```
* Breaking down tasks down into individual functions like this
* makes your code easier to follow, and it allows
* parts of your code to be re-used.
*****************************************************************/

void showSpectrum()
{
 for (int x = 0; x <= 767; x++)
 {
  RGB(x);     // Increment x and call RGB() to progress through colors.
  delay(10);  // Delay for 10 ms (1/100th of a second) - to help the "smoothing"
 }
}

/*****************************************************************
* void RGB(int color)
*
* RGB(###) displays a single color on the RGB LED.
* Call RGB(###) with the number of a color you want
* to display. For example, RGB(0) displays pure RED, RGB(255)
* displays pure green.
*
* This function translates a number between 0 and 767 into a
* specific color on the RGB LED. If you have this number count
* through the whole range (0 to 767), the LED will smoothly
* change color through the entire spectrum.
*
* The "base" numbers are:
* 0   = pure red
* 255 = pure green
* 511 = pure blue
* 767 = pure red (again)
*
* Numbers between the above colors will create blends. For
* example, 640 is midway between 512 (pure blue) and 767
* (pure red). It will give you a 50/50 mix of blue and red,
* resulting in purple.
*****************************************************************/
void RGB(int color)
{
 int redIntensity;
 int greenIntensity;
 int blueIntensity;

 color = constrain(color, 0, 767); // constrain the input value to a range of values from 0 to 767

  // if statement breaks down the "color" into three ranges:
 if (color <= 255)       // RANGE 1 (0 - 255) - red to green
 {
  redIntensity = 255 - color;    // red goes from on to off
  greenIntensity = color;        // green goes from off to on
  blueIntensity = 0;             // blue is always off
 }
```

```
 else if (color <= 511)  // RANGE 2 (256 - 511) - green to blue
 {
  redIntensity = 0;                  // red is always off
  greenIntensity = 511 - color;          // green on to off
  blueIntensity = color - 256;          // blue off to on
 }
 else                  // RANGE 3 ( >= 512)- blue to red
 {
  redIntensity = color - 512;          // red off to on
  greenIntensity = 0;                // green is always off
  blueIntensity = 767 - color;          // blue on to off
 }

 // "send" intensity values to the Red, Green, Blue Pins using analogWrite()
 analogWrite(RED_PIN, redIntensity);
 analogWrite(GREEN_PIN, greenIntensity);
 analogWrite(BLUE_PIN, blueIntensity);
}
```

Experiment 4: Dancing LED

```
int ledPins[] = {2,3,4,5,6,7,8,9};   // Defines an array to store the pin numbers of the 8 LEDs
// An array is like a list variable that can store multiple numbers.
// Arrays are referenced or "indexed" with a number in the brackets [ ]. See the examples in
// the pinMode() functions below

void setup()
{
  // setup all 8 pins as OUTPUT - notice that the list is "indexed" with a base of 0
  pinMode(ledPins[0],OUTPUT);  // ledPins[0] = 2
  pinMode(ledPins[1],OUTPUT);  // ledPins[1] = 3
  pinMode(ledPins[2],OUTPUT);  // ledPins[2] = 4
  pinMode(ledPins[3],OUTPUT);  // ledPins[3] = 5
  pinMode(ledPins[4],OUTPUT);  // ledPins[4] = 6
  pinMode(ledPins[5],OUTPUT);  // ledPins[5] = 7
  pinMode(ledPins[6],OUTPUT);  // ledPins[6] = 8
  pinMode(ledPins[7],OUTPUT);  // ledPins[7] = 9
}

void loop()
{

  // This loop() calls functions that we've written further below.
  // We've disabled some of these by commenting them out (putting
  // "//" in front of them). To try different LED displays, remove
  // the "//" in front of the ones you'd like to run, and add "//"
  // in front of those you don't to comment out (and disable) those
  // lines.

  oneAfterAnother();  // Light up all the LEDs in turn

  //oneOnAtATime();       // Turn on one LED at a time
```

```
  //pingPong();          // Same as oneOnAtATime() but change direction once LED reaches edge
  //marquee();          // Chase lights like you see on theater signs
  //randomLED();        // Blink LEDs randomly
}


/**************************************************************************
 * oneAfterAnother()
 *
 * This function turns all the LEDs on, pauses, and then turns all
 * the LEDS off. The function takes advantage of for() loops and
 * the array to do this with minimal typing.
 **************************************************************************/
void oneAfterAnother()
{
 int index;
 int delayTime = 100; // milliseconds to pause between LEDs
                      // make this smaller for faster switching

 // Turn all the LEDs on:
 for(index = 0; index <= 7; index = ++index)  // step through index from 0 to 7
 {
  digitalWrite(ledPins[index], HIGH);
  delay(delayTime);
 }

 // Turn all the LEDs off:
 for(index = 7; index >= 0; index = --index)  // step through index from 7 to 0
 {
  digitalWrite(ledPins[index], LOW);
  delay(delayTime);
 }
}

/**************************************************************************
 * oneOnAtATime()
 *
 * This function will step through the LEDs, lighting only one at
 * a time. It turns each LED ON and then OFF before going to the
 * next LED.
 **************************************************************************/

void oneOnAtATime()
{
 int index;
 int delayTime = 100; // milliseconds to pause between LEDs
                      // make this smaller for faster switching

 for(index = 0; index <= 7; index = ++index)  // step through the LEDs, from 0 to 7
 {
```

```
  digitalWrite(ledPins[index], HIGH);  // turn LED on
  delay(delayTime);                    // pause to slow down
  digitalWrite(ledPins[index], LOW);   // turn LED off
 }
}

/*****************************************************************
 * pingPong()
 *
 * This function will step through the LEDs, lighting one at at
 * time in both directions. There is no delay between the LED off
 * and turning on the next LED. This creates a smooth pattern for
 * the LED pattern.
 *****************************************************************/
void pingPong()
{
 int index;
 int delayTime = 100; // milliseconds to pause between LEDs

 for(index = 0; index <= 7; index = ++index)   // step through the LEDs, from 0 to 7
 {
  digitalWrite(ledPins[index], HIGH);  // turn LED on
  delay(delayTime);                    // pause to slow down
  digitalWrite(ledPins[index], LOW);   // turn LED off
 }

 for(index = 7; index >= 0; index = --index)   // step through the LEDs, from 7 to 0
 {
  digitalWrite(ledPins[index], HIGH);  // turn LED on
  delay(delayTime);                    // pause to slow down
  digitalWrite(ledPins[index], LOW);   // turn LED off
 }
}

/*****************************************************************
 * marquee()
 *
 * This function will mimic "chase lights" like those around
 * theater signs.
 *****************************************************************/
void marquee()
{
 int index;
 int delayTime = 200; // milliseconds to pause between LEDs

 // Step through the first four LEDs.
 // (We'll light up one in the lower 4 and one in the upper 4)

 for(index = 0; index <= 3; index++) // Step from 0 to 3
 {
  digitalWrite(ledPins[index], HIGH);   // Turn a LED on
  digitalWrite(ledPins[index+4], HIGH); // Skip four, and turn that LED on
  delay(delayTime);                     // Pause to slow down the sequence
```

```
    digitalWrite(ledPins[index], LOW);    // Turn the LED off
    digitalWrite(ledPins[index+4], LOW);  // Skip four, and turn that LED off
  }
}

/**********************************************************************
 * randomLED()
 *
 * This function will turn on random LEDs. Can you modify it so it
 * also lights them for random times?
 **********************************************************************/
void randomLED()
{
 int index;
 int delayTime;

 index = random(8);  // pick a random number between 0 and 7
 delayTime = 100;

 digitalWrite(ledPins[index], HIGH);  // turn LED on
 delay(delayTime);                    // pause to slow down
 digitalWrite(ledPins[index], LOW);   // turn LED off
}
```

Experiment 5: Running Motor

```
const int motorPin = 9;  // Connect the base of the transistor to pin 9.
                         // Even though it's not directly connected to the motor,
                         // we'll call it the 'motorPin'

void setup()
{
 pinMode(motorPin, OUTPUT);  // set up the pin as an OUTPUT
 Serial.begin(9600);         // initialize Serial communications
}


void loop()
{ // This example basically replicates a blink, but with the motorPin instead.
 int onTime = 3000;  // milliseconds to turn the motor on
 int offTime = 3000; // milliseconds to turn the motor off

 analogWrite(motorPin, 255); // turn the motor on (full speed)
 delay(onTime);              // delay for onTime milliseconds
 analogWrite(motorPin, 0);   // turn the motor off
 delay(offTime);             // delay for offTime milliseconds

 // Uncomment the functions below by taking out the //. Look below for the
 // code examples or documentation.

 // speedUpandDown();
 // serialSpeed();
}
```

```
// This function accelerates the motor to full speed,
// then decelerates back down to a stop
void speedUpandDown()
{
 int speed;
 int delayTime = 20; // milliseconds between each speed step

 // accelerate the motor
 for(speed = 0; speed <= 255; speed++)
 {
  analogWrite(motorPin,speed);          // set the new speed
  delay(delayTime);                     // delay between speed steps
 }
 // decelerate the motor
 for(speed = 255; speed >= 0; speed--)
 {
  analogWrite(motorPin,speed);          // set the new speed
  delay(delayTime);                     // delay between speed steps
 }
}

// Input a speed from 0-255 over the Serial port
void serialSpeed()
{
 int speed;

 Serial.println("Type a speed (0-255) into the box above,");
 Serial.println("then click [send] or press [return]");
 Serial.println();   // Print a blank line

 // In order to type out the above message only once,
 // we'll run the rest of this function in an infinite loop:

 while(true)  // "true" is always true, so this will loop forever.
 {
  // Check to see if incoming data is available:
  while (Serial.available() > 0)
  {
   speed = Serial.parseInt();  // parseInt() reads in the first integer value from the Serial Monitor.
   speed = constrain(speed, 0, 255); // constrains the speed between 0 and 255
                                     // because analogWrite() only works in this range.
   Serial.print("Setting speed to ");  // feedback and prints out the speed that you entered.
   Serial.println(speed);

   analogWrite(motorPin, speed);  // sets the speed of the motor.
  }
 }
}
```

Experiment 6: Potentiometer

```
int sensorPin = A0;    // select the input pin for the potentiometer
```

```
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

Experiment 7: Scrolling LED

```
int timer = 100;          // The higher the number, the slower the timing.

void setup() {
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }

  // loop from the highest pin to the lowest:
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
    // turn the pin on:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(thisPin, LOW);
  }
}
```

Experiment 8: Potentiometer

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

Experiment 9: LED with PWM

```
int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

```
Experiment 10: To measure the temperature sensor's
// signal pin.

const int temperaturePin = A0;

void setup()
{

        Serial.begin(9600); //Initialize serial port & set baud rate to 9600 bits per second (bps)
}

void loop()
{

        float voltage, degreesC, degreesF; //Declare 3 floating point variables

        voltage = getVoltage(temperaturePin); //Measure the voltage at the analog pin

        degreesC = (voltage - 0.5) * 100.0; // Convert the voltage to degrees Celsius

        degreesF = degreesC * (9.0 / 5.0) + 32.0; //Convert degrees Celsius to Fahrenheit

        //Now print to the Serial monitor. Remember the baud must be 9600 on your monitor!
        // These statements will print lines of data like this:
        // "voltage: 0.73 deg C: 22.75 deg F: 72.96"

        Serial.print("voltage: ");
        Serial.print(voltage);
        Serial.print("  deg C: ");
        Serial.print(degreesC);
        Serial.print("  deg F: ");
        Serial.println(degreesF);

        delay(1000); // repeat once per second (change as you wish!)
}
float getVoltage(int pin)           //Function to read and return
                                                        //floating-point value (true voltage)
                                                        //on analog pin
{

        return (analogRead(pin) * 0.004882814);
        // This equation converts the 0 to 1023 value that analogRead()
        // returns, into a 0.0 to 5.0 value that is the true voltage
        // being read at that pin.
}

// Other things to try with this code:

//  Turn on an LED if the temperature is above or below a value.

//  Read that threshold value from a potentiometer - now you've
//  created a thermostat!
```

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956.
NAAC Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE ON IMPLEMENTATION OF IMAGE PROCESSING CONCEPTS FOR REALTIME APPLICATIONS USING MATLAB

STARTS ON September 19, 2021

SLOT-I REGISTRATION OPEN

Registration : Rs.150
Course Duration : 24 Hours
Weekend Course (Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 25 Participants/Slot
Resource Persons: In-house Trainers

Coordinators
Dr.N.C.Sendhilkumar
Dr.P.Mukunthan

Convener
Prof.k.Ashok Babu

Principal
Dr.G.Suresh

Contact: 9443968958,9894145701

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

IMPLEMENTATION OF IMAGE PROCESSING CONCEPTS FOR REALTIME APPLICATIONS USING MATLAB

Date: From 19.09.2021 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	Topics	Resource Person
Week 1	Basics on Image Processing	Dr G.Suresh
	Introduction to Image Processing Tools	
	Types of Image Representation	
	Waveform and Amplitude Spectrum	
	False Contouring	
	Circular correlation between two signals	
	Assignment-1	
Week 2	Program to Interchage phase between two images	Dr.N.C.Senthilkumar
	Program to adjust brightness and contrast level of an image	
	Histogram Analysis of an Image	
	Types of noises and removal	
	Assignment-2	
Week 3	Bit-plane slicing of an Image	Dr G.Suresh
	Analysis of Zoom Factors	
	Image blending	
	Assignment-3	
MODULE -2		
Durations	Topics	Resource Person
	Program to compute the edges	

Week 4	watershed transform	Dr.N.C.Senthilkumar
	Program for erosion and dilation then edge detection	
	Program to separate R-G-B from RGB	
	Program to separate Missing R-G-B from RGB	
	Code that runs conversion of color image to YCbCr	
	Assignment-4	
MODULE -3		
Durations	Topics	Resource Person
Week 5	DWT based compression	Dr. G. Suresh
	Implementation of Arithmetic Coding	
	Implementation of Wavelet Transform	
	Assessment -1	
	Assignment-5	
Week 6	Implementation of Image Retrieval Schemes	Dr.G.Suresh Dr.N.C.Senthilkumar
	Implementation of Image Segmentation Schemes	
	Assessment -2	
	Conclusion	

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

HANDS ON TRAINING COURSE

ON

IMPLEMENTATION OF IMAGE PROCESSING CONCEPTS FOR REALTIME APPLICATIONS USING MATLAB

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

IMPLEMENTATION OF IMAGE PROCESSING CONCEPTS FOR REALTIME APPLICATIONS USING MATLAB

Date: From 19.09.2021 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	Topics	Resource Person
Week 1	Basics on Image Processing	Dr.G.Suresh
	Introduction to Image Processing Tools	
	Types of Image Representation	
	Waveform and Amplitude Spectrum	
	False Contouring	
	Circular correlation between two signals	
	Assignment-1	
Week 2	Program to Interchage phase between two images	Dr.N.C.Senthilkumar
	Program to adjust brightness and contrast level of an image	
	Histogram Analysis of an Image	
	Types of noises and removal	
	Assignment-2	
Week 3	Bit-plane slicing of an Image	Dr.G.Suresh
	Analysis of Zoom Factors	
	Image blending	
	Assignment-3	
MODULE -2		
Durations	Topics	Resource Person
	Program to compute the edges	

Week 4	watershed transform	Dr.N.C.Senthilkumar
	Program for erosion and dilation then edge detection	
	Program to separate R-G-B from RGB	
	Program to separate Missing R-G-B from RGB	
	Code that runs conversion of color image to YCbCr	
	Assignment-4	
MODULE -3		
Durations	Topics	Resource Person
Week 5	DWT based compression	Dr. G. Suresh
	Implementation of Arithmetic Coding	
	Implementation of Wavelet Transform	
	Assessment -1	
	Assignment-5	
Week 6	Implementation of Image Retrieval Schemes	Dr.G.Suresh Dr.N.C.Senthilkumar
	Implementation of Image Segmentation Schemes	
	Assessment -2	
	Conclusion	

DEMONSTRATIVE MODE

SIGNAL AND IMAGE PROCESSING

Fourier Transforms

Every signal can be written as a sum of sinusoids with different amplitudes and frequencies. The MATLAB command to compute the Fourier Transform and its inverse are respectively fft and ifft, for example:

```
>> x = rand(1,10); % suppose 10 samples of a random signal
>> y = fft(x); % Fourier transform of the signal
>> iy = ifft(y); % inverse Fourier transform
>> x2 = real(iy); % chop off tiny imaginary parts
>> norm(x-x2); % compare original with inverse of transformed
```

The fft is the abbreviation of Fast Fourier Transform. This algorithm implements the discrete Fourier transform to transform data from time into the frequency domain. The study of this algorithm is normally covered in a good linear algebra course. First we give an example of the meaning of the Fourier transform before showing how Fourier transforms can be used to filter noise from signals.

Waveform and Amplitude Spectrum

Suppose we sample a signal during 4 seconds, at a sampling rate of 0.01:

```
>> dt = 1/100; % sampling rate
>> et = 4; % end of the interval
>> t = 0:dt:et; % sampling range
>> y = 3*sin(4*2*pi*t) - 5*sin(2*2*pi*t); % sample the signal
```

A natural plot is that of amplitude versus time:

```
>> subplot(2,1,1); % first of two plots
>> plot(t,y); grid on % plot with grid
>> axis([0 et -8 8]); % adjust scaling
>> xlabel('Time (s)'); % time expressed in seconds
>> ylabel('Amplitude'); % amplitude as function of time
```

With the Fourier Transform we can visualize what characterizes this signal the most. From the Fourier transform we compute the amplitude spectrum:

```
>> Y = fft(y); % compute Fourier transform
>> n = size(y,2)/2; % 2nd half are complex conjugates
>> amp_spec = abs(Y)/n; % absolute value and normalize
```

To visualize the amplitude spectrum, we execute the following commands

```
>> subplot(2,1,2); % second of two plots
>> freq = (0:79)/(2*n*dt); % abscissa viewing window
>> plot(freq,amp_spec(1:80)); grid on % plot amplitude spectrum
>> xlabel('Frequency (Hz)'); % 1 Herz = number of cycles/second
>> ylabel('Amplitude'); % amplitude as function of frequency
```

On the amplitude spectrum we see two peaks: at 2 and 4. The location of the peaks occurs at the two frequencies in the signal. The heights of the peaks (5 and 3) are the amplitudes of the sines in the signal.

%Program:

```
x = rand(1,10);          % suppose 10 samples of a random signal
y = fft(x);              % Fourier transform of the signal
iy = ifft(y);            % inverse Fourier transform
x2 = real(iy);           % chop off tiny imaginary parts
norm(x-x2);              % compare original with inverse of transformed
dt = 1/100;              % sampling rate
et = 4;                  % end of the interval
t = 0:dt:et;             % sampling range
y = 3*sin(4*2*pi*t) + 5*sin(2*2*pi*t);   % sample the signal
subplot(2,1,1);                          % first of two plots
plot(t,y); grid on                       % plot with grid
axis([0 et -8 8]);                       % adjust scaling
xlabel('Time (s)');                      % time expressed in seconds
ylabel('Amplitude');                     % amplitude as function of time
Y = fft(y);                              % compute Fourier transform
n = size(y,2)/2;                         % 2nd half are complex conjugates
amp_spec = abs(Y)/n;                     % absolute value and normalize
subplot(2,1,2);                          % second of two plots
freq = (0:79)/(2*n*dt);                  % abscissa viewing window
plot(freq,amp_spec(1:80)); grid on        % plot amplitude spectrum
xlabel('Frequency (Hz)');                % 1 Herz = number of cycles/second
ylabel('Amplitude');                     % amplitude as function of frequency
```

* Filtering Noise from Signals

```
noise = randn(1,size(y,2)); % random noise
ey = y + noise;             % samples with noise
eY = fft(ey);               % Fourier transform of noisy signal
n = size(ey,2)/2;           % use size for scaling
amp_spec = abs(eY)/n;       % compute amplitude spectrum
figure                      % plots in new window
subplot(2,1,1);             % first of two plots
plot(t,ey); grid on         % plot noisy signal with grid
axis([0 et -8 8]);          % scale axes for viewing
xlabel('Time (s)');         % time expressed in seconds
```

```
ylabel('Amplitude');          % amplitude as function of time
subplot(2,1,2);               % second of two plots
freq = (0:79)/(2*n*dt);                    % abscissa viewing window
plot(freq,amp_spec(1:80)); grid on         % plot amplitude spectrum
xlabel('Frequency (Hz)');  % 1 Herz = number of cycles per second
ylabel('Amplitude');          % amplitude as function of frequency
figure                    % new window for plot
plot(Y/n,'r*')            % Fourier transform of original
hold on                   % put more on same plot
plot(eY/n,'bx')                % Fourier transform of noisy signal
fY = fix(eY/100)*100;          % set numbers < 100 to zero
ifY = ifft(fY);                % inverse Fourier transform of fixed data
cy = real(ifY);
figure                    % new window for plot
plot(t,cy); grid on        % plot corrected signal
axis([0 et -5 5]);         % adjust scale for viewing
xlabel('Time (s)');       % time expressed in seconds
ylabel('Amplitude');
```

```
% Matlab code for White Gaussian Noise
clc;
clear all;
close all;
randn('state',0);
x=randn(100,1);
subplot(2,1,1)
plot(x)
xlabel('n')
ylabel('x[n]')
grid
subplot(2,1,2)
hist(x)
xlabel('x')
ylabel('no of outcome out of 100')
title('white gaussian noise')
figure
N=100;
nbins=10;
xmin=-3;
xmax=3;
ymax=1;
```

```
[y,xx]=hist(x(1:N),nbins);
delx=xx(2)-xx(1);
bar(xx,y/(N*delx))
grid
axis([xmin xmax 0 ymax]);
xlabel('x')
ylabel('PDF,p(x)')
title('white gaussian noise')
```

1. Consider the following sequence of instructions:

```
>> t = 0:0.1:10;
>> y1 = sin(2*pi*t);
>> y2 = sin(20*pi*t);
>> plot(t,y1);
>> hold on;
>> plot(t,y2);
```

   Why is the output of the second plot like this? Find a better range for `t` to plot `sin(20*pi*t)` right. Can you find a good lower bound for the sampling interval in terms of the frequency?

2. Give the MATLAB commands to plot the amplitude spectrum for the signal

$$f(t) = \sum_{k=1}^{20} (20-k)\sin(2\pi kt).$$

   In addition, plot the waveform spectrum of this signal.

3. Make a function to plot waveform and amplitude spectrum of a signal. The function has prototype:

```
function specplot ( t, dt, et, y )
%
%  Opens a new figure window with two plots:
%  the waveform and amplitude spectrum of a signal.
%
%  On entry :
%    t          sampling range of the signal;
%    dt         sampling rate;
%    et         end of the range;
%    y          samples of the signal over the range t.
%
```

   So `specplot` computes the amplitude spectrum of the signal. For the abscissa viewing window you may take half of the range of `t`.

   Test your `specplot` with the signal of the previous assignment.

4. With fft we can decompose a signal in low and high frequencies. Take the example signal from page 1. As noise we now add a sine of amplitude 4 and with frequency 50. Plot the waveform and amplitude spectrum of the new signal. Use fft and ifft to remove this high frequency noise.

Example 1:

```
%this program illustrates false contouring
clc
clear all
close all
a=imread('boat.jpg');
subplot(3,2,1);
imshow(a)
title('original image')
%using 128 gray level
%figure,
subplot(3,2,2);
imshow(grayslice(a,128),gray(128)),
title('image with 128 gray level')
%using 64 gray level
subplot(3,2,3);
imshow(grayslice(a,64),gray(64)),
title('image with 64 gray level')
%using 32 gray level
%figure,
subplot(3,2,4);
imshow(grayslice(a,32),gray(32)),
title('image with 32 gray level')
%using 16 gray level
%figure,
 subplot(3,2,5);
imshow(grayslice(a,16),gray(16)),
title('image with 16 gray level')
%using 8 gray level
%figure,
subplot(3,2,6);
imshow(grayslice(a,8),gray(8)),
title('image with 8 gray level')
```

Output:

Example 2:

```
%frequency response
clc
clear all
close all
[x y]=meshgrid(-pi:0.09:pi);
z=2*cos(x)+2*cos(y);
surf(x,y,z)
axis([-4 4,-4 4,-4 3])
```

Example 3:

```
%frequency response
clc
clear all
close all
[x y]=meshgrid(-pi:0.05:pi);
z=2-cos(x)-cos(y);
surf(x,y,z)
axis([-4 4,-4 4,-0.5 4])
```

Example 4:

```
%application of circular convolution
x=[1 0;0 0]
h=[1 1;1 1]
x1=fft2(x)
h1=fft2(h)
y1=x1.*h1
res=ifft2(y1)
```

Example 5:%circular correlation between two signals

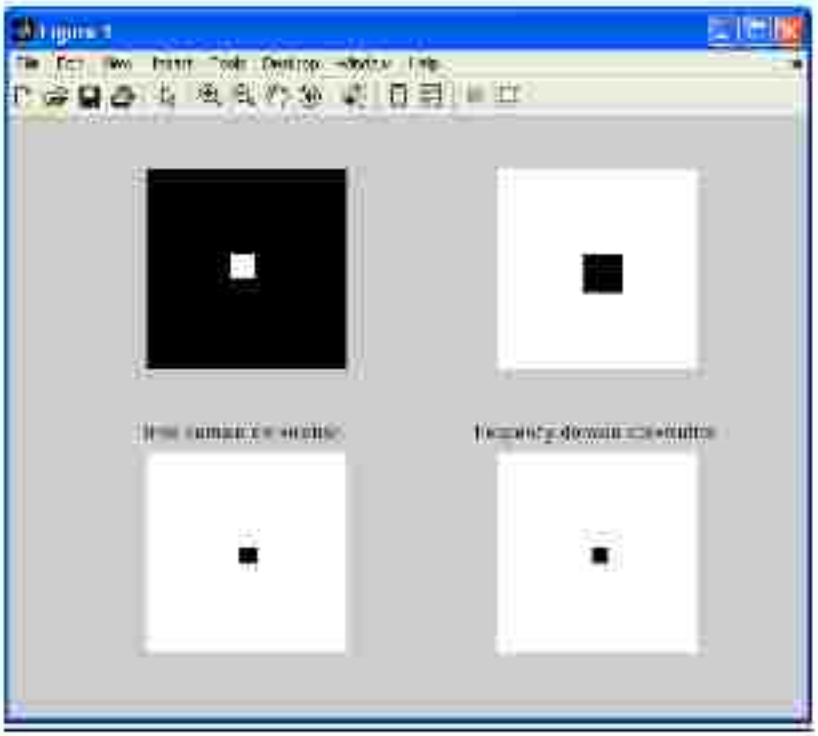
```
clc
clear all
close all
x=[5 10;15 20]
h=[3 6;9 12]
h1=fliplr(h)%fold signal along column-wise
h2=flipud(h1)%fold signal along row-wise
x1=fft2(x);
h3=fft2(h2);
y1=x1.*h3
y2=ifft2(y1)
```

Example 6:

```
clc
clear all
close all
%generation of first image A
a=zeros(256);
[m n]=size(a);
for i=110:140
   for j=110:140
      a(i,j)=255;
   end
end
subplot(2,2,1)
imshow(a)
%generation of second  image B
b=ones(256);
[m n]=size(b);
for i=110:160
   for j=110:160
      b(i,j)=0;
   end
```

```
end
subplot(2,2,2)
imshow(b)
%convolution in time domain
c=conv2(a,b,'same');
%multiplication in frequency domain
a1=fft2(a);
b1=fft2(b);
c1=a1.*b1;
d1=fftshift(ifft2(c1));
subplot(2,2,3)
imshow(c)
title('time domain convolution')
subplot(2,2,4)
imshow(d1)
title('frequency domain convolution')
```

output:



Example 8:

```
clc
clear all
close all
%generation of first image A
a=imread('boat.jpg');
```

```
subplot(3,1,1);
imshow(a)
title('original image')
b=imrotate(a,45,'bilinear','crop');
subplot(3,1,2);
imshow(b)
title('45 degree rotational image')
c=imcrop(b);
%figure,
subplot(3,1,3);
imshow(c)
title('cropped image')
```

Output:

Example 8:

```
%program to interchage phase between two images
clc
clear all
close all
%generation of first image A
a=imread('boat.jpg');
b=imread('lena.jpg');
ffta=fft2(double(a));
fftb=fft2(double(b));
%get the magnitude and phase components
mag_a=abs(ffta);
ph_a=angle(ffta);
mag_b=abs(fftb);
ph_b=angle(fftb);
```

```
%determine new FFT by interchanging the phase
newfft_a=mag_a.*(exp(i*ph_b));
newfft_b=mag_b.*(exp(i*ph_a));
%reconstruct the original image using inverse FFT
rec_a=ifft2(newfft_a);
rec_b=ifft2(newfft_b);
subplot(2,2,1)
imshow(a)
title('original imageA');
subplot(2,2,2)
imshow(b)
title('original imageB');
subplot(2,2,3)
imshow(uint8(rec_a))
title('phase shifted imageA');
subplot(2,2,4)
imshow(uint8(rec_b))
title('phase shifted imageB');
```

Output:

Example 9:

```
% Fourier transform of Fourier Transform
clc
clear all
```

```
close all
%generation of first image A
%a=imread('boat.jpg');
a=imread('lena.jpg');
[m n]=size(a);
b=fft2(a);
%spectrum of spectrum
c=(1/(m*n))*fft2(b);
subplot(2,2,1),imshow(a),title('input image');
subplot(2,2,2),imshow(uint8(c)+40),title('spectrum of spectrum');
```

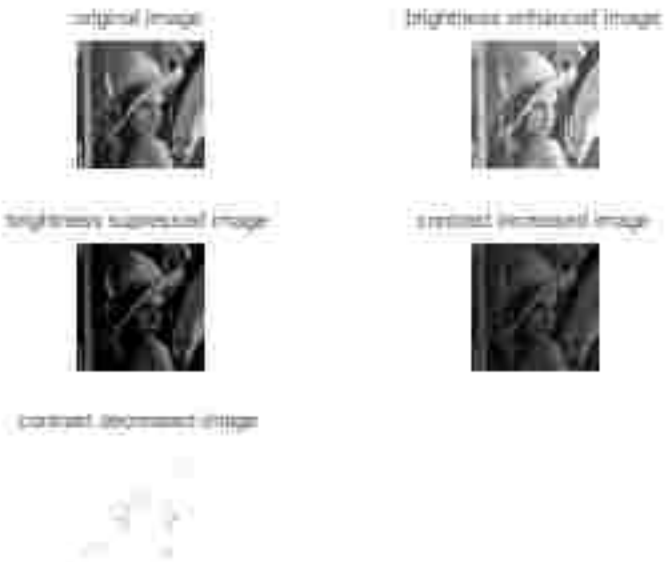
Output:

Example 10

```
%program to adjust brightness and contrast level of an image

clc
clear all
close all
a=imread('lena.jpg');
[m n]=size(a);
b=double(a)+50;
c=double(a)-70;
subplot(3,2,1);
imshow(a)
title('original image');
subplot(3,2,2);
imshow(uint8(b))
title('brightness enhanced image');
subplot(3,2,3);
```

```
imshow(uint8(c))
title('brightness supressed image');
d=a*.5;
e=a*20;
subplot(3,2,4);
imshow(uint8(d))
title('contrast increased image');
subplot(3,2,5);
imshow(uint8(e))
title('contrast decreased image');
```

output:

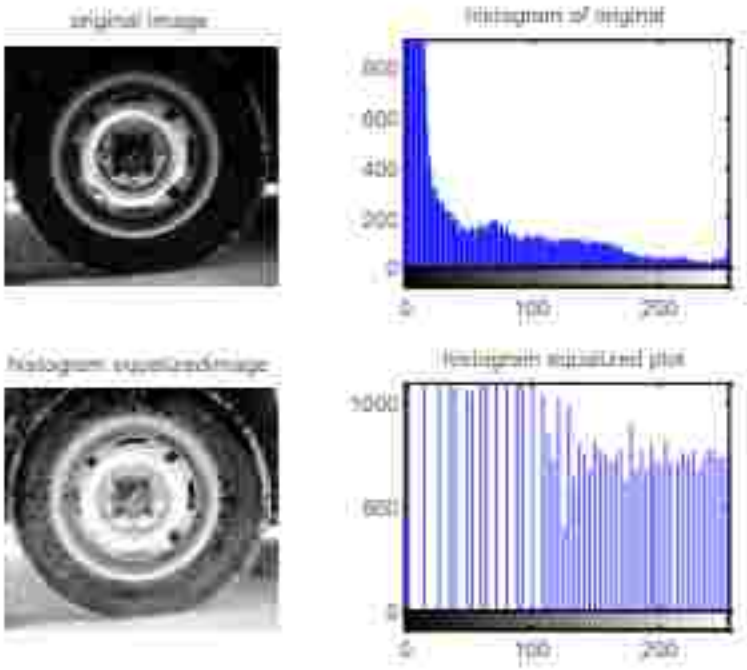


Example 11:

```
clc
clear all
close all
I = imread('tire.tif');
K = histeq(I);
subplot(2,2,1);
imshow(I)
title('original image');
subplot(2,2,2);
imhist(I)
title('histogram of original');
subplot(2,2,3);
```

```
imshow(K)
title('histogram equalizedimage');
subplot(2,2,4);
imhist(K)
title('histogram equalized plot');
```

output:

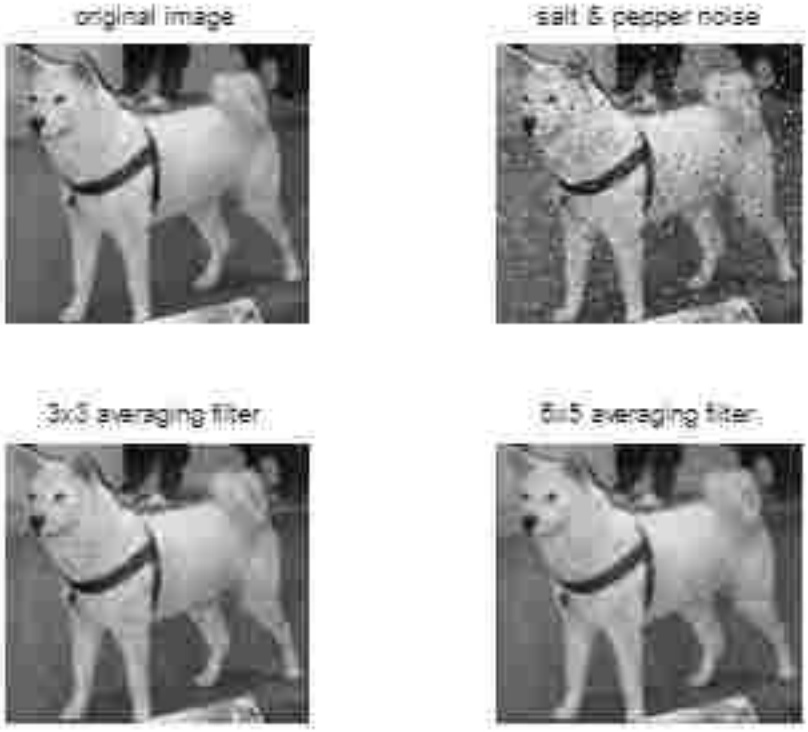


Example 12:

```
% Types of noises and removal
a=imread('dog.jpg');
a=rgb2gray(a);
b=imnoise(a,'salt & pepper');
c=imnoise(a,'gaussian');
d=imnoise(a,'speckle');
%defining 3x3 and 5x5 kernal
h1=1/9*ones(3,3);
h2=1/25*ones(5,5);
%attempt to recover the image
b1=conv2(b,h1,'same');
b2=conv2(b,h2,'same');
c1=conv2(c,h1,'same');
c2=conv2(c,h2,'same');
d1=conv2(d,h1,'same');
d2=conv2(d,h2,'same');
```

```
figure,subplot(2,2,1),imshow(a),title('original image'),
subplot(2,2,2),imshow(b),title('salt & pepper noise'),
subplot(2,2,3),imshow(uint8(b1)),title('3x3 averaging filter'),
subplot(2,2,4),imshow(uint8(b2)),title('5x5 averaging filter')
%-------------------------------------------------------------------
figure,subplot(2,2,1),imshow(a),title('original image'),
subplot(2,2,2),imshow(c),title('Gaussian noise'),
subplot(2,2,3),imshow(uint8(c1)),title('3x3 averaging filter'),
subplot(2,2,4),imshow(uint8(c2)),title('5x5 averaging filter')
%-------------------------------------------------------------------
figure,subplot(2,2,1),imshow(a),title('original image'),
subplot(2,2,2),imshow(d),title('speckle noise'),
subplot(2,2,3),imshow(uint8(d1)),title('3x3 averaging filter'),
subplot(2,2,4),imshow(uint8(d2)),title('5x5 averaging filter')
```

output:



original image

Gaussian noise

3x3 averaging filter

5x5 averaging filter

---

original image

speckle noise

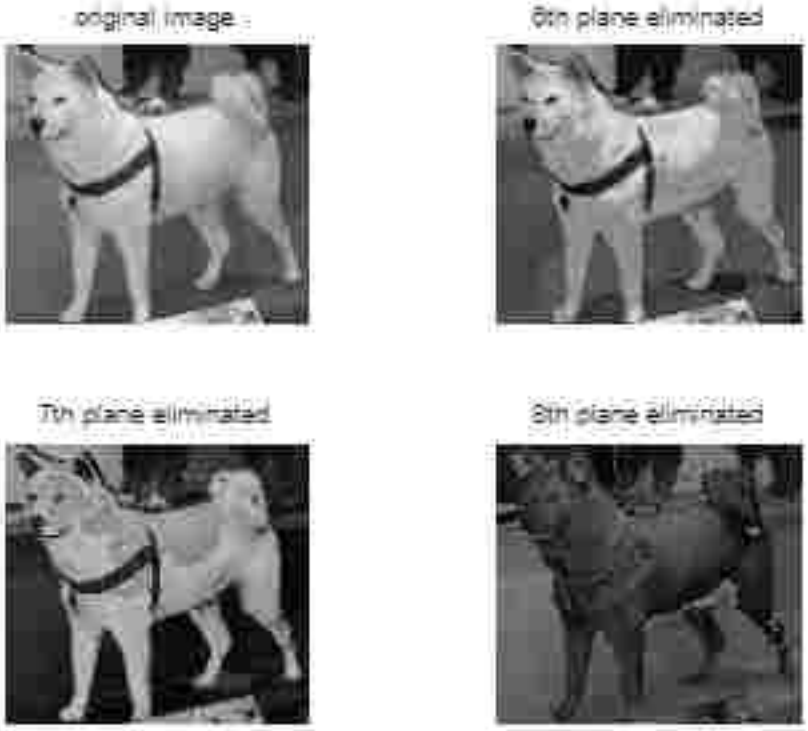
3x3 averaging filter

5x5 averaging filter

Example 13:

```
% bitplane slicing
clc
clear all
close all
a=imread('dog.jpg');
a=rgb2gray(a);
subplot(2,2,1)
imshow(a);
title('original image')
[m n]=size(a);
n1=input('enter the bit plane number (8 to 1 that to be removed:');
s=255-(2^(n1-1));
for i=1:m,
   for j=1:n,
      out_I(i,j)=bitand(a(i,j),s);
   end
end
subplot(2,2,2)
imshow(uint8(out_I));title(sprintf(' plane eliminated is %g',n1))
n1=input('enter the bit plane number (8 to 1 that to be removed:');
s=255-(2^(n1-1));
for i=1:m,
   for j=1:n,
      out_I(i,j)=bitand(a(i,j),s);
   end
end
subplot(2,2,3)
imshow(uint8(out_I));title( sprintf('plane eliminated is %g',n1));
n1=input('enter the bit plane number (8 to 1 that to be removed:');
s=255-(2^(n1-1));
for i=1:m,
   for j=1:n,
      out_I(i,j)=bitand(a(i,j),s);
   end
end
subplot(2,2,4)
imshow(uint8(out_I));title(sprint(' plane eliminated is %g',n1));
```

Output:



Example 14:

```
clc;
clear all;
close all;
a=imread('boat.jpg');a=imresize(a,[32 32]);
[m n]=size(a);
p=input('Enter the size you want: ');
for i=1:m %loop to extract every row
  for j=1:n %loop to extract every column
    for k=1:p %loop to control the number of replication
       b(i,(j-1)*p+k)=a(i,j); %replication of pixels in row wise
    end
  end
end
c=b;
[m n]=size(c);
for i=1:n %loop to extract every column
  for j=1:m %loop to extract every row
    for k=1:p %loop to control the number of replication
       b((j-1)*p+k,i)=c(j,i); %replication of pixels in column wise
```

```
    end
  end
end
imshow(a),title('original image')
figure,imshow(b),title('zoomed image')
xlabel(sprintf('zooming factor is %g',p))
```

Output:

Example 15:

```
a=imread('lena.jpg');%a=imresize(a,[64 64]);
zooming_factor=input('enter the zooming factor:');
num=zooming_factor;den=1;
while(num-floor(num)~=0)
   num=num*2;den=den*2;
end
[m n]=size(a);s1=num*m;
re=zeros(s1,num*n);
for i=1:m,
   for j=1:n,
      k=num*(i-1);
      l=num*(j-1);
      re(k+1,l+1)=a(i,j);
   end
end
   i=1;
   while(i<=(s1))
      j=1;
      while(j<=(num*n))
```

```
        x=ones(num,num);
        for p=1:num,
           for q=1:num,
              c(p,q)=re(i,j);
              j=j+1;
           end
           i=i+1;j=j-num;
        end
        z=ifft2(fft2(c).*fft2(x));
        i=i-num;
        for p=1:num,
           for q=1:num,
              re(i,j)=z(p,q);
              j=j+1;end
           i=i+1;j=j-num;end
        i=i-num;j=j+num;end
     i=i+num;end
  if(den>1)
     m=den;[p q]=size(re);
     a=double(re);
     for i=1:ceil(p/m),
        for j=1:ceil(q/m),
           if(((m*i)<p)&((m*j)<q))
              b(i,j)=re(m*i,m*j);
           else b(i,j)=0;
           end
        end
     end
  else b=re;end
  figure,imshow(uint8(b)),
```

Output:

Example 16:

```
% Image blending
clc
clear all
close all
%c=(1-x)a+xb
a=imread('lena.jpg');
a=rgb2gray(a);subplot(2,2,1);
imshow(a)
[m n]=size(a);
title('Image 1');
b=imread('boat.jpg');
b=rgb2gray(b);
b1=imresize(b,[256 256]);subplot(2,2,2);
imshow(b1)
title('Image 2');
c1=a+b1;
subplot(2,2,3);
imshow(c1)
title('blended Image');
x=input('enter x value:')
for i=1:m,
   for j=1:n,
      c2(i,j)=(1-x)*a(i,j)+x*b1(i,j);
   end
end
subplot(2,2,4);
imshow(c2)
title(sprintf('blended Image of %g',x));
```

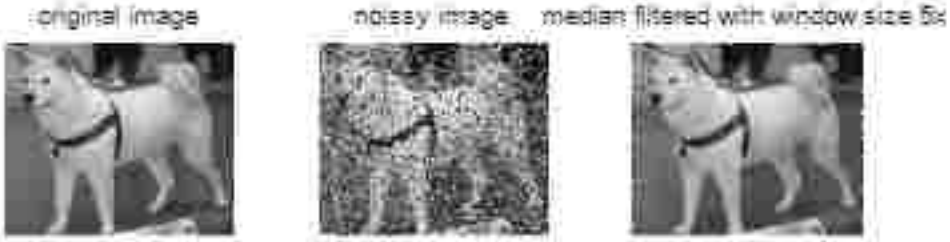
output:

Example 17:

```
%this program is to perform median filtering of the image
clc
clear all
close all
a=imread('dog.jpg');
a=rgb2gray(a);
b=imnoise(a,'salt & pepper',0.2);
b=double(b);
[m n]=size(b);
N=input('enter the window size:');
out_img=b;
if(mod(N,2)==1)
   Start=(N+1)/2;
   End=Start;
else
   Start=N/2;
End=Start+1;
end
if(mod(N,2)==1)
   limit1=(N-1)/2;
```

```
    limit2=limit1;
else
    limit1=(N/2)-1;
    limit2=limit1+1;
end
for i=Start:(m-End+1),
    for j=Start:(n-End+1),
        I=1;
        for k=-limit1:limit2,
            for l=-limit1:limit2,
                mat(I)=a(i+k,j+l);
                I=I+1;
            end
        end
        mat=sort(mat);
        if(mod(N,2)==1)
            out_img(i,j)=(mat(((N^2)+1)/2));
        else
            out_img(i,j)=(mat((N^2)/2)+mat(((N^2)/2)+1))/2;
        end
    end
end
subplot(1,3,1)
imshow(a)
title('original image');
subplot(1,3,2)
imshow(uint8(b))
title('noissy image')
subplot(1,3,3)
imshow(uint8(out_img))
title(sprintf('median filtered with window size %gx%g',N));
```

Output:

