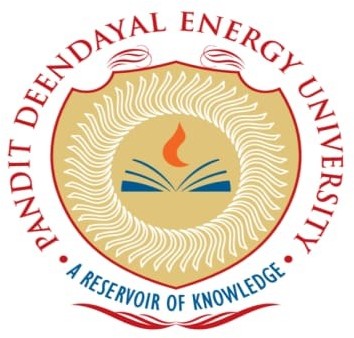
**PANDIT DEENDAYAL ENERGY UNIVERSITY SCHOOL OF TECHNOLOGY**

**DEPARTMENT OF ENGINEERING**



**LAB MANUAL INDUSTRY 4.0 (20IF201T) B.TECH - SEMESTER IV**

**NAME :**

**ROLL NO:**

**BATCH :**

**BRANCH:**

**CERTIFICATE**

**This is to certify that roll no of B.Tech 3rd year**

**branch has completed the laboratory sessions for the subjects of Industry 4.0 (20IF201T) satisfactorily.**

**Subject Coordinator**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr No** | **Experiment** | **Date of submission** | **Marks obtained** | **Sign** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 | Introduction to MATLAB programming and SIMULINK |  |  |  |
| 4 | Design of smart meter for recording the electricity consumption |  |  |  |
| 5 | Design of Ultrasonic proximity sensors using Arduino |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |

Experiment No:-03

Introduction to MATLAB programming and SIMULINK

Aim: To understand the basics about MATLAB software and learn basic programming and simulation.