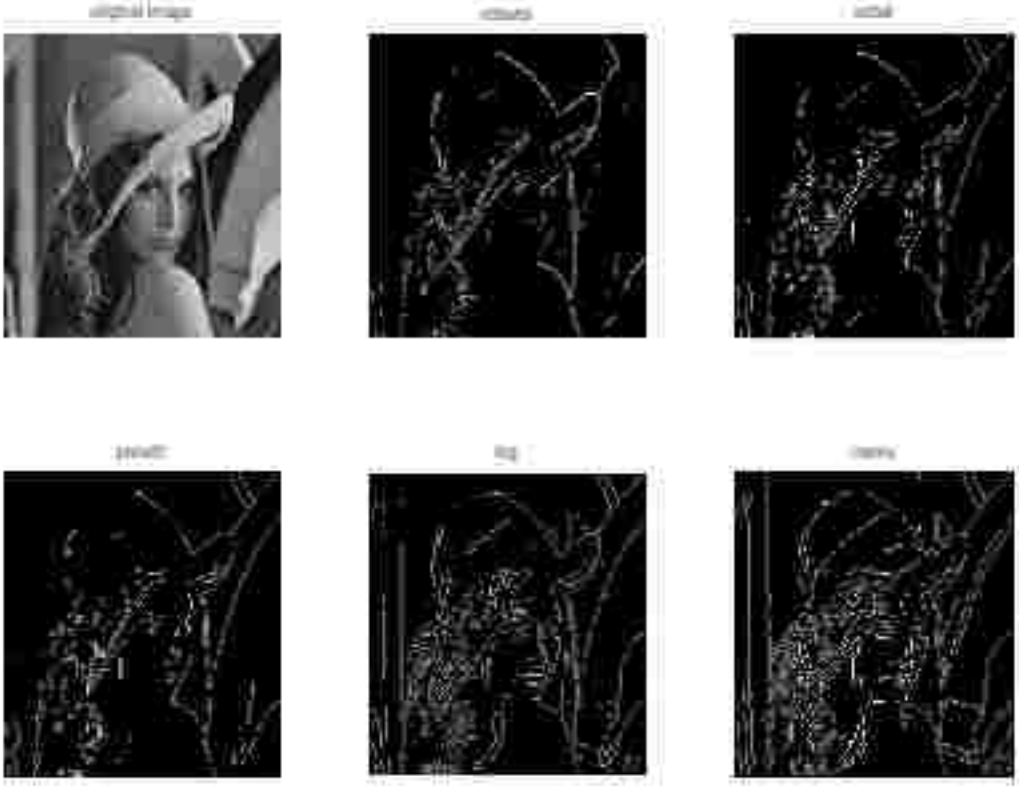


Example 18:

```
%program to compute the edges
clc
clear all
close all
a=imread('lena.jpg');
a=rgb2gray(a);
b=edge(a,'roberts');
c=edge(a,'sobel');
d=edge(a,'prewitt');
e=edge(a,'log');
f=edge(a,'canny');
%b=edge(a,'roberts');
subplot(2,3,1)
imshow(a)
title('original image')
subplot(2,3,2)
imshow(b)
title('roberts')
subplot(2,3,3)
imshow(c)
title('sobel')
subplot(2,3,4)
imshow(d)
```

```
title('prewitt')
subplot(2,3,5)
imshow(e)
title('log')
subplot(2,3,6)
imshow(f)
title('canny')
```

Output:

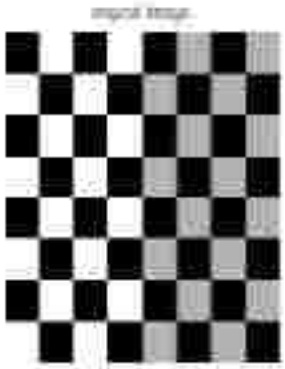


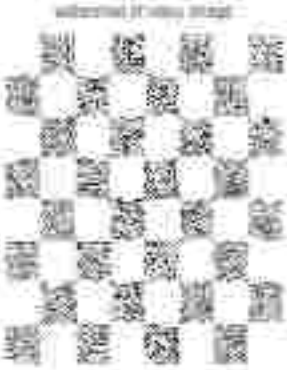
Example 19:

```
%watershed transform
clc
clear all
close all
a=checkerboard(32);
a1=imnoise(a,'salt & pepper',0.1);
b=watershed(a,4);
b1=watershed(a1,4);
subplot(2,2,1)
```

```
imshow(a),title('original image');
subplot(2,2,2);
imshow(a1),title('noisy image');
subplot(2,2,3);
imshow(b),title('watershed of original');
subplot(2,2,4);
imshow(b1),title('watershed of noisy image');
```

Output:



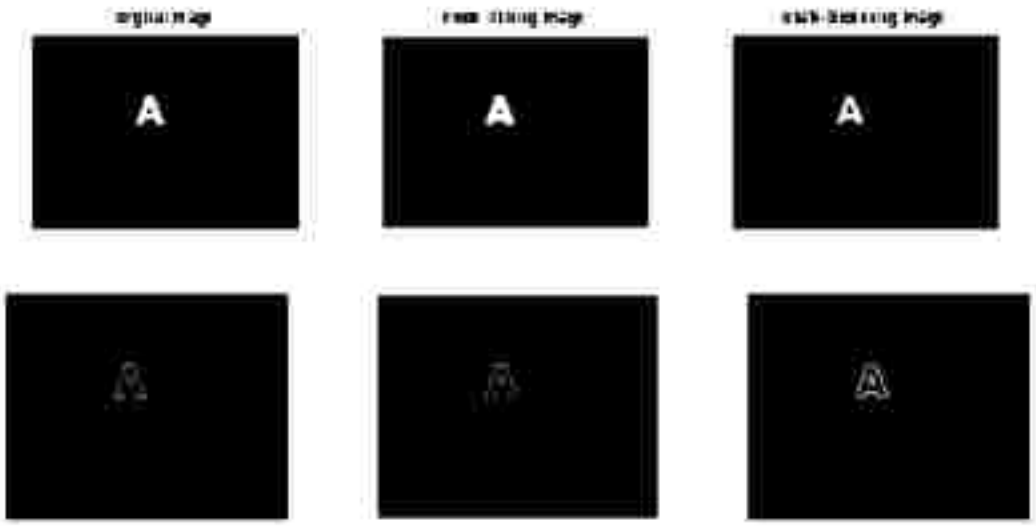


Example 20:

```
%program for erosion and dilation then edge detection
clc
clear all
close all
a=imread('sur.jpg');
b=[1 1 1;1 1 1;1 1 1];
a1=imdilate(a,b);
a2=imerode(a,b);
```

```
subplot(1,3,1)
imshow(a),title('original image');
subplot(1,3,2)
imshow(a1),title('erode -thining image');
subplot(1,3,3)
imshow(a2),title('dilate-thickening image');
a3=a-a2;
a4=a1-a;
a5=a1-a2;
figure
subplot(1,3,1)
imshow(a3),%title('');
subplot(1,3,2)
imshow(a4),%title('erode -thining image');
subplot(1,3,3)
imshow(a5),%title('dilate-thickening image');
```

Output:

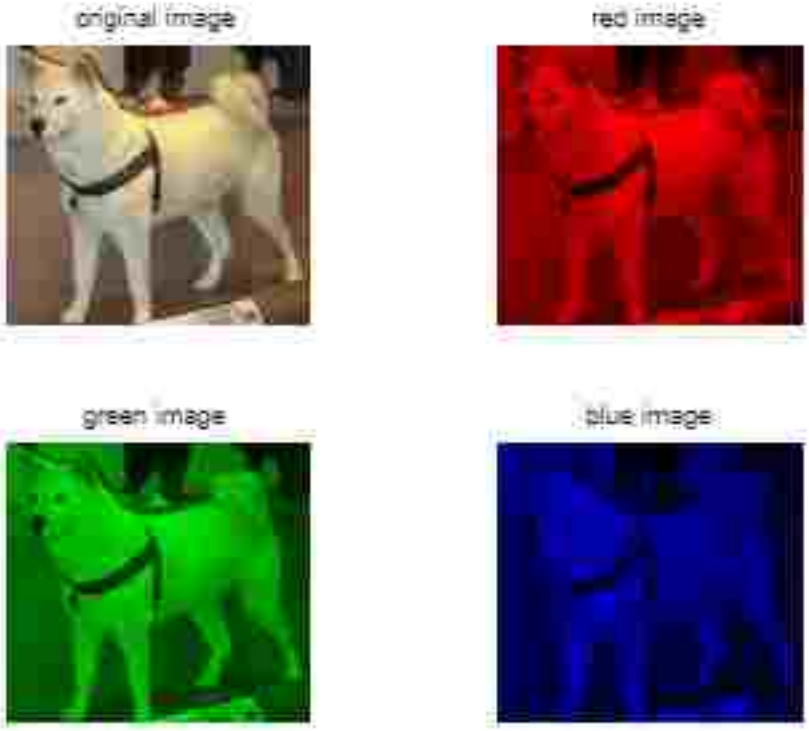


Example 21:

```
%program to separate R-G-B from RGB
RGB=imread('dog.jpg');
R=RGB;
G=RGB;
B=RGB;
R(:,:,2)=0;
R(:,:,3)=0;
G(:,:,1)=0;
```

```
G(:,:,3)=0;
B(:,:,1)=0;
B(:,:,2)=0;
subplot(2,2,1),imshow(RGB),title('original image')
subplot(2,2,2),imshow(R),title('red image')
subplot(2,2,3),imshow(G),title('green image')
subplot(2,2,4),imshow(B),title('blue image')
```

output:



Example 22:

```
%program to separate Missing R-G-B from RGB
RGB=imread('dog.jpg');
R=RGB;
G=RGB;
B=RGB;
R(:,:,1)=0;
G(:,:,2)=0;
B(:,:,3)=0;
subplot(2,2,1),imshow(RGB),title('original image')
subplot(2,2,2),imshow(R),title('red missing image')
subplot(2,2,3),imshow(G),title('green missing image')
```

```
subplot(2,2,4),imshow(B),title('blue missing image')
```

Example 23:

```
%Code that runs conversion of color image to YCbCr

%read in image filename
%inimage = input('Enter image file name with extension (like jennifer.bmp): ', 's');

%open image file
inimage = imread('dog.jpg');

%display on screen the image
figure(1), imshow(inimage), title('Original Image');


%the command size returns the size of the matrix/image
%A semi-colon suppresses the screen output of the variable
%values, while the lack of semi-colon prints it to the screen


size(inimage)
```

```
U = rgb2ycbcr(inimage);
figure(1), imshow(inimage); title('RGB image');
figure(2), imshow(U); title('YCBCR Image');
size(U)

%Here pick off the 256x256 luminance part of the ycbcr image
Y = U(:,:,1);
figure(3), imshow(Y); title('Y part of Image');
size(Y)

%Here pick off the 256x256 Cb part of the ycbcr image
CB = U(:,:,2);
figure(4), imshow(CB); title('Cb part of Image');
size(CB)

%Here pick off the 256x256 Cr part of the ycbcr image
CR = U(:,:,3);
figure(5), imshow(CR); title('Cr part of Image');
size(CR)
```

Example 24:

```
%dwt based compression
clc
clear all
close all
a=imread('dog.jpg');
[p q r t]=dwt2(a,'db1')
b=[uint8(p),q;r t];
[p1 q1 r1 t1]=dwt2(p,'db1');
b1=[p1 q1; r1 t1];imshow(b);
% b2=[b1,q;r,t];
% imshow(b2);
```

Output:

Example 25:

```
%C:\Documents and Settings\gsuresh\Desktop\Desktop 09-08-2012\code1
%boat.jpg %lena.jpg

%This program hides a message image in the lower
%bit planes of a cover image
%read in cover image filename
covername = input('Enter image file name with extension (like jennifer.bmp): ', 's');
%read in message image filename
messagename = input('Enter message image file name with extension: ', 's');
%open cover and message image files
cover = imread(covername);
message = imread(messagename);
%display on screen the two images
figure(1), imshow(cover); title('Original Image (Cover Image)');
figure(2), imshow(message);title('Image to Hide (Message Image)');
%change to double to work with addition below
cover=double(cover);
message=double(message);
%imbed = no. of bits of message image to embed in cover image
imbed=4;
%shift the message image over (8-imbed) bits to right
messageshift=bitshift(message,-(8-imbed));
%show the message image with only embed bits on screen
%must shift from LSBs to MSBs
showmess=uint8(messageshift);
showmess=bitshift(showmess,8-imbed);
figure(3),imshow(showmess);title('4 Bit Image to Hide');
%now zero out imbed bits in cover image
coverzero = cover;
```

```
for i=1:imbed
coverzero=bitset(coverzero,i,0);
end
%now add message image and cover image
stego = uint8(coverzero+messageshift);
figure(4),imshow(stego);title('Stego image');
%save files if need to
%4 bit file that was embedded = same as file extracted
imwrite(showmess,'showmess4.bmp'); %use bmp to preserve lower bits
%jpg will get rid of them
%stego file
imwrite(stego,'stego4.bmp');
```

output:

Email Address	Hall Ticket Number	Name of the Student (as per SSC)	Department	Willing to choose to the Course	Contact Number
mitulreddy2707@gmail.com	18D41A04F5	P vivek vardhan reddy	ECE	Yes	9390392706
mohammedadnan2000.ec@gmail.com	18D41A04E0	mohammed adnan	ECE	Yes	8450074747
neeleshgoud45@gmail.com	18D41A04F2	NEELESH KUMAR GOUD	ECE	Yes	7983292706
ganeshparitala2701@gmail.com	18D41A04C0	PARITALA GANESH	ECE	Yes	8121215483
abhishekramaleeti1137@gmail.com	18D41A04G8	Ramaleeti Abhishek Raj Naik	ECE	Yes	7780898397
vamshidharreddy705@gmail.com	18D41A04L8	THALABANI VAMSHIDHAR REDDY	ECE	Yes	7095980383
tharunreddy2002@gmail.com	18D41A04E8	N Tharun kumar reddy	ECE	Yes	9652590841
meesalajay@gmail.com	18D41A04D7	Meesala ajay	ECE	Yes	6303932156
sudheervenkat2000@gmail.com	18D41A04G7	Pinla venkata sudheer	ECE	Yes	9381755075
manasareddy1213@gmail.com	17D41A0483	Gudidally manasa	ECE	Yes	9981456833
varunreddy1838@gmail.com	17D41A04J0	RAMSHETTI VARUN KUMAR	ECE	Yes	9278295275
Ramakanth1212@gmail.com	17D41A0403	Angoti Ramakanth Goud	ECE	Yes	7036705316
punnakarthik51@gmail.com	18D41A04C8	PUNNA KARTHIK	ECE	Yes	8008083834
varshiniamrutha222000@gmail.com	18D41A04H1	AMRUTHA VARSHINI PUNNA	ECE	Yes	7993788535
indrakumardasari571@gmail.com	18D45A0418	D INDRA KUMAR	ECE	Yes	7893960312
chandu8ooo@gmail.com	18D41A04H0	POLOJU POORNA CHANDAR	ECE	Yes	7995008838
vaishnavipandillreddy27@gmail.com	18D41A04F6	Vaishnavi	ECE	Yes	9398361024
manasareddy1213@gmail.com	17D41A0483	Gudidally manasa	ECE	Yes	9981456633
roshanbamala4@gmail.com	18D41A04J8	BAMALA ROSHAN	ECE	Yes	9989447931
umeshchandra2010@gmail.com	18D41A04F3	P UMESHCHANDRA	ECE	Yes	7674933730
vasudhasamwika21@gmail.com	18D41A04M5	Vasudha samvika	ECE	Yes	7207300753
manithamupola82@gmail.com	18D41A04C2	M Manitha	ECE	Yes	9949437834
sureeallymounika123@gmail.com	18D41A04L2	SUREPALLY MOUNIKA	ECE	Yes	9347128880
sureeallymounika123@gmail.com	18D41A04L2	Surepally Mounika	ECE	Yes	9347128880
bhavaniyerram167@gmail.com	18D41A0426	Bhavani yerram	ECE	Yes	9912007186
karthikpolasani92@gmail.com	18D41A04H6	Polasani karthik	ECE	Yes	9963441138
rachiraju1438@gmail.com	18D41A04H3	Pudari Narasimha Raju	ECE	Yes	9346437930
abhithputtoju@gmail.com	18D41A04H4	PUTTOJU ABHIRITH CHAITHANYA	ECE	Yes	8488933744
aishuaksimin01@gmail.com	18D41A04G3	PASULA AISHWARYA	ECE	Yes	9059558421
Surepallypavan888@gmail.com	18D41A04L4	Surepally Pavan	ECE	Yes	9966447987
preethiarchu18@gmail.com	18D41A04J3	Sampatl archana	ECE	Yes	6304724128
ranghamiapama@gmail.com	18D41A04J0	Ranghani Aparna	ECE	Yes	8333011362
nishureddy874@gmail.com	18D41A04F1	NARSAPOLA NISHANTH REDDY	ECE	Yes	7893212245
chandanareddy1307@gmail.com	18D41A04F0	R Chandana	ECE	Yes	7989896720
amadhukar52@gmail.com	18D41A04AE	KONDURI MADHUKAR	ECE	Yes	8488891730
naveenkumar84242@gmail.com	18D41A04H2	Pishamaddila Naveen Kumar	ECE	Yes	8247565534
sairohith027@gmail.com	18D41A04A0	KODAMARTHI SAI ROHITH	ECE	Yes	9963311862
devakrishnakanth18@gmail.com	18D41A0481	Deva Krishna Kanth	ECE	Yes	9390652985
chandanareddy1307@gmail.com	18D41A04F0	R Chandana	ECE	Yes	7989896720
sarayured27@gmail.com	18D41A04F4	PAKALA SARAYU	ECE	Yes	7675914556
sumanthreddybegamnagari@gmail.com	17D41A0418	Begamnagari Sumanth Reddy	ECE	Yes	8192339823
preetham78333@gmail.com	18D41A04E4	Muragada Sai Preetham	ECE	Yes	8309102820

ruchikareddy03@gmail.com	18D41A04R4	NANNURI RUCHIKA REDDY	ECE	Yes	7997073368
harshasai...@gmail.com	17D41A0411	Arasili Venkata Chaitra Sai Sree Harsha	ECE	Yes	9121271806
kondalreddy738@gmail.com	17D41A0401	MALLA KONDAL REDDY	ECE	Yes	7702251394
abhinavreddy1191@gmail.com	18D41a0431	B ABHINAV REDDY	ECE	Yes	7673811201
sandeepreddy3363@gmail.com	17D41A0469	KALLURI SANDEEP REDDY	ECE	Yes	9100883364
pavankasina23@gmail.com	17D41A04A2	K PAVAN KUMAR	ECE	Yes	9374483524
nitishnagini@gmail.com	17D41A0487	Kotagiri nitish	ECE	Yes	8096802477
aarutharu@gmail.com	18D41A0487	AAVULA SAIRUTHVIK VAMSHIKAR	ECE	Yes	8309638622
sandeepmandadi677@gmail.com	18D41A04C3	MANDADI SANDEEP	ECE	Yes	9347103348
pasulaakhila2@gmail.com	18D41A0494	PASULA AKHILA	ECE	Yes	9381793869
ajayreddyaj1143@gmail.com	17D41A0488	JANGA AJAY	ECE	No	9381968831
saismitha66@gmail.com	18D41A04X9	Samhitha	ECE	Yes	9378886863
bharathsimhareddy708@gmail.com	17D41A0483	Kuppula Bharath Simha Reddy	ECE	Yes	7799638869
sejalsetti142@gmail.com	18D41A04K0	Sejal setti	ECE	Yes	9383529464
bharathguptha120@gmail.com	17D41A04N4	YASA BHARATH KUMAR	ECE	Yes	7036484741
bhavaniyamsani37@gmail.com	18D41A0423	Bhavani yamsani	ECE	Yes	9912007168
amithareddyhunoorguda@gmail.com	17D41A0488	Hunoorguda Amitha	ECE	Yes	7036417613
dandjeevanprakashreddy@gmail.com	18D41A0487	Jeevan Prakash Reddy	ECE	Yes	9398719612
neeraja2442@gmail.com	17D41A0478	Golla Neeraja	ECE	Yes	8304449840
kallakurihemanthsai@gmail.com	18D45A0401	KALLAKURI HEMANTH SAI	ECE	Yes	7382041883
gouraiannu123@gmail.com	17D41A0460	Gouri Srividya	ECE	Yes	9963316468
meddepallypranay@gmail.com	17D41A04H4	Pranay Reddy M	ECE	Yes	9010696339
ani2839z@gmail.com	17D41a0489	INUKOLLU ANIRUDH REDDY	ECE	Yes	9989810626
bhuvanasri273@gmail.com	17D41A04D8	Lingala Bhuvana Sri	ECE	Yes	9849479333
amithkumar7286@gmail.com	17D41A0404	Ageer Amith kumar	ECE	Yes	7036595216
akshayalaya93@gmail.com	17D41A0414	B Laya	ECE	Yes	7330618770
Nenavathpraveen10@gmail.com	17D41A04F6	NENAVATH PRAVEEN	ECE	Yes	9188368133
sujithnaik001@gmail.com	17D41A04F7	NENAVATH SUJITH NAIK	ECE	Yes	6303034211
prashanthera7@gmail.com	17D41A0464	Erra Prashanth	ECE	Yes	9949383639
madireddivarshith.39@gmail.com	17D41A04D0	MADIREDDY VARSHITH	ECE	Yes	7989734637
harshapawar1369@gmail.com	18D41A04L2	Suntake Harsha pawar	ECE	Yes	7989930648
venkat.nagavelli@gmail.com	17D41A0485	Nagavelli Sai Venkat	ECE	Yes	7382383799
kolukulapallybikku333@gmail.com	18D45A0408	KOLKULAPALLY BIKKU	ECE	Yes	7993049610
sainath.reddy4343@gmail.com	17D41A04D4	MANDADI SAINATH REDDY	ECE	Yes	9301605418
vaishnavi.goud79@gmail.com	18D41A04L8	Thalla Vaishnavi	ECE	Yes	9989223383
seemanthas616@gmail.com	17D41A04H1	Peddapalli sai seemantha	ECE	Yes	7036920436
narendraraju7003@gmail.com	17D41A04J3	REDDICHARLA NARENDRA RAJU	ECE	Yes	8367487003
pavansaisai321@gmail.com	17D41a0481	Dunaberg Pavan sai	ECE	Yes	9110747787
reddyrohith0917@gmail.com	17D41A0469	Annapureddy Rohithreddy	ECE	Yes	9182773849
akhilkogitala163@gmail.com	17D41A04C7	KOGITLA AKHIL KUMAR	ECE	Yes	7036944612
dandjeevanprakashreddy@gmail.com	18D41A0487	Dandi Jeevan Prakash Reddy	ECE	Yes	9398719612
bethapudiraghuvarma@gmail.com	17D41A0427	Bethapudi Raghu varma	ECE	Yes	9374602621
nagatipavankalyan@gmail.com	17D41A04F2	NAGATI PAVAN KLAYAN	ECE	Yes	9989308441
katragoniprasanna123@gmail.com	17D41A04AE	Katragoni Prasanna	ECE	Yes	7989421223
kotireddy6666@gmail.com	18D41A04A3	Koti Leela Krishna Reddy	ECE	Yes	6301958927
kumarnikhil0606@gmail.com	18D41A04AE	KORRU NIKHIL KUMAR	ECE	Yes	9676669972

kamalsingh331@gmail.com	18D41A0433	K Amarsingh	ECE	Yes	9340391337
nivas.mg@gmail.com	18D41A0404	Maddela Srinivas	ECE	Yes	8.19642E+11
mohammadsameer1503@gmail.com	18D41A0409	MOHAMMAD SAMEER	ECE	Yes	9949325190
mahalasai333@gmail.com	18D41A040E	MAHALSA SAI DONTHA	ECE	Yes	9398405172
mnagasahitya6434@gmail.com	17D41A0488	Munagapati naga sahitya	ECE	Yes	8332982442
pauldayakar5123@gmail.com	17D41A0420	BALLUGULA PAUL DAYAKAR	ECE	Yes	9963363481
bvenkatesh0432@gmail.com	17D41A0432	BOGGKUKABARU VENKATESH	ECE	Yes	9966278613
koresandhya64@gmail.com	17D41A0484	Kore Sandhya Rani	ECE	Yes	9100687924
madduri.sameer39@gmail.com	18D41A0408	Madduri Venkata Sameer Kumar	ECE	Yes	9849888822
madhavaramsriram@gmail.com	18D41A04F5	Madhavaram sriram	ECE	Yes	9949328887
anilkumardasari222@gmail.com	17D41A0447	D Anil Kumar	ECE	Yes	9676315946
rajithasayam2@gmail.com	18D41A0482	Kopputa Rajitha	ECE	Yes	9618101271
sushmareddy1530@gmail.com	17D41A0403	Madde sushma	ECE	Yes	7013347368
kunchalavenkatesh007@gmail.com	17D41A0401	Kunchala venkatesh	ECE	Yes	9182447549
madduri.sameer39@gmail.com	18D41A0408	Madduri Venkata Sameer Kumar	ECE	Yes	9849888822
lasyaleasla@gmail.com	18D41A0492	Y Lasya	ECE	Yes	8331836408
nimisha30@gmail.com	17D41A0491	Nimisha Reddy	ECE	Yes	9398008406
pathupraneeth@gmail.com	17D41A0488	TATAVARTHY SATYADATHA PRANEETH	ECE	Yes	9989481748
chilukurisaikrishnamaye@gmail.com	18D41A0439	C. Sai Krishnamayi	ECE	No	8304631337
vinishareddy1237@gmail.com	18D41A0409	Vinisha	ECE	Yes	7780579008
bharadwajaenumula@gmail.com	18D41A0488	BHARADWAJA ENUMULA	ECE	No	9398898873
gayathri0808@gmail.com	17D41A0434	Bommidi Gayathri	ECE	Yes	8333938837
gellaharini2000@gmail.com	17D41A0474	GELLA HARINI	ECE	Yes	9100623867
pureeshwaryerra@gmail.com	17D41A04N5	Yerra Pureeshwar	ECE	Yes	9652865897
tmraju1999@gmail.com	17D41A04H1	TANGELLA MALLAM RAJU	ECE	Yes	9993352746
ramadugupranathi@gmail.com	18D41A04H7	R.Pranathi	ECE	Yes	9290902331
preethiche1001@gmail.com	18D41A0465	AERRAGANNELA PREETHI	ECE	Yes	7093322883
Saichanderrao72@gmail.com	18D41A04N1	VELGAPURI SAI CHANDER RAO	ECE	Yes	9989486360
begumnidhi12@gmail.com	18D45A0420	Begum Nidhi	ECE	Yes	9381428638
kokkulapallybikku538@gmail.com	18D45A0406	KOKKULAPALLY BIKKU	ECE	Yes	7989049810
Shivaramallenki2001@gmail.com	18D41A0408	ALLENKI SHIVA RAM	ECE	Yes	9390392078
guntivarun2000@gmail.com	17D41A0489	Gunti Varun	ECE	Yes	7330782010
bhavanareddy3333@gmail.com	18D41A0420	Bejjuri Bhavana reddy	ECE	Yes	9381283388
vickyvikas@gmail.com	18D41A0432	Pasupula Vikas	ECE	Yes	7036478271
pramodpadala260@gmail.com	17D41A0436	PADALA PRAMOD	ECE	Yes	9640546215
tejaswireddy0515@gmail.com	17D41A0408	Alugubelly Tejaswini	ECE	Yes	9121396308
chinnamaheshh17@gmail.com	18D45A0403	CH. MAHESH	ECE	Yes	9392368888
anilnani1912@gmail.com	18D41A0482	MORTHALA ANIL	ECE	Yes	8305416622
goudsri712@gmail.com	18D41A0481	J SHIVA PRASAD	ECE	Yes	9640442234
sairohith27@gmail.com	18D41A04A2	KOCHARTHI SAI ROHITH	ECE	Yes	9989311936
anthamsravani@gmail.com	17D41A04N8	Antham Sravani	ECE	Yes	8106334279
poojithachilukuri2000@gmail.com	17D41A0442	CHILUKURI POOJITHA	ECE	Yes	9987342181
manoharraj4122@gmail.com	18D41A04H6	R MANOHAR	ECE	Yes	9703833036
boggirajesh@gmail.com	18D41A0430	Boggiri rajesh raman	ECE	Yes	9963313631
bharat1981@gmail.com	18D41A04K5	Bharat Chandra	ECE	Yes	9391152329

nagasai1118@gmail.com	18D41A04J1	REDROWTHU NAGA SAI SRAVAN	ECE	Yes	9133227318
bsowmya1830@gmail.com	18D41A0423	Sowmya Boddu	ECE	Yes	9973321880
umamaheswariyadav1978@gmail.com	18D41A0474	KADARI SOWPATHI	ECE	Yes	9399424409
manireddy2365@gmail.com	17D41A0493	Sanireddy Mani kumar reddy	ECE	Yes	9059327266
maanyaraj21@gmail.com	18D41A04N3	Yata Maanya Raj	ECE	Yes	9490330204
pallavinari1999@gmail.com	17D41A04P4	NARAYANAPOBAPU PALLAVI NARI	ECE	Yes	9032935496
mounikabolle411@gmail.com	18D41A0434	Bolle Mounika	ECE	Yes	9398272130
shivangoud683@gmail.com	17D41A0452	P Shivani	ECE	Yes	7676977663
sanithreddy2628@gmail.com	17D41A0457	Dede Sanithi	ECE	Yes	8301918279
divyabolle12@gmail.com	18D41A0432	Bolle Divya	ECE	Yes	9390067371
shyamreddy6369@gmail.com	18D41A0458	KOTA MEGHA SHYAM REDDY	ECE	Yes	9848536303
maheshgoud578@gmail.com	18D41A0465	Talupunoori Mahesh Goud	ECE	Yes	9618741883
saikumarpeddala3@gmail.com	17D41A04K2	Pedala Saikumar	ECE	Yes	8700092155
bapputmmu0419@gmail.com	18D45A0409	Bajjuri Supriya	ECE	Yes	6309683336
sailakshmanmadala@gmail.com	18D41A0403	MADALA SAI LAKSHMAN	ECE	Yes	8309065804
himabindugoud0@gmail.com	18D41A04J3	ROLLA HIMA BINDU GOUD	ECE	Yes	7386784612
thummalaabhiruthreddy@gmail.com	18D41A04H4	THUMMALA ABHIRUTH REDDY	ECE	Yes	7032223528
gundagouthamreddy@gmail.com	17D41A0458	GUNDA GOUTHAM REDDY	ECE	Yes	9700303339
gundagouthamreddy@gmail.com	17D41A0458	GUNDA GOUTHAM REDDY	ECE	Yes	9700303339
dineshreddydomala@gmail.com	17D41A0460	Domala Dinesh Reddy	ECE	Yes	8300810637
gopaldasupavanipavani@gmail.com	18D41A0457	Gopaldasu pavani	ECE	Yes	9390693893
nandinirenthala@gmail.com	18D41A04J3	RENTHALA NANDINI	ECE	Yes	9391985984
archanapuppireddy@gmail.com	17D41A04H6	Puppireddy Archana	ECE	Yes	8309922502
saikota246@gmail.com	17D41A0486	Kota sai chandu	ECE	Yes	8701285420
Tejavardhan1?@gmail.com	18D41A0447	CHILKUR TEJA VARDHAN REDDY	ECE	Yes	8639428148
etikyalaarunkumar@gmail.com	18D45A0403	ETIKYALA ARUNKUMAR	ECE	Yes	7093352239
vinay0188@gmail.com	18D41A0490	KASSETTY VINAY	ECE	No	9110332839
sudhesh143nai@gmail.com	17D41A0452	Jatavath Sudhesh Naik	ECE	Yes	8300161932
sharathkumarsalluvala003@gmail.com	18D41A04J7	SALLUVALA SHARATH KUMAR	ECE	No	7893335412
sanjanabheemreddy21@gmail.com	18D41A0427	BHEEMREDDY SANJANA	ECE	Yes	8491319071
sindhunamani788@gmail.com	18D41A0405	NAMANI SINDHU	ECE	Yes	9915591428
bsowmya1830@gmail.com	18D41A0423	Sowmya Boddu	ECE	Yes	9573321880
maheshbabukalvakuntla2000@gmail.com	17D41A0458	Kalvakuntl Mahesh Babu	ECE	Yes	8309915419
kammareddy811@gmail.com	17D41A04A4	Katti Kamleek Reddy	ECE	Yes	8.19808E+11
tandrasanjanareddy4028@gmail.com	17D41A04L8	T sanjana	ECE	Yes	9704420972
muppidipramod813@gmail.com	17D41A0483	Muppidi pramod	ECE	Yes	9494055540
jangiliraghavendra2@gmail.com	18D45A0432	JANGILI RAGHAVENDRA	ECE	Yes	9100481432
saiakhilesh.rangu@gmail.com	17D41A04H9	R Sai Akhilesh Goud	ECE	Yes	9177823834
arshadsiddique191@gmail.com	17D41A04H7	Syed Mohammed Abdul Raheem Siddique	ECE	Yes	9963436774
mosesabhishek197@gmail.com	17D41A04M2	Vemuri Moses Abhishek	ECE	Yes	7036372748
anemoniashloka1234@gmail.com	18D41A0412	Anemoni shloka mudhiraj	ECE	Yes	9390982265
tkavya2000@gmail.com	17D41A0427	T Kavya	ECE	Yes	9643310614
rahulrangarau2000@gmail.com	17D41A04J6	Rangarau Rahul	ECE	Yes	7386052615
saisridachepally03@gmail.com	18D41A0441	Dachepally Sai	ECE	Yes	7995322328

[illegible]@gmail.com	19D41A04A0	LUNAVATH RAHUL NAIK	ECE	Yes	9177137942
[illegible]@gmail.com	19D41A0439	E Sahith	ECE	Yes	9542763976
[illegible]@gmail.com	17D41A0438	[illegible]	ECE	Yes	9989902522
gayathrisagar2001@gmail.com	19D41A0438	BUDURU GAYATRI	ECE	Yes	7036425456
[illegible]@gmail.com	19D41A0417	[illegible] Madhuri	ECE	Yes	9390425147
[illegible]@gmail.com	19D41A04A3	MAHESH BHUKYA	ECE	Yes	7286770204
Ruchitha[illegible]@gmail.com	19D41A0451	Ruchitha	ECE	Yes	9121746936
[illegible]@gmail.com	17D41A0491	Nimisha Reddy	ECE	Yes	9396003406
[illegible]@gmail.com	19D41A0424	BEEMAGANI DEDEEPYA	ECE	Yes	9542013059
[illegible]@gmail.com	17D41A0405	L.[illegible]	ECE	Yes	9398817784
[illegible]@gmail.com	17D41A0482	KONTHAM LASYA REDDY	ECE	Yes	9133073913
[illegible]@gmail.com	19D45A0422	KESOJI MANISH KUMAR	ECE	Yes	7013394005
[illegible]@gmail.com	17D41A0438	E. SNEHA	ECE	Yes	9374965426
[illegible]@gmail.com	19D41A0402	MADALA SAI LAKSHMAN	ECE	Yes	9303038804
[illegible]2204@gmail.com	17D41A0484	MONDRU DIVYA SREE	ECE	Yes	9247446306
[illegible]@gmail.com	17D41A0405	MADHURI NERAVETLA	ECE	Yes	9398164161

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

IMPLEMENTATION OF IMAGE PROCESSING CONCEPTS FOR REALTIME APPLICATIONS USING MATLAB

Date: From 19.09.2021 (6 Week Course, Only on Saturdays)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
1	18D41A04F9	P VIVEK VARDHAN REDDY						
2	18D41A04E0	MOHAMMED ADNAN				← A →		
3	18D41A04F2	NEELESH KUMAR GOUD						
4	18D41A04G0	PARITALA GANESH						
5	18D41A04N8	RANAVATH ABHISHEK RAJ NAIK						
6	18D41A04L5	THALASANI VAMSHIDHAR REDDY						
7	18D41A04E9	N.THARUN KUMAR REDDY						
8	18D41A04D7	MASADI AJAY						
9	18D41A04G7	PIRLA VENKATA SUDHEER						← A →
10	17D41A0483	GUDIPALLY.MANASA						
11	17D41A04J0	RAMISHETTI VARUN KUMAR						
12	17D41A0408	ANGOLI RISHWANTH GPUD		← A →				
13	18D41A04Q0	PUNNA KARTHIK						
14	18D41A04H1	AMRUTHA VARSHINI PONNA						
15	19D45A0418	D INDRA KUMAR						
16	18D41A04H0	POLOJU POORNA CHANDAR						
17	18D41A04P8	VAISHNAVI						

18	17D41A0483	GUDIPALLY.MANASA							
19	18D41A04J8	SAMALA ROSHAN							
20	18D41A04F3	P UNESHCHANDRA							
21	18D41A04M5	VADYALA SATHWIKA							
22	18D41A04C2	M NISHITHA							
23	18D41A04L3	SUREPALLY.MOUNIKA							
24	18D41A04L3	SUREPALLY.MOUNIKA							
25	18D41A0425	BHAVANI YAMSANI							
		Present →	25/25	24/25	25/25	23/25	23/25	24/25	

1.

2.

Coordinator

Convener

HOD/ECE

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
NAAC Approved by AICTE &
Permanently Affiliated to JNTUH

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VBB enabled Projects using Arduino

STARTS ON March 12, 2021

SLOT-I REGISTRATION OPEN

Registration : Free
Course Duration : 12 Hours
Weekend Course (Friday & Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 25 Participants/Slot
Resource Persons: In-house Trainers

Coordinators
Dr.P.Mukunthan
Contact: 9894145701

Convener
Prof.K.Ashok Babu

Principal
Dr.G.Suresh

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

VBB Enabled Projects using Arduino

Date: From 12.03.2021 (3 Week Course, Only on Friday & Saturday)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1		Week 2		Week 3	
1	18D41A04H5	Pyarasani Karthik	Karthik	Karthik	Karthik	Karthik	Karthik	Karthik
2	18D41A04H3	Pusapati Narasimha Raju	P. Raju	P. Raju	P. Raju	P. Raju	P. Raju	P. Raju
3	18D41A04H4	Puttoju Ashrith Chaithanya	Chaithanya	Chaithanya	Chaithanya	Chaithanya	Chaithanya	Chaithanya
4	18D41A04G3	Pasula Aishwarya	Aishwarya	Aishwarya	Aishwarya	Aishwarya	Aishwarya	Aishwarya
5	18D41A04L4	Surepally Pavan	S Pavan	S Pavan	S Pavan	S Pavan	S Pavan	S Pavan
6	18D41A04J9	Sampati Archana	S Archana	S Archana	S Archana	S Archana	S Archana	S Archana
7	18D41A04J0	Rangineni Aparna	Aparna	Aparna	Aparna	Aparna	← A →	← A →
8	18D41A04F1	Narsapoia Nishanth Reddy	N Reddy	N Reddy	N Reddy	N Reddy	N Reddy	N Reddy
9	18D41A04P2	R.Chandana	Chandana	Chandana	Chandana	Chandana	Chandana	Chandana
10	18D41A04A5	Konduri Madhukar	Madhukar	Madhukar	Madhukar	Madhukar	Madhukar	Madhukar
11	18D41A04H2	Poshanaboina Naveen Kumar	P. Kumar	P. Kumar	P. Kumar	P. Kumar	P. Kumar	P. Kumar
12	18D41A04A0	Kodamarthi Sai Rohith	Rohith	Rohith	Rohith	Rohith	Rohith	Rohith
13	18D41A0461	Deva Krishna Kanth	Krishnakanth	Krishnakanth	Krishnakanth	Krishnakanth	Krishnakanth	Krishnakanth
14	~~18D41A04P2~~	~~R.Chandana~~						
15	18D41A04F4	Pakala Sarayu	P. Sayu	P. Sarayu	P. Sarayu	P. Sarayu	P. Sarayu	P. Sarayu
16	17D41A0419	Bagannagari Sumanth Reddy	S Reddy	S Reddy	S Reddy	S Reddy	S Reddy	S Reddy
17	18D41A04E4	Mu ala Sai Preetham	Preetham	← A →	Preetham	Preetham	Preetham	Preetham

18	18D41A04P4	Nannuri Ruchika Reddy	Present	Present	Present	Present	Present	Present
19	17D41A0411	Aravilli Venkata Chakra Shree Harsha						
20	17D41A04D1	Malla Kondal Reddy						
21	18d41a0431	B Abhinav Reddy						
22	17D41A0499	Kalluri Sandeep Reddy						
23	17D41A04A3	K Pavan Kumar						
24	17D41A04B7	Kotagiri Nitish						
25	18D41A0497	Aavula Sairuthwik Vamshikar						
			(24)	(23)	(23)	(23)	(23)	(23)

Coordinator

Convener

HOD/ECE

Sri Indu

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VBB enabled Projects using Arduino

STARTS ON March 12, 2021

SLOT-I REGISTRATION OPEN

Registration : Free
Course Duration : 12 Hours
Weekend Course (Friday & Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 25 Participants/Slot
Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Dr.P.Mukunthan	Prof.K.Ashok Babu	Dr.G.Suresh
Contact: 9894145701		

Introducing Virtual Breadboard Windows App

The Virtual Breadboard (VBB) modern App for the Windows Universal Platform (UWP) uses the Fluent Design System modelled on the Windows Paint 3D App that we all know and love.

In VBB the main design area

(1) is a Design Sheet where you layout Virtual Breadboard circuits which can be virtualized by powering on the circuit (2). The menu (3) opens a navigation view where standard file dialogs along with examples, trainings and account management can be accessed. While designing the toolbar ribbon (4) selects the design mode which shows context sensitive tool panes in the right hand panel (5).

Main functional regions of the App:

1. Design Sheets : Are where you design your Breadboards
2. Power On : Power Up the circuit to start the interactive Virtualization
3. Menu : Open and save projects, find examples and take training and access your account
4. Tools Ribbon : Access the tools for editing and managing your project from the tools ribbon
5. Tools Pane : Tools from the Tools Ribbon are accessed from the Tools Pane

Avatar Hardware

Your real microcontroller is inserted into a Virtual Breadboard via an Avatar Hardware interface.

Your microcontroller cannot tell the difference between virtual or real.

Other real components can also be connected to your microcontroller creating micro-mixed-reality.

App Menu Navigation View

The Menu Navigation view slide out panel provides access to the file management, account, settings and other management features.

The platform workspace environment features:

1. File Menu : New, Open, Save, Save as file options
2. Remotify : Publish virtual hardware to the Cloud
3. Student Manager : Create and manage student accounts
4. Export : Export and exchange VBB designs in SVG and KiCad formats
5. Examples : Browse reference examples for quickstarting a project.
6. Training : Awards based training system for getting started.
7. Keyboard Shortcuts : Awards based training system for getting started.
8. Software Store : Make In-App purchases from the Software Store.
9. Hardware Store : Browse the available Avatar interface modules.
10. What's New : Displays the Splash Screen which contains news and quick access tutorials.
11. Learn and Feedback : Access Documentation, YouTube and CodeLab Tutorials and Forum.
12. Account : Account status and login.
13. Settings : Project wide settings and App information.

Files

Virtual Breadboard projects are stored in .VBB files. Standard file dialogs are used to open and save .VBB files.

New

Click New to create a New blank project. If you already have a project open you will be prompted to save the project before the New project is created.

Open

Click Open displays the Open panel with tools to open existing project. If you already have a project

open you will be prompted to save the project before the existing project is opened.

1. Browse Files : Click to open a file dialog to locate .vbb files to open
2. Import Previous Project : Click to open a folder dialog to locate and import project folders from earlier versions of VBB
3. Recent Breadboard Files : A listing of recently used .vbb project files. Click a project from the list to open directly

Save

Saves the current project to the currently selected .vbb file. If this is the first time you are saving the project the Save as will be activate instead.

Save as

Click to open a Save File Dialog to provide a new name and location for your .vbb file and save to the new file.

Remotify

Remotify is a publishing system that enables virtual circuits to be published to the internet and played in a Browser.

You can think of Remotify as a Cloud File and Folder manager where the Files are VBB Projects and the folders are Groups of VBB Projects.

Remotify manager maintains a tree heirachy view of the Groups and Projects.

The basic procedure is

1. Create Group
2. Add Project to Group
3. Publish Group or Individual Projects in the group
4. Paste the publish Link to the browser

When publishing url link is copied to the clipboard which can be used in a Browser to Play the virtual hardware project.

HTML5 WebPlayer

The WebPlayer is a lightweight Html javascript client that connects a Html5 Canvas renderer to a docker container instance of the .Net CORE version of the VBB runtime connected over SignalR. The docker container is hosted as an Azure container instance and spun up on demand per user session so all users have their own dedicated instances giving consistent scaleable performance for each user.

Despite being an economical approach there is a cost associated with hosting cloud sessions and hence the basic subscriptions covers development and casual personal use of the WebPlayer only. If useage grows beyond fair use a seperate usage based susbcription will apply.

HTML5 Canvas currently works on the latest versions of

- Chrome
- Edge

Published Group

When a group is published a url will be saved to the clipboard. The url can then be pasted directly to a browser or an alternate such as notepad. The url can then be shared by email or a link in a content management system. When the url is opened or pasted into a browser it will load the group access page with the Name, Description and Thumbnail taken from the group editor fields and the projects.

There are two types of Browser view

- Collections of Groups
- Collections of Projects

Group Collections Viewer

Project Collections Viewer

Published Project

When a published project is opened from a Group view or by directly pasting or linking the project url the project will load into a project Viewer. If this is the first project accessed in a browser session then a session container will be spun up which currently takes around 30 seconds. In the future standby containers will be used to reduce this intial spin up time. If when switching to a new project within the same session the browser uses can track the current session and reuses the same container. Hence there is no additional spin up time when switching between projects as long as the same browser window is used and standard navigate back button is used to navigate the groups collection heirachy. Each session has a time limit after which it will expire. This can depend on the user account.

Project Viewer Load

Project Viewer Run

Remotify Manager Tools

1. Subscription : A subscription and Virtual Breadboard account is required to activate Remotify
2. Remotify : Remotify is accessed from the Navigation View Menu
3. WebPlayer Play Lists : The root directory of the web group collections
4. Group : A Group is a collection of Groups or Projects
5. Project : A Virtual Breadboard Project stored in a Group
6. Group/Project Toolbar : The Toolbar of the currently selected Group or Project
7. Current Project : The VBB project currently open when Remotify Menu was selected

1. Subscription Status

To activate the Remotify Manager requires :

- PRO or CLASSROOM Subscription
- Virtual Breadboard Account

For more information about accounts see here : Account

If not activated the Remotify Manager will show the following message:

2. Remotify Menu

Remotify is accessed from the Navigation View Menu. This

3. WebPlayer Play Lists

The root remotify group contains all the Group or Project collections. There can be only groups or projects at the same level of the heirachy. The Name, Description and Thumbnail are not editable for the root remotify group.

Icon	Function
	Adds a new Group Child to the root directtory
	Publishes the project, making it public and copying a url link to the clipboard
	UnPublishes the project making it private and not visible in the browser

4. Group

A Group is a collection of Groups or Projects. When selected a group will be highlighted with a blue strip in the heirachy and the group information will be displayed in the Group pane.

5. Project

A Virtual Breadboard Project stored in a Group. When selected a project will be highlighted with a blue strip in the heirachy and the project information will be displayed in the Project pane.