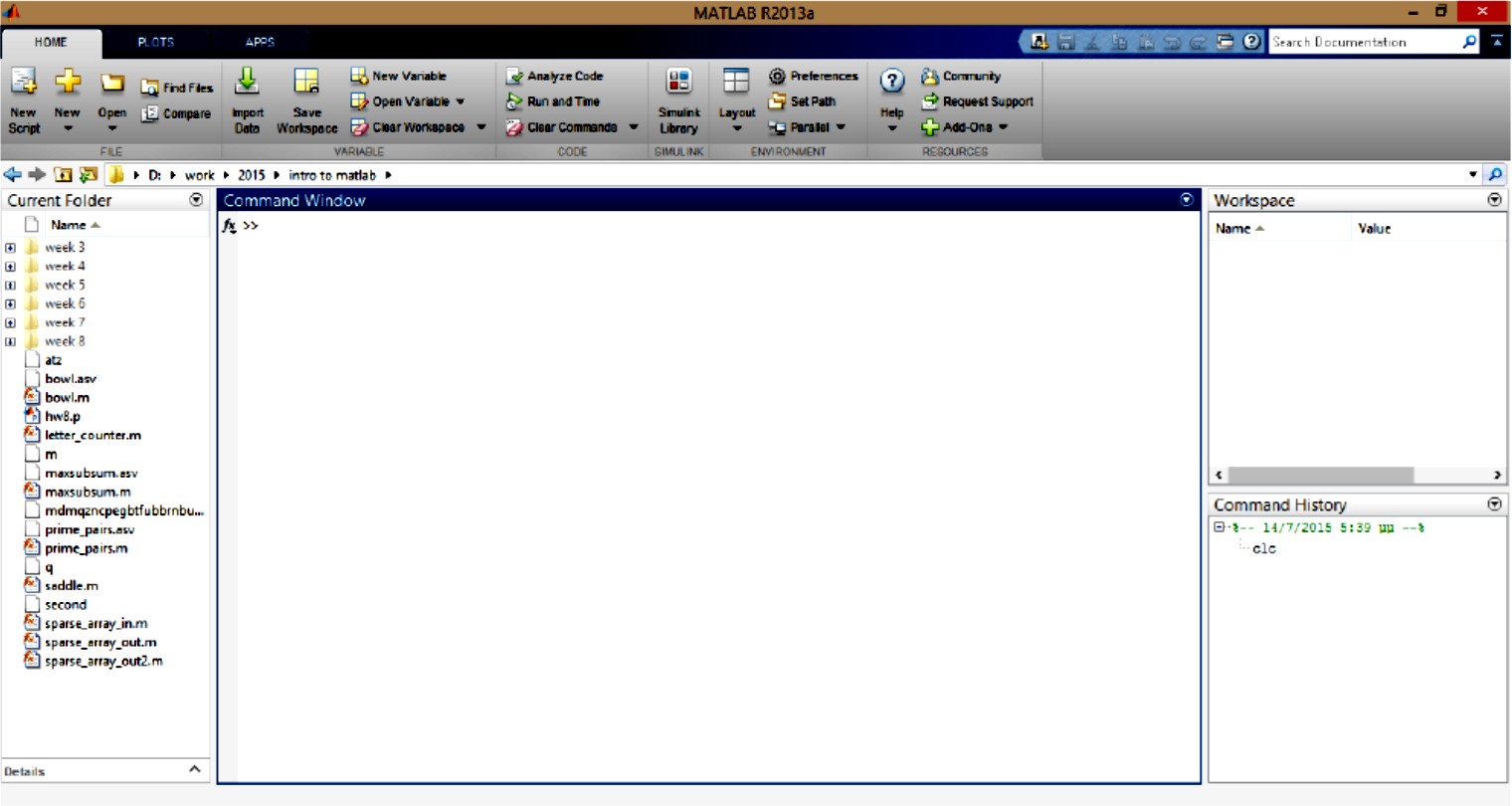
### Introduction

"MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation." - from Mathworks MATLAB is available on all three major operation systems.

SIMULINK is a powerful simulator for modeling and simulating dynamic systems.

One of the biggest advantages of using MATLAB is one does not need to declare ahead of time, the type of variable being used. The base version of MATLAB/SIMULINK is aptly supported by a variety of toolboxes.

The window is divided into three main parts.



* The Command Window is the main window where the commands are input.
* The Current Directory shows the directory from which MatLab runs the files and functions we have created.
* Command History shows the history of all the commands that we input into the Command Window.
* The main menu has three different tabs, HOME, PLOTS and APPS. From the Home tab the appearance and layout of MatLab can be changed.

### Script File

A script file in MATLAB is any set of MATLAB commands which are executed in that sequence. Script files have ".m" extension. It is recommended to use a meaningful name for the filename without including "space".

A script file can be run by simply typing the filename without its extension.

The scope of the variables used in script files is the general workspace. They can be accessed even after the execution.

A script file can contain functions which can be used within that script.

MATLAB executes the script file by interpreting every line, which makes its somewhat slow at times. A script file can be put on path so that it can be called from any directory.

Note: Do not assign a script file name which is already a MATLAB in built function.

### Matrix Algebra

**Create a row vector** A = [1 3 6 8]

**Create a column vector** A = [1;3;6;8]

**Transpose of matrix** A = [1 3 6 8]’ **Or use transpose(**A)

**Create a matrix** A = [1 3; 6 8]

**Assign each element to matrix** A(1, 1) = 1; A(2, 1) = 6; A(1, 2) = 3; A(2, 2) = 8;

* Both matrices and vectors are enclosed in square brackets
* Elements are accessed using parentheses
* **For creating uniformly spaced vector** B = (0 : 1 : 99) B has 100 elements from 0 to 99 with spacing of 1.

Access first two and last two elements of B: B=([1 2 end-1 end]) Take every third element of B and store in C: C=B(1:3:end)

### Special Matrices/Array

Matlab allows to create a variety of special matrices that appear in several problems.

* **Create an Identity matrix** Imat = eye(3; 3)
* **Create a matrix of ones and zeros** A = ones(3; 3) B = zeros(3; 3)
* **Hadamard Matrix** H = hadamard(2)
* **Magic Matrix** M = magic(3)
* **Toeplitz Matrix** T = toeplitz([34]; [32])
* **Simple checks on matrices** Check if the matrix is empty A=[ ]; isempty(A)

### Check if two matrices are equal

A=exp([1 2 ; 3 4])

B=expm([1 2; 3 4])

isequal(A,B)

### Check if a matrix contains real elements

A = [1 2; 3 4]

B = ones(2; 2) C = A + j \* B

isreal(C)

### Check if matrix elements are NaN (Not a Number)

A = [1 2; 3 inf]

isnan(A)

### Mathematical Operations on Matrices

Transpose of a matrix A = [1 8 3; 5 6 0] A’

### Extract triangular part

tril(A)

### Find indices of elements

find(A) find(A >= 4)

### Matrix Multiplication

A = [1 8; -5 0; 9 2] B = [2 6; 8 3; -1 5]

### Element wise multiplication: A.\*B

* **Product of A and B:** A\*B

### Maximum and minimum of A

max(A) min(abs(A))

* **Inverse of matrix A:** inv(A)

### Determinant of matrix A: det(A)

* **Matrix division** B B \*inv(A)