6. Selected Group/Project Toolbar

6.1 Selected Group Toolbar

The Group definition is edited in the Group Pane. The Name, Description can be edited to be displayed in the Browser Group Viewer to inform the nature of the projects in the group. The Thumbnail is also editable and must be a 290x200 png image.

Icon	Function
	Adds a new Group as a child of this group.
	Decrements the selected group order moving it up one place in the list
	Increments the selected group order moving it down one place in the list
	Deletes the selected group if empty removing the group from the cloud.
	Publishes the project, making it public and copying a url link to the clipboard
	UnPublishes the project making it private and not visible in the browser
Update Thu	Opens a dialog to select an 290x200 png image to be the new thumbnail.
Save Ch	Saves the new group definition to the remotify cloud.

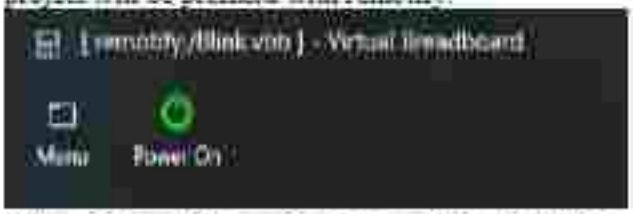
6.2 Selected Project Toolbar

The Project definition is viewed in the Project Pane. The Name, Description are copied over from the Info designsheet in the VBB Project. The remotify project can be opened and later saved directly to the remotify cloud making remotify a type of cloud drive for VBB projects. The toolbar tools are used to manage the visibility and membership of the project in it's group.

Icon	Function
	Decrements the selected project order moving it up one place in the list
	Increments the selected project order moving it down one place in the list
	Deletes the selected project removing it from the cloud store
	Publishes the project, making it public and copying a url link to the clipboard
	UnPublishes the project making it private and not visible in the browser
Open P	Opens the remote remotify project as the current project

Opening Remote Project

When opening a remote project the title of the project will be prefixed with remotify.



When a remotify project is saved it it will be updated directly in the cloud. In this way remotify acts as a cloud drive for VEB projects.

7. Current Project

When navigating to the Remotify Manager there is a project currently open in the background. A snapshot of this project is shown in the Current Project pane. The Name and Summary are taken from the project information of the current project and the Thumnail is automatically generated from the active Breadboard. The Project tools are used to connect the project with a Remotify Group.

Icon	Function
	Add Project to the currently selected group

Troubleshooting

- Firefox does not seem to work with the way VEB uses the Html5 Canvas at this time
- Projects should be single Breadboard projects.
- A Group should contain only projects or groups but not both otherwise the web player will not function correctly.

Roadmap

- ESP8266 is not currently available – backend version needs to be updated.
- Connect Avatars to Remotify Browsers via RasPi Server to create remote labs

Student Manager

Classroom administrators use the Student Manager to create and manage Student Accounts. The task of the Student Manager is to enable an Administrator to register Student Accounts by following these steps.

1. Edit or import the list of Student Names
2. Set a password for all Students.
3. Click the Register Students Button

Notes:

- Student Names should be unique names.
- The password should be unique to your classroom.
- Student names and accounts can be changed and updated as required.
- Students logged on when registrations are updated will have to logon again.
- Only one student can be logged onto each student name at a time.

Student Accounts

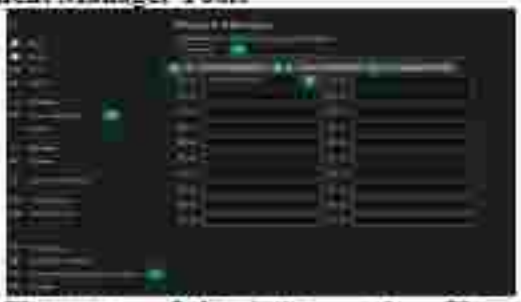
Student Accounts are suitable for school use where there are often privacy concerns. A Student license only uses a nickname and is maintained by classroom administrator so there is no student information, email or microsoft account information stored with a Student Account. A Student Account has full access to VEB except with only a few restrictions.

Student Account Restricted Access:

- Student Manager

- Remotify Publishing
- CDK Publishing

Student Manager Tools



1. Classroom Subscription : A Classroom subscription is required to activate the Student Manager
2. Student Manager : The Student Manager is access from the Student Manager Navigation Menu
3. Student Names : The Student Manager maintains a list of editable student names representing a class
4. PassPhrase : A shared password is created by editing the password textbox
5. Import Student Names Button : Imports a list of student names
6. Export Student Names Button : Export the current list of students can be d using this button
7. Register Students Button : Create a Virtual Breadboard Student account for each named student with the shared passphrase

1. Classroom Subscription

To activate the Student Manager requires two accounts

- Microsoft Classroom Subscription
- Virtual Breadboard Account

For more information about accounts see here : Account

If not activated the Student Manager will show the following message

Microsoft Classroom Subscription

The Microsoft Classroom Subscription account manages the subscription information. There are 4 classroom subscriptions available to suit different class sizes of 10,20,30 and 60 students. These subscriptions are annual subscription and we have partnered with Microsoft using their In-App purchasing system to simplify the aquistion, invoicing, localised sales taxes and management of these licenses. When you start the Virtual Breadboard App your account information is already known from your Microsoft Windows 10 Logon and so the subscription information is queried automatically using this account.

A classroom subscriber will have the CLASSROOM name shown in their Account status

Note : There are actually 2 account required to work with the Student Manager so if only the subscription account is available '*' is shown in the status name to show the account is only partially activated.

Virtual Breadboard Account

The Virtual Breadboard Account manages the access to Virtual Breeadboard Cloud backend. The Student Accounts are registered in the cloud which enables students to logon from any location. For more information on creating a Virtual Breadboard account see : Account

When both accounts are registered your account status will display CLASSROOM without the '*' showing the CLASSROOM account is fully activated.

2. Student Manager

The Student Manager is access from the Navigation Menu.

3. Student Names

An editable collection of student names is managed by the Student Manager.

- Clicking a name will highlight the text ready for editing
- Use the Arrow keys to navigate between names

4. Pass Phrase

A single password is shared by all users. The Password should be longer than 6 letters and should be easy to remember but not super obvious.

5. Import Student Names

To make it easy to maintain class lists you can import a named list of students.

- Should be a .txt file
- One student per line
- If more students are in the list than are available in the subscription the list will be truncated.

6. Export Student Names

To make it easy to maintain class lists you can Export a named list of students.

- The file will be .txt file
- One student per line

7. Register Students
Registers the current list of Student names creating a unique account for each student.
If you have students names that are not uniqtie or there is some other error then you will receive a warning.
If the registration is successful you will receive a success message.

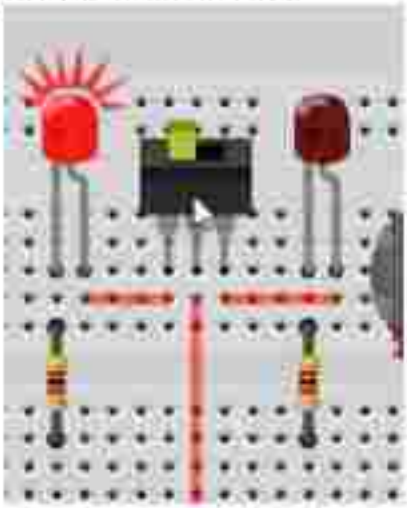
VBB101
Contents:

- Interactive

Toggle a Switch

A 3 pin switch has two possible configurations. It can connect the command center pin with the left pin or the right pin. When connected to one pin the other pin is disconnected or open circuit. The switch position toggles to show which pin is connected to the center pin.

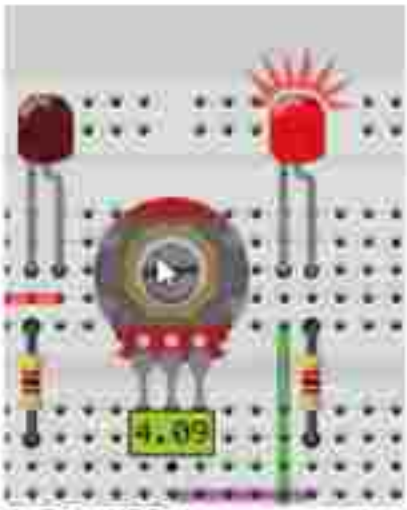
Award : While Powered Up locate the Toggle Switch and click the component on the switch location. The position of the switch slider will toggle and the alternate circuit will be completed toggling which L.E.D is activated.



Rotate a Potentiometer

A potentiometer is a variable resistor which when you rotate the dial it's resistance changes which changes the output voltage of the central pin. Virtual Breadboard models the voltage output of the potentiometer voltage divider which can be read by instruments like the digital voltmeter and visualised by components like the L.E.D which are sensitive to analog voltage levels.

Award: While Powered Up locate the Potentiometer and press and hold the mouse down on the potentiometer dial. Rotate the dial around the center for the full range of voltage values. To receive this reward you need to exercise the full potentiometer range. You will see the L.E.D dim when the voltage is low and brighten when the voltage is high.



Advanced TIP: VBB is not an analog PSPICE simulator but instead models high level circuit behaviour. For example a PSPICE simulator will model the reduced L.E.D current and bandgap and deduce the L.E.D should dim but VBB models common circuit behaviours directly.

Junctions

Learn how to join links with junctions.

Junction Mode

Links are drawn between component pins. However there is often the need to link multiple pins together on the same wire. You can draw multiple individual wires from pin to pin but you can also use junctions to simplify circuit layout and make it easier to understand what the connections are.

Award: Select Junction Mode from Wires ToolTab

Place Junctions

When in Junction Mode the cursor changes to a cross-hair and when the cursor is pressed a junction is placed at the cursor location. Junctions should be placed at the end point of a wire and a wire segment. This joins all the wires into a single wire.

Award: Place a junction at each of the 3 wire-t-junctions.

Exercise : Power Up and Verify

When correctly placed the wire becomes a single wire leading from the DIP power source to the LED lights. All the LEDS should light up when the LED is in the 'on' position.

Wire Links

Wires

Wires link component pins to form circuits. Real breadboard 'wires' are usually coloured with plastic protection and are stripped back to wire only at the ends. Virtual Breadboard (VBB) wires are the also only active at the ends and snap to pins at either end. When successfully snapped the wire thickens slightly to give a visual clue the link has been successfully made.

Enter Wire Mode

Wire mode is a design mode and is set from the Wires tool tab. When in wire mode the cursor changes to a colored cross-hair and cursor actions draw wires.

Award: Select the Wires Tab and press a colored Wire Button to enter Wire Mode.

Draw a Link

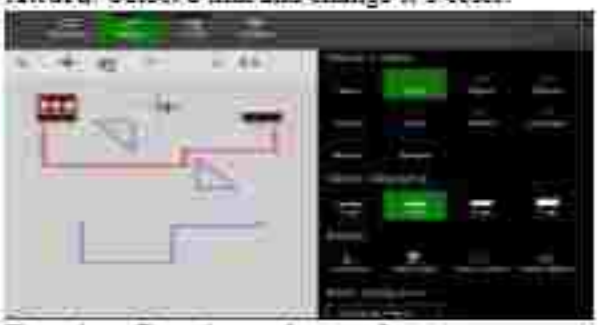
A link is active only at it's ends but it can have multiple 'joint's along it's length to enable better layout organisation than just have a wire running from end to end. Links are drawing by clicking the joints with the left button then either **clicking the right mouse button to end** the link at the last joint **or double clicking** to make the current joint the final link.

Award: Draw a link from DIP pin 1 to LED pin 3 avoiding the obstacles with pin joints. To get this award you need to link the correct pins (DIP Pin 1, LED Pin 3).

Change the Link Color

If you change your mind about the color of the link you can change it by first selecting it either in move or select mode. When selected the links properties will be shown in the properties pane. Select a new color from color property.

Award: Select a link and change it's color.



Exercise : Practice makes perfect

Repeat the previous step drawing a link from DIP Pin 2, to LED pin 2. This time use the alternative link ending. So if you used right click method, use double click this time. Or if you used double click use the right click method this time.

Reposition Link Corner

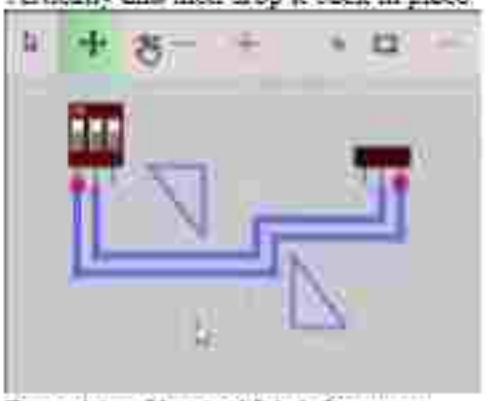
When in Move Mode links can be moved around in several different ways. You can move joints, sections or move the link as a whole.

Award: Enter move mode and drag a corner of a link, move it around and then drop it back in place.

Reposition Horizontal Link Segment

If you drag a link section from the middle of the section instead of the corners the whole section will move. If the section is horizontal it can be moved vertically.

Award: Drag a horizontal link section up and down vertically and then drop it back in place.



Reposition Vertical Link Segment

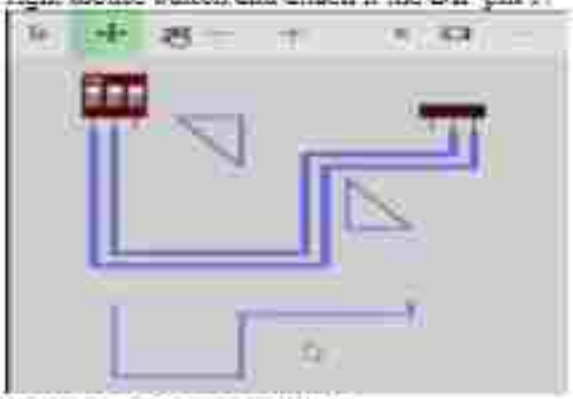
If the link section is vertical it can be moved horizontally.

Award: Drag a vertical link section left and right horizontally and then drop it back in place.

Moving Links

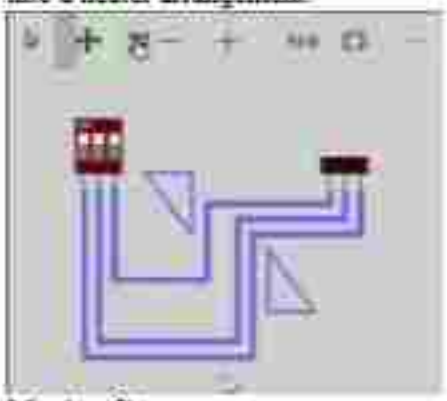
You can also move a link either as part of a group or as a single link by dragging with the **right mouse** button.

Award: Drag the whole unattached link using the right mouse button and attach it the DIP pin 5.



Exercise : Reorganise Links

A neatly organized layout is easier to understand. Practice your new skills to re-arrange the layout into a neater arrangement.



Moving Components

Moving Components

Virtual Breadboard (VBB) components can be moved around a design by dragging and dropping them while in **Move Mode** .

Select Move Mode

Move mode (**ShortCut V**) is one of the Editor Design Modes which can be selected from the Edit mode toolbar. Move Mode is used when you want to move components around.

Award: Enter move mode by selecting the move icon from the mode toolbar.

Move Component

When in move mode components can be dragged and dropped into position by pressing and holding and moving to the new location. VBB dynamically calculates the connections the component is making with other components and highlights the shared pin contacts. This is a useful visual guide for making sure components are correctly connected.

Award: Drag and Drop the L.E.D to the left most wire/resistor connection using the contact point

highlighting to make sure the L.E.D is correctly aligned.

Copy and Paste

Use **Copy** and **Paste** to duplicate the currently selected breadboard elements. Copy copies to the Clipboard and Paste creates new versions of the copied components with a small offset to the originally selected elements and selects the newly created elements.

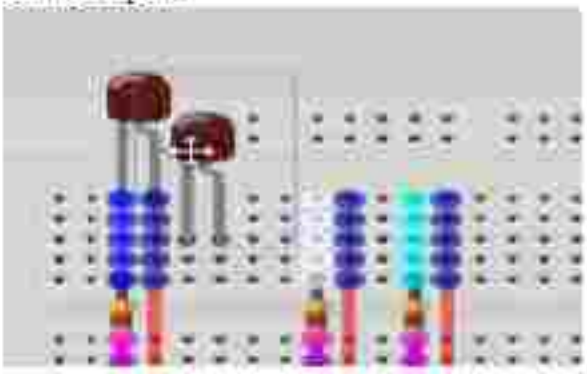
The Select toolbar contains functions that work with the currently selected elements including Copy and Paste.

Award: From the Selected toolbar first press Copy and then press Paste to duplicate the L.E.D.

Advanced Tip: Copy stores the currently selection to the Clipboard and can be pasted to other Breadboards in the project or even Breadboards in other Project.

Move Duplicate

Move the duplicate L.E.D to the next resistor link connector point.



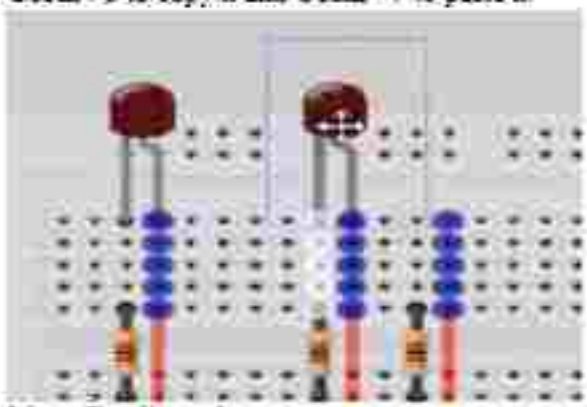
Copy and Paste with ShortCut keys

In the previous award the Copy and Paste functions were activated from the toolbar. The VBB App is designed with touchscreens in mind so all functions are available with touchable sized buttons.

However for keyboard users Copy and Paste have well defined shortcut keys in windows || Function || ShortCut | Copy | CTRL+C | Paste | CTRL+V | Cut | CTRL+X

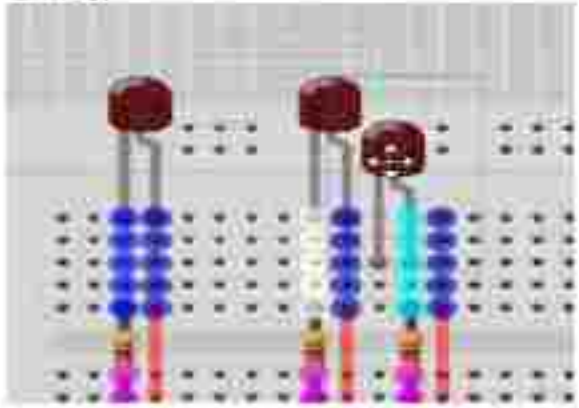
These shortcuts can often be faster to use when working in desktop environment.

Award: Copy and Paste the duplicate L.E.D using the short cut keys by first selecting it then using the CTRL+C to copy it and CTRL+V to paste it.



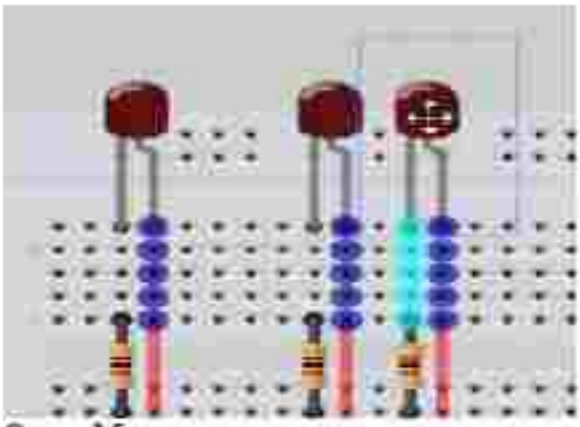
Move Duplicate 2

Move the second duplicate L.E.D to the next resistor link connector point inline with the first two L.E.D's.



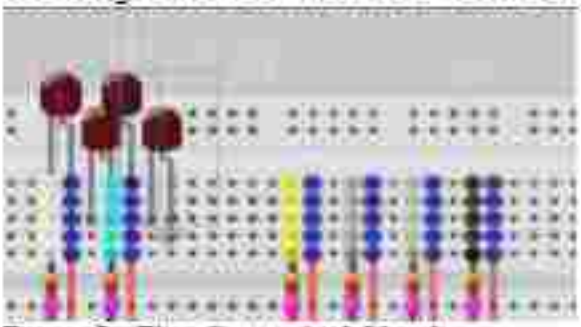
Group Selection Append

In an earlier award a group of components were selected by drawing a selection window around the whole group. You can also create groups by appending new components to the group by holding down the **SHIFT** key and then selecting other components to add to the selection group.



Group Move

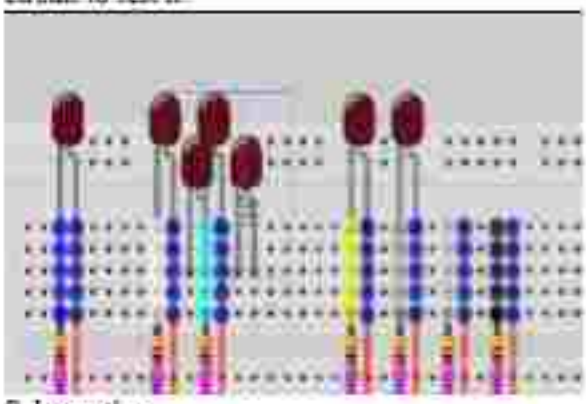
Moving a group of selected elements can be much faster than moving components one at a time. To move the whole group hold down the **SHIFT** key and press and hold one of the selected components. You can then drag and drop the group. You can also use the **right mouse** button without the SHIFT key.



Power On First Customined Circuit

Repeat the previous steps 2 steps to practice the copy and moving components.

When ready you can then Power On your circuit to test it.



Schematics

Schematics

Schematics are abstract representations of physical components. It can be easier to understand how a circuit functions when represented in schematic format. Often schematics are used to describe a circuits netlist in a separate design step. In Virtual Breadboard (VBB) Schematics and Breadboard components are merged into one design sheet creating hybrid designs where all the information about the design is in one place.

- Components with the same ID are linked
- Components have schematic/breadboard representations
- When a circuit has a schematic and breadboard component with the same ID the schematic component creates the netlist for the breadboard component.

Edit 'ID' property

When a component shares the same 'ID' property the pin the pins of both components are linked with virtual links.

Award : Edit the property of the LED to be D1

Common 'ID' Common Netlist

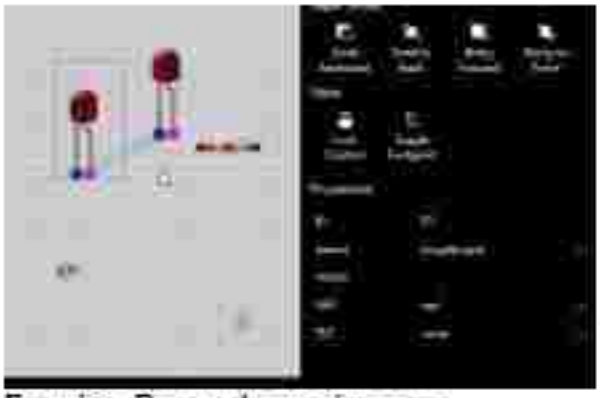
Components with the same ID share a netlist between them. To see this copy and paste a component with the same ID and Show the Nets to see the hidden links between the components.

Exercise: Duplicate LED D1 and Show Nets

Edit 'Layout' property

The Layout property determines which type of view of the component is rendered. So far you have only seen the Breadboard layout. Many components also support a Schematic layout.

Award: Select the duplicate LED and set it's layout property to Schematic



Exercise : Repeat the previous steps

Repeat the above steps for the Resistor * Edit the Resistor property to be R1 Copy and Paste the Resistor Set the duplicate Resistor layout property to be Schematic.

Wire Up the Schematic

When the schematic twins are wired to form valid circuits the matching Breadboard component twin is also wired with the same netlist.

Award: Snap the schematic elements together to form a powered LED circuit

Exercise : 'Power Up' and verify

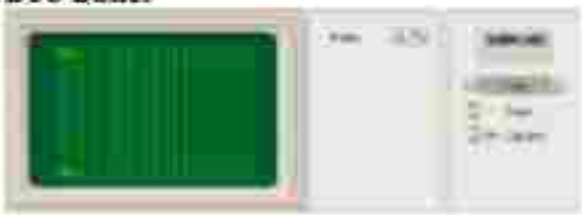
At runtime only the Breadboard component is activated. The schematic component can be used to create an understandable netlist but it does not participate in the runtime circuit.

Note: One advantage of this approach is the physical Breadboard components can be placed anywhere without the confusion of connecting wires running all over the place.

Exercise: 'Power Up' the circuit and verify the LED brightens as powered by the virtual links.

DSO Basics

DSO Basics



The **Digital Storage Oscilloscope** (DSO) is an essential tool for visualising and analysing circuits. In this training you will learn the basic steps of adding a DSO to your project and linking it to a probe.

The DSO Design Sheet

The DSO has two elements, the DSO design sheet and logic probes. The DSO design sheet is used to visualise the signals and configure the triggers and timebase view parameters. Probes are Breadboard components and are inserted into Breadboard circuits to capture signals and are linked to the DSO to send and visualise the captured signals.

Award : Add a DSO Design Sheet to the project

Viewing the DSO

Like all Design Sheets the DSO needs to be dragged and dropped into a view pane to actually view it. This allows different configurations of circuits and instrument views to be created and easily switched between.

Award : Drag and Drop the DSO Design sheet into the bottom view panel

Attaching a probe

To capture signal data to display in the DSO you add one or more probes to the circuit.

Award :Add a probe and link it to the frequency generator

Power On to start tracing

The default mode is Trace Mode which continuously captures signals with the current timebase settings. The signal sampled will be the signal shown. Without a trigger there is no specific event that starts the sampling and the signal will appear to jitter.

A Trigger is a specific event that the sampler waits for before it begins to sample the signal. The effect of this is to place the signal in known location on the screen allowing visual comparison with previous signals better suited for detecting signal changes.

Triggers are a property of the probes. When powered up the probes in the Breadboard are scanned and added to the DSO Triggers panel. These triggers are logically AND'd together to create a single filter trigger event

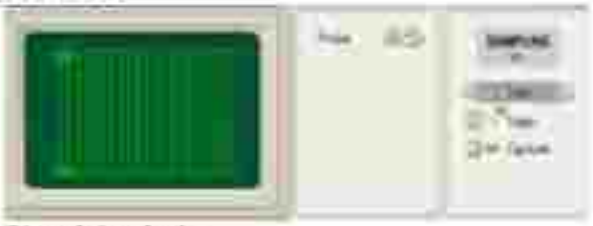
Trigger	Event	Description
	Don't care	Not used in filter
	Is LOW	True if this signal is LOW
	Is RISING	True if this signal is Rising Edge
	Is HIGH	True if this signal is HIGH
	Is FALLING	True if this signal is Falling Edge

Award :Change the Trigger to Rising Edge to stabilise the signal

Setting the Timebase

The Time base is the amount of time shown by each grid unit in the DSO display. Changing the Time base has the effect of zooming in and out of the signal.

Award :Change the Time base zooming the display in and out



Signal Analysis

The DSO is used to sample and display signals to assist in circuit analysis and troubleshooting. For example you can use it to visualise and measure the frequency generated by a frequency generator.

Exercise : Change frequency by sliding the slider of the frequency generator and visualise the signal changes in the DSO

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

HANDS ON TRAINING COURSE

ON

HANDS ON TRAINING IN NS2

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

HANDS ON TRAINING IN NS2

Date: From 25-11-2021 to 31-12-21 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	**Topics**	**Resource Person**
Week 1	Introduction to NS2	Dr. C.Kotteeswaran
	Overview	
	Why TCL. Installation of NS-2	
	Network Component	
	Node and Routing	
	Packet Flow	
	Assignment-1	
Week 2	Overvaew of ns-2 simulation test bed	Dr. C Kotteeswaran
	ns architecture	
	NS programming	
	TCL interpreter, characteristics, X Graph	
	Assignment-2	
Week 3	Basic Linux and Ns2	Dr. C.Kotteeswaran
	Node Commands	
	WIRELESS NETWORK PROGRAMS	
	Assignment-3	
MODULE -2		
Durations	**Topics**	**Resource Person**
	WSN program	
	Creation of TCP	
	AODV routing protocol	
	Multicast	

Week 4	Link	Dr. C.Kotteeswaran
	Assignment-4	
MODULE -3		
Directions	Topics	Reference Period
Week 5	Channel – Wireless Channel	Dr. C.Kotteeswaran
	Propagation Two Ray Ground Propagation	
	Queue Type – Drop Tail	
	Assessment -1	
	Conclusion	

1 NS 2 INTRODUCTION

Network Simulation (version 2) is one of the object-oriented language based discrete event-driven introduced at UC Berkely developed in two languages, namely C++ and OTcl

(Object Tool Command Language). Network Simulation is first and foremost used in the simulation of LAN and WAN network.

1.1 Overview

Network Simulation 2 is an event-driven simulator that simulates various kinds of IP networks. It implements network protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), behavior of traffic source such as File Transfer Protocol (FTP), Telnet, Web, Constant Bit Rate (CBR) and Variable Bit Rate (VBR), queue management methods such as Drop Tail, RED and CBQ, and some of the routing algorithm are used. NS also works with multicasting network oriented programs and some of the Medium Access Control (MAC) layer protocols for local area network simulations. Network Simulation project is currently working for the VINT project that introduce tools for simulation results display, analysis and converters that convert network topologies generated by well-known generators to NS formats. At present, Network Simulation (version 2) developed in C++ and OTcl is on hand. This manual discusses briefly about the basic construction of NS, and explains detaily how to make use of NS frequently by giving examples.

As shown in Figure (Simplified user view of NS2), in a simplified user's view, Network simulation is an Object-oriented tool script interpretered with simulation event scheduler and the libraries of network component object, and the libraries of network setup (plumbing) module (in fact it is the plumbing modules which are implemented as member functions of the base simulator object). Otherwise to use NS, we have to program in OTcl script language.

To create and run a simulation, OTcl script should be written by the user that creates an event, initiate the network topology set up using the objects of the network and comment the traffic sources and fix the transmission time and stop time of transmitting packets through the event scheduler.

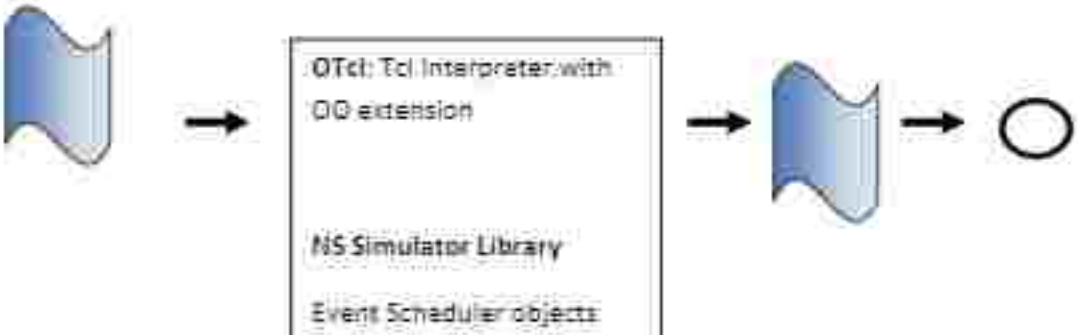


Figure Simplified User's View of NS

One more imporant component of NS beside network objects is the event scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler continuously tracking of simulation time period and fires all the simulation events in the event queue programmed for the present time by invoking suitable network components, which more often than not are the ones who issued the events in the simulation, and let them perform the suitable action connected with packet pointed by the event.

Network components communicates with one another transitory packet, however this does not devour real simulation time. Each and every network components that require spending a little simulation time for handling a packet (i.e. essential delay) use the event scheduler by providing an event for the packet and to come for the event to be fired to itself previous to doing additional action handling the packet. For example, a network switch component that handles the simulation which switch with 20 microseconds of switching delay issues an event for a data packet to be switched to the scheduler as an event 20 microsecond afterward. The scheduler following 20 microseconds handle the process of dequeue the event and fires it to the switch component, which subsequently send the packet to a suitable output link component.

One more work of an event scheduler is timer. For example, Transmission Control Protocol (TCP) needs to make use of a timer to keep tracking transmission time of a packet out for further transmission (transmission of a packet with similar TCP-packet number but dissimilar network packet identication). Timers will make use of the event schedulers in a same manner that delay does. The one and only difference in that timer is, it measures a time linked with a

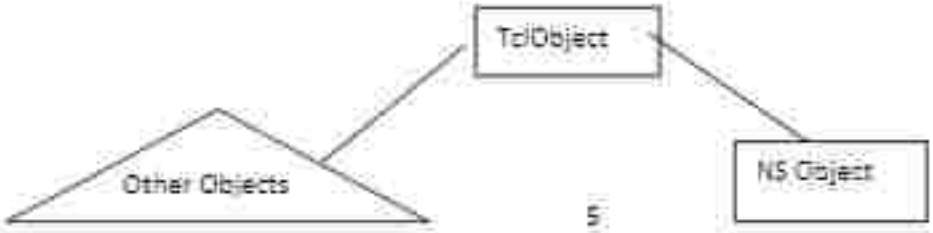
packet and does a suitable action connected to that packet after a firm time goes by, and does not simulate a time delay.

1.2 Why TCL

A user inscribe an OTcl script that's creates an event scheduler, sets up the topology of the network by make use of an objects of the network and the libraries plumbing functions, and control the traffic sources when to initiate and finalize the transmission of the packets through the event scheduler. Here plumbing is defined as a network setup, for the reason that setting up a network is plumbing possible paths for data transfer between network objects by locating the "neighbor" pointer of an object to the address of a suitable object. When a user needs to create a new network object, he or she with no trouble can make an object either by creating a fresh object or by constructing a compound object from the object library, and plumb the path of the data through the object. This may resonance like difficult job, but the plumbing OTcl modules in reality make the job trouble-free. The influence of NS comes from this plumbing.

NS is created not only in tool command language but in C++ also. For efficiency reason, NS data path implementations are separated from control path implementations. Most importantly the packet and event processing time has to be reduced, for that purpose the C++ language is used to write and compile the event scheduler and the component objects. These compiled objects are ready accessible to the OTcl interpreter through an OTcl linkage that creates a corresponding OTcl object for each and every C++ objects and makes the control functions and the configurable variables specified by the C++ object take action as member functions and member variables of the corresponding OTcl object. By the way, the controls of the C++ objects are prearranged to OTcl. It is also probable to insert member functions and variables to a C++ linked OTcl object.

1.2.1 Network Component



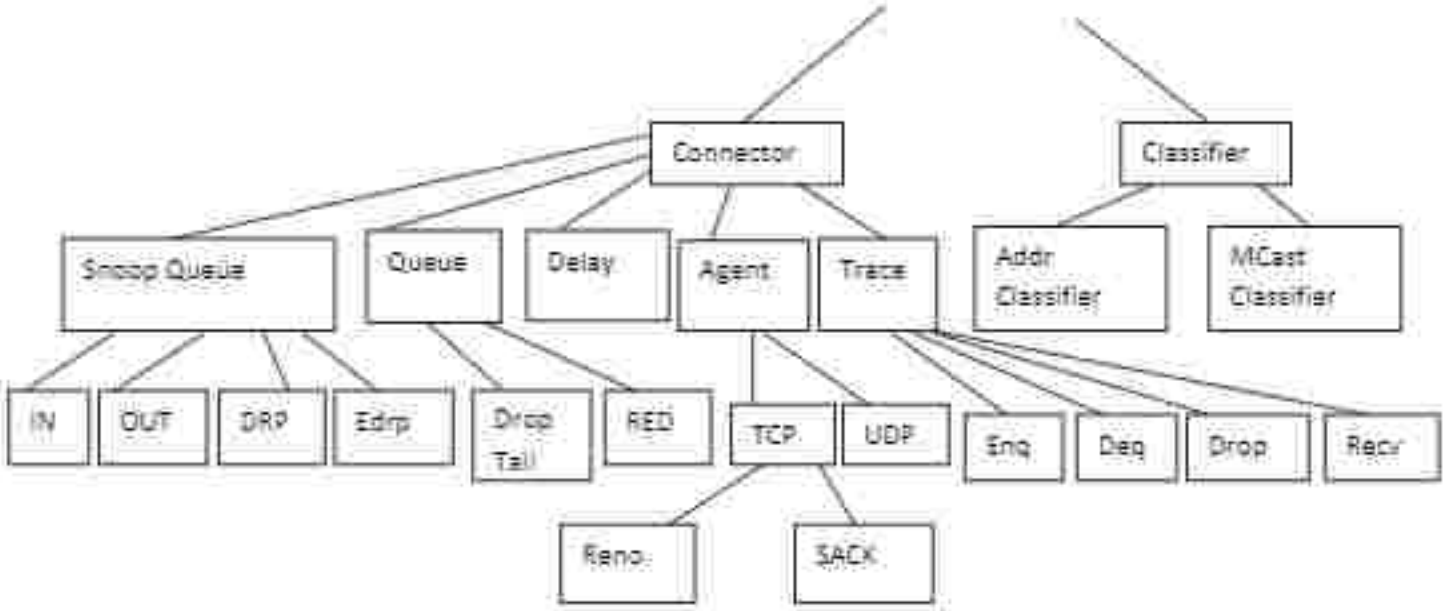


Figure Class Hierarchies (Partial)

1.2.2 Node and Routing

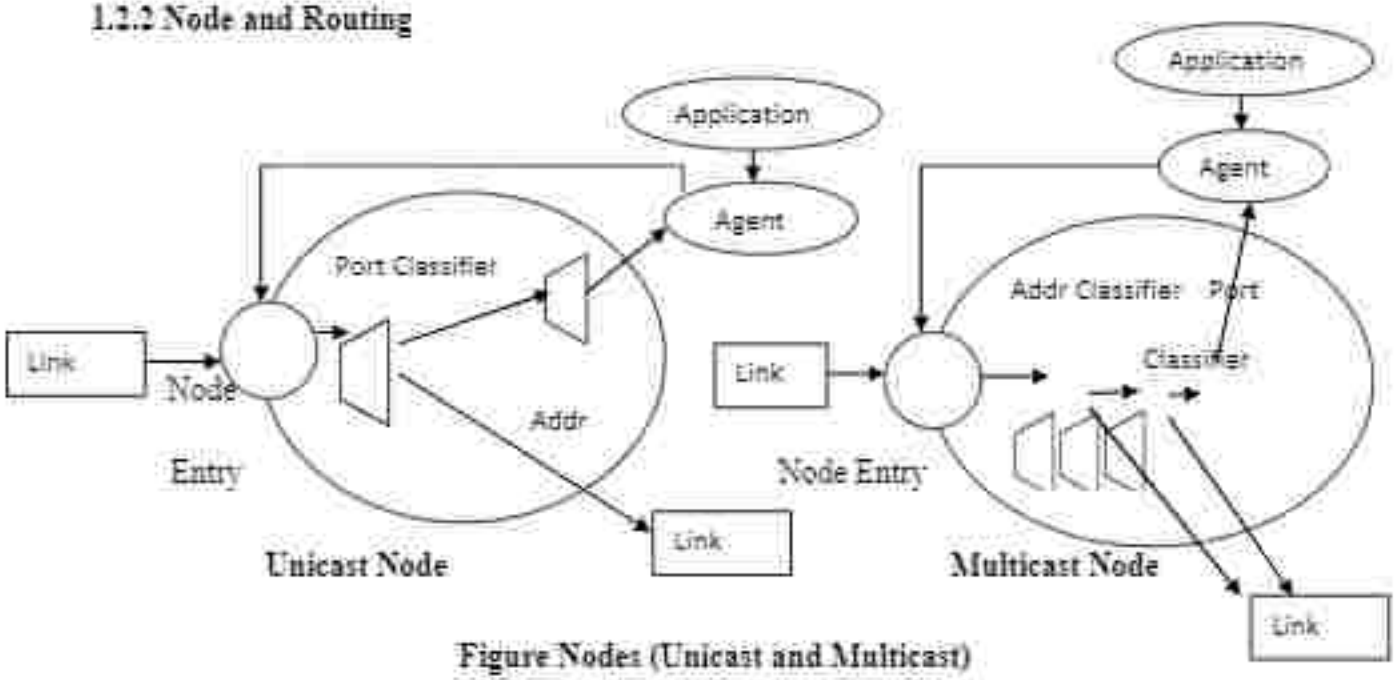


Figure Nodes (Unicast and Multicast)

A node of a network is one of a compound object composed by both the node entry object and classifiers as shown in Figure (Unicast and Multicast). There are two important types of nodes in network simulation. First is unicast node, it has an address classifier with it that does the operation of unicast routing and a port classifier. Next is multicast node, in addition to routing

and port classification, it has a classifier which classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

In network simulation, Unicast nodes are in default condition. In order to create Multicast nodes once the user must clearly notify in the input of the OTcl script, after the exact creation of scheduler object, the nodes which are created that will be in the form of multicast nodes. Next process is specification, once it get done then specify the node type, the user can also select a exact routing protocol other than using a predifined one.

Unicast

- *$ns* rtproto *type*
- *type*: Static, Session, DV, cost, multi-path

Multicast

- *$ns* multicast (right after set *$ns* [new Scheduler])

1.2.3 Link

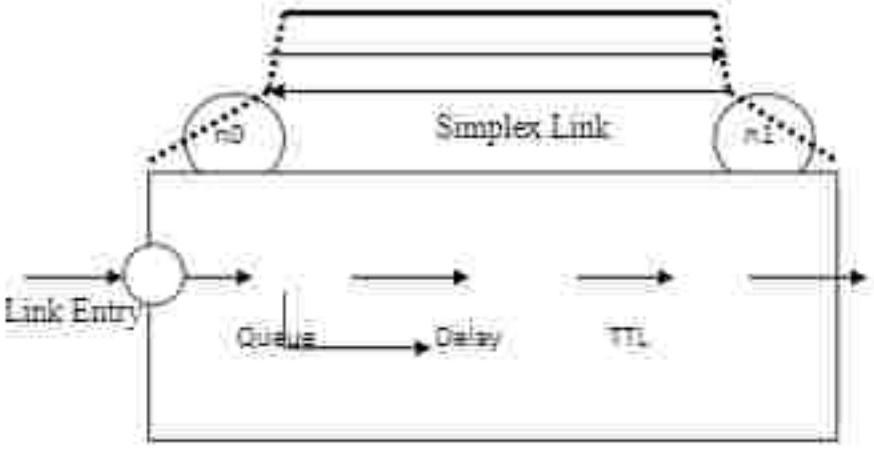


Figure Link

First notify that a node's output queue is actually implemented in the form of simplex link object. Once after completing the process of dequeued packets from a queue are passed to the Delay object that again simulates the link delay, and the dropped packets at a queue are transferred to a Null Agent and are made freed there. Finally, the TTL object calculates Time To Live (TTL) parameters for each received packets and updates.

1.2.4 Tracing

In NS2, activities of the network are traced around simplex links. If the simulator is intended for to the trace network activities (specifically make use of $ns trace-all *file* or $ns namtrace-all *file*), the links created after this commands will be followed by the upcoming trace objects inserted as shown in Figure (Inserting Trace Objects). Users creates a trace object of type *type* between the given *source* and *destination* nodes using the create-trace (*type file src dst*) command.

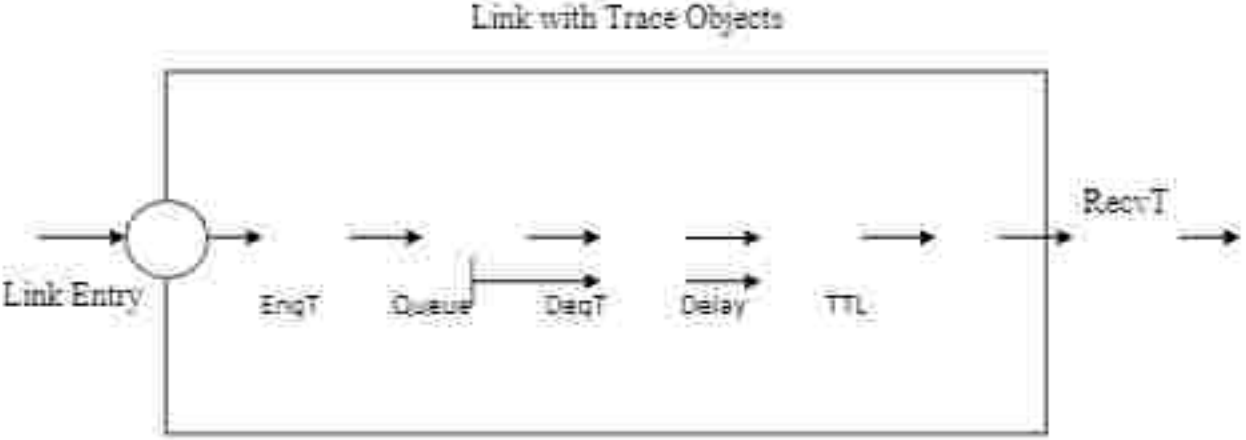


Figure Inserting Trace Objects

1.2.5 Queue Monitor

On the whole, tracing objects are intended to trace packet arrival time at which the localization is done. Even though a user gets sufficient information from the trace files, he or she might be concerned with what is going inside the output queue. For example, a user paying attention in RED queue behavior may want to calculate the dynamics of average and current queue size of a exact RED queue (i.e. need for queue monitoring). Queue monitoring can be successfully achieved using queue monitor objects and snoop queue objects as shown in Figure (Monitoring Queues).

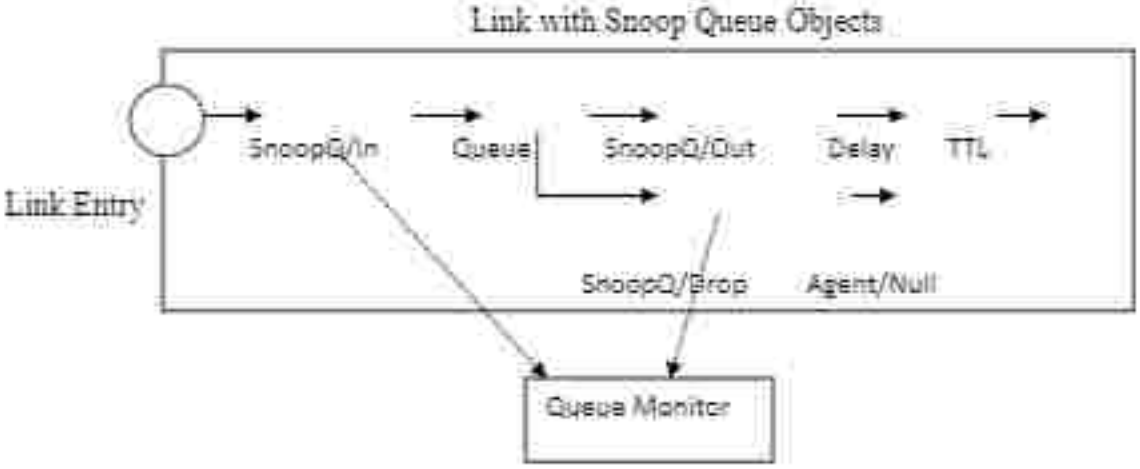


Figure Monitoring Queues

When a data packet arrives, the queue monitor object is notified by the snoop queue object of this event. Using this information the queue is monitored by the queue monitor. RED Queue Monitor Example section shows some examples for RED queue monitoring. Note that snoop queue objects can be second-hand in parallel with tracing objects although it is not shown in the above figure (Figure Monitoring Queues).

1.2.6 Packet Flow Example

Until now, the examination of two most important network components (node and link) is done. Figure (Packet Flow Examples) shows internals of an example simulation network setup and packet flow. The number of nodes network is two which is node 0 (n0) and node 1 (n1) of which the network addresses are 0 and 1 respectively. A TCP agent (sender agent) attached to n0 using port 0 communicates with a TCP sink object (destination agent) attached to n1 port 0. Finally, an FTP application layer (or traffic source) is attached to the TCP agent (sender application), asking to send some amount of data to the destination which is node 1.

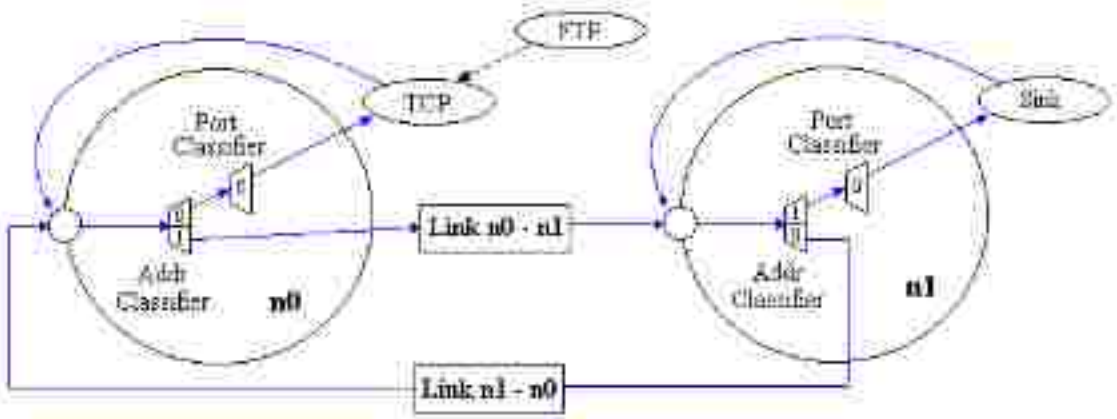


Figure Packet Flow Examples

1.3 Overview of ns-2 simulation test bed

NS-2 has many and expanding uses included.

- The performance of existing network protocols is evaluated.
- Before the usage the new network protocols are evaluated.
- To run large scale experiments and it is not possible in the real time environment.
- Various kinds of internet protocols IP are possible to simulate in network simulation 2.

NS-2 is an object oriented discrete event simulator which works to calculate the performance and behavior of the network. Simulator maintains list of events in the queue and executes one event after another event.

Features:

- Protocols mostly used
- Fast to run, with high network control
- Front end is OTCL – Object tool command language
- BACK end is C++ - Creating scenarios, extensions to C++ protocols
- fast to create and modify

1.3.1 Characteristics of NS-2

- NS-2 implementation consist of the following features
- Multicasting is employed here.

- Simulation of various kinds of wireless networks
- Terrestrial (cellular, Adhoc, GPRS, WLAN, BLUETOOTH), satellite network are used
- IEEE 802.11 standard can be simulated, Mobile Internet Protocols and Ad hoc protocols such as DSR, TORA, DSDV and AODV Routing are simulated

1.3.2 Software Tools used with NS-2

In the simulation, there are the two tools are used.

- **NAM(Network Animator)**
- **xGraph**

1.3.3 NS ARCHITECTURE

- Object-oriented (C++, OTCL)
- Modular approach
- Fine –grained object composition
- Reusability
- Maintenance
- Perfomanace(speed and memory)
- Careful planning of modularity

1.3.4 NS PROGRAMMING

- Create the event scheduler
- Turn on tracing
- Create network
- Setup routing
- Insert errors
- Create transport connection
- Create traffic
- Transmit application-level data

1.3.5 TCL INTERPRETER

TclCL is the language used to provide a linkage between C++ and OTcl. Toolkit Command Language(Tcl OTcl) scripts are written to set up configure network topologies. TclCL provides linkage for class hierarchy,object instantiation, variable binding and command dispatching. OTcl is used for periodic or triggered events

The following is written and compiled with C++

1. Events Scheduler
2. NAM- The Network Animator
3. Xgraph- For plotting
4. Pre Processing- Traffic & Topology generator
5. Post Processing- Simple Trace Analysis often used TCL and Pearl

1.3.6 CHARACTERISTICS

NS-2 implements the following features

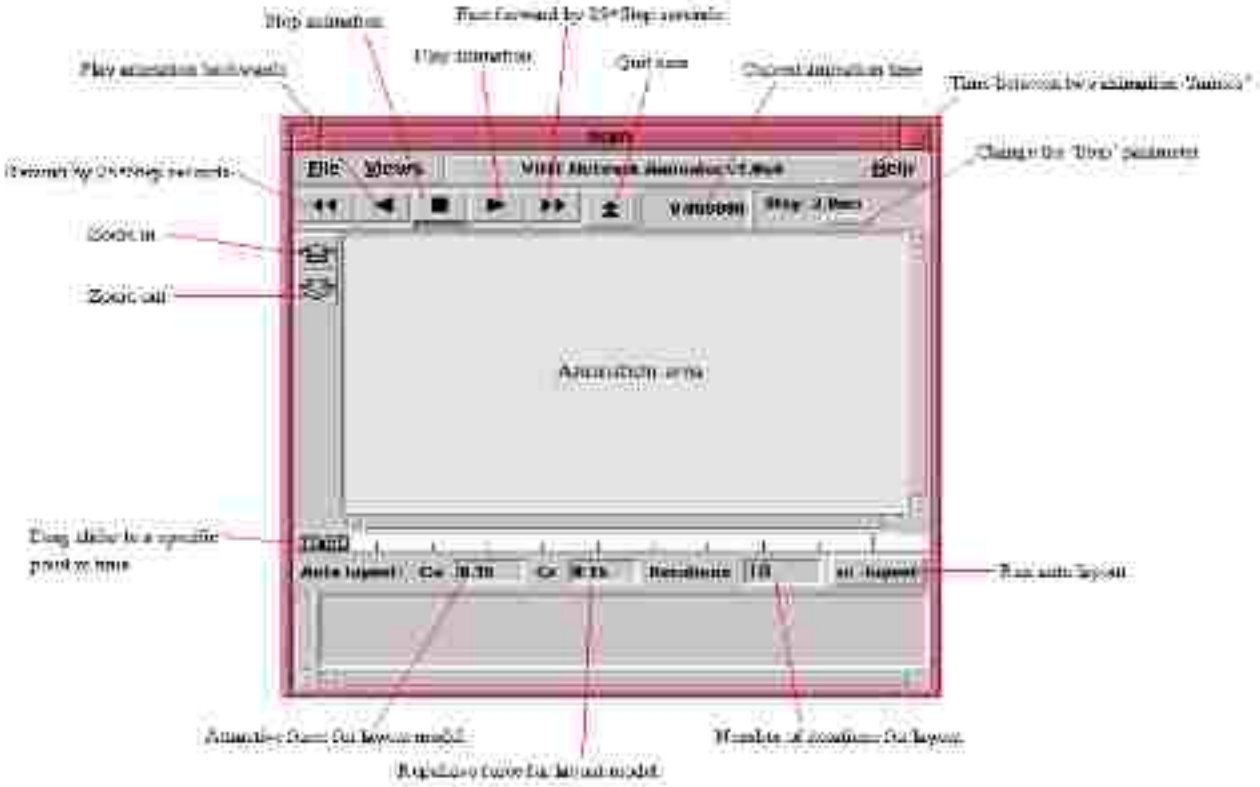
1. Router queue Management Techniques Drop Tail, RED, CBQ,
2. Multicasting
3. Simulation of wireless networks
4. Developed by Sun Microsystem + UC Berkeley (Daedalus project)
5. Terrestrial (Cellular, Ad-hoc, GPRS, WLAN, BLUETOOTH), Satellite

1.3.7 NAM (Network Animator)

NAM provides a visual interpretation of the network topology created. The application was developed as part of the VINT project. Its feature is as follows.

- Provides a visual interpretation of the network created
- Can be executed directly from a Tcl script

- Controls include play, stop fast forward, rewind, pause, a display speed controller button and a packet monitor facility.
- Presented information such as throughput, number packets on each link



1.3.8 X Graph

X- Graph is an X-Window application that includes:

Interactive plotting and graphing Animated and derivatives, to use Graph in NS-2 the executable can be called within a TCL script. This will then load a graph displaying the information visually displaying the information of the file produced from the simulation. The output is a graph of size 800 x 400 displaying information on the traffic flow and time.

1.3.9 Simulation tool

NS2 are often growing to include new protocols. LANs need to be updated for new wired/wireless support. ns are an object oriented simulator, written in C++, with an OTcl interpreter as a front-end. The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy). The two hierarchies

are closely related to each other; from the user's perspective, there is a one-to-one correspondence between classes in the interpreted.

2 Basic Linux and N22

2.1 Linux Commands

cd : change directory

- Syntax: cd directoryname

ls : list the files in current directory

- Syntax: ls

rm : Remove a file from directory

- Syntax: rm filename

cp : Copying file from one directory to another

- Syntax: cp filename directoryname

pwd. For checking current directory

- Syntax: pwd

ps: For viewing currently running processes on system

- Syntax: ps

kill: For killing a process

- Syntax: kill processid

cat: For viewing file contents on terminal

- Syntax: cat filename

clear: clear the contents on terminal

- Syntax: clear

gcc: For compliling c and c++ programs.

- Syntax: gcc programname.c

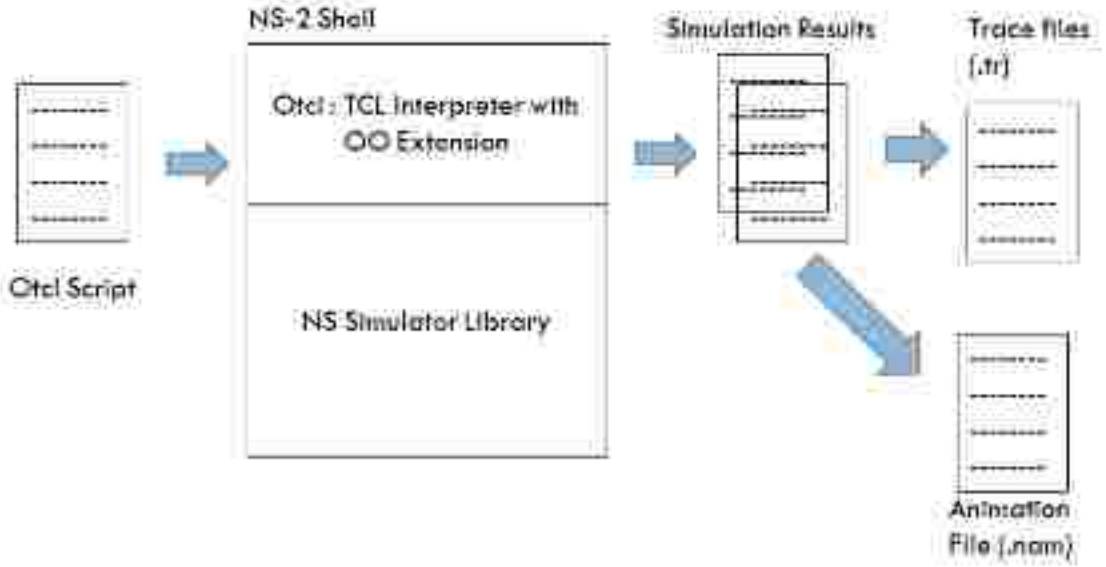
gedit: Create and open the file in text editor.

- Syntax: gedit filename

./ : For running object file.

- Syntax: ./ objectfilename

2.2 Simulation System Architecture



2.3 Installation of NS-2

2.3.1 Installation on Linux

Copy ns-allinone-2.34.tar_1.gz into /usr/local folder

Extract ns-allinone-2.34.tar_1.gz, you will get ns-allinone-2.34.tar_1.

Extract ns-allinone-2.34.tar_1, you will get ns-allinone-2.34 folder.

Go to ns-allinone-2.34 folder and say – (./install).

Go to ns-2.34 folder,

Do (./configure)

Do make all.

Do make install.

2.3.2 Bash file setting (option 2)

Open Terminal

Type on terminal following command

gedit ~/.bashrc

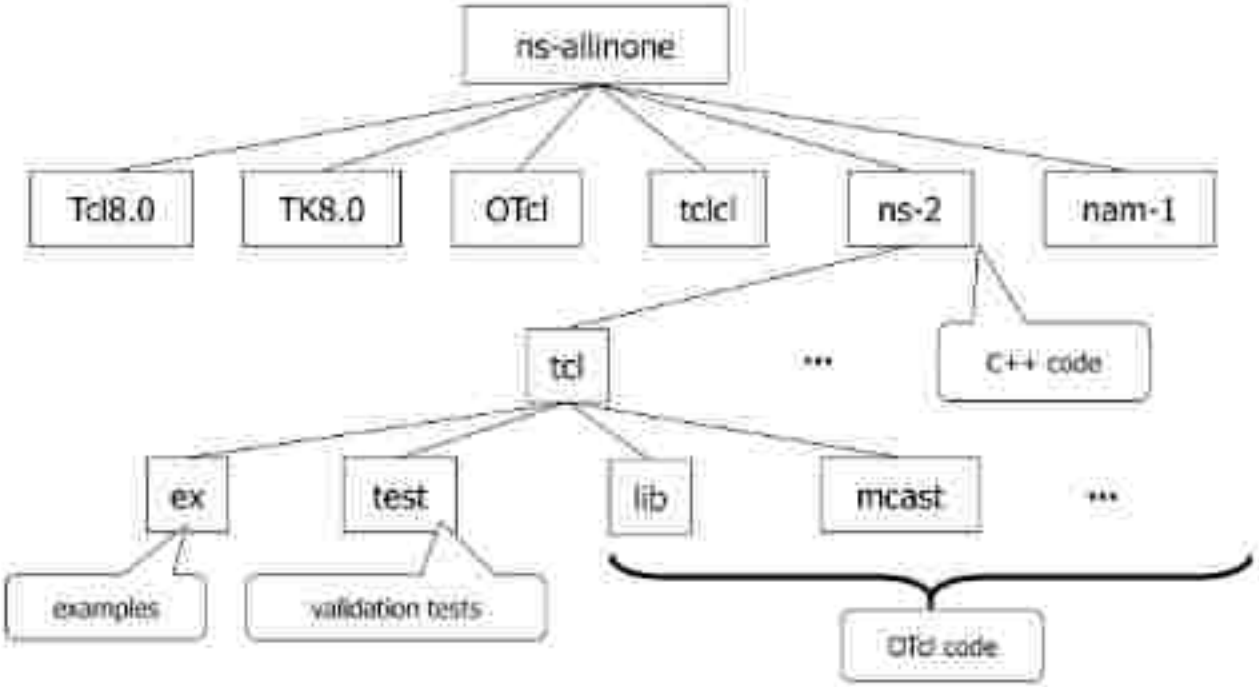
Add the TCL library, LD library and ns library path in .bashrc file.

Save the changes

Type on terminal following command

source ~/.bashrc

2.4 NS-2 Directory Structure



3 Scenarios

3.1 First Simulation Scenario

Simulation Script

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Open the event trace file
set tf1 [open out.tr w]
$ns trace-all $tf1
#Define a 'finish' procedure
proc finish { }
{
    global ns nf
    $ns flush-trace
        #Close the nam trace file
    close $nf
#Close the event trace file
    close $tf1
  #Execute nam
    exec nam out.nam &
    exit 0
}
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
[Create a UDP agent
set udp0 [new Agent/UDP]
#Attach udp agent to node n0
$ns attach-agent $n0 $udp0
# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
#Create a Null agent (a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
#Connect the traffic source with the traffic sink
$ns connect $udp0 $null0
#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

Save the simulation script in specific folder.

Open the terminal and go up to specific folder.

Run the simulation script,

ns: command to run simulation script.

Syntax: ns filename.tcl

e.g. ns First_script_wired.tcl

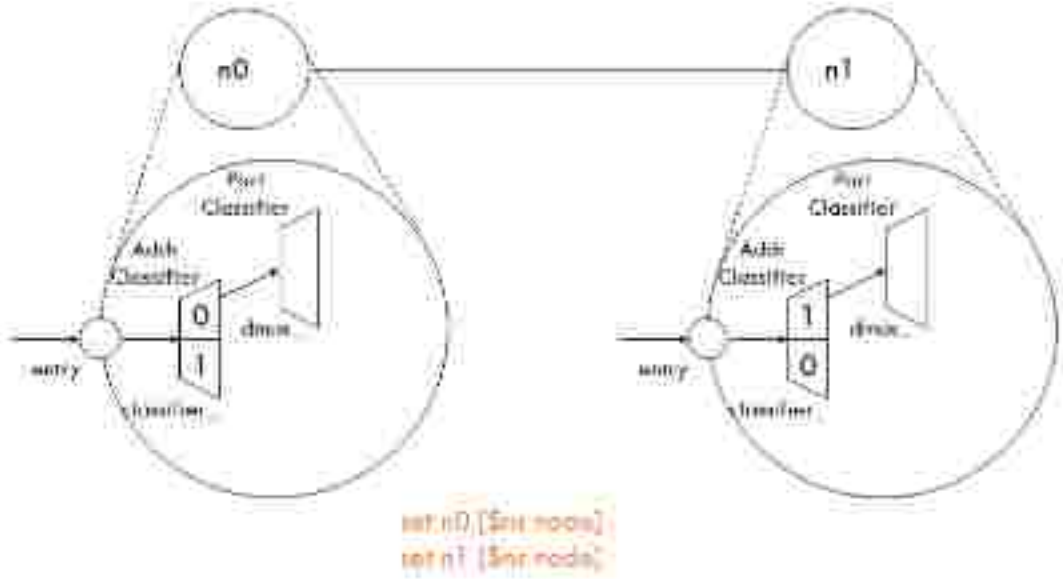
Run the nam file,

nam: command to run animation file.

Syntax: nam filename.nam

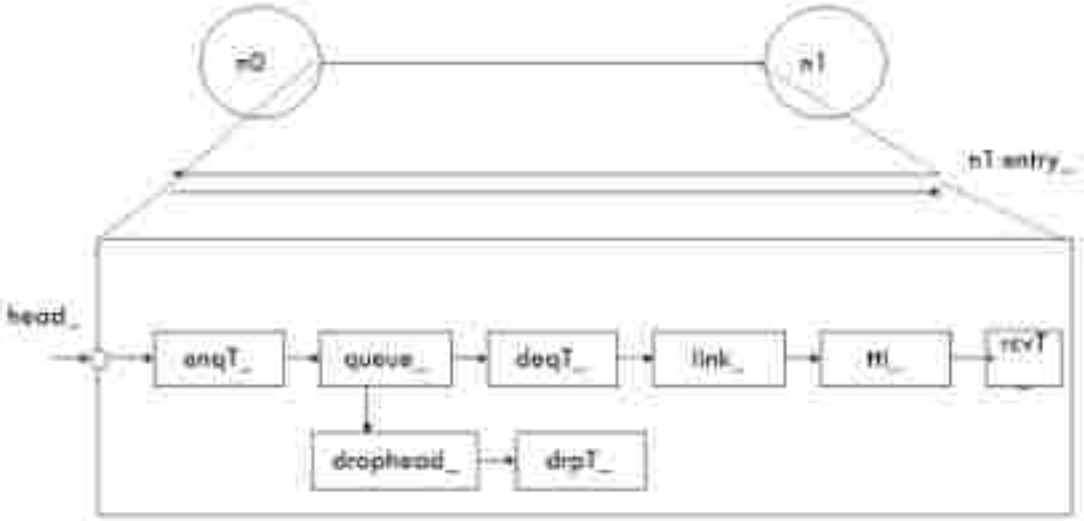
e.g. nam s1.nam

3.1.1 Flow of Simulation (NS-Node)



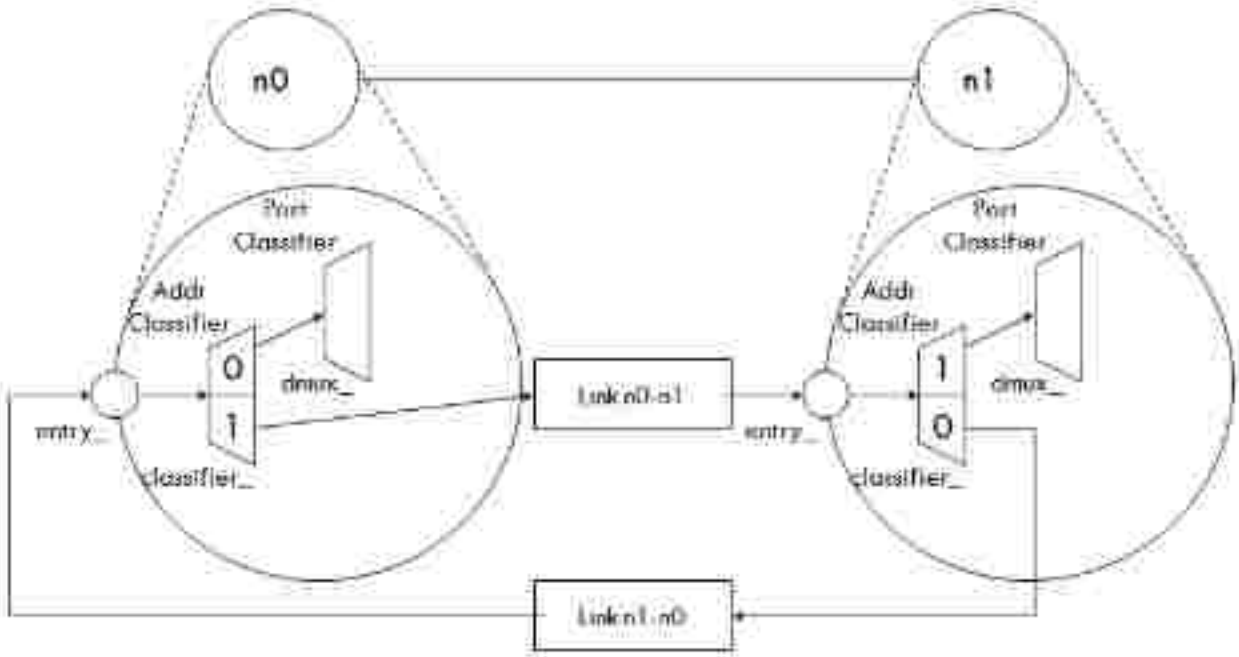
set n0 [$ns node]
set n1 [$ns node]

3.1.2 Flow of Simulation (Network Topology – Link)

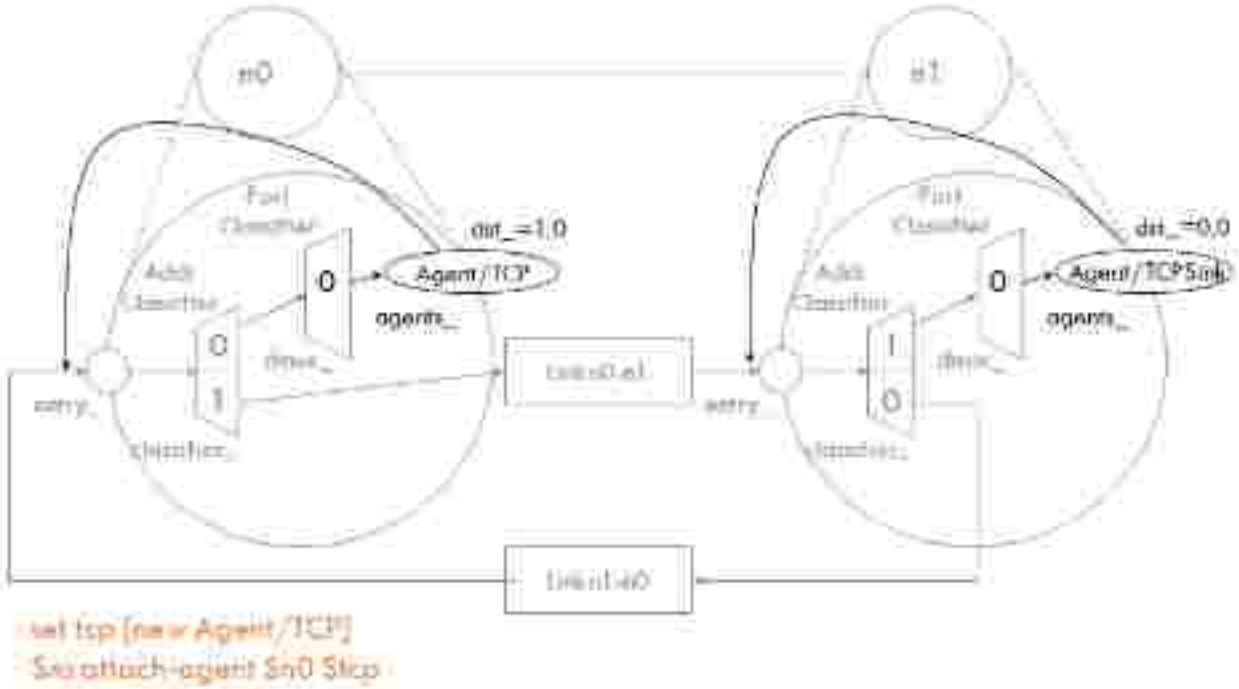


$ns duplex-link $n0 $n1 1Mb 10ms DropTail

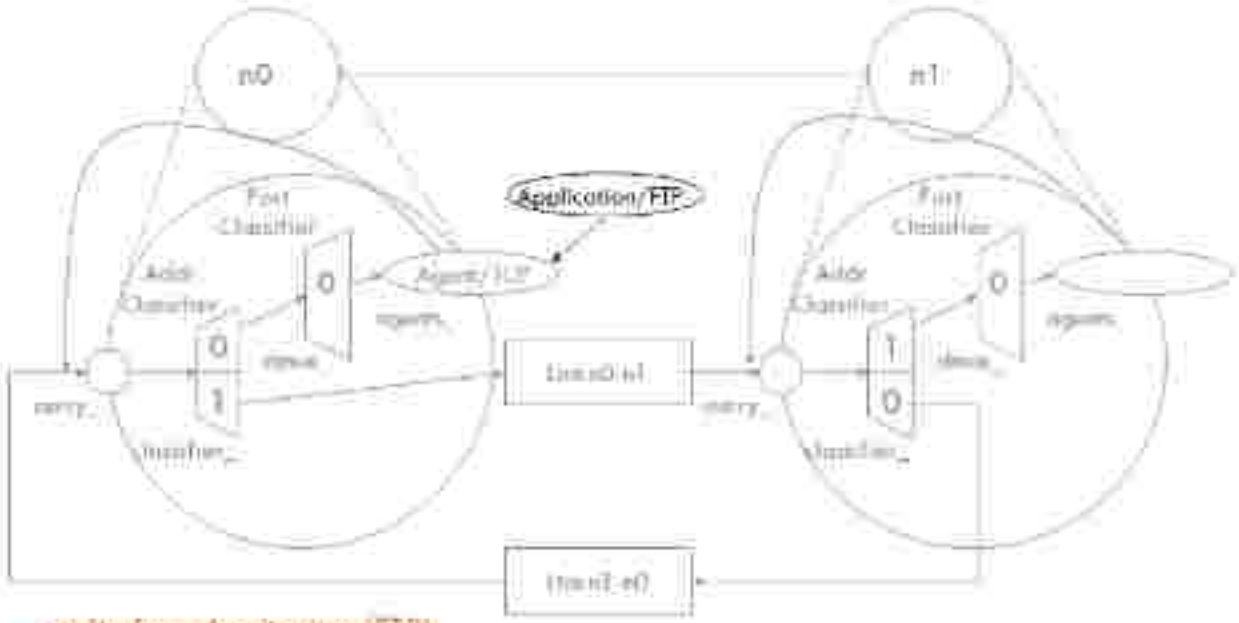
3.1.3 Flow of Simulation (Routing)



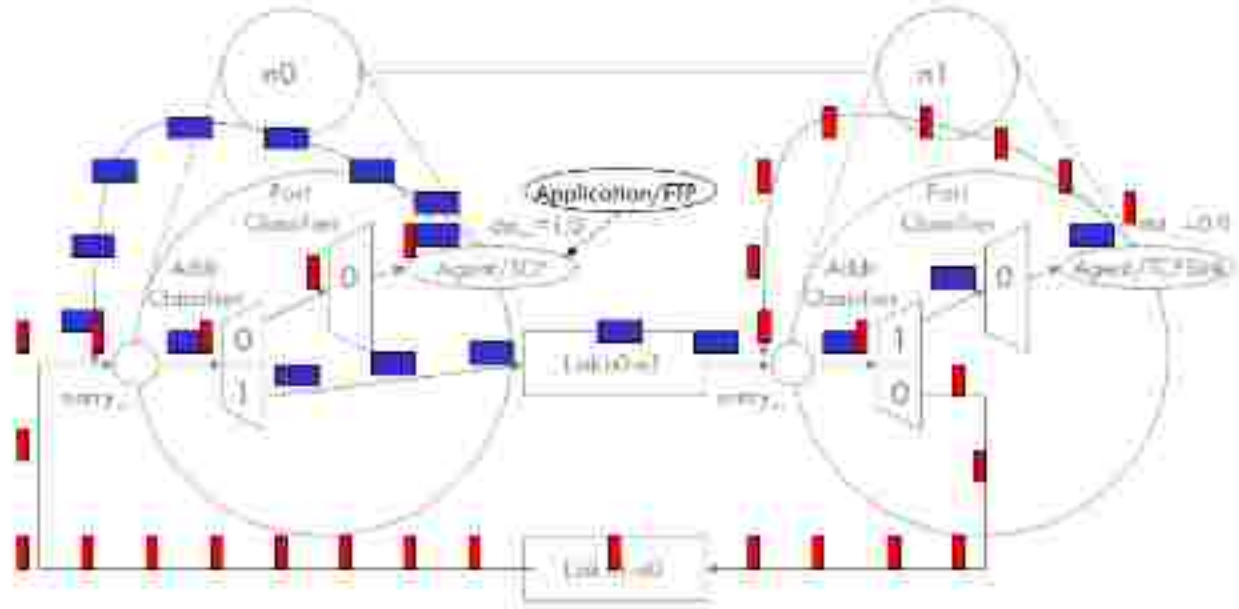
3.1.4 Flow of Simulation (Transport)



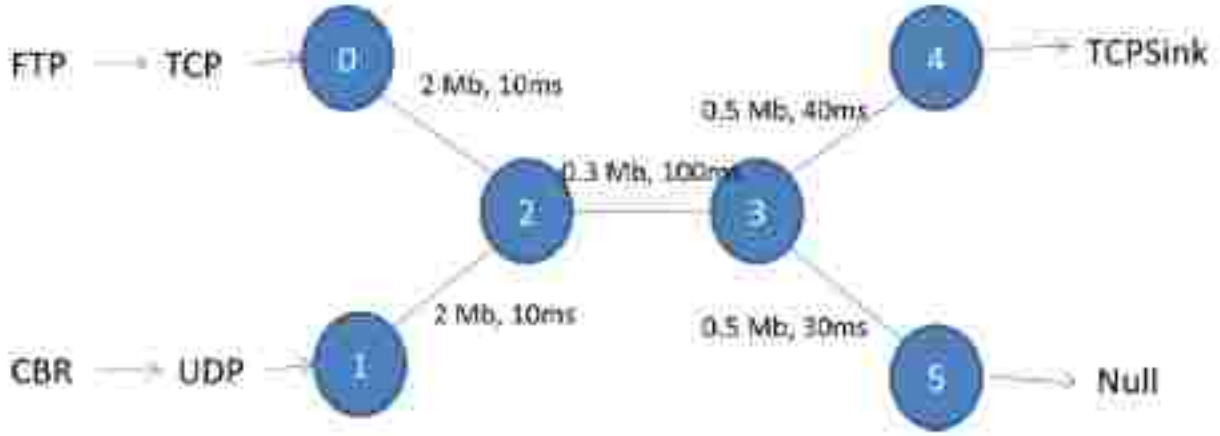
3.1.5 Flow of Simulation (Application)



3.1.6 Flow of Simulation (Packet Flow)



3.2 Second Simulation Scenario



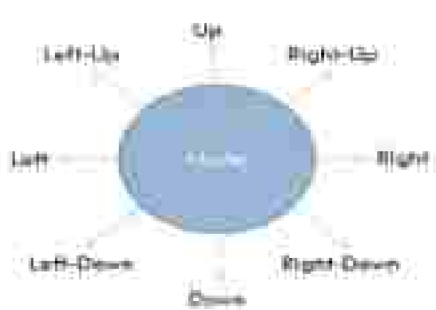
3.2.1 Simulation Script 2

```
#Create Simulator Object
    set ns [new Simulator]
#Define different colors for data flows (for NAM)
    $ns color 1 Blue
    $ns color 2 Red
#Open the Event trace file
    set file1 [open out.tr w]
    $ns trace-all $file1
#Open the NAM trace file
    set file2 [open out.nam w]
    $ns namtrace-all $file2
#Define a 'finish' procedure
    proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam out.nam &
        exit 0 }
#Create six nodes
    set n0 [$ns node]
    set n1 [$ns node]
    set n2 [$ns node]
    set n3 [$ns node]
    set n4 [$ns node]
```

```
    set n1 [$ns node]
#Label to the node n1 and node n2
$ns at 0.1 "$n1 label "CBR""
$ns at 1.0 "$n1 label "FTP""
#Create links between the nodes
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
$ns duplex-link $n2 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail
#Give node position (for NAM)
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
#Set Queue Size of link (n2-n3) to 40
$ns queue-limit $n2 $n3 40
#Setup a TCP connection
    set tcp [new Agent/TCP]
    $ns attach-agent $n3 $tcp
    set sink [new Agent/TCPSink]
    $ns attach-agent $n4 $sink
```

```
        $ns connect $tcp $sink
        $tcp set fid_ 1
        $tcp set window_ 8000
        $tcp set packetSize_ 552
#Setup a FTP over TCP connection
        set ftp [new Application/FTP]
        $ftp attach-agent $tcp
        $ftp set type_ FTP
#Setup a UDP connection
        set udp [new Agent/UDP]
        $ns attach-agent $n1 $udp
        set null [new Agent/Null]
        $ns attach-agent $n5 $null
        $ns connect $udp $null
        $udp set fid_ 2
#Setup a CBR over UDP connection
        set cbr [new Application/Traffic/CBR]
        $cbr attach-agent $udp
        $cbr set type_ CBR
        $cbr set packet_size_ 1000
        $cbr set rate_ 0.01mb
        $cbr set random_ false
# Scheduling the event
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"


$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"
# Trace Congestion Window and RTT
set file [open cwnd_rtt.tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_
# Call finish procedure
$ns at 625.0 "finish"
# Run the simulation
$ns run
```

3.3 Node Orientation



3.4 Node Commands

$ns node [<nier_addr_>]

$ns node-config -<config-parameter> <optional-val>

$node id

$node node-addr

$node reset

$node agent <port_num>

$node entry

$node attach <agent>

$node detach <agent>

$node neighbors

$node add-neighbor <neighbor_node>

$node add-route <destination_id> <target>

$node alloc-port <null_agent>

$node incr-rgtable-size

More Node Commands

Check ~ns-2.34/tcl/lib/ns-node.tcl and ~tcl/lib/ns-mobilenode.tcl

3.5 Link Commands

$ns simplex-link <node1> <node2> <bw> <delay> <qtype> <args>

$ns duplex-link <node1> <node2> <bw> <delay> <qtype> args>

$ns simplex-link-op <n1> <n2> <op> <args>

$ns duplex-link-op <n1> <n2> <p> <args>

$ns lossmodel <lossobj> <from> <to>

$link head

$link link

$link add-to-head <connector>

$link queue

$link cost <c>

$link cost?

$link if-lable?

$link up

$link down

$link up?

$link all-connectors <op>

More Node Commands

Check ~ns-2.34/tcl/lib (ns-lib.tcl,ns-link.tcl, ns-intserv.tcl, ns-namsupp.tcl, ns-queue.tcl) and ~tcl/mcast (Mcast/monitor.tcl, ns-mcast.tcl), ~ns-2.34/tcl/session/session.tcl

3.6 Simulator Commands

set ns [new Simulator]

set now [$ns now]

$ns halt

$ns run

$ns at <time> <event>

$ns cancel <event>

$ns flush-trace

$ns use - scheduler <type>

$ns after <delay> <event>

$ns clearMemTrace

$ns is-started

$ns dumpq

More functions

Check ~ns-2.34/tcl/lib/ns-lib.tcl, ~ns-2.34/common/scheduler.{cc,h} and ~ns-2.34/heap.h

3.7 Trace Related Commands

$ns trace-all <trace-file>

$ns namtrace-all <namtracefile>

$ns namtrace-all-wireless <namtracefile> <X> <Y>

$ns nam-end-wireless <stoptime>

$ns flush-trace

$ns create-trace <type> <file> <src> <dst> <optional:op>

$ns trace-queue <n1> <n2> <optional : file>

$ns namtrace-queue <n1> <n2> <optional : file>

$ns drop-trace <n1> <n2> <trace>

$ns monitor-queue <n1> <n2> <qtrace> <optional : sampleinterval>

$link trace-dynamics <ns> <fileID>

More Functions,

Check ~ns-2.34/trace.{cc,h}, ~ns-2.34/tcl/lib/ns-trace.tcl, ~ns/queue-monitor.{cc,h}, ~ns-2.34/tcl/ns-link.tcl, ~ns-2.34/packet.h, ~ns-2.34/flowmon.cc and ~ns-2.34/classifier-hash.cc

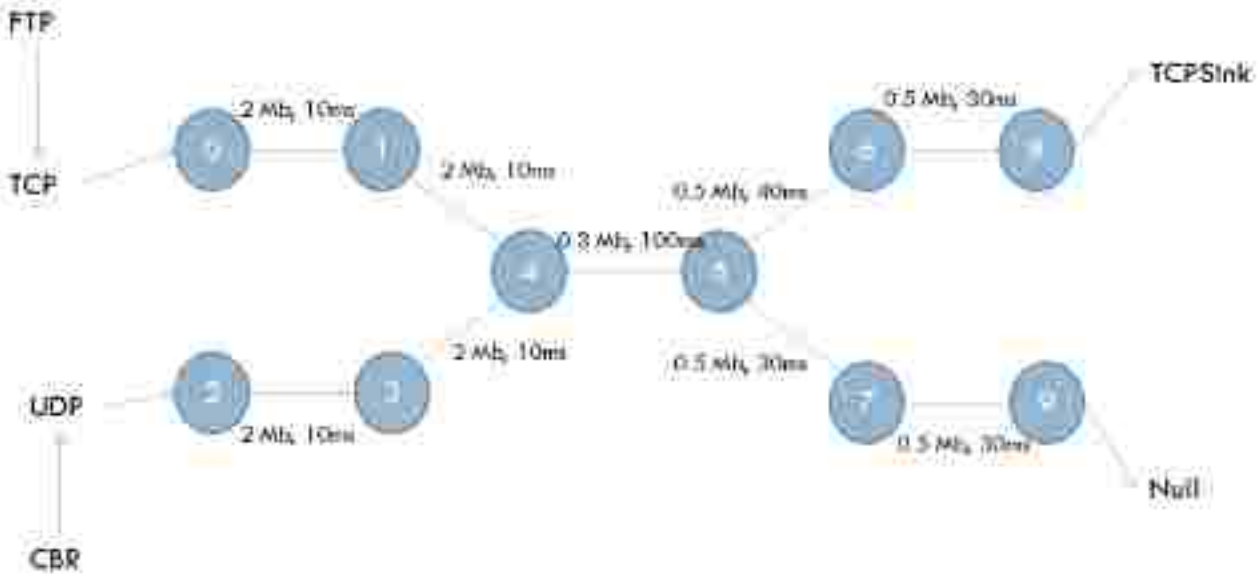
3.8 NAM Commands

$ns color <color-id>

$ns trace-annotate <annotation>

$ns set-animation-rate <timestep>

3.9 Third Simulation Scenario



3.9.1 Simulation Script 3

 # Create Simulator Object

```
set ns [new Simulator]
```

#Define different colours for data flows (for NAM)

```
$ns color 1 Blue
$ns color 2 Red
```

#Open the Event trace files

```
set file1 [open out.tr w]
$ns trace-all $file1
```

#Open the NAM trace file

```
set file2 [open out.nam w]
$ns namtrace-all $file2
```

#Define a 'finish' procedure

```
proc finish {} {
        global ns file1 file2
         $ns flush-trace
        close $file1
        close $file2
        exec nam out.nam &
        exit 0 }
```

#Create ten nodes

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
```

```
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
#Label to the node n1 and node n2
$ns at 0.1 "$n1 label "CBR""
$ns at 1.0 "$n0 label "FTP""
#Create links between the nodes
$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 2Mb 10ms DropTail
$ns duplex-link $n3 $n4 2Mb 10ms DropTail
$ns simplex-link $n4 $n5 0.3Mb 100ms DropTail
$ns simplex-link $n5 $n4 0.3Mb 100ms DropTail
$ns duplex-link $n5 $n6 0.5Mb 40ms DropTail
$ns duplex-link $n6 $n8 0.5Mb 40ms DropTail
$ns duplex-link $n5 $n7 0.5Mb 30ms DropTail
$ns duplex-link $n7 $n9 0.5Mb 30ms DropTail
#Set Queue Size of link (n4-n5) to 10
$ns queue-limit $n4 $n5 10
```

```
#Setup a TCP connection
    set tcp [new Agent/TCP]
    $ns attach-agent $n0 $tcp
    set snk [new Agent/TCPSink]
    $ns attach-agent $n1 $snk
    $ns connect $tcp $snk
    $tcp set fid_ 1
    $tcp set window_ 8000
    $tcp set packetSize_ 552
#Setup a FTP over TCP connection
    set ftp [new Application/FTP]
    $ftp attach-agent $tcp
    $ftp set type_ FTP
#Setup a UDP connection
    set udp [new Agent/UDP]
    $ns attach-agent $n2 $udp
    set null [new Agent/Null]
    $ns attach-agent $n9 $null
    $ns connect $udp $null
    $udp set fid_ 2
#Setup a CBR over UDP connection
    set cbr [new Application/Traffic/CBR]
    $cbr attach-agent $udp
```

```
    $cbr set type_ CBR
    $cbr set packet_size_ 1000
    $cbr set rate_ 0.01mb
    $cbr set random_ false
#Scheduling the event
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 624.0 "$ftp stop"
$ns at 624.5 "$cbr stop"
#Trace Congestion Window and RTT
set file [open cwnd_rtt.tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_
#Call finish procedure
    $ns at 625.0 "finish"
#Run the simulation
    $ns run
```

3.10 Wired file format

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop    (at queue)
```

```
src_addr : node.port (3.0)
dst_addr : node.port (0.0)
```

Event	Time	From-Node	To-Node	Pkt-Type	Pkt-Size	Flags	Fid	Src-addr	Dest-addr	Seq-num	Pkt-id
-	1.06	0	2	tcp	1040	-------	1	0.0	3.0	2	124
r	1.07	1	2	cbr	1000	-------	2	1.0	3.1	120	122
+	1.07	2	3	cbr	1000	-------	2	1.0	3.1	120	122
d	1.07	2	3	cbr	1000	-------	2	1.0	3.1	120	122

4 WIRELESS NETWORK PROGRAMS

4.1 SIMULATION PROGRAM FOR LAN NETWORK

```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam out.nam &
        exit 0
}

#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n1 color red
$n1 shape box
```

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right
$ns simplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

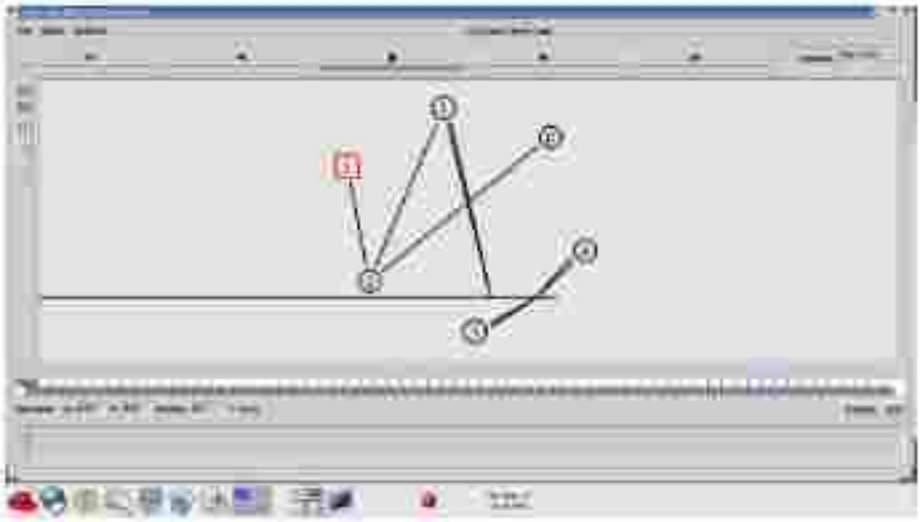
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"

# next procedure gets two arguments: the name of the
# tcp source node. will be called here "tcp",
# and the name o foutput file.

proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]


set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"


$ns at 5 "$ns trace-annotate \"packet drop\""


# PPP

$ns at 125.0 "finish"
$ns run
```

OUTPUT

NETWORK FORMATION



DATA TRANSMISSION

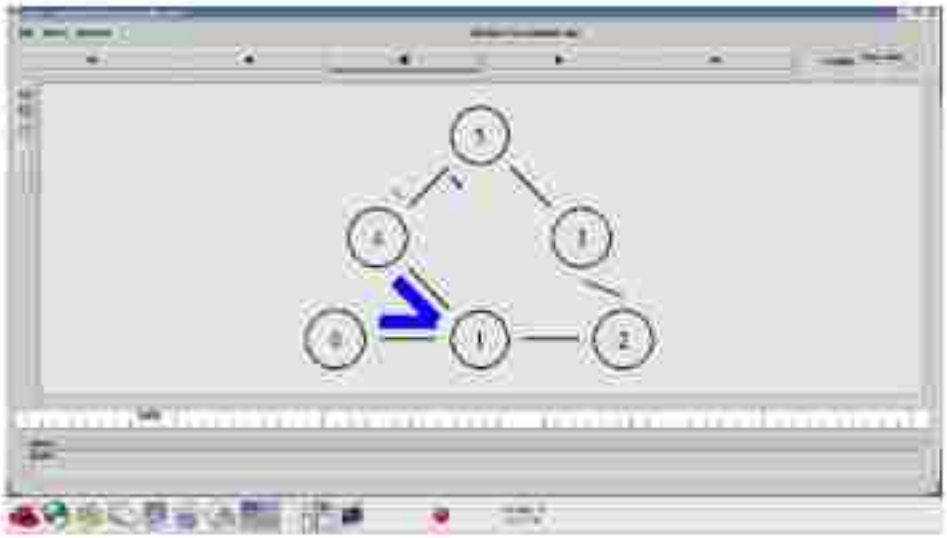
4.2 UNICAST PROGRAM

```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the Trace file
set file1 [open unicastDV.tr w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open unicastDV.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam unicastDV.nam &
        exit 0
}

# Next line should be commented out to have the static routing
$ns rtproto DV

#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
```

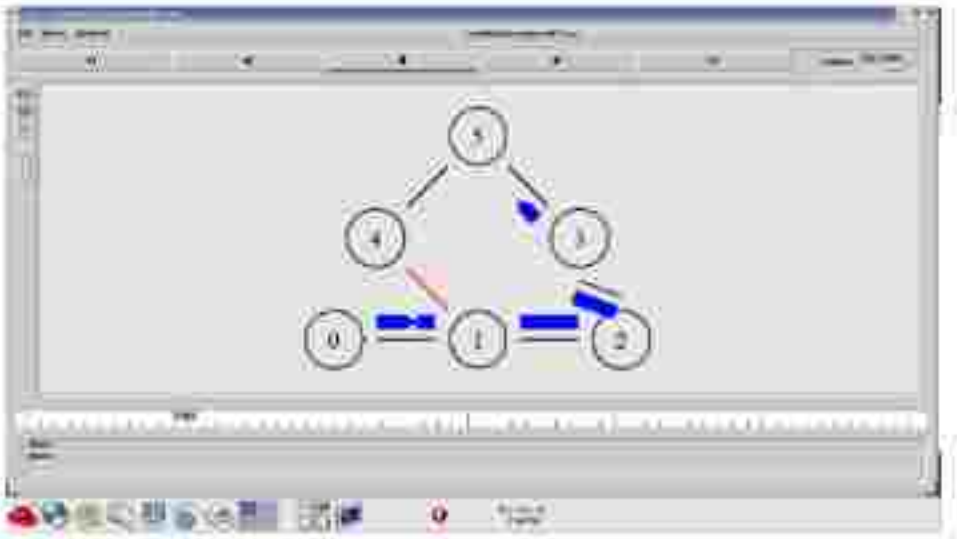
```
#Create links between the nodes
$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n2 0.3Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail
$ns duplex-link $n3 $n5 0.3Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient up
$ns duplex-link-op $n1 $n4 orient up-left
$ns duplex-link-op $n3 $n5 orient left-up
$ns duplex-link-op $n4 $n5 orient right-up


#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

$ns rtmodel-at 1.0 down $n1 $n4
$ns rtmodel-at 4.5 up $n1 $n4
$ns at 0.1 "$ftp start"
$ns at 6.0 "finish"
$ns run
```

OUTPUT

DATA TRANSFER



PATH CHANGE DUE TO LINK FAILURE

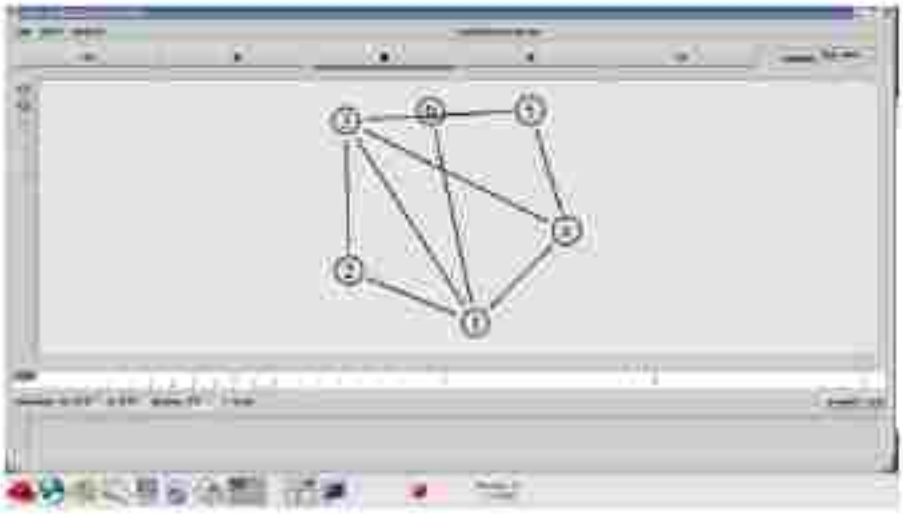


4.3 MULTICAST PROGRAM 1

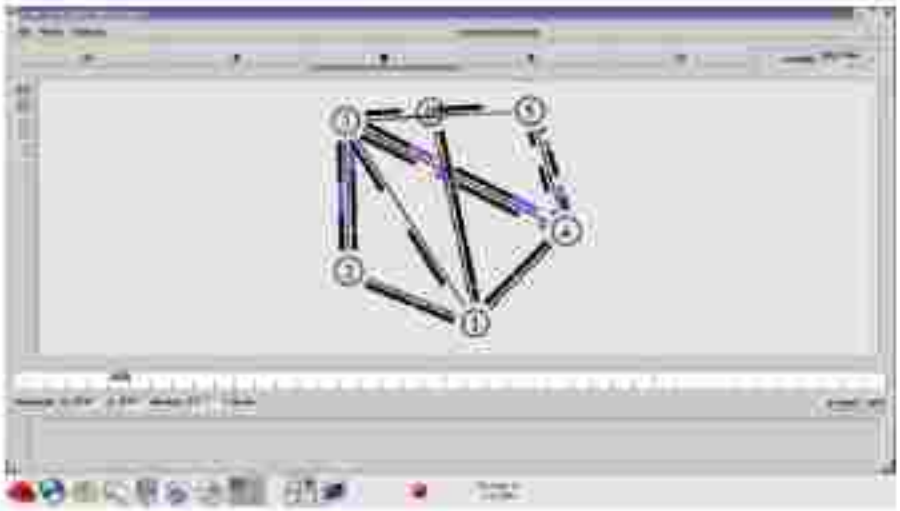
```
set ns [new Simulator]
$ns multicast

set f [open out.tr w]
$ns trace-all $f
$ns namtrace-all [open out.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# allocate a multicast address;
set group [Node allocaddr]

# nod is the number of nodes
set nod 6

# create multicast capable nodes;
for {set i 1} {$i <= $nod} {incr i} {
  set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 0.5Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 0.3Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 0.3Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 0.5Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 0.5Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 0.5Mb 10ms DropTail
```

```
# configure multicast protocol:
set mproto DM

# all nodes will contain multicast protocol agents:
set mrthandle [$ns mrtproto $mproto]

set udp1 [new Agent/UDP]
set udp2 [new Agent/UDP]

$ns attach-agent $n(1) $udp1
$ns attach-agent $n(2) $udp2

set src1 [new Application/Traffic/CBR]
$src1 attach-agent $udp1
$udp1 set dst_addr_ $group
$udp1 set dst_port_ 0
$src1 set random_ false


set src2 [new Application/Traffic/CBR]
$src2 attach-agent $udp2
$udp2 set dst_addr_ $group
$udp2 set dst_port_ 1
$src2 set random_ false

# create receiver agents
set rcvr [new Agent/LossMonitor]

# joining and leaving the group:
$ns at 0.6 "$n(3) join-group $rcvr $group"
$ns at 1.3 "$n(4) join-group $rcvr $group"
$ns at 1.6 "$n(5) join-group $rcvr $group"
$ns at 1.9 "$n(4) leave-group $rcvr $group"
$ns at 2.3 "$n(6) join-group $rcvr $group"


$ns at 3.5 "$n(3) leave-group $rcvr $group"

$ns at 0.4 "$src1 start"
$ns at 2.0 "$src2 start"

$ns at 4.0 "finish"

proc finish {} {
        global ns
        $ns flush-trace
        exec nam out.nam &
        exit 0
}

$ns run
```

OUTPUT



DATA TRANSMISSION

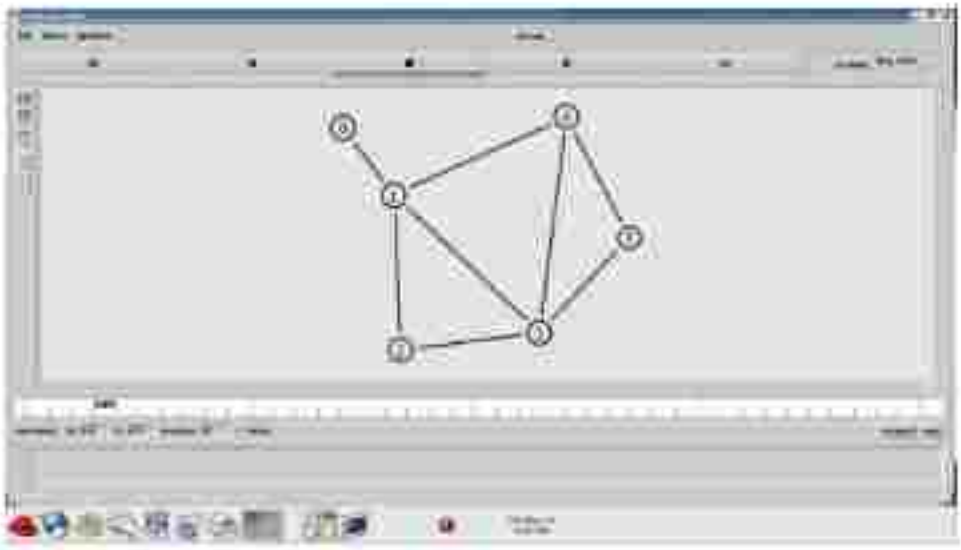


4.4 MULTICAST PROGRAM 2

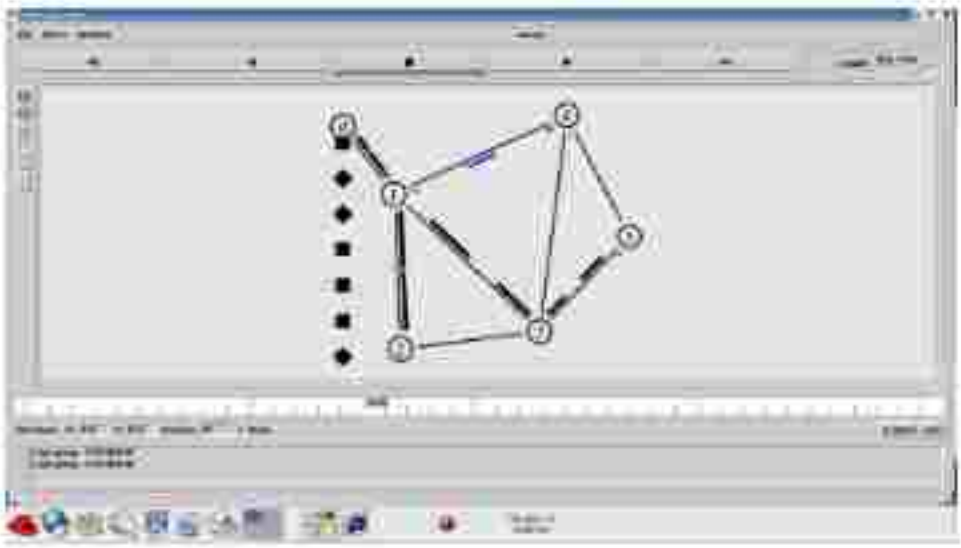
```
set ns [new Simulator]
$ns multicast

set f [open out.tr w]
$ns trace-all $f
$ns namtrace-all [open out.nam w]

$ns color 1 red
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

# allocate a multicast address.
set group [Node allocaddr]

# nod is the number of nodes
set nod 6

# create multicast capable nodes.
for {set i 1} {$i <= $nod} {incr i} {
  set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 0.5Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 0.3Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 0.3Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 0.5Mb 10ms DropTail
$ns duplex-link $n(4) $n(6) 0.5Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 0.5Mb 10ms DropTail
```

```
# configure multicast protocol;
DM set CacheMissMode dvmrp
set mproto DM

# all nodes will contain multicast protocol agents
set mrthandle [$ns mrtproto $mproto]

set udp1 [new Agent/UDP]
set udp2 [new Agent/UDP]

$ns attach-agent $n(1) $udp1
$ns attach-agent $n(2) $udp2

set src1 [new Application/Traffic/CBR]
$src1 attach-agent $udp1
$udp1 set dst_addr_ $group
$udp1 set dst_port_ 0
$src1 set random_ false


set src2 [new Application/Traffic/CBR]
$src2 attach-agent $udp2
$udp2 set dst_addr_ $group
$udp2 set dst_port_ 1
$src2 set random_ false

# create receiver agents
set rcvr [new Agent/LossMonitor]

# joining and leaving the group
$ns at 0.6 "$n(3) join-group $rcvr $group"
$ns at 1.3 "$n(4) join-group $rcvr $group"
$ns at 1.6 "$n(5) join-group $rcvr $group"
$ns at 1.9 "$n(4) leave-group $rcvr $group"

$ns at 2.3 "$n(6) join-group $rcvr $group"
$ns at 3.5 "$n(3) leave-group $rcvr $group"

$ns at 0.4 "$src1 start"
$ns at 2.0 "$src2 start"

$ns at 4.0 "finish"

proc finish {} {
        global ns
        $ns flush-trace
        exec nam out.nam &
        exit 0
}

$ns run
```

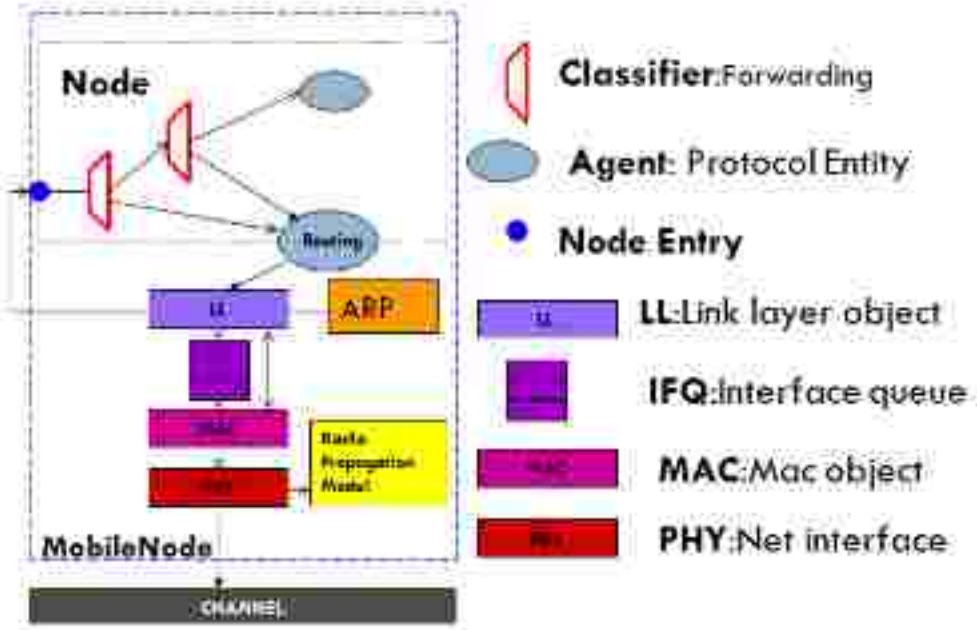
OUTPUT



DATA TRANSMISSION



4.5 Mobile/Wireless Node Structure



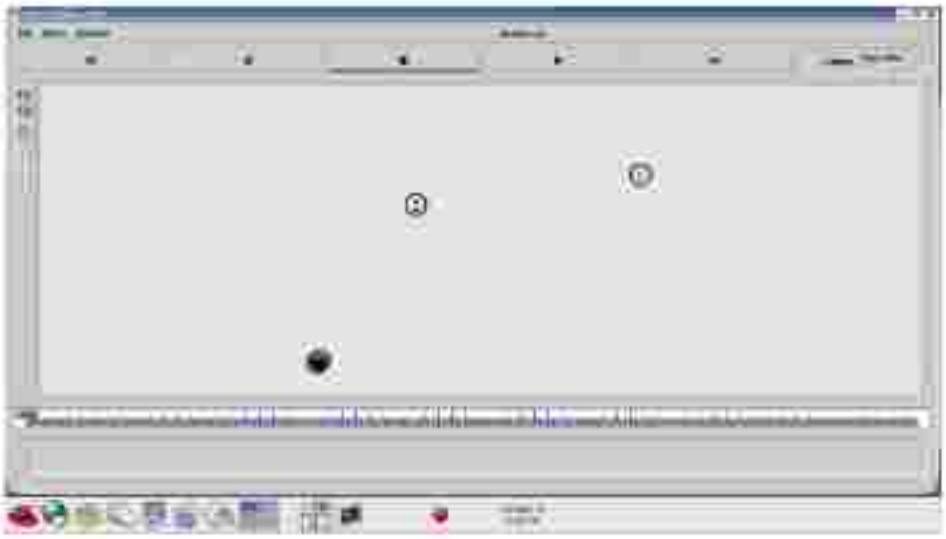
4.6 WIRELESS PROGRAM 1

```
# Define options
set val(chan)         Channel/WirelessChannel    ;# channel type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation
set val(netif)        Phy/WirelessPhy            ;# network interface
set val(mac)          Mac/802_11                 ;# MAC type
set val(ifq)          Queue/DropTail/PriQueue    ;# interface queue
set val(ll)           LL                         ;# link layer type
set val(ant)          Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)       50                         ;# max packet in ifq
set val(nn)           20                         ;# number of nodes
set val(rp)           AODV                       ;# routing protocol
set val(x)            500                        ;# X dimension
set val(y)            400                        ;# Y dimension
set val(stop)         110                        ;# simulation end

set ns              [new Simulator]
set tracefd     [open wireless.tr w]
set namtrace    [open wireless.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
```

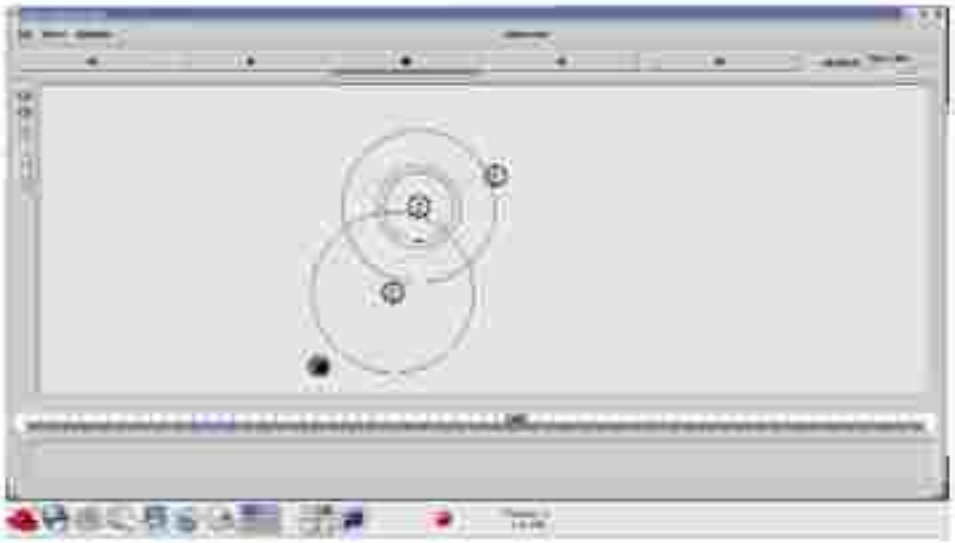
```
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace OFF \
                -routerTrace OFF \
                -macTrace ON \
                -movementTrace ON

        for {set i 0} {$i < $val(nn) } { incr i } {
                set node_($i) [$ns node]
        }

# Provide initial location of nodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 300.0 5.0"
```

```
# Set a TCP connection between node_(0) and node_(1)
set udp [new Agent/UDP]
set sink [new Agent/Null]
$ns attach-agent $node_(0) $udp
$ns attach-agent $node_(1) $sink
$ns connect $udp $sink
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set interval_ 1
$cbr set maxpkts_ 100
$ns at 10.0 "$cbr start"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 110.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
}

$ns run
```

OUTPUT



DATA TRANSFER



4.7 WSN PROGRAM – 802.11

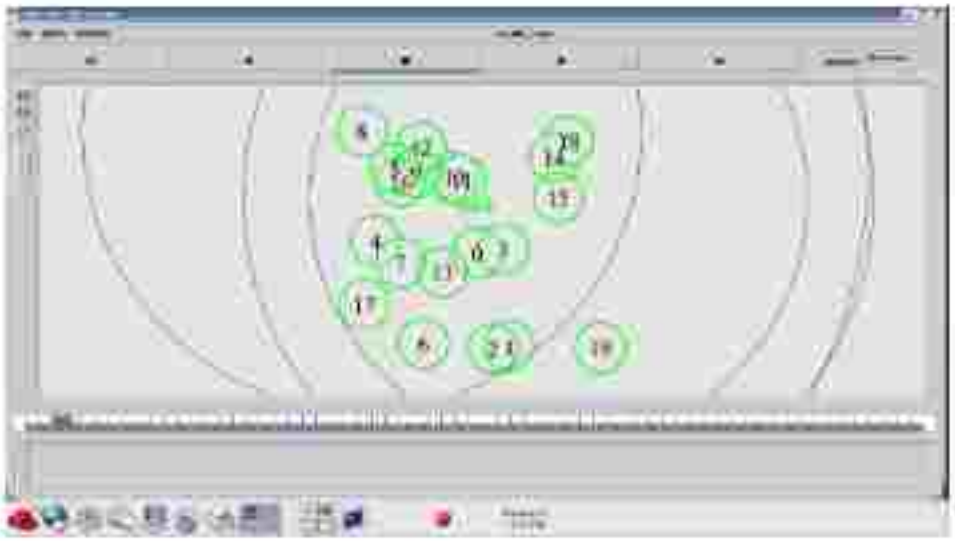
```
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
set val(netif)          Phy/WirelessPhy            ;# network interface
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         100                        ;# max packet in ifq
set val(nn)             20                         ;# number of nodes
set val(rp)             AODV                       ;# protocol type
set val(x)              50                         ;# X dimension
set val(y)              50                         ;# Y dimension
set val(stop)           500                        ;# simulation period
set val(energymodel)    EnergyModel                ;# Energy Model
set val(initialenergy)  100                        ;# value

set ns              [new Simulator]
set tracefd         [open sim_802_11.tr w]
set namtrace        [open sim_802_11.nam w]
$ns use-newtrace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
# configure the nodes
$ns node-config -adhocRouting $val(rp) \
           -llType $val(ll) \
           -macType $val(mac) \
           -ifqType $val(ifq) \
           -ifqLen $val(ifqlen) \
           -antType $val(ant) \
           -propType $val(prop) \
           -phyType $val(netif) \
```

```
        -channel [new $val(chan)]
        -topoInstance $topo
        -agentTrace OFF
        -routerTrace OFF
        -macTrace ON
        -movementTrace OFF
        -energyModel $val(energymodel)
        -initialEnergy $val(initialenergy)
        -txPower 35.28e-3
        -rxPower 31.32e-3
        -idlePower 712e-6
        -sleepPower 144e-9
for {set i 0} {$i < $val(nn)} {incr i} {
    set nnode_($i) [$ns node]
}
for {set i 1} {$i < $val(nn)} {incr i} {
    $nnode_($i) set X_ [expr {$val(x) * rand()}]
    $nnode_($i) set Y_ [expr {$val(y) * rand()}]
    $nnode_($i) set Z_ 0
}
#Position of Sink
$nnode_(0) set X_ [expr {$val(x)/2}]
$nnode_(0) set Y_ [expr {$val(y)/2}]
$nnode_(0) set Z_ 0.0
$nnode_(0) label "Sink"
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns initial_node_pos $nnode_($i) 10
}
#Setup a UDP connection
for {set i 1} {$i < $val(nn)} {incr i} {
set udp($i) [new Agent/UDP]
$ns attach-agent $nnode_($i) $udp($i)
}
set sink [new Agent/Null]
$ns attach-agent $nnode_(0) $sink
```

```
for {set i 1} {$i < $val(nn)} {incr i} {
$ns connect $udp($i) $sink
}
#Setup a CBR over UDP connection
for {set i 1} {$i < $val(nn)} {incr i} {
set cbr($i) [new Application/Traffic/CBR]
$cbr($i) attach-agent $udp($i)
$cbr($i) set type_ CBR
$cbr($i) set packet_size_ 100
$cbr($i) set maxpkts_ 100
#$cbr($i) set rate_ 0.1Mb
$cbr($i) set interval_ 1
$cbr($i) set random_ false
}
for {set i 1} {$i < $val(nn)} {incr i} {
$ns at [expr {$i + 5}] "$cbr($i) start"
}
for {set i 1} {$i < $val(nn)} {incr i} {
$ns at [expr $val(stop) - $i] "$cbr($i) stop"
}
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns at $val(stop) "$mnode_($i) reset;"
}
# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at [expr $val(stop) + 0.01] "puts \"end simulation\"; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
}
$ns run
```

OUTPUT

DATA TRANSFER



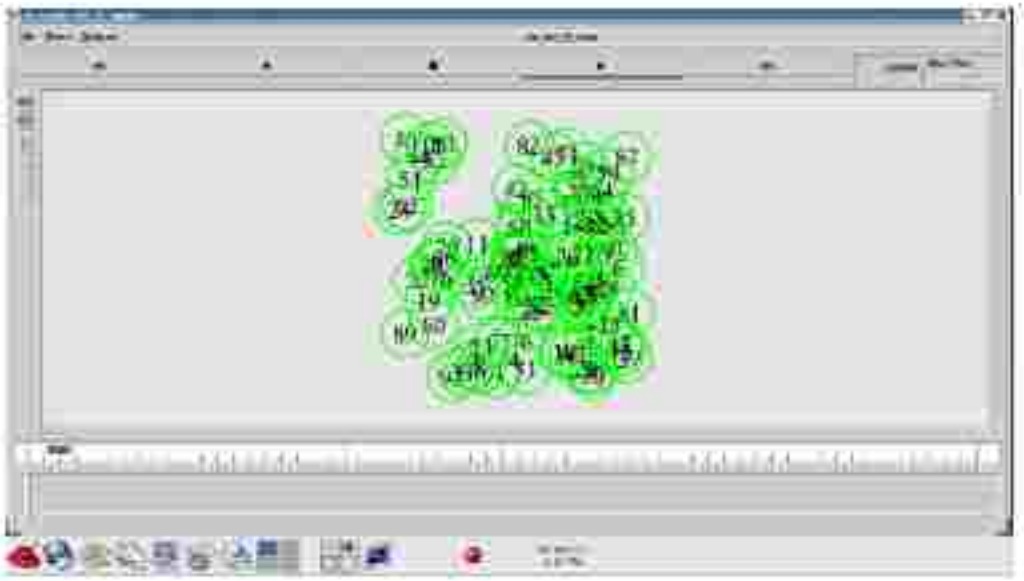
4.8 WSN PROGRAM – 802.15.4

```
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
set val(netif)          Phy/WirelessPhy/802_15_4   ;# network interface
set val(mac)            Mac/802_15_4               ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         100                        ;# max packet in ifq
set val(nn)             100                        ;# number of nodes
set val(rp)             AODV                       ;# protocol type
set val(x)              50                         ;# X dimension
set val(y)              50                         ;# Y dimension
set val(stop)           500                        ;# simulation period
set val(energymodel)    EnergyModel                ;# Energy Model
set val(initialenergy)  100                        ;# value
set ns                  [new Simulator]
set tracefd             [open simm_802_15_4.tr w]
set namtrace            [open simm_802_15_4.nam w]
$ns use-newtrace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
# configure the nodes
$ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channel [new $val(chan)] \
```

```
        -topoInstance $topo \
        -agentTrace OFF \
        -routerTrace OFF \
        -macTrace ON \
        -movementTrace OFF \
        -energyModel $val(energymodel) \
        -initialEnergy $val(initialenergy) \
        -txPower 3.5e-3 \
        -txPower 3.1e-3 \
        -idlePower 3.1e-3 \
        -sleepPower 1.5e-3
for {set i 0} {$i < $val(nn)} {incr i} {
    set nnode_($i) [$ns node]
}
for {set i 1} {$i < $val(nn)} {incr i} {
    $nnode_($i) set X_ [expr {$val(x) * rand()}]
    $nnode_($i) set Y_ [expr {$val(y) * rand()}]
    $nnode_($i) set Z_ 0
}
#Position of Sink
$nnode_(0) set X_ [expr {$val(x)/2}]
$nnode_(0) set Y_ [expr {$val(y)/2}]
$nnode_(0) set Z_ 0.0
$nnode_(0) label "Sink"
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns initial_node_pos $nnode_($i) 10
}
#Setup a UDP connection
for {set i 1} {$i < $val(nn)} {incr i} {
set udp($i) [new Agent/UDP]
$ns attach-agent $nnode_($i) $udp($i)
}
set sink [new Agent/Null]
$ns attach-agent $nnode_(0) $sink
```

```
for {set i 1} {$i < $val(nn)} {incr i} {
$ns connect $udp($i) $sink
}
#Setup a CBR over UDP connection
for {set i 1} {$i < $val(nn)} {incr i} {
set cbr($i) [new Application/Traffic/CBR]
$cbr($i) attach-agent $udp($i)
$cbr($i) set type_ CBR
$cbr($i) set packet_size_ 100
$cbr($i) set maxpkts_ 100
#$cbr($i) set rate_ 0.1Mb
$cbr($i) set interval_ 1
$cbr($i) set random_ false
}
for {set i 1} {$i < $val(nn)} {incr i} {
$ns at [expr {$i + 5}] "$cbr($i) start"
}
for {set i 1} {$i < $val(nn)} {incr i} {
$ns at [expr $val(stop) - $i] "$cbr($i) stop"
}
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns at $val(stop) "$mnode_($i) reset;"
}
# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at [expr $val(stop) + 0.01] "puts \"end simulation\"; $ns halt"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
}
$ns run
```

OUTPUT



5 Protocol Works

5.1 Procedure to construct Malicious Node in TCL Script and C++

- Modification in AODV PROTOCOL
- LOCATION – ns-allinone-2.33/ns2.33/aodv/aodv.cc
- LOCATION – ns-allinone-2.33/ns2.33/aodv/aodv.h

aodv.h file changes

Declare a boolean variable malicious as shown below in the protected scope in the class AODV

```
bool malicious;
```

aodv.cc file changes

1. Initialize the **malicious** variable with a value "false". Declare it inside the constructor as shown below

```
AODV::AODV(nsaddr_t id):Agent(PT_AODV)...
{
.....
malicious = false;
}
```

2. Add the following statement to the aodv.cc file in the "if(argc==2)" statement

```
if(strcmp(argv[1], "malicious") == 0) {
  desyn = true;
  return TCL_OK;
}
```

3. Implement the behavior of the **malicious** node by setting the following code in the rt_resolve(Packet *p) function. The **malicious** s node will simply drop the packet as indicated below.

```
if(malicious ==true)
{
drop(p,DROP_RTR_ROUTE_LOOP);
}
```

Once done, recompile ns2 as given below

Open Terminal -> Go to ~ns-2.33/ directory and type the command make to compile

S] cd /ns-allinone-2.33/ns-2.33/

S] make

Once the compilation is done. Check the **malicious** behavior using the Tcl Script by setting any four node as **malicious** node. The command to set the **malicious** node is

$ns at 2.0 "[$n0 set ragent_] malicious "

$ns at 2.0 "[$n8 set ragent_] malicious "

$ns at 2.0 "[$n23 set ragent_] malicious "

$ns at 2.0 "[$n19 set ragent_] malicious "

5.2 How to generate random mobility in ns2?

Procedure

Open the new terminal

cd ns-allinone-2.34

cd ns-2.34

cd indep-utils

pwd

ls

cd cmu-scen-gen

ls

cd setdest

ls

./setdest

./setdest -v 2 -n 10 -s 1 -m 10 -M 50 -t 30 -P 1 -p 1 -x 500 -y 500
./setdest -v 2 -n 10 -s 1 -m 10 -M 50 -t 30 -P 1 -p 1 -x 500 -y 500 >usersetdest.tcl
gedit usersetdest.tcl

5.3 How to generate random agent and application creation in ns2?

Procedure

cd ns-allinone-2.34
cd ns-2.34
cdindep-utils
cdcmu-scen-gen
ls
nsbrgen.tcl
nsbrgen.tcl -type cbr -nn 10 -seed 1 -mc 5 -rate 5.0
nsbrgen.tcl -type cbr -nn 10 -seed 1 -mc 5 -rate 5.0 > cbr-10.tcl
gedit cbr-10.tcl

6. PROGRAMS

6.1 PROGRAMS 1 – Wireless Network Construction using TCL script

Program Description

Basic wirless construction with number of nodes contained is three. The procedure to create nam file and trace file is given in this program. Topology is created by giving the position to the nodes and is specified by X, Y and Z coordinates. Here initial size of each and every nodes are built using initial_node_pos.The routing protocol which is used in this program is AODV (Adhoc On-demand Vector Routing Protocol). And simulation end time is 10ms.

File Name – program1.tcl

- Channel Type – Wireless Channel
- Propogation – Two Ray Ground Model
- X dimension – 500
- Y dimension – 400

```
# Define options
set val(chan)       Channel/WirelessChannel    ;# channel type
set val(prop)       Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)      Phy/WirelessPhy            ;# network interface type
set val(mac)        Mac/802_11                 ;# MAC type
set val(ifq)        Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)         LL                         ;# link layer type
set val(ant)        Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)     50                         ;# max packet in ifq
set val(nn)         3                          ;# number of mobilenodes
set val(rp)         AODV                       ;# routing protocol
set val(x)          500                        ;# X dimension of topography
set val(y)          400                        ;# Y dimension of topography
set val(stop)       10                         ;# time of simulation end
```

```
#----Event scheduler object creation----#

set ns [new Simulator]

# Creating trace file and nam file

set tracefd [open wireless1.tr w]
set namtrace [open wireless1.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace ON

## Creating node objects
for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }

# Provide initial location of mobile nodes
$node_(0) set X_ 50.0
```

```
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# Ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\"; $ns halt"
#stop procedure
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program1.tcl

OUTPUT

6.2 PROGRAM 2 – Code for the construction of wireless nodes with fixed colors

Program Description

Number of nodes in the network is eight which are created and configured as mobile wireless nodes. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y and Z coordinates and the Z coordinates are always remains zero. Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes are created in cyan color.

- Channel Type – Wireless Channel
- Propogation – Two Ray Ground Model
- Queue Type – DropTail
- Antenna Type – Omni Directional Antenna
- Number of nodes – 8
- Routing protocol - AODV
- X dimension – 500
- Y dimension – 400
- Stop time – 10ms
- Color – cyan color

File Name – program2.tcl

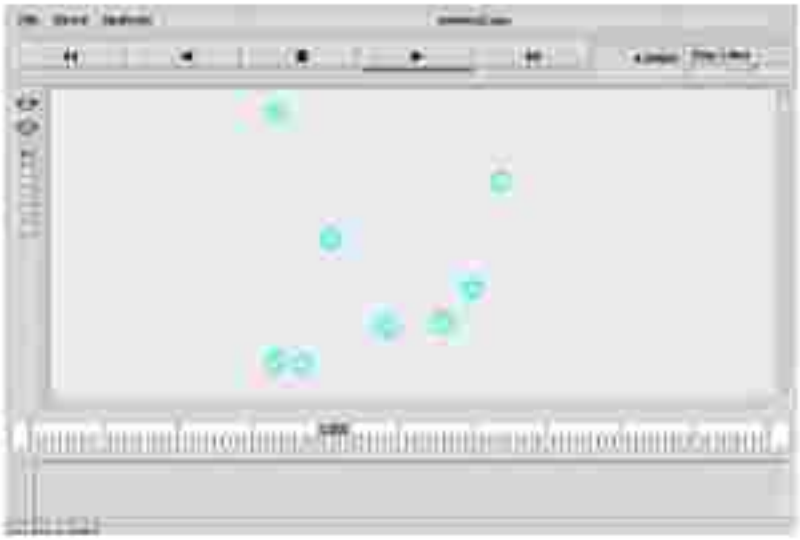
```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   50                         ;# max packet in ifq
set val(nn)       8                          ;# number of mobilenodes
set val(rp)       AODV                       ;# routing protocol
set val(x)        500                        ;# X dimension of topography
set val(y)        400                        ;# Y dimension of topography
set val(stop)     10                         ;# time of simulation end
```

```
#-----Event scheduler object creation-----#

set ns          [new Simulator]

#Creating trace file and nam file
set tracefd     [open wireless2.tr w]
set namtrace    [open wireless2.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# configure the nodes
    $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON

#Creating node objects
for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color cyan
        $ns at 0.0 "$node_($i) color cyan"
    }
```

```
# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 30.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 60.0
$node_(1) set Y_ 25.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 90.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 350.0
$node_(3) set Y_ 160.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 100.0
$node_(4) set Y_ 250.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 300.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 400.0
$node_(6) set Y_ 360.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 3s50.0
$node_(7) set Y_ 470.0
$node_(7) set Z_ 0.0

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless2.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program2.tcl

OUTPUT



6.3 PROGRAM 3 – Dynamic node creation program using AODV protocol TCL script

Program Discription

Number of nodes in the network is not static in this program. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y and Z coordinates and the Z coordinates are always remains zero. Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes are created in yellow color.

File Name – program3.tcl

```
if {$argc != 1} {
    error "\nCommand: ns wireless1.tcl <no.of.mobile-nodes>\n\n"
}
```

```
# Define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             [lindex $argv 0]           ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)              600                        ;# X dimension of topography
set val(y)              600                        ;# Y dimension of topography
set val(stop)           10                         ;# time of simulation end
```

```
#----------Event scheduler object creation----------#

set ns          [new Simulator]

#creating the trace file and nam file

set tracefd     [open wireless1.tr w]
set namtrace    [open wireless1.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#set up topography object
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

#configure the nodes
    $ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON

## Creating node objects ...
    for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)} {incr i} {
        $node_($i) color yellow
        $ns at 0.0 "$node_($i) color yellow"
    }
```

```
#Provide initial location of mobilenodes
$node_(0) set X_ 27.0
$node_(0) set Y_ 266.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 137.0
$node_(1) set Y_ 346.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 294.0
$node_(2) set Y_ 235.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 414.0
$node_(3) set Y_ 342.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 562.0
$node_(4) set Y_ 287.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 279.0
$node_(5) set Y_ 447.0
$node_(5) set Z_ 0.0

$node_(6) set X_ -128.0
$node_(6) set Y_ 260.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 727.0
$node_(7) set Y_ 269.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 130.0
$node_(8) set Y_ 126.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 318.0
$node_(9) set Y_ 45.0
$node_(9) set Z_ 0.0
```

```
$node_(10) set X_ 505.0
$node_(10) set Y_ 445.0
$node_(10) set Z_ 0.0

$node_(11) set X_ 421.0
$node_(11) set Y_ 153.0
$node_(11) set Z_ 0.0



# Define node initial position in nam
for {set i 0} {$i < $val(nn) } { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program3.tcl

OUTPUT

6.4 PROGRAM 4 – Dynamic node creation program and its initial location using AODV protocol TCL script

Program Discription

Number of nodes in the network is not static in this program. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y and Z coordinates and the Z coordinates are always remains zero. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes are created in yellow color.

File Name – program4.tcl

- X dimension – 600
- Y dimension – 600
- Stop time – 10ms
- Color – Yellow color
- Initial Node Position - 30

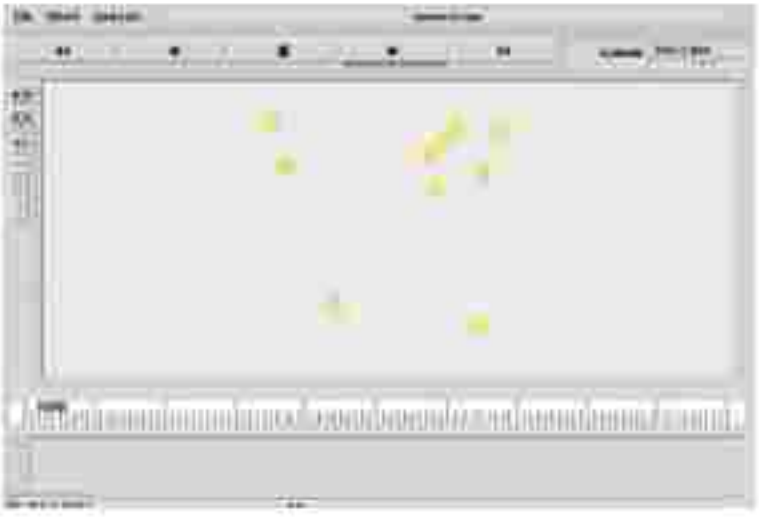
```
if {$argc != 1} {
    error "\nCommand: ns wireless3.tcl <no of mobile-nodes>\n\n"
}

# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   50                         ;# max packet in ifq
set val(nn)       [lindex $argv 0]           ;# number of mobilenodes
set val(rp)       AODV                       ;# routing protocol
set val(x)        600                        ;# X dimension of topography
set val(y)        600                        ;# Y dimension of topography
set val(stop)     10                         ;# time of simulation end
```

```
#----Event scheduler object creation----#

set ns [new Simulator]

#creating the trace file and nam file

set tracefd [open wireless3.tr w]
set namtrace [open wireless3.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace ON

## Creating node objects...
    for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)} {incr i} {
        $node_($i) color gold
        $ns at 0.0 "$node_($i) color gold"
    }
```

```
## Provide initial location of mobilenodes
        for {set i 0} {$i < $val(nn) } {incr i } {
            set xx [expr rand()*600]
            set yy [expr rand()*600]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }

#Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

#Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns at $val(stop) "$node_($i) reset";
}

#ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"

#stop procedure
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless3.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program4.tcl

OUTPUT



6.5 PROGRAM 5 – Dynamic color creation program and its initial location of nodes using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static in this program. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is not static. X and Y coordinates are randomly selected and the Z coordinates are always remains zero. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here all the nodes colors will get modified dynamically according to the time period

File Name – program5.tcl

- Number of nodes - 4
- X dimension – 750
- Y dimension – 550
- Stop time – 3.0ms
- Color – Yellow color
- Initial Node Position - 30

```
## Setting The wireless Channels..
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   5                          ;# max packet in ifq
set val(nn)       4                          ;# number of mobilenodes
set val(rp)       AODV                       ;# routing protocol
set val(x)        750                        ;# X dimension of topography
set val(y)        550                        ;# Y dimension of topography
set val(stop)     3.0                        ;# time of simulation end
```

```
#----Event scheduler object creation----#

    set ns [new Simulator]

## Create a trace file and nam file...
    set tracefd [open wireless1.tr w]
    set namtrace [open wireless1.nam w]

## Trace the nam and trace details from the main simulation...
    $ns trace-all $tracefd
    $ns namtrace-all-wireless $namtrace $val(x) $val(y)

## set up topography object...
    set topo [new Topography]

    $topo load_flatgrid $val(x) $val(y)

    set god_ [create-god $val(nn)]

## Color Descriptions...
    $ns color 1 dodgerblue
    $ns color 2 blue
    $ns color 3 cyan
    $ns color 4 green
    $ns color 5 yellow
    $ns color 6 black
    $ns color 7 magenta
    $ns color 8 gold
    $ns color 9 red

## Array for dynamic color settings...
    set colorname(0) blue
    set colorname(1) cyan
    set colorname(2) green
    set colorname(3) red
    set colorname(4) gold
    set colorname(5) magenta

## Setting The Distance Variables...
# For model TwoRayGround
    set dist(5m)  7.69113e-06
    set dist(9m)  2.37381e-06
    set dist(10m) 1.92278e-06
    set dist(11m) 1.58908e-06
```

```
set dist(12m) 1.33527e-06
set dist(13m) 1.13774e-06
set dist(25m) 3.07545e-07
set dist(30m) 2.13643e-07
set dist(35m) 1.56962e-07
set dist(40m) 1.56962e-10
set dist(45m) 1.56962e-11
set dist(50m) 1.20174e-13
#Phy/WirelessPhy set CSThresh_ $dist(50m)
#Phy/WirelessPhy set RXThresh_ $dist(50m)

## Setting node config event with set of inputs...
    $ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON


## Creating node objects...
    for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
    }

## Provide initial location of mobilenodes...
        for {set i 0} {$i < $val(nn) } {incr i} {
            set xx [expr rand()*500]
            set yy [expr rand()*500]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy
            $node_($i) set Z_ 0.0
        }
```

```
## Define node initial position in nam
    for {set i 0} {$i < $val(nn)} {incr i} {
    # 30 defines the node size for nam
        $ns initial_node_pos $node_($i) 30
    }

## Dynamic color procedure
$ns at 0.0 "dynamic-color"
proc dynamic-color {} {
    global ns val node_colorname
    set time 0.3
    set now [$ns now]
    set Rand [expr round(rand()*5)]
    for {set i 0} {$i < $val(nn)} {incr i} {
        $node_($i) color $colorname($Rand)
        $ns at $now "$node_($i) color $colorname($Rand)"
    }
    $ns at [expr $now+$time] "dynamic-color"
}
## stop procedure
$ns at $val(stop) "stop"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    puts "running nam..."
    exec nam wireless1.nam &
    exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program5.tcl

OUTPUT

6.6 PROGRAM 6 – Node mobility construction program using DSR routing protocol TCL script

Program Discription

Number of nodes in the network is static. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and specified receiver address which is randomly generated and also the mobility will get change according to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSR and the stop time of the simulation is 10ms.

File Name – program6.tcl

- X dimension – 750
- Y dimension – 550
- Stop time – 3.0ms
- Color – Yellow color
- Initial Node Position - 30

```
if {$argc != 1} {
    error "\nCommand: ns program6.tcl <no of mobile-nodes>\n\n"
}

## Setting The wireless Channels
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   5                          ;# max packet in ifq
set val(nn)       [lindex $argv 0]           ;# number of mobilenodes
set val(rp)       DSR                        ;# routing protocol
set val(x)        750                        ;# X dimension of topography
set val(y)        550                        ;# Y dimension of topography
set val(stop)     10.0                       ;# time of simulation end
```

```
#----Event scheduler object creation----#

    set ns [new Simulator]

## Create a trace file and nam file
    set traceId [open wireless2.tr w]
    set namtrace [open wireless2.nam w]

## Trace the nam and trace details from the main simulation
    $ns trace-all $traceId
    $ns namtrace-all-wireless $namtrace $val(x) $val(y)

## set up topography object
    set topo [new Topography]

    $topo load_flatgrid $val(x) $val(y)

    set god_ [create-god $val(nn)]

## Color Descriptions
    $ns color 1 dodgerblue
    $ns color 2 blue
    $ns color 3 cyan
    $ns color 4 green
    $ns color 5 yellow
    $ns color 6 black
    $ns color 7 magenta
    $ns color 8 gold
    $ns color 9 red
## Setting The Distance Variables
# For model 'TwoRayGround'
    set dist(5m) 7.69113e-06
    set dist(9m) 2.37381e-06
    set dist(10m) 1.92278e-06
    set dist(11m) 1.58908e-06
    set dist(12m) 1.33527e-06
    set dist(13m) 1.13774e-06
    set dist(25m) 3.07645e-07
    set dist(30m) 2.13643e-07
    set dist(35m) 1.56962e-07
    set dist(40m) 1.58962e-10
    set dist(45m) 1.58962e-11
    set dist(50m) 1.20174e-13
    #Phy/WirelessPhy set CSThresh_ $dist(50m)
    #Phy/WirelessPhy set RXThresh_ $dist(50m)
```

```
## Setting node config event with set of inputs

    $ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON


## Creating node objects
    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < 4} {incr i} {
        $node_($i) color yellow
        $ns at 0.0 "$node_($i) color yellow"
    }
    for {set i 4} {$i < 10 } {incr i } {
        $node_($i) color red
        $ns at 3.0 "$node_($i) color red"
    }
    for {set i 10} {$i < 15 } {incr i } {
        $node_($i) color blue
        $ns at 5.0 "$node_($i) color blue"
    }

## Provide initial location of mobilenodes
    $node_(0) set X_ 27.0
    $node_(0) set Y_ 260.0
    $node_(0) set Z_ 0.0

    $node_(1) set X_ 137.0
    $node_(1) set Y_ 348.0
    $node_(1) set Z_ 0.0
```

```
$node_(2) set X_ 264.0
$node_(2) set Y_ 235.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 414.0
$node_(3) set Y_ 342.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 562.0
$node_(4) set Y_ 267.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 279.0
$node_(5) set Y_ 447.0
$node_(5) set Z_ 0.0
$node_(6) set X_ -128.0
$node_(6) set Y_ 260.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 727.0
$node_(7) set Y_ 259.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 139.0
$node_(8) set Y_ 126.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 318.0
$node_(9) set Y_ 45.0
$node_(9) set Z_ 0.0

$node_(10) set X_ 605.0
$node_(10) set Y_ 446.0
$node_(10) set Z_ 0.0

$node_(11) set X_ 421.0
$node_(11) set Y_ 158.0
$node_(11) set Z_ 0.0

$node_(12) set X_ 72.0
$node_(12) set Y_ 397.0
$node_(12) set Z_ 0.0

if {$val(nn) > 12} {
    for {set i 13} {$i < $val(nn)} { incr i } {
        set xx [expr rand()*600]
```

```
            set yy [expr rand()*500]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy
            $node_($i) set Z_ 0.0
        }
    }
## Define node initial position in nam
    for {set i 0} {$i < $val(nn) } {incr i } {
    # 30 defines the node size for nam
        $ns initial_node_pos $node_($i) 30
    }
## Stop procedure
$ns at 0.0 "destination"
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incr i} {
        set xx [expr rand()*600]
        set yy [expr rand()*500]
        $ns at $now "$node_($i) setdest $xx $yy 20.0"
    }
    $ns at [expr $now+$time] "destination"
}

$ns at $val(stop) "stop"

#stop procedure
proc stop {} {
  global ns tracefd namtrace
  $ns flush-trace
  close $tracefd
  close $namtrace
  puts "running nam..."
  exec nam wireless2.nam &
  exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program6.tcl

OUTPUT

6.7 PROGRAM 7 – Creation of TCP (Transmission Control Protocol) communication between the nodes using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectified receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 150ms. Three nodes are created which are node0, node 1 and node 2. Send TCP agent is created and attached to node0, destination TCPsink agent is created and attached to node1. Then the TCP agent and the TCPsink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now the communication is initiated.

File Name – program7.tcl

- X dimension – 500
- Y dimension – 400
- Stop time – 150 ms

```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   50                         ;# max packet in ifq
set val(nn)       3                          ;# number of mobilenodes
set val(rp)       AODV                       ;# routing protocol
set val(x)        500                        ;# X dimension of topography
set val(y)        400                        ;# Y dimension of topography
set val(stop)     150                        ;# time of simulation end
```

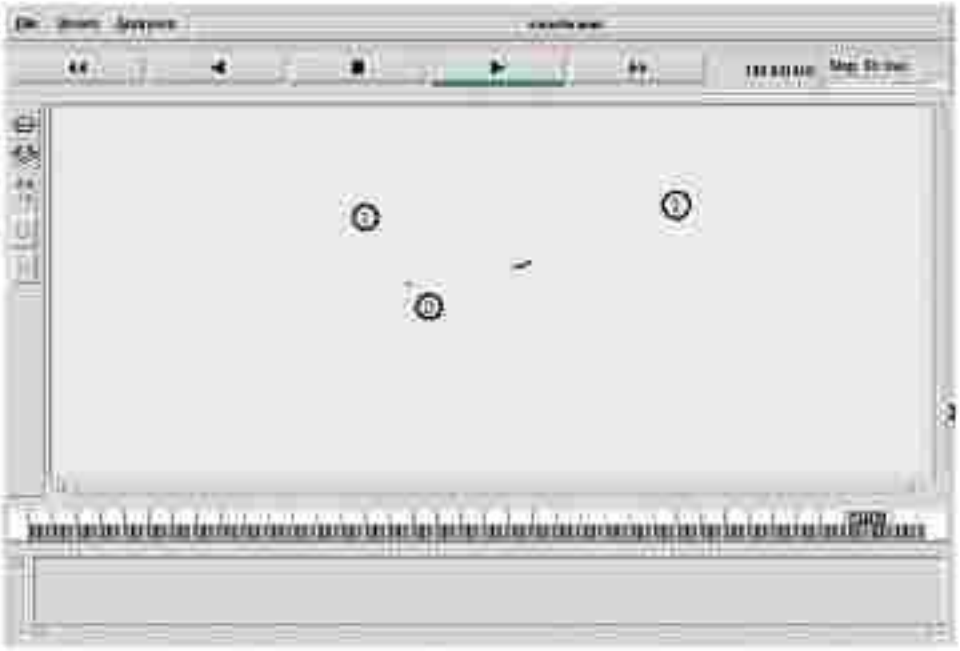
```
#----Event scheduler object creation----#

set ns          [new Simulator]
#creating trace file and nam file
set traceId     [open wireless1.tr w]
set windowVsTime2 [open win.tr w]
set namtrace    [open wireless1.nam w]

$ns trace-all $traceId
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#set up topography object
set topo    [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#configure the nodes
    $ns node-config -adhocRouting $val(rp)\
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen)\
            -antType $val(ant) \
            -propType $val(prop)\
            -phyType $val(netif) \
            -channelType $val(chan)\
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON

  for {set i 0} {$i < $val(nn)} {incr i} {
      set node_($i) [$ns node]
  }

#Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 19.0 "$node_(2) setdest 480.0 300.0 5.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(2) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file"
$ns at 10.0 "plotWindow $tcp $windowVsTime2"
```

```
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam simwris.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program7.tcl

OUTPUT



6.8 PROGRAM 8 – Creation of TCP (Transmission Control Protocol) communication between the nodes using DSR routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectified receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSR and the stop time of the simulation is 150ms. Three nodes are created which are node0, node 1 and node 2. Send TCP agent is created and attached to node0, destination TCPsink agent is created and attached to node1. Then the TCP agent and the TCPsink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now the communication is initiated.

File Name – program8.tcl

- X dimension – 500
- Y dimension – 400
- Stop time – 150 ms

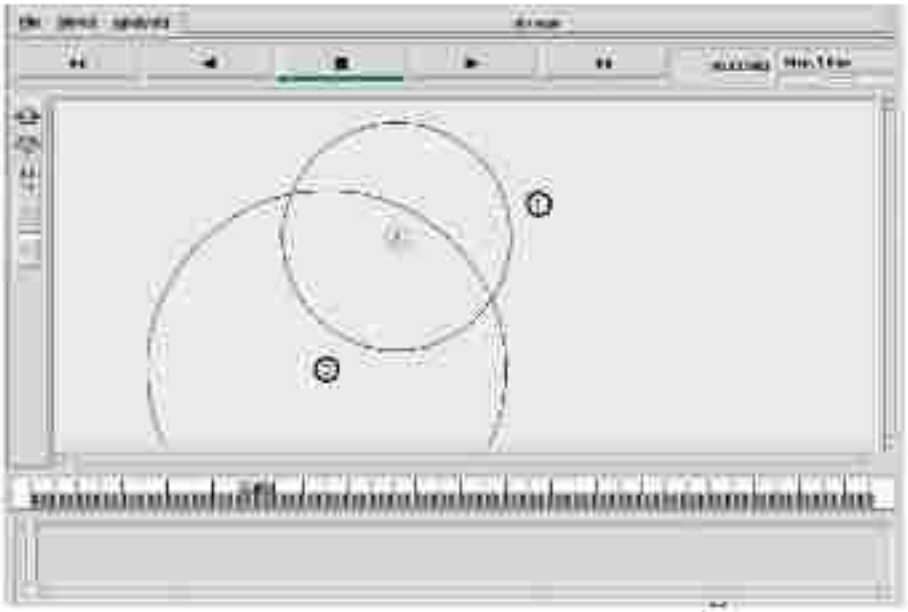
```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   50                         ;# max packet in ifq
set val(nn)       3                          ;# number of mobilenodes
set val(rp)       DSR                        ;# routing protocol
set val(x)        500                        ;# X dimension of topography
set val(y)        400                        ;# Y dimension of topography
set val(stop)     150                        ;# time of simulation end
```

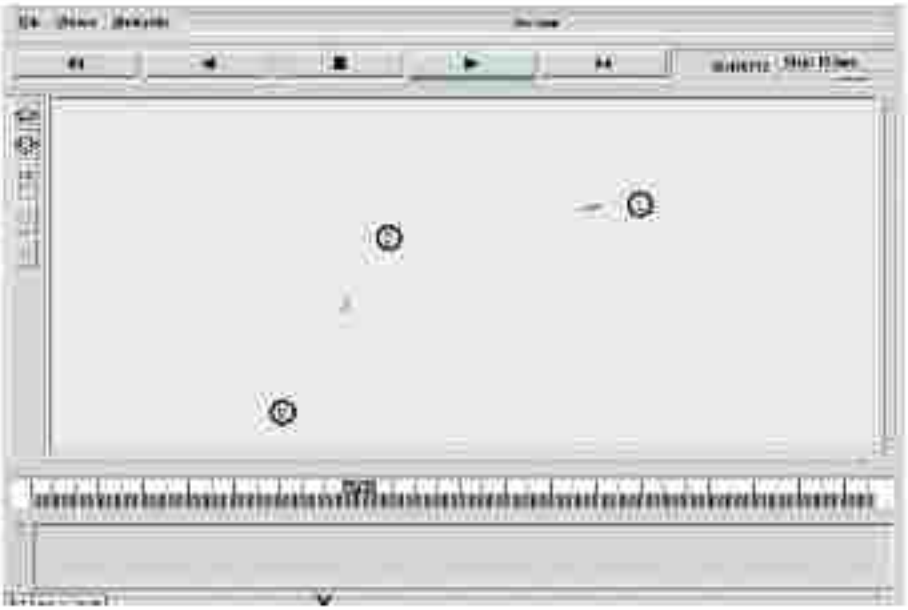
```
#-------Event scheduler object creation------#

set ns          [new Simulator]
#Creating trace file and nam file
set tracefd     [open dsr.tr w]
set windowVsTime2 [open win.tr w]
set namtrace    [open dsr.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#set up topography object
set topo        [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

#configure the nodes
        $ns node-config -adhocRouting $val(rp)\
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen)\
                        -antType $val(ant) \
                        -propType $val(prop)\
                        -phyType $val(netif) \
                        -channelType $val(chan)\
                        -topoInstance $topo\
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace ON

    for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns node]
    }

#Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 480.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 300.0 5.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.1 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}


# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam dsr.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program8.tcl

OUTPUT





6.9 PROGRAM 9 – Creation of UDP (User Datagram Protocol) communication between nodes with CBR traffic using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as 22 in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 150ms. Send UDP agent is created and attached to sender node, destination UDPNull agent is created and attached to destination node. Then the UDP agent and the UDPNull agent are connected. In the next level, CBR application is created and attached to the sender UDP agent. Now the communication is initiated.

File Name – program9.tcl

- Number or nodes - 22
- X dimension – 1800
- Y dimension – 840
- Stop time – 150 ms

```
#Define setting option
set val(chan)       Channel/WirelessChannel     ;#Channel Type
set val(prop)       Propagation/TwoRayGround    ;# radio-propagation model
set val(netif)      Phy/WirelessPhy             ;# network interface type
set val(mac)        Mac/802_11                  ;# MAC type
set val(ifq)        Queue/DropTail/PriQueue     ;# interface queue type
set val(ll)         LL                          ;# link layer type
set val(ant)        Antenna/OmniAntenna         ;# antenna model
set val(ifqlen)     50                          ;# max packet in ifq
set val(nn)         22                          ;# number of mobilenodes
set val(rp)         AODV                        ;# routing protocol
set val(x)          1800
set val(y)          840
```

```
### Setting The Simulator Objects

    set ns_ [new Simulator]
#create the nam and trace file
    set tracefd [open aodv.tr w]
    $ns_ trace-all $tracefd

    set namtrace [open aodv.nam w]
    $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

    set topo [new Topography]
    $topo load_flatgrid $val(x) $val(y)
    create-god $val(nn)
    set chan_1_ [new $val(chan)]

#### Setting The Distance Variables


    # For model='TwoRayGround'
    set dist(5m) 7.69113e-06
    set dist(6m) 2.37381e-06
    set dist(10m) 1.92278e-06
    set dist(11m) 1.58908e-06
    set dist(12m) 1.33527e-06
    set dist(13m) 1.13774e-06
    set dist(14m) 9.81011e-07
    set dist(15m) 8.54570e-07
    set dist(16m) 7.51087e-07
    set dist(20m) 4.80698e-07
    set dist(25m) 3.07645e-07
    set dist(30m) 2.13643e-07
    set dist(35m) 1.56962e-07
    set dist(40m) 1.56962e-10
    set dist(45m) 1.56962e-11
    set dist(50m) 1.20174e-13
    Phy/WirelessPhy set CSThresh_ $dist(50m)
    Phy/WirelessPhy set RXThresh_ $dist(50m)

# Defining Node Configuration

            $ns_ node-config -adhocRouting $val(rp) \
             -llType $val(ll) \
             -macType $val(mac) \
             -ifqType $val(ifq) \
             -ifqLen $val(ifqlen) \
             -antType $val(ant) \
```

```
        -propType $val(prop) \
        -phyType $val(netif) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace ON \
        -movementTrace ON \
        -channel $chan_1_
```

Creating The WIRELESS NODES

```
set Server1 [$ns_ node]
set Server2 [$ns_ node]
set n2 [$ns_ node]
set n3 [$ns_ node]
set n4 [$ns_ node]
set n5 [$ns_ node]
set n6 [$ns_ node]
set n7 [$ns_ node]
set n8 [$ns_ node]
set n9 [$ns_ node]
set n10 [$ns_ node]
set n11 [$ns_ node]
set n12 [$ns_ node]
set n13 [$ns_ node]
set n14 [$ns_ node]
set n15 [$ns_ node]
set n16 [$ns_ node]
set n17 [$ns_ node]
set n18 [$ns_ node]
set n19 [$ns_ node]
set n20 [$ns_ node]
set n21 [$ns_ node]
set n22 [$ns_ node]

set opt(seed) 0.1
set a [ns-random $opt(seed)]
set i 0
while {$i < 5} {
incr i
}
```

Setting The Initial Positions of Nodes

```
$Server1 set X_ 513.0
$Server1 set Y_ 517.0
$Server1 set Z_ 0.0

$Server2 set X_ 1445.0
$Server2 set Y_ 474.0
$Server2 set Z_ 0.0

$n2 set X_ 36.0
$n2 set Y_ 529.0
$n2 set Z_ 0.0

$n3 set X_ 143.0
$n3 set Y_ 666.0
$n3 set Z_ 0.0

$n4 set X_ 201.0
$n4 set Y_ 552.0
$n4 set Z_ 0.0

$n5 set X_ 147.0
$n5 set Y_ 403.0
$n5 set Z_ 0.0

$n6 set X_ 230.0
$n6 set Y_ 291.0
$n6 set Z_ 0.0

$n7 set X_ 295.0
$n7 set Y_ 415.0
$n7 set Z_ 0.0

$n8 set X_ 363.0
$n8 set Y_ 335.0
$n8 set Z_ 0.0

$n9 set X_ 334.0
$n9 set Y_ 647.0
$n9 set Z_ 0.0

$n10 set X_ 304.0
$n10 set Y_ 777.0
$n10 set Z_ 0.0

$n11 set X_ 412.0
$n11 set Y_ 194.0
```

```
$n11 setZ_0.0

$n12 setX_619.0
$n12 setY_361.0
$n12 setZ_0.0

$n13 setX_569.0
$n13 setY_167.0
$n13 setZ_0.0

$n14 setX_349.0
$n14 setY_546.0
$n14 setZ_0.0

$n15 setX_456.0
$n15 setY_668.0
$n15 setZ_0.0

$n16 setX_489.0
$n16 setY_794.0
$n16 setZ_0.0

$n17 setX_606.0
$n17 setY_711.0
$n17 setZ_0.0

$n18 setX_630.0
$n18 setY_626.0
$n18 setZ_0.0

$n19 setX_666.0
$n19 setY_347.0
$n19 setZ_0.0

$n20 setX_741.0
$n20 setY_152.0
$n20 setZ_0.0

$n21 setX_882.0
$n21 setY_264.0
$n21 setZ_0.0

$n22 setX_761.0
$n22 setY_441.0
$n22 setZ_0.0
```

Giving Mobility to Nodes

```
$ns_ at 0.75 "$n2 setdest 379.0 349.0 20.0"
$ns_ at 0.75 "$n3 setdest 566.0 302.0 20.0"
$ns_ at 0.20 "$n4 setdest 309.0 211.0 20.0"
$ns_ at 1.25 "$n5 setdest 179.0 333.0 20.0"
$ns_ at 0.75 "$n6 setdest 139.0 63.0 20.0"
$ns_ at 0.75 "$n7 setdest 320.0 27.0 20.0"
$ns_ at 1.50 "$n8 setdest 505.0 124.0 20.0"
$ns_ at 1.25 "$n9 setdest 274.0 487.0 20.0"
$ns_ at 1.25 "$n10 setdest 494.0 475.0 20.0"
$ns_ at 1.25 "$n11 setdest 899.0 757.0 25.0"
$ns_ at 0.50 "$n12 setdest 598.0 728.0 25.0"
$ns_ at 0.25 "$n13 setdest 561.0 624.0 25.0"
$ns_ at 1.25 "$n14 setdest 387.0 547.0 25.0"
$ns_ at 1.25 "$n15 setdest 748.0 689.0 25.0"
$ns_ at 1.25 "$n16 setdest 842.0 623.0 25.0"
$ns_ at 1.25 "$n17 setdest 578.0 548.0 25.0"
$ns_ at 0.75 "$n18 setdest 741.0 805.0 20.0"
$ns_ at 0.75 "$n19 setdest 437.0 799.0 20.0"
$ns_ at 0.20 "$n20 setdest 159.0 722.0 20.0"
$ns_ at 1.25 "$n21 setdest 700.0 350.0 20.0"
$ns_ at 0.75 "$n22 setdest 839.0 444.0 20.0"
```

Setting The Node Size

```
$ns_ initial_node_pos $Server1 75
$ns_ initial_node_pos $Server2 75
$ns_ initial_node_pos $n2 40
$ns_ initial_node_pos $n3 40
$ns_ initial_node_pos $n4 40
$ns_ initial_node_pos $n5 40
$ns_ initial_node_pos $n6 40
$ns_ initial_node_pos $n7 40
$ns_ initial_node_pos $n8 40
$ns_ initial_node_pos $n9 40
$ns_ initial_node_pos $n10 40
$ns_ initial_node_pos $n11 40
$ns_ initial_node_pos $n12 40
$ns_ initial_node_pos $n13 40
$ns_ initial_node_pos $n14 40
$ns_ initial_node_pos $n15 40
$ns_ initial_node_pos $n16 40
$ns_ initial_node_pos $n17 40
$ns_ initial_node_pos $n18 40
$ns_ initial_node_pos $n19 40
```

```
$ns_ initial_node_pos $n20 40
$ns_ initial_node_pos $n21 40
$ns_ initial_node_pos $n22 40


#### Setting The Labels For Nodes

$ns_ at 0.0 "$Server1 label Server1"
$ns_ at 0.0 "$Server2 label Server2"

#Setting Color For Server

$Server1 color maroon
$ns_ at 0.0 "$Server1 color maroon"

$Server2 color maroon
$ns_ at 0.0 "$Server2 color maroon"

## SETTING ANIMATION RATE
$ns_ at 0.0 "$ns_ set-animation-rate 15.0ms"

 # COLORING THE NODES
$n9 color blue
$ns_ at 4.71 "$n9 color blue"
$n8 color blue
$ns_ at 7.0 "$n8 color blue"
$n2 color blue
$ns_ at 7.29 "$n2 color blue"

$n16 color blue
$ns_ at 7.53 "$n16 color blue"

$n9 color maroon
$ns_ at 7.44 "$n9 color maroon"

$ns_ at 7.43 "$n9 label TTLover"
$ns_ at 7.55 "$n9 label ""

$n12 color blue
$ns_ at 7.85 "$n12 color blue"

#### Establishing Communication

set udp0 [$ns_ create-connection UDP $Server1 LossMonitor $n16 0]
$udp0 set fid_ 1
set cbr0 [$udp0 attach-app Traffic/CBR]
```

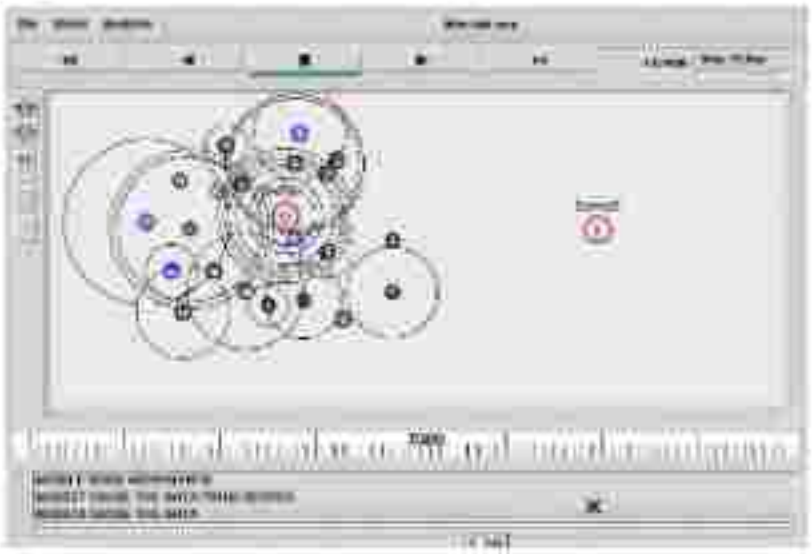
```
$cbr0 set packetSize_ 1000
$cbr0 set interval_ .07
$ns_ at 0.0 "$cbr0 start"
$ns_ at 4.0 "$cbr0 stop"

set udp1 [$ns_ create-connection UDP $Server1 LossMonitor $n220]
$udp1 set fid_ 1
set cbr1 [$udp1 attach-app Traffic/CBR]
$cbr1 set packetSize_ 1000
$cbr1 set interval_ .07
$ns_ at 0.1 "$cbr1 start"
$ns_ at 4.1 "$cbr1 stop"


set udp2 [$ns_ create-connection UDP $n21 LossMonitor $n20-0]
$udp2 set fid_ 1
set cbr2 [$udp2 attach-app Traffic/CBR]
$cbr2 set packetSize_ 1000
$cbr2 set interval_ .07
$ns_ at 2.4 "$cbr2 start"
$ns_ at 4.1 "$cbr2 stop"

set udp3 [$ns_ create-connection UDP $Server1 LossMonitor $n150]
$udp3 set fid_ 1
set cbr3 [$udp3 attach-app Traffic/CBR]
$cbr3 set packetSize_ 1000
$cbr3 set interval_ .5
$ns_ at 4.0 "$cbr3 start"
$ns_ at 4.1 "$cbr3 stop"

set udp4 [$ns_ create-connection UDP $Server1 LossMonitor $n140]
$udp4 set fid_ 1
set cbr4 [$udp4 attach-app Traffic/CBR]
$cbr4 set packetSize_ 1000
$cbr4 set interval_ .5
$ns_ at 4.0 "$cbr4 start"
$ns_ at 4.1 "$cbr4 stop"

set udp5 [$ns_ create-connection UDP $n15 LossMonitor $n16-0]
$udp5 set fid_ 1
set cbr5 [$udp5 attach-app Traffic/CBR]
$cbr5 set packetSize_ 1000
$cbr5 set interval_ .5
$ns_ at 4.0 "$cbr5 start"
$ns_ at 4.1 "$cbr5 stop"
```

```
set udp6 [$ns_ create-connection UDP $n15 LossMonitor $n17 0]
$udp6 set fid_ 1
set cbr6 [$udp6 attach-app Traffic/CBR]
$cbr6 set packetSize_ 1000
$cbr6 set interval_ 5
$ns_ at 4.0 "$cbr6 start"
$ns_ at 4.1 "$cbr6 stop"

set udp7 [$ns_ create-connection UDP $n14 LossMonitor $n4 0]
$udp7 set fid_ 1
set cbr7 [$udp7 attach-app Traffic/CBR]
$cbr7 set packetSize_ 1000
$cbr7 set interval_ 5
$ns_ at 4.0 "$cbr7 start"
$ns_ at 4.1 "$cbr7 stop"

set udp8 [$ns_ create-connection UDP $n14 LossMonitor $n9 0]
$udp8 set fid_ 1
set cbr8 [$udp8 attach-app Traffic/CBR]
$cbr8 set packetSize_ 1000
$cbr8 set interval_ 5
$ns_ at 4.0 "$cbr8 start"
$ns_ at 4.1 "$cbr8 stop"

set udp9 [$ns_ create-connection UDP $n4 LossMonitor $n3 0]
$udp9 set fid_ 1
set cbr9 [$udp9 attach-app Traffic/CBR]
$cbr9 set packetSize_ 1000
$cbr9 set interval_ 5
$ns_ at 4.0 "$cbr9 start"
$ns_ at 4.1 "$cbr9 stop"

set udp10 [$ns_ create-connection UDP $n4 LossMonitor $n2 0]
$udp10 set fid_ 1
set cbr10 [$udp10 attach-app Traffic/CBR]
$cbr10 set packetSize_ 1000
$cbr10 set interval_ 5
$ns_ at 4.0 "$cbr10 start"
$ns_ at 4.1 "$cbr10 stop"

set udp11 [$ns_ create-connection UDP $n9 LossMonitor $n16 0]
$udp11 set fid_ 1
set cbr11 [$udp11 attach-app Traffic/CBR]
$cbr11 set packetSize_ 1000
$cbr11 set interval_ 5
$ns_ at 4.0 "$cbr11 start"
```

```
$ns_ at 4.1 "$cbr11 stop"

set udp12 [$ns_ create-connection UDP $n9 LossMonitor $n10 0]
$udp12 set fid_ 1
set cbr12 [$udp12 attach-app Traffic/CBR]
$cbr12 set packetSize_ 1000
$cbr12 set interval_ 6
$ns_ at 4.0 "$cbr12 start"
$ns_ at 4.1 "$cbr12 stop"

#ANNOTATIONS DETAILS

$ns_ at 0.0 "$ns_ trace-annotate \"MOBILE NODE MOVEMENTS\""
$ns_ at 4.1 "$ns_ trace-annotate \"NODE27 CACHE THE DATA FRO SERVER\""
#$ns_ at 4.58 "$ns_ trace-annotate \"PACKET LOSS AT NODE27\""
$ns_ at 4.71 "$ns_ ttrace-annotate \"NODE10 CACHE THE DATA\""

### PROCEDURE TO STOP

proc stop {} {

        global ns_ tracefd
        $ns_ flush-trace
        close $tracefd
        exec nam datacache.nam &
        exit 0

    }

puts "Starting Simulation....."
$ns_ at 26.0 "stop"
$ns_ run
```

Procedure to run the program in the terminal window - $ns program9.tcl

OUTPUT



6.10 PROGRAM 10 – Creation of TCP (Transmission Control Protocol) communication between the nodes using DSDV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectified receiver address which is randomly generated and also the mobility will get change accoding to the time period.

Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSDV and the stop time of the simulation is 150ms. Three nodes are created which are node0, node 1 and node 2. Send TCP agent is created and attached to node0, destination TCPsink agent is created and attached to node1. Then the TCP agent and the TCPsink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now the communication is initiated.

File Name – program10.tcl

- Channel Type – Wireless Channel
- Propogation – Two Ray Ground Model
- Queue Type – DropTail
- Antenna Type – Omni Directional Antenna
- Number of nodes – 3
- Routing protocol - DSDV
- X dimension – 500
- Y dimension – 400
- Stop time – 150ms
- AGENT – TCP
- Application - FTP

```
# Define setting option
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   50                         ;# max packet in ifq
set val(nn)       3                          ;# number of mobilenodes
set val(rp)       DSDV                       ;# routing protocol
set val(x)        500                        ;# X dimension of topography
set val(y)        400                        ;# Y dimension of topography
set val(stop)     150                        ;# time of simulation end
```

```
#Creating trace file and nam file
set tracefd     [open dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace    [open dsdv.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

# configure the nodes
      $ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON

    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
```

```
# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 300.0 5.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.1 "plotWindow $tcp $windowVsTime2"
```

```
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam dsdv.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program10.tcl

OUTPUT

6.11 PROGRAM 11 – Mobile node energy model construction TCL script

Program Discription

Number of nodes in the network is static and is declared as six in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero.

Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the mobility will get change acceding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is DSR and the stop time of the simulation is 18ms.

File Name – program11.tcl

- Channel – Wireless Channel
- Propagation – Two Ray Ground Propagation
- Queue Type – Drop Tail
- Antenna – Omni Directional Antenna
- Initial Energy – 20 J
- Transmission Power – 0.9 J
- Receiver Power – 0.8 J
- Idle Power – 0.0 J
- Sense Power – 0.0175 J
- Routing Protocol – DSR
- Simulation Time – 18ms
- Number of nodes – 6 nodes
- X dimension – 750
- Y dimension – 550
- Initial Node Position - 30

```
## Setting The wireless Channels.

set val(chan)         Channel/WirelessChannel    ;# channel type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)        Phy/WirelessPhy            ;# network interface type
set val(mac)          Mac/802_11                 ;# MAC type
set val(ifq)          Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)           LL                         ;# link layer type
set val(ant)          Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)       5                          ;# max packet in ifq
set val(nn)           6                          ;# number of mobilenodes
set val(rp)           DSR                        ;# routing protocol
set val(x)            750                        ;# X dimension of topography
set val(y)            550                        ;# Y dimension of topography
set val(stop)         18.0                       ;# time of simulation end

## Create a simulator object(nothing but a scheduler's object).
    set ns [new Simulator]

## Create a trace file and nam file
    set tracefd [open wireless1.tr w]
    set namtrace [open wireless1.nam w]

## Trace the nam and trace details from the main simulation
    $ns trace-all $tracefd
    $ns namtrace-all-wireless $namtrace $val(x) $val(y)

## set up topography object
    set topo [new Topography]

    $topo load_flatgrid $val(x) $val(y)

    set god_ [create-god $val(nn)]
```

```
## Color Descriptions
    $ns color 1 dodgerblue
    $ns color 2 blue
    $ns color 3 cyan
    $ns color 4 green
    $ns color 5 yellow
    $ns color 6 black
    $ns color 7 magenta
    $ns color 8 gold
    $ns color 9 red

# Setting The Distance Variables.
# For model 'TwoRayGround'
    set dist(5m)  7.69113e-06
    set dist(9m)  2.37381e-06
    set dist(10m) 1.92278e-06
    set dist(11m) 1.58908e-06
    set dist(12m) 1.33527e-06
    set dist(13m) 1.13774e-06
    set dist(14m) 9.81011e-07
    set dist(15m) 8.54570e-07
    set dist(16m) 7.51087e-07
    set dist(20m) 4.80696e-07
    set dist(25m) 3.07645e-07
    set dist(30m) 2.13643e-07
    set dist(35m) 1.56962e-07
    set dist(40m) 1.56962e-10
    set dist(45m) 1.56962e-11
    set dist(50m) 1.20174e-13
    #Phy/WirelessPhy set CSThresh_ $dist(50m)
    #Phy/WirelessPhy set RXThresh_ $dist(50m)
```

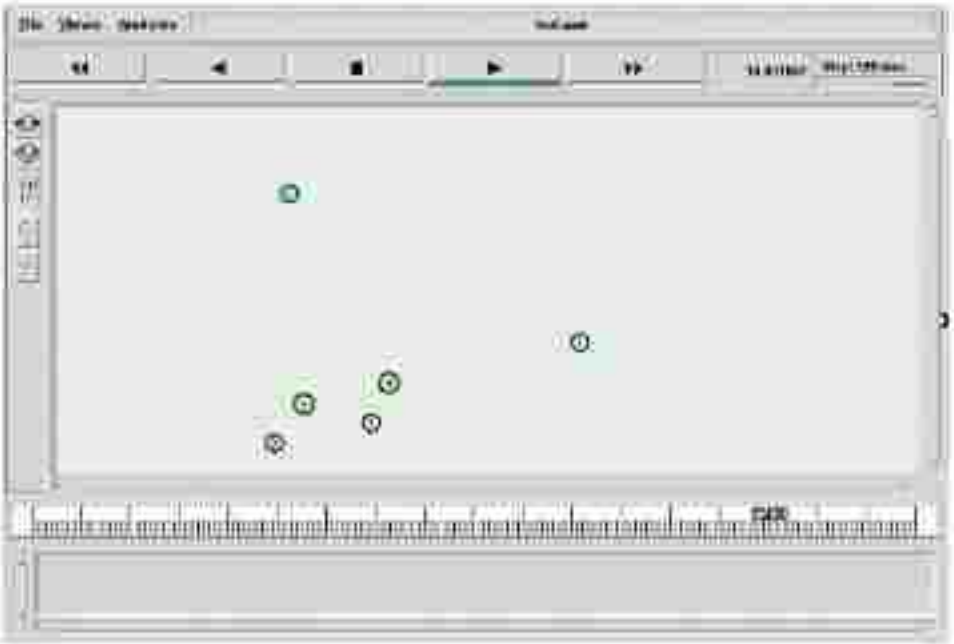
```
## Setting node config event with set of inputs
    puts "Node Configuration Started here ..\n\
            -channel $val(chan) \n \
            -adhocRouting $val(rp) \n \
            -llType $val(ll) \n \
            -macType $val(mac) \n \
            -ifqType $val(ifq) \n \
            -ifqLen $val(ifqlen) \n \
            -antType $val(ant) \n \
            -propType $val(prop) \n \
            -phyType $val(netif) \n"


     $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON
# Energy model
    $ns node-config  -energyModel EnergyModel \
                -initialEnergy 20 \
                -txPower 0.9 \
                -rxPower 0.8 \
                -idlePower 0.0 \
                -sensePower 0.0175
```

```
    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }

for {set i 0} {$i < $val(nn)} {incr i} {
    $node_($i) color darkgreen
    $ns at 0.0 "$node_($i) color darkgreen"
    }

## Provide initial location of mobilenodes

    if {$val(nn) >0} {
        for {set i 1} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*600]
            set yy [expr rand()*500];
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy
        }
    }

## set god distance
    $god_ set-dist 0 1 2
    $god_ set-dist 0 2 2
    $god_ set-dist 0 3 2
    $god_ set-dist 0 4 1
    $god_ set-dist 0 5 2
    $god_ set-dist 1 2 3
    $god_ set-dist 1 3 3


## Define node initial position in nam
    for {set i 0} {$i < $val(nn)} { incr i } {
    # 30 defines the node size for nam
        $ns initial_node_pos $node_($i) 30
    }
```

```
## Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

## Ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 16.01 "puts \"end simulation\" " ;# $ns halt

## Stop procedure

proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace

    exec nam wireless1.nam &
    exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program11.tcl

OUTPUT



6.12 PROGRAM 12 – Creation of nodes at random destination at particular time interval using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is not static and is declared as six in the network. Nodes are configured in the mobile wireless node format. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the detination location will get change accoding to the time period.

File Name – program12.tcl

```
# Define options
set val(chan)         Channel/WirelessChannel    # channel type
set val(prop)         Propagation/TwoRayGround   # radio-propagation model
set val(netif)        Phy/WirelessPhy            # network interface type
set val(mac)          Mac/802_11                 # MAC type
set val(ifq)          Queue/DropTail/PriQueue    # interface queue type
set val(ll)           LL                         # link layer type
set val(ant)          Antenna/OmniAntenna        # antenna model
set val(ifqlen)       50                         # max packet in ifq
set val(nn)           5                          # number of mobilenodes
set val(rp)           AODV                       # routing protocol
set val(x)            500                        # X dimension of topography
set val(y)            400                        # Y dimension of topography
set val(stop)         10                         # time of simulation end

set ns          [new Simulator]

#Creating nam and trace file

set tracefd     [open wireless1.tr w]
set namtrace    [open wireless1.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo        [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

```
# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace ON

    ## Creating node objects
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)  } {incr i } {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }

 ## Provide initial location of mobilenodes
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*500]
            set yy [expr rand()*400]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }
```

```
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

#Destination procedure:
$ns at 0.0 "destination"
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incr i} {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $ns at $now "$node_($i) setdest $xx $yy 10.0"
    }
    $ns at [expr $now+$time] "destination"
}

$ns at $val(stop) "stop"

#Stop procedure

proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program12.tcl

OUTPUT

6.13 PROGRAM 13 – Creation of nodes destination and random coloring using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is not static and is declared as eight in the network. Nodes are configured in the mobile wireless node format. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Movement for each and every node is built with static speed and spectified receiver address which is randomly generated and also the detination location will get change acceding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here each and every group of the nodes is constructed with different type of colors.

File Name – program13.tcl

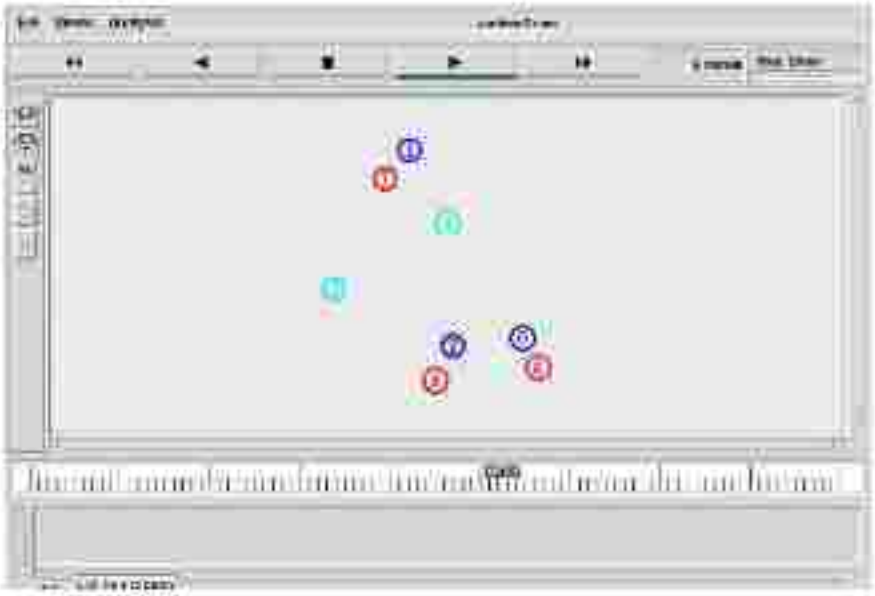
```
# Define setting options
set val(chan)         Channel/WirelessChannel    # channel type
set val(prop)         Propagation/TwoRayGround   # radio-propagation model
set val(netif)        Phy/WirelessPhy            # network interface type
set val(mac)          Mac/802_11                 # MAC type
set val(ifq)          Queue/DropTail/PriQueue    # interface queue type
set val(ll)           LL                         # link layer type
set val(ant)          Antenna/OmniAntenna        # antenna model
set val(ifqlen)       50                         # max packet in ifq
set val(nn)           3                          # number of mobilenodes
set val(rp)           AODV                       # routing protocol
set val(x)            500                        # X dimension of topography
set val(y)            400                        # Y dimension of topography
set val(stop)         10                         # time of simulation end

set ns            [new Simulator]

#Creating nam and trace file
set tracefd       [open wireless2.tr w]
set namtrace      [open wireless2.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo      [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# configure the nodes
        $ns node-config -adhocRouting $val(rp) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace ON
```

```
## Creating node objects...
for {set i 0} {$i < 3 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < 3 } {incr i } {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
    }
for {set i 3} {$i < 6 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 3} {$i < 6 } {incr i } {
        $node_($i) color cyan
        $ns at 0.0 "$node_($i) color cyan"
    }
for {set i 6} {$i <=8 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 6} {$i < 8 } {incr i } {
        $node_($i) color red
        $ns at 0.0 "$node_($i) color red"
    }

## Provide initial location of mobilenodes...
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*500]
            set yy [expr rand()*400]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at 5.0 "destination"
```

```
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incr i} {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $ns at $now "$node_($i) setdest $xx $yy 10.0"
    }
    $ns at [expr $now+$time] "destination"
}

#stop procedure
$ns at $val(stop) "stop"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless2.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program13.tcl

OUTPUT



6.14 PROGRAM 14 – Creation of nodes with the initial and destination position in random manner using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is not static and is declared as eight in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remams zero.

Movement for each and every node is built with static speed and spectifed receiver address which is randomly generated and also the detination location will get change accoding to the time period. Here initial size of each and every node is created by the use of the command (initial_node_pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here each and every group of the nodes is constructed with different type of colors.

File Name – program14.tcl

- Channel – Wireless Channel
- Propagation – Two Ray Ground Propagation
- Queue Type – Drop Tail
- Antenna – Omni Directional Antenna
- Routing Protocol –AODV
- Simulation Time – 18ms
- Number of nodes – 8 nodes
- X dimension – 500
- Y dimension – 400
- Initial Node Position - 30

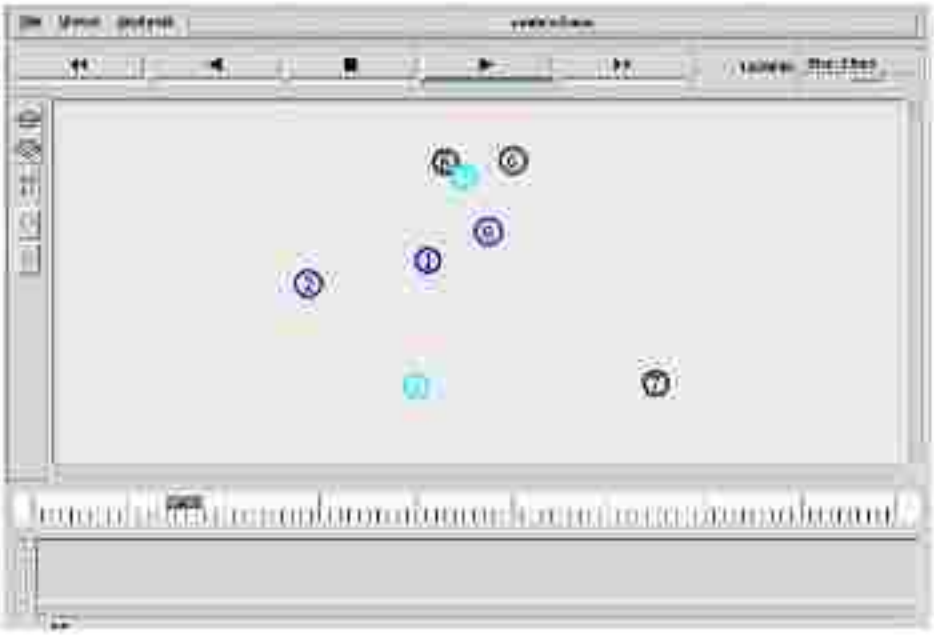
```
#Define options
set val(chan)         Channel/WirelessChannel          # channel type
set val(prop)         Propagation/TwoRayGround         # radio-propagation model
set val(netif)        Phy/WirelessPhy                  # network interface type
set val(mac)          Mac/802_11                       # MAC type
set val(ifq)          Queue/DropTail/PriQueue          # interface queue type
set val(ll)           LL                               # link layer type
set val(ant)          Antenna/OmniAntenna              # antenna model
set val(ifqlen)       50                               # max packet in ifq
set val(nn)           3                                # number of mobilenodes
set val(rp)           AODV                             # routing protocol
set val(x)            500                              # X dimension of topography
set val(y)            400                              # Y dimension of topography
set val(stop)         10                               # time of simulation end

#Creating simulation:
set ns          [new Simulator]

#Creating nam and trace file:
set tracefd     [open wireless3.tr w]
set namtrace    [open wireless3.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#set up topography object
set topo    [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

#configure the nodes
    $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
```

```
                -movementTrace ON

## Creating node objects...
for {set i 0} {$i < 3 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < 3  } {incr i } {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
    }
for {set i 3} {$i < 6 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 3} {$i < 6  } {incr i } {
        $node_($i) color cyan
        $ns at 1.0 "$node_($i) color cyan"
    }
for {set i 6} {$i < 8 } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 5} {$i < 8  } {incr i } {
        $node_($i) color red
        $ns at 2.0 "$node_($i) color red"
    }

## Provide initial location of mobilenodes...
        for {set i 0} {$i < $val(nn) } { incr i } {
            set xx [expr rand()*500]
            set yy [expr rand()*400]
            $node_($i) set X_ $xx
            $node_($i) set Y_ $yy

        }

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "$node_($i) reset";
}
```

```
# dynamic destination setting procedure
$ns at 0.0 "destination"
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incr i} {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $ns at $now "$node_($i) setdest $xx $yy 10.0"
    }
    $ns at [expr $now+$time] "destination"
}

#stop procedure
$ns at $val(stop) "stop"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless3.nam &
}

$ns run
```

Procedure to run the program in the terminal window - $ns program14.tcl

OUTPUT



6.15 PROGRAM 15 – Creation of graphs with X dimension and Y Dimension constructed randomly using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. Here the trace file acts as a input file to plot the graph in the format of trace file. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name – program15.tcl

- Channel – Wireless Channel
- Propagation – Two Ray Ground Propagation
- Queue Type – Drop Tail
- Antenna – Omni Directional Antenna
- Routing Protocol –AODV
- Simulation Time – 10ms
- Initial Node Position - 30

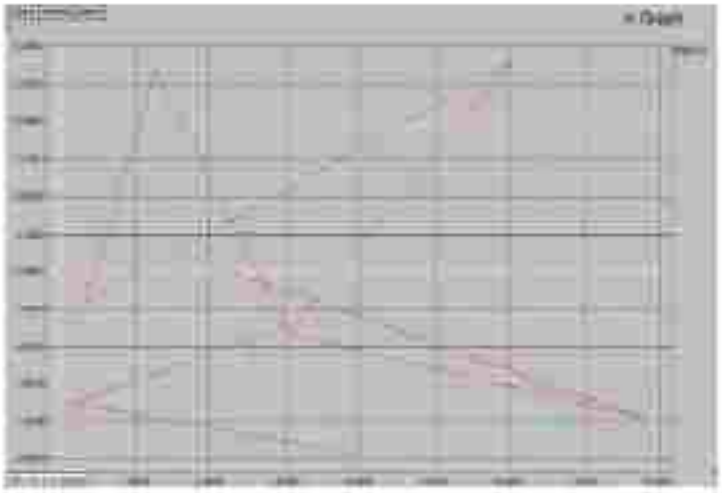
```
# Define setting options
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     3                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      500                        ;# X dimension of topography
set val(y)      400                        ;# Y dimension of topography
set val(stop)   10                         ;# time of simulation end

set ns          [new Simulator]
```

```
# Creating simulation
set ns [new Simulator]

#Creating nam and trace file
set traceId     [open Graph1.tr w]
set namtrace    [open Graph1.nam w]

$ns trace-all $traceId
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo    [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

# configure the nodes
    $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON

## Creating node objects
for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i } {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }

# Provide initial location of mobilenodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"
$ns at 1.0 "Graph"
set g [open graph.tr w]
proc Graph {} {
global ns g
set time 1.0
set now [$ns now]
puts $g "[expr rand()*6][expr rand()*6]"

$ns at [expr $now+$time] "Graph"
}

#Stop procedure
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec xgraph -M -bb -geometry 700X800 graph.tr &
exec nam Graph1.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program15.tcl

OUTPUT



6.16 PROGRAM 16 – Creation of graphs with two parameters as inputs using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. The trace file acts as input file to plot the graph in the format of trace file. Here single plotted graph consist of two trace file values. Different colors are given to each trace file during plotting. Routing protocol is AODV and the stop time of the simulation is 10ms.

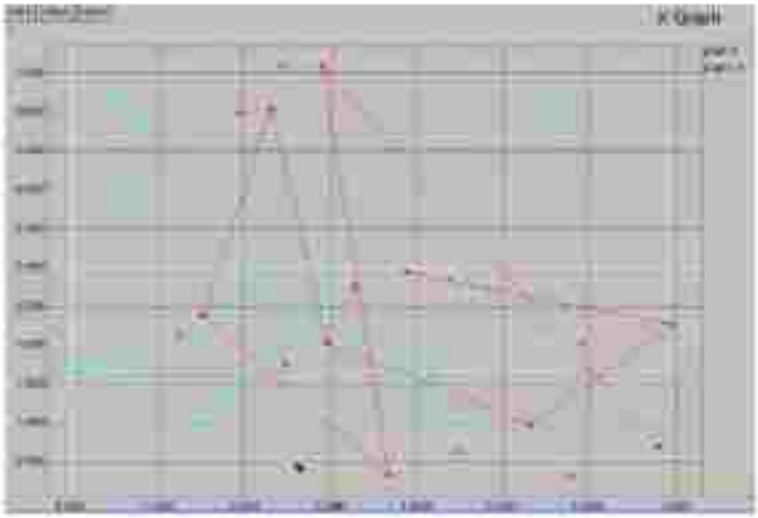
File Name – program16.tcl

```
# Define options
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     2                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      500                        ;# X dimension of topography
set val(y)      400                        ;# Y dimension of topography
set val(stop)   10                         ;# time of simulation end

set ns          [new Simulator]


# Creating simulation
set ns [new Simulator]

#Creating nam and trace file
set traceid     [open Graph2.tr w]
set namtrace    [open Graph2.nam w]

$ns trace-all $traceid
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo    [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

#configure the nodes
    $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON

## Creating node objects
for {set i 0} {$i < $val(nn) } {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

```
#Provide initial location of mobilenodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

#Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

#Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"

#Graph procedure.
#procedure.
$ns at 1.0 "Graph"
set g [open graph.tr w]
set g1 [open graph1.tr w]
proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*6] [expr rand()*6]"
puts $g1 "[expr rand()*8] [expr rand()*6]"
$ns at [expr $now+$time] "Graph"
}


#stop procedure.
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec xgraph -P -bb -geometry 700X800 graph.tr graph1.tr &

exec nam Graph2.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program16.tcl

OUTPUT



6.17 PROGRAM 17 – Creation of graphs with more than two parameter files as inputs using AODV routing protocol TCL script

Program Discription

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are always remains zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. The trace file acts as input file to plot the graph in the format of trace file. Here single plotted graph consist of more than two trace file values. Different colors are given to each trace file during plotting. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name – program17.tcl

```
# Define options
set val(chan)     Channel/WirelessChannel    ;# channel type
set val(prop)     Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)    Phy/WirelessPhy            ;# network interface type
set val(mac)      Mac/802_11                 ;# MAC type
set val(ifq)      Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)       LL                         ;# link layer type
set val(ant)      Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)   50                         ;# max packet in ifq
set val(nn)       3                          ;# number of mobilenodes
set val(rp)       AODV                       ;# routing protocol
set val(x)        500                        ;# X dimension of topography
set val(y)        400                        ;# Y dimension of topography
set val(stop)     10                         ;# time of simulation end
```

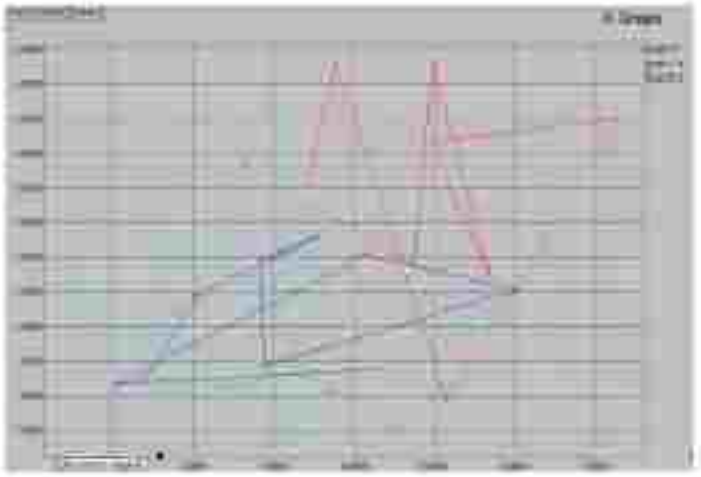
```
#Creating simulation
set ns [new Simulator]

#Creating nam and trace file
set tracefd     [open Graph3.tr w]
set namtrace    [open Graph3.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

#set up topography object
set topo    [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]


#configure the nodes
    $ns node-config -adhocRouting $val(rp) \
            -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON

#Creating node objects
for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn) } {incr i} {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

```
#Provide initial location of mobilenodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

#Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

#Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"

#Graph procedure.
$ns at 1.0 "Graph"
set g [open graph.tr w]
set g1 [open graph1.tr w]
set g2 [open graph2.tr w]
proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*6] [expr rand()*6]"
puts $g1 "[expr rand()*8] [expr rand()*6]"
puts $g2 "[expr rand()*8] [expr rand()*6]"
$ns at [expr $now+$time] "Graph"
}
```

```
#Stop procedure
proc stop {} {
    global ns tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec xgraph -M -bb -geometry 700X800 graph.tr graph1.tr graph2.tr &
exec nam Graph3.nam &
exit 0
}

$ns run
```

Procedure to run the program in the terminal window - $ns program17.tcl

OUTPUT



SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

HANDS ON TRAINING IN NS2

Date: From 25-11-2021 To 31-12-21 (6 Week Course, Only on Saturdays)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
1	19D41A05C1	MALOTH AKHIL	Akhil	MalothAkhil	Maloth Akhil	Akhil	Maloth Akhil	MalothAkhil
2	19D41A05C2	MAMIDALA POOJITHA	Poojitha	Poojitha	Poojitha	Poojitha	Poojitha	Poojithar
3	19D41A05C3	MANDADI HRSHITHA REDDY	Harshitha	Harshitha	Harshitha	Harshitha	Harshitha	Harshitha
4	19D41A05C4	MANGILIGAY VARUN KUMAR	Varun	Varun	Varun	Varun	Varun	Varun
5	19D41A05C5	MARAM HARSHITH REDDY	Harshith	Harshith	Harshith	Harshith	Harshith	Harshith
6	19D41A05C6	MARRIPELLY SHIVA	Shiva	Shiva	Shiva	Shiva	Shiva	Shiva
7	19D41A05C7	MATTIPA JAHNAWI	Jahnawi	Jahnawi	Jahnawi	Jahnawi	Jahnawi	Jahnawi
8	19D41A05C8	MEDISHETTI SAKETH	Saketh	Saketh	Saketh	Saketh	Saketh	Saketh
9	19D41A05C9	MEGHAVATH TEJASREE	Sree	Sree	Sree	Sree	Sree	Sree
10	19D41A05D0	MERAVATH SREENU	Sreenu	Sreenu	Sreenu	Sreenu	Sreenu	Sreenu
11	19D41A05D1	MITTAPALLI SHIVA GANESH	Ganesh	Ganesh	Ganesh	Ganesh	Ganesh	Ganesh
12	19D41A05D2	MODINIPALLY PRAVEENA KUMARI	Praveena	Praveena	Praveena	Praveena	Praveena	Praveena
13	19D41A05D3	MOGILI SRINIVAS	Srinivas	Srinivas	Srinivas	Srinivas	Srinivas	Srinivas
14	19D41A05D4	MOHAMMAD RAHEEM	Raheem	Raheem	Raheem	Raheem	Raheem	Raheem
15	19D41A05D5	MOHAMMAD SHEEMA FIRDOUS	Sheema	Sheema	Sheema	Sheema	Sheema	Sheema
16	19D41A05D6	MOHAMMED BUSHRA	BUSHRA	BUSHRA	BUSHRA	BUSHRA	BUSHRA	BUSHRA

		KHANAM						
17	19D41A05D7	MOHAMMED SAMEER	Sameer	Sameer	Sameer	Sameer	Sameer	Sameer
18	19D41A05D8	MOHD SAMI AHMED						
19	19D41A05D9	MOHD ZEENATH						
20	19D41A05E0	MOTHE HARI CHANDRA PRASAD						
21	19D41A05E1	MUKKA SAI KIRAN						
22	19D41A05E2	MULUGURI VAMSHI	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi
23	19D41A05E3	MURGANI VARUN	Varun	Varun	Varun	Varun	Varun	Varun
24	19D41A05E4	MUSKU SAMSRITHA						
25	19D41A05E5	MUTHYALA DINESH CHANDRA						

Coordinator

Convener

HOD/CSE

Head of the Department
Department of CSE
Sri Indu College Of Engineering & Technology
Sheriguda, Ibrahimpatnam, R.R. Dist.

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
NAAC Approved by AICTE &
Permanently Affiliated to JNTUH

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE ON IMAGE RETRIEVAL PROCESS USING MATLAB

STARTS ON February 13, 2021

Registration : Free

Course Duration : 4 Week

Weekend Course (Saturday)

Invited Participants: Third Year ECE, EEE, CSE

Restricted to 30 Participants/Slot

Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Dr.P.Epsiba	Prof.k.Ashok Babu	Dr.G.Suresh

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE

ON

IMAGE RETRIEVAL PROCESS USING MATLAB

Date: From 13.02.2021 (4 Week Course, Only on Saturdays)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4
1	16D41A04C5	Madduri Venkata Sameer Kumar				
2	16D41A04P5	Madhavaram Sriram				
3	16D41A0447	D Anil Kumar				
4	16D41A0492	Koppula Rajitha				
5	16D41A04D3	Manda Sushma				
6	16D41A04C1	Kunchala Venkatesh				
7	16D41A04C5	Madduri Venkata Sameer Kumar				
8	16D41A04M2	V.Lasya				
9	16D41A04G1	Nimisha Reddy				
10	16D41A04M2	Tatavarthy Satyadatha Praneeth				
11	16D41A0439	C. Sai Hiranmayi				
12	16D41A04C5	Vinisha				
13	16D41A0468	Bharadwaja Enumula				
14	16D41A0434	Bommidi Gayathri				
15	16D41A0474	Gella Harini				
16	16D41A04N5	Yerra Puneeshwar				

17	18D41A04M1	Tangella Mallam Raju	Mallam Raju	Mallam Raju	Mallam Raju	Mallam Raju
18	18D41A04H7	R.Pranathi	P. Pranathi	P. Pranathi	P. Pranathi	P. Pranathi
19	19D41A04D5	Aeragiri nela Preethi	A. Preethi	A. Preethi	A. Preethi	A. Preethi
20	18D41A04N1	Velgapuri Sai Chander Rao				
21	18D45A0420	Salguti Nikitha	Nikitha	Nikitha	Nikitha	Nikitha
22	19D45A0406	Kolkulapally Bikku	K. Bikku	K. Bikku	← ^ →	K. Bikku
23	19D41A0408	Alleniki Shiva Ram	Shiva Ram	Shiva Ram	Shiva Ram	Shiva Ram
24	18D41A0469	Gunti Varun	Gunti Varun	Gunti Varun	Gunti Varun	Gunti Varun
25	18D41A0420	Bajjuri Bhavana Reddy		Bhavana Reddy	Bhavana Reddy	Bhavana Reddy
26	18D41A04G2	Paspolla Vikas	Vikas	Vikas	Vikas	Vikas
27	18D41A04G5	Padala Pramod	P. Pramod	P. Pramod	P. Pramod	P. Pramod
28	18D41A0406	Alugubelly.Tejaswini	Tejaswini	Tejaswini	Tejaswini	Tejaswini
29	18D45A0406	Ch. Mahesh	ch. Mahesh	ch. Mahesh	ch. Mahesh	ch. Mahesh
30	18D41A04E2	Mokthala Anil	Anil	Anil	Anil	Anil
31	18D41A0481	J Shiva Prasad				
			29	31	30	30

Coordinator

Convener

HOD/ECE

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

HANDS ON TRAINING COURSE ON IMAGE RETRIEVAL PROCESS USING MATLAB

STARTS ON February 13, 2021

Registration : Free
Course Duration : 4 Week
Weekend Course (Saturday)
Invited Participants: Third Year ECE, EEE, CSE
Restricted to 30 Participants/Slot
Resource Persons: In-house Trainers

Coordinators	Convener	Principal
Dr.P.Epsiba	Prof.k.Ashok Babu	Dr.G.Suresh

1. Foreground Background Extraction

```
clc;
close all;
clear;
%Read Background Image
Background=imread('background.jpg');%Background=rgb2gray(Background);
Background=imresize(Background,[160,160]);
%Read Current Frame
%x1=size(Background);
CurrentFrame=imread('original.jpg');%CurrentFrame=rgb2gray(CurrentFrame)
CurrentFrame=imresize(CurrentFrame,[160 160]);
%Display Background and Foreground
subplot(2,2,1);imshow(Background);title('BackGround');
subplot(2,2,2);imshow(CurrentFrame);title('Current Frame');
%Convert RGB 2 HSV Color conversion
[Background_hsv]=round(rgb2hsv(Background));
[CurrentFrame_hsv]=round(rgb2hsv(CurrentFrame));
Out = bitxor(Background_hsv,CurrentFrame_hsv);
%Convert RGB 2 GRAY
Out=rgb2gray(Out);
%Read Rows and Columns of the Image
[rows columns]=size(Out);
%Convert to Binary Image
for i=1:rows
for j=1:columns
if Out(i,j) >0
BinaryImage(i,j)=1;
else
BinaryImage(i,j)=0;
end
end
end
%Apply Median filter to remove Noise
FilteredImage=medfilt2(BinaryImage,[5 5]);
%Boundary Label the Filtered Image
[L num]=bwlabel(FilteredImage);
STATS=regionprops(L,'all');
cc=[];
removed=0;
%Remove the noisy regions
for i=1:num
dd=STATS(i).Area;
```

```
if (dd < 500)
L(L==i)=0;
removed = removed + 1;
num=num-1;
else
end
end
[L2 num2]=bwlabel(L);
% Trace region boundaries in a binary image.
[B,L,N,A] = bwboundaries(L2);
%Display results
subplot(2,2,3),  imshow(L2);title('BackGround Detected');
subplot(2,2,4),  imshow(L2);title('Blob Detected');
hold on;
for k=1:length(B),
if(~sum(A(k,:)))
boundary = B{k};
plot(boundary(:,2), boundary(:,1), 'r','LineWidth',2);
for l=find(A(:,k))'
boundary = B{l};
plot(boundary(:,2), boundary(:,1), 'g','LineWidth',2);
end
end
end
```

2. Round Object Detection

```
%RGB = imread('10.bmp');
%RGB=imread('E:\D Drive Files\03.11.2014 epsiba phd\Project Code\fusion of local global
estimation project\Codes\snaps\075.jpg');
imshow(RGB);
I = rgb2gray(RGB);
threshold = graythresh(I);
bw = im2bw(I,threshold);
imshow(bw)
% remove all object containing fewer than 30 pixels
bw = bwareaopen(bw,30);

% fill a gap in the pen's cap
se = strel('disk',2);
bw = imclose(bw,se);

% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
%bw = imfill(bw,'holes');
figure,
```

```
imshow(bw)
[B,L] = bwboundaries(bw,'noholes');
figure;
% Display the label matrix and draw each boundary
imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
for k = 1:length(B)
  boundary = B{k};
  plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
stats = regionprops(L,'Area','Centroid');

threshold = 0.94;

% loop over the boundaries
for k = 1:length(B)

  % obtain (X,Y) boundary coordinates corresponding to label 'k'
  boundary = B{k};

  % compute a simple estimate of the object's perimeter
  delta_sq = diff(boundary).^2;
  perimeter = sum(sqrt(sum(delta_sq,2)));

  % obtain the area calculation corresponding to label 'k'
  area = stats(k).Area;

  % compute the roundness metric
  metric = 4*pi*area/perimeter^2;

  % display the results
  metric_string = sprintf('%2.2f',metric);

  % mark objects above the threshold with a black circle
  if metric > threshold
    centroid = stats(k).Centroid;
    plot(centroid(1),centroid(2),'ko');
  end

  text(boundary(1,2)-35,boundary(1,1)+13,metric_string,'Color','y',...
       'FontSize',14,'FontWeight','bold');

end

title(['Metrics closer to 1 indicate that ',...
       'the object is approximately round']);
```

3. Edge object detection

```
clc;
clear all;
k=input('Enter the file name','s'); % input image; color image
im=imread(k);
im1=rgb2gray(im);
im1=medfilt2(im1,[3 3]); %Median filtering the image to remove noise%
BW = edge(im1,'sobel'); %finding edges
[imx,imy]=size(BW);
msk=[0 0 0 0 0;
    0 1 1 1 0;
    0 1 1 1 0;
    0 1 1 1 0;
    0 0 0 0 0;];
B=conv2(double(BW),double(msk)); %Smoothing  image to reduce the number of connected
components
L = bwlabel(B,8);% Calculating connected components
mx=max(max(L))
% There will be mx connected components.Here U can give a value between 1 and mx for L or
in a loop you can extract all connected components
% If you are using the attached car image, by giving 17,18,19,22,27,28 to L you can extract the
number plate completely.
[r,c] = find(L==17);
rc = [r c];
[sx sy]=size(rc);
n1=zeros(imx,imy);
for i=1:sx
   x1=rc(i,1);
   y1=rc(i,2);
   n1(x1,y1)=255;
end % Storing the extracted image in an array
figure,imshow(im);
figure,imshow(im1);
figure,imshow(B);
figure,imshow(n1,[]);
```

4. Content Based Image Retrieval

```
[filename, pathname] = uigetfile('*.bmp', 'Pick an image');
 a=imread(filename);
 figure(1),imshow(a);
 X1=a;
```

```
 [r c]=size(X1);
 a=X1(:,:,1);
 b=X1(:,:,2);
 c=X1(:,:,3);
 [r c]=size(a);
 M=r*c;
 N=reshape(a,[1 M]);
 N=double(N);
 p=[];
 for i=1:M
    p(i)=N(i).^M;
 end
 P=sum(sum(p));
 HSVmap1 = rgb2ycbcr(X1);
 figure(2),imshow(HSVmap1);
  fid = fopen('database.txt');

resultValues = [];     % Results matrix...
resultNames = {};
i = 1;                 % Indices...
j = 1;

while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break; end      % Meaning: End of File...

%    [X, RGBmap] = imread(imagename);
%    HSVmap = rgb2hsv(RGBmap);
    [X] = imread(imagename);
    figure(3),imshow(X);
    HSVmap = rgb2ycbcr(X);
    figure(4),imshow(HSVmap);


[D1,D2,D3] = quadratic1(X1,  HSVmap1, X, HSVmap);
   resultValues1(i) = D1;
   resultValues2(i) = D2;
   resultValues3(i) = D3;
   resultNames(j) = {imagename};
   i = i + 1;
   j = j + 1;
end

fclose(fid);
[sortedValues1, index1] = sort(resultValues1);    % Sorted results... the vector index
[sortedValues2, index2] = sort(resultValues2);
```

```
[sortedValues3, index3] = sort(resultValues3); % is used to find the resulting files

%----------------RED-----------------------------------------------

fid = fopen('colourResults_R_C.txt', 'w+');        % Create a file, over-write old ones

for i = 1:10                                       % Store top 10 matches

    tempstr = char(resultNames(index1(i)));
    fprintf(fid, '%s\r', tempstr);

    disp(resultNames(index1(i)));
    disp(sortedValues1(i));
    disp(' ');
end

fclose(fid);

%----------------GREEN---------------------------------------------

fid = fopen('colourResults_G_C.txt', 'w+');        % Create a file, over-write old ones

for i = 1:10                                       % Store top 10 matches
    tempstr = char(resultNames(index2(i)));
    fprintf(fid, '%s\r', tempstr);

    disp(resultNames(index2(i)));
    disp(sortedValues2(i));
    disp(' ');
end

fclose(fid);

%----------------BLUE----------------------------------------------

fid = fopen('colourResults_B_C.txt', 'w+');        % Create a file, over-write old ones

for i = 1:10                                       % Store top 10 matches
    tempstr = char(resultNames(index3(i)));
    fprintf(fid, '%s\r', tempstr);

    disp(resultNames(index3(i)));
    disp(sortedValues3(i));
    disp(' ');
end
```

```
fclose(fid);

%return;

disp('Colour part done...');
disp('Colour results saved...');
disp(' ');

% displayResults1('colourResultsR.txt', 'Colour Results_r...');
%  displayResults1('colourResultsG.txt', 'Colour Results_g...');
%  displayResults1('colourResultsB.txt', 'Colour Results_b...');
% displayResults1('textureResults_r.txt', 'Texture Results_r...');
% displayResults2('textureResults_g.txt', 'Texture Results_g...');

filename='colourResults_R_C.txt';

fid = fopen(filename);

i = 1;              % Subplot index on the figure...

while 1
    imagename = fgetl(fid);
    if ~ischar(imagename), break; end     % Meaning: End of File...

    [x, map] = imread(imagename);

%   subplot(4,5,i);
    if i==1,
    subplot(3,10,1);
%     figure()
    imshow(x);
    end

    if i==2

    subplot(3,10,2);
    imshow(x);
    end

    if i==3
    subplot(3,10,3);
    imshow(x);
    end

    if i==4
    subplot(3,10,4);
```

```
        imshow(x);
        end

        if i==5
        subplot(3,10,5);
        imshow(x);
        end

        if i==6
        subplot(3,10,6);
        imshow(x);
        end

        if i==7
        subplot(3,10,7);
        imshow(x);
        end

        if i==8
        subplot(3,10,8);
        imshow(x);
        end

        if i==9
        subplot(3,10,9);
        imshow(x);
        end

        if i==10
        subplot(3,10,10);
        imshow(x);
        end

        i = i - 1;

end

fclose(fid);

% displayResults1('textureResults_b.txt', 'Texture Results_b...');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

filename='colourResults_G_C.txt';

fid = fopen(filename);
```

```
i = 1;                  % Subplot index on the figure...

while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break; end      % Meaning: End of File...

   [x, map] = imread(imagename);

%    subplot(4,5,i);
if i==1;
subplot(3,10,11);
imshow(x);
end

if i==2

subplot(3,10,12);

 imshow(x);
end

if i==3
subplot(3,10,13);
 imshow(x);
end
if i==4

subplot(3,10,14);
 imshow(x);
end

if i==5

subplot(3,10,15);
 imshow(x);
end
if i==6

subplot(3,10,16);
 imshow(x);
end
if i==7
subplot(3,10,17);
imshow(x);
end
```

```
if i==8

subplot(3,10,18);
 imshow(x);
end
if i==9

subplot(3,10,19);
 imshow(x);
end

if i==10

subplot(3,10,20);
imshow(x);
end

%    subimage(x, map);
% imshow(x),
%     xlabel(imagename);

   i = i + 1;

end

fclose(fid);

% displayResults1('textureResults_b.txt', 'Texture Results_b...');
filename='colourResults_B_C.txt';

fid = fopen(filename);

i = 1;                  % Subplot index on the figure...

while 1
   imagename = fgetl(fid);
   if ~ischar(imagename), break, end       % Meaning: End of File...

   [x, map] = imread(imagename);

%    subplot(4,5,i);

if i==1;
subplot(3,10,21);
imshow(x);
end
```

```
if i==2
subplot(3,10,22);
imshow(x);
end

if i==3
subplot(3,10,23);
imshow(x);
end

if i==4
subplot(3,10,24);
imshow(x);
end

if i==5
subplot(3,10,25);
imshow(x);
end

if i==6
subplot(3,10,26);
imshow(x);
end

if i==7
subplot(3,10,27);
imshow(x);
end

if i==8
subplot(3,10,28);
imshow(x);
end

if i==9
subplot(3,10,29);
imshow(x);
end

if i==10
subplot(3,10,30);
imshow(x);
end
```

```
%    subimage(x, map);
% imshow(x);
%    xlabel(imagename);

   i = i - 1;

end

fclose(fid);
```

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

HANDS ON TRAINING COURSE

ON

ANGULAR JS PROGRAMMING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

ANGULAR JS PROGRAMING

Date: From 23.02.2022 To 31-03-2022 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	Topics	Resource Person
Week 1	Introduction to Angular JS programming	Mr.D.Prasanth
	(integer, float, Boolean, string, etc)	
	Angular JS Modules:	
	Functions & Modules	
	Angular Beginner Projects	
	Angular for Adding a Directive	
	Assignment-1	
Week 2	Modules and Controllers in Files	Mr.D.Prasanth
	myApp.js	
	myCtrl.js. Angular JS Controllers,	
	Assignment-2	
Week 3	Controller Methods	Mr.D.Prasanth
	Angular JS Forms	
	Data-Binding. Checkbox	
	Angular JS $http	

MODULE -2		
Durations	Topics	Resource Person
Week 4	Angular JS Example	Mr.D.Prasanth
	Server Code Examples	
	Server Code PHP	
	MySQL	
	Reading Relational Tables	
	Assignment-4	
MODULE -3		
Durations	Topics	Resource Person
Week 5	Creating a Chart	Mr.D.Prasanth
	Formatting Line type and Colour	
	Drawing a 3D Plot	
	Applying Aggregations on Data Frame	
Week 6	Input as CSV File	Mr.D.Prasanth
	Assessment -2	
	Conclusion	

Angular JS

Angular JS is a **JavaScript framework**. It can be added to an HTML pagewith a <script> tag. Angular JS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**. Angular JS is a JavaScript Framework Angular JS is a JavaScript framework. It is a library written in JavaScript.

Angular JS is distributed as a JavaScript file, and can be added to a web pagewith a script tag:

```
<script src="https://ajax.googleapis.com/ajax/libs/Angular JS/1.4.8/angular.min.js"></script>
```

Angular JS Extends HTML

Angular JS extends HTML with **ng-directives**.

The **ng-app** directive defines an Angular JS application.

The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

The **ng-bind** directive binds application data to the HTML view. Angular JSExample

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/Angular JS/1.4.8/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

Example explained:

Angular JS starts automatically when the web page has loaded.

The **ng-app** directive tells Angular JS that the element is the "owner" ofan Angular JS **application**.

The **ng-model** directive binds the value of the input field to the applicationvariable **name**.

The **ng-bind** directive binds the **inner HTML** of the <p> element to theapplication variable **name**.

Directives

Angular JS directives are HTML attributes with an **ng** prefix.

Angular JS Example

Angular JS controllers **control the data** of Angular JS applications Angular JS controllers are regular **JavaScript Objects**.

Angular JS Modules

An Angular JS module defines an application.

The module is a container for the different parts of an application The module is a container for the application controllers.

Controllers always belong to a modul

Creating a Module

A module is created by using the Angular JS function angular.module

```
<div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);

</script>
```

The "myApp" parameter refers to an HTML element in which the application willrun.

Now you can add controllers, directives, filters, and more, to your Angular JSapplication.

Adding a Controller

Add a controller to your application, and refer to the controller with the ng-controller directive.

Example

```
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>

var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});

</script>
```

Adding a Directive

Angular JS has a set of built-in directives which you can use to add functionalityto your application.

For a full reference, visit our Angular JS directive reference.

In addition you can use the module to add your own directives to yourapplications:

Example

```
<div ng-app="myApp" w3-test-directive></div>

<script>
var app = angular.module("myApp", []);

app.directive("w3TestDirective", function() {return {
        template : "I was made in a directive constructor!"
    };
});
```

```
</script>
```

Modules and Controllers in Files

It is common in Angular JS applications to put the module and the controllers inJavaScript files.

In this example, "myApp.js" contains an application module definition, while"myCtrl.js" contains the controller:

Example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.4.8/angular.min.js"></script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
```

```
</div>
<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

myApp.js

```
var app = angular.module("myApp", []);
```

The [] parameter in the module definition can be used to define dependentmodules.

Without the [] parameter, you are not *creating* a new module, but *retrieving* anexisting one.

myCtrl.js

```
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName= "Doe";
});
```

Functions can Pollute the Global Namespace

Global functions should be avoided in JavaScript. They can easily be overwrittenor destroyed by other scripts.

Angular JS modules reduces this problem, by keeping all functions local to themodule.

When to Load the Library

While it is common in HTML applications to place scripts at the end of the <body>element, it is recommended that you load the Angular JS library either inthe <head>or at the start of the <body>

This is because calls to angular.modulecan only be compiled after the libraryhas been loaded.

Example

```
<!DOCTYPE html>
<html>
<body>
<script src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.4.8/angular.min
.js"></script>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>

</body>
</html>
```

Angular JS Controllers

Angular JS applications are controlled by controllers.

The **ng-controller** directive defines the application controller.

A controller is a **JavaScript Object**, created by a standard JavaScript **objectconstructor**.

Angular JS Example

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>Last
Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>
```

Application

The Angular JS application is defined by **ng-app="myApp"**. The applicationruns inside the .

The **ng-controller="myCtrl"** attribute is an Angular JS directive. It defines acontroller.

The **myCtrl** function is a JavaScript function.

Angular JS will invoke the controller with a **$scope** object.

In Angular JS, $scope is the application object (the owner of applicationvariables and functions).

The controller creates two properties (variables) in the scope (**firstName** and **lastName**).

The **ng-model** directives bind the input fields to the controller properties(firstName and lastName).

Controller Methods

The example above demonstrated a controller object with two properties:lastName and firstName.

A controller can also have methods (variables as functions).

Angular JS Example

```
<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>Last
Name: <input type="text" ng-model="lastName"><br>
<br>
```

```
Full Name: {{fullName()}}

</div>

<script>
var app = angular.module('myApp', []); app.controller('personCtrl',
function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    };
});
</script>
```

Controllers In External Files

In larger applications, it is common to store controllers in external files.

Just copy the code between the <script> tags into an external file named personController.js:

Angular JS Example

```
<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>Last
Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>

<script src="personController.js"></script>
```

Another Example

For the next example we will create a new controller file:

```
angular.module('myApp', []).controller('namesCtrl', function($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Hege',country:'Sweden'},
        {name:'Kai',country:'Denmark'}
    ];
});
```

Save the file as namesController.js:

And then use the controller file in an application:

Angular JS Example

```
<div ng-app="myApp" ng-controller="namesCtrl">

<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>

<script src="namesController.js"></script>
```

Angular JS Forms

Forms in Angular JS provides data-binding and validation of input controls.

Input Controls

Input controls are the HTML input elements:

- input elements
- select elements
- button elements
- textarea elements

Data-Binding

Input controls provides data-binding by using the ng-modeldirective.

```
<input type="text" ng-model="firstname">
```

The application does now have a property named firstname.

The ng-modeldirective binds the input controller to the rest of your application.The property firstname, can be referred to in a controller:

Example

```
<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.firstname = "John";
});
</script>
```

It can also be referred to elsewhere in the application:

Example

```
<form>
First Name: <input type="text" ng-model="firstname">
</form>

<h1>You entered: {{firstname}}</h1>
```

Checkbox

A checkbox has the value trueor false. Apply the ng-modeldirective to acheckbox, and use its value in your application.

Example

Show the header if the checkbox is checked:

```
<form>
    Check to show a header:
    <input type="checkbox" ng-model="myVar">
```

```
<form>

<h1 ng-show="myVar">My Header<h1>
```

Radiobuttons

Bind radio buttons to your application with the ng-modeldirective.

Radio buttons with the same ng-modelcan have different values, but only theselected one will be used.

Example

Display some text, based on the value of the selected radio button:

```
 <form>
Pick a topic:
<input type="radio" ng-model="myVar" value="dogs">Dogs
<input type="radio" ng-model="myVar" value="tuts">Tutorials
<input type="radio" ng-model="myVar" value="cars">Cars
</form>
```

The value of myVar will be either dogs, tuts, or cars.

Selectbox

Bind select boxes to your application with the ng-modeldirective.

The property defined in the ng-modelattribute will have the value of theselected option in the selectbox.

Example

Display some text, based on the value of the selected option <form>
Select a topic:
<select ng-model="myVar">
<option value="">
<option value="dogs">Dogs
<option value="tuts">Tutorials
<option value="cars">Cars
</select>
</form>

The value of myVar will be either dogs, tuts, or cars.

An Angular JS Form Example

First Name:

Last Name:

RESET

form = {"firstName":"John","lastName":"Doe"} master = {"firstName":"John","lastName":"Doe"}

Application Code

```
<div ng-app="myApp" ng-controller="formCtrl">
  <form novalidate>
    First Name:<br>
    <input type="text" ng-model="user.firstName"><br>Last Name:<br>
    <input type="text" ng-model="user.lastName">
    <br><br>
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user}}</p>
```

```
<p>master = {{master}}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.master = {firstName: "John", lastName: "Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
});
</script>
```

The **novalidate** attribute is new in HTML5. It disables any default browservalidation.

Example Explained

The **ng-app** directive defines the Angular JS application.

The **ng-controller** directive defines the application controller.

The **ng-model** directive binds two input elements to the **user** object in themodel.

The **formCtrl** controller sets initial values to the **master** object, and defines the **reset()** method.

The **reset()** method sets the **user** object equal to the **master** object.

The **ng-click** directive invokes the **reset()** method, only if the button is clicked.

The novalidate attribute is not needed for this application, but normally you willuse it in Angular JS forms, to override standard HTML5 validation.

Angular JS AJAX - $http

http://www.w3schools.com/angular/angular_http.asp

$http is an Angular JS service for reading data from remote servers.

Angular JS $http

The Angular JS $http service makes a request to the server, and returns a response.

Example

Make a simple request to the server, and display the result in a header:

```
<div ng-app="myApp" ng-controller="myCtrl">

<p>Today's welcome message is:</p>
```

```
<h1>{{myWelcome}}</h1>

</div>

<script>

var app = angular.module('myApp', []); app.controller('myCtrl',
function($scope, $http) {
    $http.get("welcome.htm")
    .then(function(response) {
        $scope.myWelcome = response.data;
    });
});

</script>
```

Methods

The example above uses the .getmethod of the $httpservice.

The .get method is a shortcut method of the $http service. There are severalshortcut methods:

- .delete()
- .get()
- .head()
- .jsonp()
- .patch()
- .post()
- .put()

The methods above are all shortcuts of calling the $http service.

Example

```
var app = angular.module('myApp', []); app.controller('myCtrl',
function($scope, $http) {
    $http({
        method : "GET",
        url : "welcome.htm"
    }).then(function mySucces(response) {
```

```
        $scope.myWelcome = response.data;
    }, function myError(response) {
        $scope.myWelcome = response.statusText;
    });
});
```

The example above executes the $http service with an object as an argument.The object is specifying the HTTP method, the url, what to do on success, andwhat to do on failure.

Properties

The response from the server is an object with these properties:

- .configthe object used to generate the request.
- .data a string, or an object, carrying the response from the server.
- .headersa function to use to get header information.
- .statusa number defining the HTTP status.
- .statusTexta string defining the HTTP status.

Example

```
var app = angular.module('myApp', []); app.controller('myCtrl',
function($scope, $http) {
    $http.get("welcome.htm")
    .then(function(response) {
        $scope. Content = response.data;
        $scope.statuscode = response.status;
        $scope.statustext = response.statustext;
    });
});
```

To handle errors, add one more functions to the .thenmethod:

Example

```
var app = angular.module('myApp', []); app.controller('myCtrl',
function($scope, $http) {
    $http.get("wrongfilename.htm")
```

```
    .then(function(response) {
        // First function handles success
        $scope.content = response.data;
    }, function(response) {
        // Second function handles error
        $scope.content = "Something went wrong";
    });
});
```

JSON

The data you get from the response is expected to be in JSON format.

JSON is a great way of transporting data, and it is easy to use within Angular JS,or any other JavaScript.

Example: On the server we have a file that returns a JSON object containing 15customers, all wrapped in array called records.

Take a look at the JSON object:
customers.php

```
{
    "records": [
        {
            "Name": "Alfreds Futterkiste","City":
            "Berlin",
            "Country": "Germany"
        },
        {
            "Name": "Ana Trujillo Emparedados y helados","City":
            "México D.F.",
            "Country": "Mexico"
        },
        {
            "Name": "Antonio Moreno Taquería","City":
            "México D.F.",
            "Country": "Mexico"
```

```
},
{
    "Name": "Around the Horn","City":
    "London",
    "Country": "UK"
},
{
    "Name": "B's Beverages",
    "City": "London",
    "Country": "UK"
},
{
    "Name": "Berglunds snabbköp","City":
    "Luleå",
    "Country": "Sweden"
},
{
    "Name": "Blauer See Delikatessen","City":
    "Mannheim",
    "Country": "Germany"
},
{
    "Name": "Blondel père et fils","City":
    "Strasbourg", "Country": "France"
},
{
    "Name": "Bólido Comidas preparadas","City":
    "Madrid",
    "Country": "Spain"
},
{
    "Name": "Bon app'",
    "City": "Marseille",
    "Country": "France"
},
{
    "Name": "Bottom-Dollar Marketse","City":
    "Tsawassen",
    "Country": "Canada"
},
{
    "Name": "Cactus Comidas para llevar","City":
    "Buenos Aires",
    "Country": "Argentina"
},
{
    "Name": "Centro comercial Moctezuma","City":
    "Mexico D.F.",
```

```
        "Country": "Mexico"
    },
    {
        "Name": "Chop-suey Chinese","City":
        "Bern",
        "Country": "Switzerland"
    },
    {
        "Name": "Comércio Mineiro","City":
        "São Paulo", "Country": "Brazil"
    }
  ]
}
```

Example

The ng-repeat directive is perfect for looping through an array:

```
<div ng-app="myApp" ng-controller="customersCtrl">

<ul>
  <li ng-repeat="x in myData">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>

</div>

<script>

var app = angular.module('myApp', []); app.controller('customersCtrl',
function($scope, $http) {
    $http.get("customers.php").then(function(response) {
        $scope.myData = response.data.records;
    });
});

</script>
```

Application Explained:

The application defines the customersCtrlcontroller, with a $scopeand $http object.

$httpis an **XMLHttpRequest object** for requesting external data

$http.get() reads **JSON data** from http://www.w3schools.com/angular/customers.php.

On success, the controller creates a property, myData, in the scope, with JSONdata from the server.

Fetching Data From a PHP Server Running
MySQL

Angular JS Example

```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []); app.controller('customersCtrl',
function($scope, $http) {
    $http.get("http://www.w3schools.com/angular/customers_mysql.php")
    .then(function (response) {$scope.names = response.data.records;});
});
</script>
```

Fetching Data From an ASP.NET ServerRunning SQL

Angular JS Example

```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
```

```
<script>
var app = angular.module('myApp', []); app.controller('customersCtrl',
function($scope, $http) {
    $http.get("http://www.w3schools.com/angular/customers_sql.aspx")
    .then(function (response) {$scope.names = response.data.records;});
});
<script>
```

Server Code Examples

The following section is a listing of the server code used to fetch SQL data.

1. Using PHP and MySQL. Returning JSON.
2. Using PHP and MS Access. Returning JSON.
3. Using ASP.NET, VB, and MS Access. Returning JSON.
4. Using ASP.NET, Razor, and SQL Lite. Returning JSON.
5. **Cross-Site HTTP Requests**

Requests for data from a different server (than the requesting page), are called **cross-site** HTTP requests.

Cross-site requests are common on the web. Many pages load CSS, images,and scripts from different servers.

In modern browsers, cross-site HTTP requests **from scripts** are restricted to **same site** for security reasons.

The following line, in our PHP examples, has been added to allow cross-siteaccess.

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Origin: *");
```

Server Code PHP and MySQL

```
<?php
header("Access-Control-Allow-Origin: *"); header("Content-Type:
application/json; charset=UTF-8");

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");

$result = $conn->query("SELECT CompanyName, City, Country FROM Customers");

$outp = "";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {if
    ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'                  . $rs["City"]                  . '",';
    $outp .= '"Country":"'. $rs["Country"]                  . '"}';
}
$outp = ('{"records":[' .$outp .']}');
$conn->close();

echo($outp);
?>
```

Server Code PHP and MS Access

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=ISO-8859-1");

$conn = new COM("ADODB.Connection");
$conn->open("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=Northwind.mdb");

$rs = $conn->execute("SELECT CompanyName, City, Country FROM Customers");

$outp = "";
while (!$rs->EOF) {
    if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'                  . $rs["City"]                  . '",';
    $outp .= '"Country":"'. $rs["Country"]                  . '"}';
    $rs->MoveNext();
}
$outp ='{"records":[' .$outp .']
```

```
$conn->close();

  echo ($outp);
  ?>
```

Server Code ASP.NET, VB and MS Access

```
  <%@ Import Namespace="System.IO"%>
  <%@ Import Namespace="System.Data"%>
  <%@ Import Namespace="System.Data.OleDb"%>
  <%
  Response.AppendHeader("Access-Control-Allow-Origin",          "*")
  Response.AppendHeader("Content-type",  "application/json")Dim conn
  As OleDbConnection
  Dim objAdapter As OleDbDataAdapter
  Dim objTable As DataTable
  Dim objRow As DataRow
  Dim objDataSet As New DataSet()
  Dim outp
  Dim c
  conn = New OledbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data
  source=Northwind.mdb")
  objAdapter = New OledbDataAdapter("SELECT CompanyName, City, Country FROM
  Customers", conn)
  objAdapter.Fill(objDataSet, "myTable")
  objTable=objDataSet.Tables("myTable")

  outp  =  ""c
  = chr(34)
  for each x in objTable.Rows
  if outp <> "" then outp = outp & ","
  outp = outp & "{" & c & "Name" & c & ":" & c & x("CompanyName") & c & ","outp = outp
  &     c & "City"    & c & ":" & c & x("City")      & c & ","outp = outp &    c & "Country" & c
  & ":" & c & x("Country")    & c & "}"next

  outp ="{" & c & "records" & c & ":[" & outp & "]}"
  response.write(outp)
  conn.close

  %>
```

Server Code ASP.NET, Razor C# and SQL Lite

```
@{
Response.AppendHeader("Access-Control-Allow-Origin",            "*")
Response.AppendHeader("Content-type",  "application/json") var  db =
Database.Open("Northwind");
var query = db.Query("SELECT CompanyName, City, Country FROM Customers");var
outp =""
var c = chr(34)
}
@foreach(var row in query)
{
if outp <> "" then outp = outp + ","
outp = outp + "{" + c + "Name" + c + ":" + c + @row.CompanyName + c + "," outp = outp +
c + "City"   + c + ":" + c + @row.City      + c + "," outp = outp +   c + "Country"  + c + ":" + c +
@row.Country   + c + "}"
}
outp = "{" + c + "records" + c + ":[" + outp + "]}" @outp
```

RAJINDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

ANGULAR JS PROGRAMING

Date: From 23-02-2022 To 31-03-2022(6 Week Course, Only on Saturdays)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
1	19D41A0501	ADDAP VAMSHI	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi	Vamshi
2	19D41A0502	AEDULLA BHARATH CHANDRA	Bharath	Bharath	Bharath	Bharath	Bharath	Bharath
3	19D41A0503	AGIRISHETTY ANUSHA	A. Anusha	A. Anusha	A. Anusha	A. Anusha	A. Anusha	A. Anusha
4	19D41A0504	AITAGONI KAVYA	A. Kavya	A. Kavya	A. Kavya	A. Kavya	A. Kavya	A. Kavya
5	19D41A0505	AKULA NIHARIKA	Niharika	Niharika	Niharika	Niharika	Niharika	Niharika
6	19D41A0506	ALAMPALLY SURYA PRAKASH REDDY	Surya	Surya	Surya	Surya	Surya	Surya
7	19D41A0507	AMBATI NAGARAJU	Nagaraju	Nagaraju	Nagaraju	Nagaraju	Nagaraju	Nagaraju
8	19D41A0508	AMIRISHETTY SUSHMITHA						
9	19D41A0509	ANKAM MEGHANA						
10	19D41A0510	ANNIGULA PAVAN KUMAR	Pavan	Pavan	Pavan	Pavan	Pavan	Pavan
11	19D41A0511	ANTHATI MADHUSHALINI	Madhu	Madhu	Madhu	Madhu	Madhu	Madhu
12	19D41A0512	ARIGELA SRIKAR	Srikar	Srikar	Srikar	Srikar	Srikar	Srikar
13	19D41A0513	ARPULA NAVANEETHA	A. Navaneetha	A. Navaneetha	A. Navaneetha	A. Navaneetha	A. Navaneetha	A. Navaneetha
14	19D41A0514	ASULA RAJITHA	A. Rajitha	A. Rajitha	A. Rajitha	A. Rajitha	A. Rajitha	A. Rajitha

16	19D41A0516	AVVARI SRIRAM	A.Sriram	A.Sriram	A.Sriram	A.Sriram	A.Sriram	A.Sriram
17	19D41A0517	BAKKAVAMANA SHARON RANI	A.Sharon	A.Sharon	A.Sharon	A.sharon	A.sharon	A.sharon
18	19D41A0518	BALAYYAGARI NAGESH	A.Nagesh	A.Nagesh	A.Nagesh	A.Nagesh	A.Nagesh	A.Nagesh
19	19D41A0519	BANDARI AVINASH	Avinash	Avinash	Avinash	Avinash	Avinash	Avinash
20	19D41A0520	BANDLA PRANAY KUMAR	Pranay	Pranay	Pranay	Pranay	Pranay	Pranay
21	19D41A0521	BANGARI MADHURI	Madhuri	Madhuri	Madhuri	Madhuri	Madhuri	Madhuri
22	19D41A0522	BANOTH TARUN	Tarun	Tarun	Tarun	Tarun	Tarun	Tarun
23	19D41A0523	BATHULA SIVA BHARGAV REDDY	Siva	Siva	Siva	Siva	Siva	Siva
24	19D41A0524	BATTU RANI	Rani	Rani	Rani	Rani	Rani	Rani
25	19D41A0525	BHUTHPUR SAI KIRAN REDDY	Saikiran	Saikiran	Saikiran	Saikiran	Saikiran	Saikiran

Coordinator

Convener

HOD/CSE

Head of the Department
Department of CSE
Sri Indu College Of Engineering & Technology
Sheriguda, Ibrahimpatnam, R.R. Dist.

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

HANDS ON TRAINING COURSE

ON

BASICS OF PYTHON PROGRAMMING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
HANDS ON TRAINING COURSE
ON
BASICS OF PYTHON PROGRAMMING

Date: From 05.01.2022 To 18-02-2022 (6 Week Course, Only on Saturdays)

COURSE CONTENTS

MODULE -1		
Durations	Topics	Resource Person
Week 1	Basics of python programming	Mr.K.Raju
	What is python programming?	
	Installation and Execution	
	Assignment -1	
Week 2	Applications of python programming	Mr.K.Raju
	An Introduction to Python	
	Installing Python	
	Python – Understanding Data with Visualization	
	Peppering Data	
Week 3	Python 2.x or Python 3.x ?	Mr.K.Raju
	Python interactive:	
	using Python as a calculator	
	Using variables	
MODULE -2		
Durations	Topics	Resource Person
Week 4	Mathematical functions	Mr.K.Raju
	Python scripts (programs)	
	Variables and objects	
	User input, Iterating with indexing	
	Assignment 3	
MODULE -3		
Durations	Topics	Resource Person
Week 5	Understanding Data with Statistics	Mr.K.Raju
	Methods for python programming	
	when to use closures?	

Basics of python programming

1. Python installation

On Linux systems, Python 2.x is already installed. To download Python for Windows and OSx, and for documentation see http://python.org. It might be a good idea to install the Bethought distribution Canopy that contains already the very useful modules Numpy, Scipy and Matplotlib: https://www.enthought.com/downloads/

2. Python 2.x or Python 3.x ?

The current version is 3.x. Some libraries may not yet be available for version 3, and Linux Ubuntu comes with 2.x as astandard. Many improvements from 3 have been back ported to 2.7. The main differences for basic programming are in the print and input function. We will use Python 2.x in this tutorial.

3. Python interactive: using Python as a calculator

Start Python (or IDLE, the Python IDE). A
prompt is showing up:
>>>Display version:>>>help()
Welcome to Python 2.7! This is the online help utility help>

Help commands:

modules:	available modules
keywords:	list of reserved Python keywords
quit:	leave help

To get help on a keyword, just enter it's name in help.

Simple calculations in Python

```
>>> 3.14*5
15.700000000000001
```

Supported operators:

Operator		Example	Explication
+, - *, /	add, substract, multiply, divide		
%	modulo	25 % 5 = 0 84 % 5 = 4	25/5 = 5, remainder = 0 84/5 = 16, remainder = 4
**	exponent	2**10 = 1024	
//	floor division	84//5 = 16	84/5 = 16, remainder = 4

Take care in Python 2.x if you divide two numbers.
Isn't this strange:

```
>>> 35/65
```

Obviously the result is wrong!
But:

```
>>> 35.0/6
5.833333333333333
>>> 35/6.0
5.833333333333333
```

In the first example, 35 and 6 are interpreted as integer numbers, so integer division is used and the result is an integer.
This uncanny behavior has been abolished in Python 3, where 35/6 gives 5.833333333333333.

In Python 2.x, use floating point numbers (like 3.14, 3.0 etc...) to force floating point division!

Another workaround would be to import the Python 3 like division at the beginning:

```
>>> from __future__ import division
>>> 3/4
0.75
```

Builtin functions:

```
>>> hex(1024)
'0x400'

>>> bin(1024)
'0b10000000000'
```

Expressions:

```
>>> (20.0+4)/64
>>> (2+3)*525
```

4. Using variables

To simplify calculations, values can be stored in variables, and and these can be used as in normal mathematics.

```
>>> a=2.0
>>> b = 3.36
>>> a+b
5.359999999999999
>>> a-b
-1.3599999999999999
>>> a**2 = b**2 15.289599999999998
>>> a>b
False
```

The name of a variable must not be a Python keyword!

Keywords are:

and	elif	if	print
as	else	import	raise
assert	except	in	return
break	exec	is	try
class	finally	lambda	while
continue	for	not	with
def	from	or	yield
del	global	pass	

5. Mathematical functions

Mathematical functions like square root, sine, cosine and constants like pi etc. are available in Python. To use them it is necessary to import them from the math module:

```
>>> from math import *
>>> sqrt(2)
1.4142135623730951
```

Note
There is more than one way to import functions from modules. Our simple method imports all functions available in the math module. For more details see appendix.

Other examples using math:

Calculate the perimeter of a circle

```
>>> from math import *
>>> diameter = 5
>>> perimeter = 2 * pi * diameter
>>> perimeter
31.415926535897931
```

Calculate the amplitude of a sine wave:

```
>>> from math import *
>>> Ueff = 230
>>> amplitude = Ueff * sqrt(2)
>>> amplitude
325.26911934581119
```

6. Python scripts (programs)

If you have to do more than a small calculation, it is better to write a script (a program in Python).

This can be done in IDLE, the Python editor.
A good choice is also Geany, a small freeware editor with symax colouring, from which you can directly start your script.

To write and run a program in IDLE:

- Menu File – New Window
- Write script
- File – Save (name with extension .py, for example myprogram.py)
- Run program: <F5> or Menu Run – Run Module

Take care:

- In Python white spaces are important!
  The indeniation of a source code is important!
  A program that is not correctly indented shows either errors or does not what you want!

- Python is case sensitive!
  For example x and X are two different variables.

7. A simple program

This small program calculates the area of a circle:

```
from math import *
d = 10.0                    # diameter
A = pi * d**2 / 4
print "diameter =", d
print "area = ", A
```

Note: everything behind a "#" is a comment.
Comments are important for others to understand what the program does (and for yourself if you look at your program a long time after you wrote it).

8. User input

In the above program the diameter is hard coded in the program.
If the program is started from IDLE or an editor like Geany, this is not really a problem, as it is easy to edit the value if necessary.

In a bigger program this method is not very practical.

This little program in Python 2.7 asks the user for his name and greets him:

```
s = raw_input("What is your name?")
print "HELLO ", s
```

```
What is your name?Tom
HELLO  Tom
```

Take care:
The raw_input function gives back a string, that means a list of characters. If the input will be used as a number, it must be converted.

9. Variables and objects

In Python, values are stored in objects.
If we do
d = 10.0
a new object d is created. As we have given it a floating point value (10.0) the object is of type floating point. If we had defined d = 10, d would have been an integer object.

In other programming languages, values are stored in variables. This is not exactly the same as an object, as an object has "methods", that means functions that belong to the object.
For our beginning examples the difference is not important.

There are many object types in Python.

The most important to begin with are:

Object type	Type class name	Description	Example
Integer	int	Signed integer, 32 bit	a = 5
Float	float	Double precision floating point number, 64 bit	b = 3.14
Complex	complex	Complex number	c = 3 + 5j c= complex(3,5)
Character	chr	Single byte character	d = chr(65) d = 'A' d = "A"
String	str	List of characters, text string	e = 'LTAM' e = "LTAM"

10. Input with data conversion

If we use the raw_input function in Python 2.x or the input function in Python 3, the result is always a string. So if we want to input a number, we have to convert from string to number.

```
x = int(raw_input("Input an integer: "))
y = float(raw_input("Input a float: "))
print x, y
```

Now we can modify our program to calculate the area of a circle, so we can input the diameter:

```
""" Calculate area of a circle"""
from math import *
d = float(raw_input("Diameter: "))
A = pi * d** 2 / 4
print "Area = ", A
```

Diameter: 25
Area = 490.873852123

Note:
The text at the beginning of the program is a description of what it does. It is a special comment enclosed in triple quote marks that can spread over several lines.
Every program should have a short description of what it does.

11. While loops

We can use the computer to do tedious tasks, like calculating the square roots of all integers between 0 and 100. In this case we use a while loop as

```
0               0.0
1               1.0
2               1.41421356237
3               1.73205080757
...
98              9.89949493661
99              9.94987437107
100             10.0
READY'
```

The syntax is :

```
while <condition> :
        <....
        block of statements
        ...>
```

The block of statements is executed as long as <condition> is True, in our example as long as i <= 100.

Take care:

- Don't forget the ":" at the end of the while statement
- Don't forget to indent the block that should be executed inside the while loop!

The indentation can be any number of spaces (4 are standard), but it must be consistent for the whole block.

Avoid endless loops!

In the following example the loop runs infinitely, as the condition is always true:

```
i = 0
while i<= 5 :
    print i
```

The only way to stop it is by pressing <Ctrl>-C

Examples of conditions:

Example	
x == 3	True if x = 3
x != 5	True if x is not equal to 5
x < 5 x > 5	
x <= 5 x >= 5	

Note:
i = i +1 can be written in a shorter and more "Pythonic" way as i += 1

12. Testing conditions: if, elif, else

Sometimes it is necessary to test a condition and to do different things, depending on the condition.
Examples: avoiding division by zero, branching in a menu structure etc.

The following program greets the user with "Hello Tom", if the name he inputs is Tom:

```
s = raw_input ("Input your name: ")

if s == "Tom":
    print "HELLO ", s
```

Note the indentation and the ":" behind the if statement!

The above program can be extended to do something if the testing condition is not true:

```
s = raw_input ("Input your name: ")
if s == "Tom":
    print "Hello ", s
else:
    print "Hello unknown"
```

It is possible to test more than one condition using the elif statement:

```
s = raw_input ("Input your name: ")
if s == "Tom":
    print "Hello ", s
elif s == "Carmen":
    print "I'm so glad to see you ", s
elif s == "Sonia":
    print "I didn't expect you ",s
else:
    print "Hello unknown"
```

Note the indentation and the ":" behind the if, elif and else statements!

13. Tuples

In Python, variables can be grouped together under one name. There are different ways to do this, and one is to use tuples.

Tuples make sense for small collections of data, e.g. for coordinates:

```
(x,y) = (3, 3)
coordinates = (x,y)
print coordinates
```

```
dimensions = (8, 5.0, 3.14)
print dimensions
print dimensions[0]
print dimensions[1]
print dimensions[2]
```

```
(3, 3)
(8, 5.0, 3.14)
8
5.0
3.14
```

Note:
The brackets may be omitted, so it doesn't matter if you write x, y or (x, y)

14. Lists (arrays)

Lists are ordered sequences of objects.
It can for example be very practical to put many measured values, or names of an address book, into a list, so they can be accessed by one common name.

```
nameslist = ["Sam", "Lizy", "Bib"]
numberslist = [1, 2, 3.14]
mixedlist = ["ham", 'eggs', 3.14, 3]
```

Note:
Unlike other programming languages Python's arrays may contain different types of objects in one list.

New elements can be appended to a list:

```
a=[0,1,2]
print a

a.append(5)
a.append("Zapzoo")
print a
```

```
[0, 1, 2]
[0, 1, 2, 5, 'Zapzoo']
```

An empty list can be created this way:

x=[]

Sometimes we need an array that is initialized with zero values.

This is done with:

```
y= [0]*10       # array of integers with 10 zero elements
x = [0.0]*20    # array of floats with 20 zero elements
```

The number of elements can be determined with the len (length) function:

```
a=[0,1,2]
print len(a)

3
```

The elements of a list can be accessed one by one using an index:

```
mylist = ["black", "red", "orange"]
print mylist[0]
print mylist[1]
print mylist[2]
```

```
black
red
orange
```

15. Range: producing lists of integer numbers

Often you need a regularly spread list of numbers from a beginning value to an end value.

This is done by the range function:

```
""" range gives a list of int numbers
    note that end value is NOT included! """

r1 = range(11)          # 0...10
print r1                # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

r2 = range(5,16)        # 5...15
print r2                # [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

r3 = range(4,21,2)      # 4...20  step 2
print r3                # [4, 6, 8, 10, 12, 14, 16, 18, 20]

r4 = range(15, 4, -3)   # 15....5 step -3
print r4                # [15, 12, 9]
```

The general syntax is

```
range (<startvalue>, <endvalue>, <stepsize>)
```

Take care:

- A strange (and somewhat illogical) detail of the range function is that the end value is excluded from the resulting list!
- The range function only works for integers!

16. Producing lists of floating point numbers

If you need floating point numbers, use linspace from the Numpy module, a package that is very useful for technical and scientific applications. This package must be installed first, it is available at http://www.numpy.org.
Don't forget to import the module in your script!

Note:
Here we use a slightly different method of import that avoids confusion between names of variables and numpy funtions. There are 3 ways to import functions from a module, see appendix.

```
*** for floating point numbers use linspace and logspace from numpy!***
import numpy as np
x5 = np.linspace(0,2,9)
print x5
```

```
[0.    0.25  0.5   0.75  1.    1.25  1.5   1.75  2.  ]
```

The syntax for linspace is

```
linspace( <startvalue>, <stopvalue>, <number of values> )
```

The next example gives 9 logarithmically spaced values between $100 = 10^2$ and $1000 = 10^3$:

```
x6 = np.logspace(2, 3, 9)
print x6
```

```
[  100.           133.35214322   177.827941     237.13737057   316.22776602
   421.69650343   562.34132519   749.89420933  1000.        ]
```

17. Iterating through a list: the for loop

If we have to do something with all the elements of a list (or another sequence like a tuple etc.) one after the other, we use a for loop.

The example uses the names of a list of names, one after one:

```
mynames = [ "Sam", "Pit", "Misch" ]

for n in mynames:
    print "HELLO ", n
```

```
HELLO  Sam
HELLO  Pit
HELLO  Misch
```

This can also be done with numbers:

```
from math import *
for i in range (0, 5):
    print i, "\t", sqrt(i)
```

```
0       0.0
1       1.0
2       1.41421356237
3       1.73205080757
4       2.0
```

Notes:

- Python's for loop is somewhat different of the for ... next loops of other programming languages. in principle it can iterate through anything that can be cut into slices. So it can be used on lists of numbers, lists of text,mixed lists, strings, tuples etc.
- In Python the for ... next construction is often not to be missed, if we think in a "Pythonic" way. Example: if we need to calculate a lot of values, it is not a good idea to use a for loop, as this is very time consuming. It is better to use the Numpy module that provides array functions that can calculate a lot of values inone bunch (see below).

18. Iterating with indexing

Sometimes you want to iterate through a list and have access to the index (the numbering) of the items of the list.

The following example uses a list of colour codes for electronic parts and prints their index and the colour. As the colours list is well ordered, the index is also the colour value.

```
""" Display resistor colour code values"""
colours = [ "black", "brown", "red", "orange", "yellow",
            "green", "blue", "violet", "grey","white" ]

cv = list (enumerate (colours))

for c in cv:
    print c[0], "\t", c[1]
```

The list(enumerate (...)) function gives back a list of tuples cv that contain each an index (the numbering) and the color value as text. If we print this we see

[(0, 'black'), (1, 'brown'), (2, 'red'), (3, 'orange'), (4, 'yellow'), (5, 'green'), (6, 'blue'), (7, 'violet'), (8, 'grey'), (9, 'white')]

Now we iterate on this, so we get the different tuples one after the other.

From these tuples we print c[0], the index and c[1], the colour text, separated by a tab.

So as a result we get

```
0       black
1       brown
2       red
...
8       grey
9       white
```

19. Functions

It is a good idea to put pieces of code that do a clearly defined task into separate blocks that are called functions. Functions must be defined before they are used.

Once they are defined, they can be used like native Python statements.

A very simple example calculates area and perimeter of a rectangle:

```
# function definitions
def area(b, h):
""" calculate area of a rectangle"""
A = b * h return A

def perimeter(b, h)
""" calulates perimeter of a rectangle"""
P = 2 * (b+ h) return P

# main program using defined functions
width = 5
height = 3
print "Area = ", area(width, height)
print "Perimeter = ", perimeter(width, height)
```

The syntax of a function definition is:

```
def <function_name(<argument1>, <argument2>, ..... ):
        <statements>
        ....
        return <returnvalue(s)>
```

The arguments are the values passed to the function.

the return value is the value that the function gives back to the calling program statement.

Don't forget the ":" and the indentation !

A function can return more than one value:

```
# function definition
def area_and_perimeter (b, h):
    A = b * h
    P = 2 * (b+h)
    return A, P

# main program using defined function
ar, per = area_and_perimeter ( 4, 3)
print ar
print per
```

Here the return values are returned as a tuple:

If the function doesn't need to return a value, the return statement can simply be omitted.

Example

```
# function definition
def greeting():
    print "HELLO"


# main program using defined functions
greeting()
```

One good thing about functions is that they can be easily reused in another program.

Notes:

- Functions that are used often can be placed in a separate file called a module. Once this module is importedthe functions can be used in the program.
- It is possible to pass a variable number of arguments to a function.
For details see here: http://en.wikibooks.org/wiki/Python_Programming/Functions
- It is possible to pass named variables to a function.

20. Avoiding for loops: vector functions

For loops tend to get slow if there are many iterations to do.

They are not necessary for calculations on numbers, if the **Numpy** module is used. It can befound here http://www.numpy.org/ and must be installed before using it.

In this example we get 100 values of a sine function in one line of code:

```
import numpy as np

# calculate 100 values for x and y without a for loop
x = np.linspace(0, 2* np.pi, 100)
y = np.sin(x)

print x
print y
```

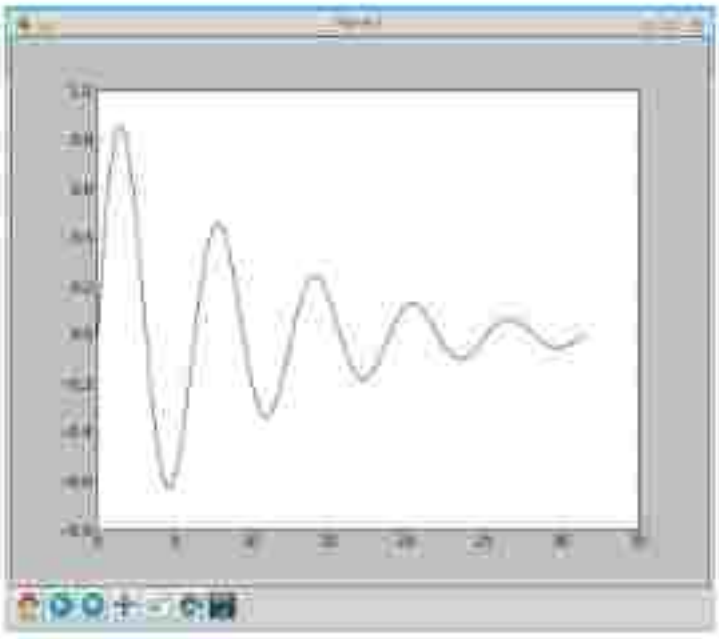
21. Diagrams

Once you have calculated the many function values, it would be nice to display them in adiagram. This is very simple if you use Matplotlib, the standard Python plotting library.

Matplotlib can be found here: http://matplotlib.org/downloads

The following program calculates the values of a function and draws a diagram:

```
from numpy import linspace, sin, exp, pi
import matplotlib.pyplot as mp

# calculate 500 values for x and y without a for loop
x = linspace(0, 10*pi, 500)
y = sin(x) * exp(-x/10)

# make diagram
mp.plot(x,y)
mp.show()
```



Notes:

- Matplotlib offers much more than this, see online documentation.
- There are two ways to use Matplotlib: a simple functional way that we have just used, and a more complicated object oriented way, that allows for example to embed a diagram into a GUI.

22. What next?

This tutorial covered just a minimal part of the Python basics. There are many, many interesting possibilities to discover:

Object oriented programming, programs with a graphical user interface (GUI), connecting to hardware, signal processing, image and sound processing etc. etc.

The Python package index is a good place to look for interesting modules:

https://pypi.python.org/pypi

Appendix

Importing functions from a module
Three ways to import functions:

1 the simplest way: import everything from a moduleadvantage: simple usage e.g. of math functions disadvantage: risk of naming conflicts when a variable has the same name as a module function

```
from numpy import *
print sin(pi/4)

# With this import method the following would give an error:
#sin = 5              # naming conflict!
#print sin(pi/4)
```

2 import module under an alias name that is short enough to enhance code clarityadvantage: it is clear to see which function belongs to which module

```
import numpy as np
print np.sin(np.pi/4)
```

3 import only the functions that are needed advantage: simple usage e.g. of math functions naming conflict possible, but less probable than with

1 disadvantage: you must keep track of all the used functions and adapt the import statement if anew function is used

```
from numpy import linspace, sin,
exp, pi#print sin(pi/4)
```

Python Directory

If there are a large number of files to handle in our Python program, we can arrange our code within different directories to make things more manageable. A directory or folder is a collection of files and subdirectories. Python has the os module that provides us with many useful methods to work with directories (and files as well).

Get Current Directory

We can get the present working directory using the getcwd() method of the os module.
This method returns the current working directory in the form of a string. We can also use the getcwdb() method to get it as bytes object.

```
>>> import os

>>> os.getcwd()
'C:\\Program Files\\PyScripter'

>>> os.getcwdb()
b'C:\\Program Files\\PyScripter'
```

The extra backslash implies an escape sequence. The print() function will render this properly.

```
>>> print(os.getcwd())
```

C:\Program Files\PyScripter

Changing Directory

We can change the current working directory by using the chdir() method.
The new path that we want to change into must be supplied as a string to this method. We can use both the forward-slash / or the backward-slash \ to separate the path elements.
It is safer to use an escape sequence when using the backward slash.

```
>>> os.chdir('C:\Python33')

>>> print(os.getcwd())
C:\Python33
```

List Directories and Files

All files and sub-directories inside a directory can be retrieved using the listdir() method.
This method takes in a path and returns a list of subdirectories and files in that path. If no path is specified, it returns the list of subdirectories and files from the current working directory.

```
>>> print(os.getcwd())
C:\Python33

>>> os.listdir()
['DLLs',
'Doc',
'include',
'Lib',
'libs',
'LICENSE.txt',
'NEWS.txt',
'python.exe',
'pythonw.exe',
'README.txt',
'Scripts',
'tcl',
'Tools']

>>> os.listdir('G:\')
['$RECYCLE.BIN',
'Movies',
'Music',
'Photos',
'Series',
'System Volume Information']
```

Exceptions in Python

Python has many built-in exceptions that are raised when your program encounters an error (something in the program goes wrong).

When these exceptions occur, the Python interpreter stops the current process and passes it to the calling process until it is handled. If not handled, the program will crash.

For example, let us consider a program where we have a function A that calls function B, which in turn calls function C. If an exception occurs in function C but is not handled in C, the exception passes to B and then to A.

If never handled, an error message is displayed and our program comes to a sudden unexpected halt.

Catching Exceptions in Python

In Python, exceptions can be handled using a try statement.

The critical operation which can raise an exception is placed inside the try clause. The code that handles the exceptions is written in the except clause.

We can thus choose what operations to perform once we have caught the exception. Here is a simple example

= import module sys to get the type of exception
import sys

randomList = ['a', 0, 2]

for entry in randomList:
    try:
        print("The entry is", entry)
        r = 1/int(entry)
        break
    except:
        print("Oops!", sys.exc_info()[0], "occurred.")
        print("Next entry.")
        print()
print("The reciprocal of", entry, "is", r)

Output

The entry is a

Oops! <class 'ValueError'> occurred.

Next entry.

The entry is 0

Oops! <class 'ZeroDivisionError'> occured.

Next entry.

The entry is 2

The reciprocal of 2 is 0.5

In this program, we loop through the values of the randomList list. As previously mentioned, the portion that can cause an exception is placed inside the try block.

If no exception occurs, the except block is skipped and normal flow continues(for last value). But if any exception occurs, it is caught by the except block (first and second values).

Here, we print the name of the exception using the exc_info() function inside sys module. We can see that a causes ValueError and 0 causes ZeroDivisionError.

Catching Specific Exceptions in Python

In the above example, we did not mention any specific exception in the except clause.

This is not a good programming practice as it will catch all exceptions and handle every case in the same way.

We can specify which exceptions an except clause should catch.

A try clause can have any number of except clauses to handle different exceptions, however, only one will be executed in case an exception occurs.

We can use a tuple of values to specify multiple exceptions in an except clause. Here is an example pseudo code.

```
try:
  # do something
  pass

except ValueError:
  # handle ValueError exception
  pass

except (TypeError, ZeroDivisionError):
  # handle multiple exceptions
  # TypeError and ZeroDivisionError
  pass

except:
  # handle all other exceptions
  pass
```

Object Oriented Programming

Python is a multi-paradigm programming language. It supports different programming approaches.

One of the popular approaches to solve a programming problem is by creating objects. This is known as Object-Oriented Programming (OOP).

An object has two characteristics:

- attributes
- behavior

Let's take an example:

A parrot is an object, as it has the following properties:

- name, age, color as attributes
- singing, dancing as behavior

The concept of OOP in Python focuses on creating reusable code. This concept is also known as DRY (Don't Repeat Yourself).

In Python, the concept of OOP follows some basic principles:

Class

A class is a blueprint for the object.

We can think of class as a sketch of a parrot with labels. It contains all the details about the name, colors, size etc. Based on these descriptions, we can study about the parrot. Here, a parrot is an object.

The example for class of parrot can be :

```
class Parrot:
  pass
```

Here, we use the class keyword to define an empty class Parrot. From class, we construct instances. An instance is a specific object created from a particular class.

Object

An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.

The example for object of parrot class can be:

```
obj = Parrot()
```

Here, obj is an object of class Parrot.

Suppose we have details of parrots. Now, we are going to show how to build the class and objects of parrots.

Example 1: Creating Class and Object in Python

```
class Parrot:

  # class attribute
  species = "bird"

  # instance attribute
  def __init__(self, name, age):
    self.name = name
    self.age = age

# instantiate the Parrot class
blu = Parrot("Blu", 10)
woo = Parrot("Woo", 15)

# access the class attributes
print("Blu is a {}".format(blu.__class__.species))
print("Woo is also a {}".format(woo.__class__.species))

# access the instance attributes
print("{} is {} years old".format( blu.name, blu.age))
print("{} is {} years old".format( woo.name, woo.age))
```

Output

```
Blu is a bird
Woo is also a bird
Blu is 10 years old
Woo is 15 years old
```

In the above program, we created a class with the name Parrot. Then, we define attributes. The attributes are a characteristic of an object.

These attributes are defined inside the __init__ method of the class. It is the initializer method that is first run as soon as the object is created.

Then, we create instances of the Parrot class. Here, blu and woo are references (value) to our new objects.

We can access the class attribute using class__species. Class attributes are the same for all instances of a class.

Similarly, we access the instance attributes using blu.name and blu.age. However, instance attributes are different for every instance of a class.

To learn more about classes and objects, go to Python Classes and Objects

Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

Example 2 : Creating Methods in Python

```
class Parrot:

    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)

# instantiate the object
blu = Parrot("Blu", 10)

# call our instance methods
print(blu.sing("'Happy'"))
print(blu.dance())
```

Output

```
Blu sings 'Happy'
Blu is now dancing
```

In the above program, we define two methods i.e sing() and dance(). These are called instance methods because they are called on an instance object i.e blu.

Inheritance

Inheritance is a way of creating a new class for using details of an existing class without modifying it. The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).

Example 3: Use of Inheritance in Python

```
# parent class
class Bird:

    def __init__(self):
        print("Bird is ready")
```

```
    def whoisThis(self):
        print("Bird")

    def swim(self):
        print("Swim faster")

# child class
class Penguin(Bird):

    def __init__(self):
        # call super() function
        super().__init__()
        print("Penguin is ready")

    def whoisThis(self):
        print("Penguin")

    def run(self):
        print("Run faster")

peggy = Penguin()
peggy.whoisThis()
peggy.swim()
peggy.run()
```

Output

```
Bird is ready
Penguin is ready
Penguin
Swim faster
Run faster
```

In the above program, we created two classes i.e. Bird (parent class) and Penguin (child class). The child class inherits the functions of parent class. We can see this from the swim() method.

Again, the child class modified the behavior of the parent class. We can see this from the whoisThis() method.

Furthermore, we extend the functions of the parent class, by creating a new run() method.

Additionally, we use the super() function inside the __init__() method. This allows us to run the __init__() method of the parent class inside the child clas

Encapsulation

Using OOP in Python, we can restrict access to methods and variables. This prevents data from direct modification which is called encapsulation. In Python, we denote private attributes using underscore as the prefix i.e single _ or double __.

Example 4: Data Encapsulation in Python

```
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
```

```
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()
```

Output

```
Selling Price: 900
Selling Price: 900
Selling Price: 1000
```

In the above program, we defined a Computer class.

We used __init__() method to store the maximum selling price of Computer. Here, notice the code
c.__maxprice = 1000

Here, we have tried to modify the value of __maxprice outside of the class. However, since __maxprice is a private variable, this modification is not seen on the output.

As shown, to change the value, we have to use a setter function i.e setMaxPrice() which takes price as a parameter.

Polymorphism

Polymorphism is an ability (in OOP) to use a common interface for multiple forms (data types).

Suppose, we need to color a shape, there are multiple shape options (rectangle, square, circle). However we could use the same method to color any shape. This concept is called Polymorphism.

Example 5: Using Polymorphism in Python

```
k

# instantiate objects
blu = Parrot()
peggy = Penguin()

# passing the object
flying_test(blu)
flying_test(peggy)
```

Output

```
Parrot can fly
Penguin can't fly
```

In the above program, we defined two classes Parrot and Penguin. Each of them have a common fly() method. However, their functions are different.

To use polymorphism, we created a common interface i.e flying_test() function that takes any object and calls the object's fly() method. Thus, when we passed the blu and peggy objects in the flying_test() function, it ran effectively.

Iterators in Python

Iterators are everywhere in Python. They are elegantly implemented within for loops, comprehensions, generators etc. but are hidden in plain sight.

Iterator in Python is simply an object that can be iterated upon. An object which will return data, one element at a time.

Technically speaking, a Python **iterator object** must implement two special methods, __iter__() and __next__(), collectively called the **iterator protocol**.

An object is called **iterable** if we can get an iterator from it. Most built-in containers in Python like: list, tuple, string etc. are iterables.

The iter() function (which in turn calls the __iter__() method) returns an iterator from them.

Iterating Through an Iterator

We use the next() function to manually iterate through all the items of an iterator. When we reach the end and there is no more data to be returned, it will raise the StopIteration Exception. Following is an example.

```
# define a list
my_list = [4, 7, 0, 3]

# get an iterator using iter()
my_iter = iter(my_list)

# iterate through it using next()

# Output: 4
print(next(my_iter))

# Output: 7
print(next(my_iter))

# next(obj) is same as obj.__next__()

# Output: 0
print(my_iter.__next__())

# Output: 3
print(my_iter.__next__())

# This will raise error, no items left
next(my_iter)
```

Output

```
4
7
0
3
Traceback (most recent call last):
  File "<string>", line 24, in <module>
    next(my_iter)
StopIteration
```

A more elegant way of automatically iterating is by using the for loop. Using this, we can iterate over any object that can return an iterator, for example list, string, file etc.

```
>>> for element in my_list:
...     print(element)
...
4
7
0
3
```

Generators in Python

There is a lot of work in building an iterator in Python. We have to implement a class with __iter__() and __next__() method, keep track of internal states, and raise StopIteration when there are no values to be returned.
This is both lengthy and counterintuitive. Generator comes to the rescue in such situations.

Python generators are a simple way of creating iterators. All the work we mentioned above are automatically handled by generators in Python.

Simply speaking, a generator is a function that returns an object (iterator) which we can iterate over (one value at a time).

Create Generators in Python

It is fairly simple to create a generator in Python. It is as easy as defining a normal function, but with a yield statement instead of a return statement. If a function contains at least one yield statement (it may contain other yield or return statements), it becomes a generator function. Both yield and return will return some value from a function. The difference is that while a return statement terminates a function entirely, yield statement pauses the function saving all its states and later continues from there on successive calls.

Differences between Generator function and Normal function

Here is how a generator function differs from a normal function.

- Generator function contains one or more yield statements.
- When called, it returns an object (iterator) but does not start execution immediately.
- Methods like __iter__() and __next__() are implemented automatically. So we can iterate through the items using next().
- Once the function yields, the function is paused and the control is transferred to the caller.
- Local variables and their states are remembered between successive calls.
- Finally, when the function terminates, StopIteration is raised automatically on further calls.

Here is an example to illustrate all of the points stated above. We have a generator function named my_gen() with several yield statements.

```
# A simple generator function
def my_gen():
    n = 1
    print('This is printed first')
    # Generator function contains yield statements
    yield n

    n += 1
    print('This is printed second')
    yield n

    n += 1
    print('This is printed at last')
    yield n
```

An interactive run in the interpreter is given below. Run these in the Python shell to see the output.

```
>>> # It returns an object but does not start execution immediately.
>>> a = my_gen()

>>> # We can iterate through the items using next().
>>> next(a)
This is printed first
1 >>> # Once the function yields, the function is paused and the control is transferred to the caller.

>>> # Local variables and their states are remembered between successive calls.
>>> next(a)
This is printed second
2 >>> next(a)
This is printed at last
3 >>> # Finally, when the function terminates, StopIteration is raised automatically on further calls.
>>> next(a)
Traceback (most recent call last):
...
StopIteration
>>> next(a)
Traceback (most recent call last):
...
StopIteration
```

One interesting thing to note in the above example is that the value of variable n is remembered between each call.

Unlike normal functions, the local variables are not destroyed when the function yields. Furthermore, the generator object can be iterated only once.

To restart the process we need to create another generator object using something like a = my_gen().

One final thing to note is that we can use generators with for loops directly.

This is because a for loop takes an iterator and iterates over it using next() function. It automatically ends when StopIteration is raised. Check here to know how a for loop is actually implemented in Python.

```
# A simple generator function
def my_gen():
    n = 1
    print('This is printed first')
    # Generator function contains yield statements
    yield n

    n += 1
    print('This is printed second')
    yield n

    n += 1
    print('This is printed at last')
    yield n


# Using for loop
for item in my_gen():
    print(item)
```

Python Closures

Nonlocal variable in a nested function

Before getting into what a closure is, we have to first understand what a nested function and nonlocal variable is.

A function defined inside another function is called a nested function. Nested functions can access variables of the enclosing scope.

In Python, these non-local variables are read-only by default and we must declare them explicitly as non-local (using nonlocal keyword) in order to modify them.

Following is an example of a nested function accessing a non-local variable.

```
def print_msg(msg):
    # This is the outer enclosing function

    def printer():
        # This is the nested function
        print(msg)

    printer()

# We execute the function
# Output: Hello
print_msg("Hello")
```

Output

Hello

We can see that the nested printer() function was able to access the non-local msg variable of the enclosing function.

Defining a Closure Function

In the example above, what would happen if the last line of the function print_msg() returned the printer() function instead of calling it? This means the function was defined as follows:

```
def print_msg(msg):
    # This is the outer enclosing function

    def printer():
        # This is the nested function
        print(msg)

    return printer  # returns the nested function


# Now let's try calling this function.
# Output: Hello
another = print_msg("Hello")
another()
```

Output

Hello

That's unusual.

The print_msg() function was called with the string "Hello" and the returned function was bound to the name another. On calling another(), the message was still remembered although we had already finished executing the print_msg() function.

This technique by which some data ("Hello in this case) gets attached to the code is called **closure in Python**.

This value in the enclosing scope is remembered even when the variable goes out of scope or the function itself is removed from the current namespace.

Try running the following in the Python shell to see the output.

```
>>> del print_msg
>>> another()
Hello
>>> print_msg("Hello")
Traceback (most recent call last):
...
NameError: name 'print_msg' is not defined
```

Here, the returned function still works even when the original function was deleted.

When do we have closures?

As seen from the above example, we have a closure in Python when a nested function references a value in its enclosing scope.

The criteria that must be met to create closure in Python are summarized in the following points.

- We must have a nested function (function inside a function).
- The nested function must refer to a value defined in the enclosing function.
- The enclosing function must return the nested function.

When to use closures?

So what are closures good for?

Closures can avoid the use of global values and provides some form of data hiding. It can also provide an object oriented solution to the problem.

When there are few methods (one method in most cases) to be implemented in a class, closures can provide an alternate and more elegant solution. But when the number of attributes and methods get larger, it's better to implement a class.

Python RegEx

In this tutorial, you will learn about regular expressions (RegEx), and use Python's re module to work with RegEx (with the help of examples).

A Regular Expression (RegEx) is a sequence of characters that defines a search pattern. For example,

```
^a...s$
```

The above code defines a RegEx pattern. The pattern is: **any five letter string starting with a and ending with s**.

A pattern defined using RegEx can be used to match against a string.

Expression	String	Matched?
	abc	No match
	alias	Match
^a...s$	abyss	Match
	Alias	No match
	An abacus	No match

Python has a module named `re` to work with RegEx. Here's an example:

```
import re

pattern = '^a...s$'
test_string = 'abyss'
result = re.match(pattern, test_string)

if result:
  print("Search successful.")
else:
  print("Search unsuccessful.")
```

Here, we used `re.match()` function to search pattern within the test_string. The method returns a match object if the search is successful. If not, it returns None.

There are other several functions defined in the re module to work with RegEx. Before we explore that, let's learn about regular expressions themselves.

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HANDS ON TRAINING COURSE

ON

BASICS OF PYTHON PROGRAMMING

Date: From 05-01-2022 To 18-02-2022(6 Week Course, Only on Saturdays)

SHORTLISTED STUDENTS

ATTENDANCE SHEET

S. No	Hall Ticket Number	NAME OF THE STUDENT	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
1	20D41A0501	AASHRITHA POLAMPALLI	P. Aashritha	P. Aashritha	P. Aashritha	P. Aashritha	P. Aashritha	P. Aashritha
2	20D41A0502	ABDUL BASEER	Abdul	Abdul	Abdul	Abdul	Abdul	Abdul
3	20D41A0503	ABHINAY SAGI	S. Abhinay	S. Abhinay	S. Abhinay	S. Abhinay	S. Abhinay	S. Abhinay
4	20D41A0504	ADUSUMALLI SAI DHYEYA	Sai Dhyeya	Sai Dhyeya	Sai Dhyeya	Sai Dhyeya	Sai Dhyeya	Sai Dhyeya
5	20D41A0505	AENUGU DURGA PRASAD REDDY	Durga Reddy	Durga Reddy	Durga Reddy	Durga Reddy	Durga Reddy	Durga Reddy
6	20D41A0506	AITI AKSHAY	A	A	A	A	A	A
7	20D41A0507	ALETI UDAY GANESH	UG	UG	UG	UG	UG	UG
8	20D41A0508	ALLA VASANTH KUMAR	Vasanth	Vasanth	Vasanth	Vasanth	Vasanth	Vasanth
9	20D41A0509	ALLURI SAI KUMAR	Sai	Sai	Sai	Sai	Sai	Sai
10	20D41A0510	AMMIREDDY SHASHANKA	Shashanka	Shashanka	Shashanka	Shashanka	Shashanka	Shashanka
11	20D41A0511	ANNOJU SAI KIRAN	SK	SK	SK	SK	SK	SK
12	20D41A0512	ANUGU YASHWANTH REDDY	Y Reddy	Y Reddy	Y Reddy	Y Reddy	Y Reddy	Y Reddy
13	20D41A0513	ATHAPU SRILATHA	Srilatha	Srilatha	Srilatha	Srilatha	Srilatha	Srilatha
14	20D41A0514	BAGULI JAYA SREE	Jaya	Jaya	Jaya	Jaya	Jaya	Jaya
15	20D41A0515	BALAM SRIVANI	Srivani	Srivani	Srivani	Srivani	Srivani	Srivani
16	20D41A0516	BALU SAI KRISHNA	Sai Krishna	Sai Krishna	Sai Krishna	Sai Krishna	Sai Krishna	Sai Krishna
17	20D41A0517	BANDARI KOMALI	Komali	Komali	Komali	Komali	Komali	Komali
18	20D41A0518	BANDARI SHRAVYA	Shravya	Shravya	Shravya	Shravya	Shravya	Shravya

19	20041A0519	BANDARU DIVYA	Divya	Divya	Divya	Divya	Divya	Divya
20	20041A0520	BAVANIYULA ESHWAR	Eshwar	Eshwar	Eshwar	Eshwar	Eshwar	Eshwar
21	20041A0521	BAYNDLA VIJAY KUMAR	Vijay	Vijay	Vijay	Vijay	Vijay	Vijay
22	20041A0522	BEJJANKI NITHIN	Nithin	Nithin	Nithin	Nithin	Nithin	Nithin
23	20041A0523	BHAVANARI MURALI KRISHNA	Murali Krishna	Murali Krishna	Murali Krishna	Murali Krishna	Murali Krishna	Murali Krishna
24	20041A0524	BHIMISETTY MANASA	Manasa	Manasa	Manasa	Manasa	Manasa	Manasa
25	20041A0525	BHUPATHIGALLA NITHIN	Nithin	Nithin	Nithin	Nithin	Nithin	Nithin

Coordinator

Convener

HOD/CSE

Head of the Department
Department of CSE
Sai Indu College Of Engineering & Technology
Sheriguda, Ibrahimpatnam, R.R. Dist.