### Element wise division: B./A

* **Eigen values of matrix A:** eig(A)
* **Rank of matrix A:** rank(A)
* **Characteristics equation** eqA = poly([1 2; 3 4]) roots(eqA),

Note: compare this with eigen values of A

* **LU factorization** [L,U] = lu([1 2; 3 4])
* **Orthogonalization** Q = orth([1 2; 3 4])

### Special Numbers and variables

pi: The value of pi

inf: Infinity (or a very very large number) eps: Floating point relative accuracy

i or j: Imaginary number,

NaN: Not-a-Number

ans: The most recent answer

end: The last element of a vector; OR to indicate end of a loop or a conditional statement all: Used with clear command to clear all variables

### Elementary functions

Trigonometric: sin, sinh, cos, atan, sec Exponential: exp, log, log2, pow2, sqrt

Complex: abs, imag, conj, unwrap, angle Rounding: fix, floor, ceil, mod, rem, sign Specialized: bessel, beta, erf, dot, gamma Number theoretic: factor, primes, factorial, gcd

### Create 100 samples of sine and cosine

x = sin(2 \*pi \* 0:2 \* (0 : 99)) ; y = cos(2\* pi \* 0:2 \*(0 : 99))

**PART A: MATLAB Programming**

(1) **Sum of series 1 + 1/2 + 1/4 + 1/6 +... for 100 terms**

clc

% clear all

% sum of series 1 + 1/2 + 1/4 + 1/6 +... for 100 terms x=1; %initialising the first term

n=input('The number of terms in series to be summed:') for i=1:n

y=1/(2\*i);

x=x+y;

end display(x)

## (2) Square roots of odd positive integers

clc

%clear all

% square roots of odd positive integers:

n=input('The total number of odd positive terms whose roots are to be displayed:')

for i=1:n

y=(2\*i)-1;

x(i)=sqrt(y);

|  |  |  |
| --- | --- | --- |
| end display(x') |  | |
| (3) **Find the smallest 'k' so that** | **1 +** | **2 + 3 + ....+ k >= 30** |
| clc  % smallest 'k' so that 1 + 2 | + 3 | + ....+ k >= 30 |

for k=1:100

sum = k\*(k+1)/2; if sum>=30

display(k)

display('is the smallest value for which sum is >= 30') break

end end

## Create a function file to input the radius & area of circle is returned as output:

% Input the radius & output area of circle: function ar=cir\_area(r)

r=input('Enter the radius of the circle: ') ar=pi\*r\*r;

disp('Area of the circle is:')

## To check whether the number is odd or even

clc

% To check whether the number is odd or even

x=input(‘Enter any number: ’) z=rem(x,2)

if z==0

display(‘The entered number is an even number.’) else

display(‘The entered number is an odd number.’) end

## To display all prime numbers from 50-150

clc

clear all

% display all primes from 50 to 150 x1=input('Enter the lower limit of range') % 50

x2=input('Enter the upper limit of range') % 150 y=primes(x2)

for k=1:length(y) if y(k)>50

disp(y(k)) end

end

## Display if number is divisible by 100

clc

% Display if number is divisible by 100

x=input(‘Enter any number: ’) z=rem(x,100)

if z==0

disp(‘The number is divisible by 100.’)

end

else

disp(‘The number is not divisible by 100.’)

## Swap the values of given numbers

clc

% swap the values of given numbers

x1=input('Enter any number:') x2=input('Enter any number:')

a=x2; b=x1;

x1=a x2=b

sprintf('X1 is swapped by X2..!')

1. (9)

clc

clear all

% sum of all entries of a 6x5 matrix &

% maximum entry in a row

% column wise total sum

r=input(‘The number of rows ’) % 6 c=input(‘The number of columns ’) % 5 n=r\*c % total number of elements m=[];

for i=1:n

x(i)=input(‘Row-wise elements of the matrix’); m(i)=x(i);

end display(m)

q=max(m); % display max entry of a matrix f=sort(q,’descend’);

maxentry=f(1) a=[sum(m)]

b=[sum(a)]

A=m(r,:)

B=m(:,c)

sumrow=[sum(A)] % sum of row wise entries sumcol=[sum(B)] % sum of column entries k=sort(A,’descend’)

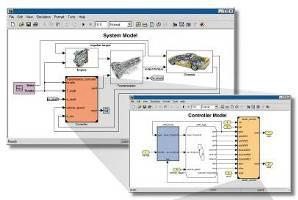
rowmax=k(1)

**PART B: MATLAB Simulink**

**Simulink**

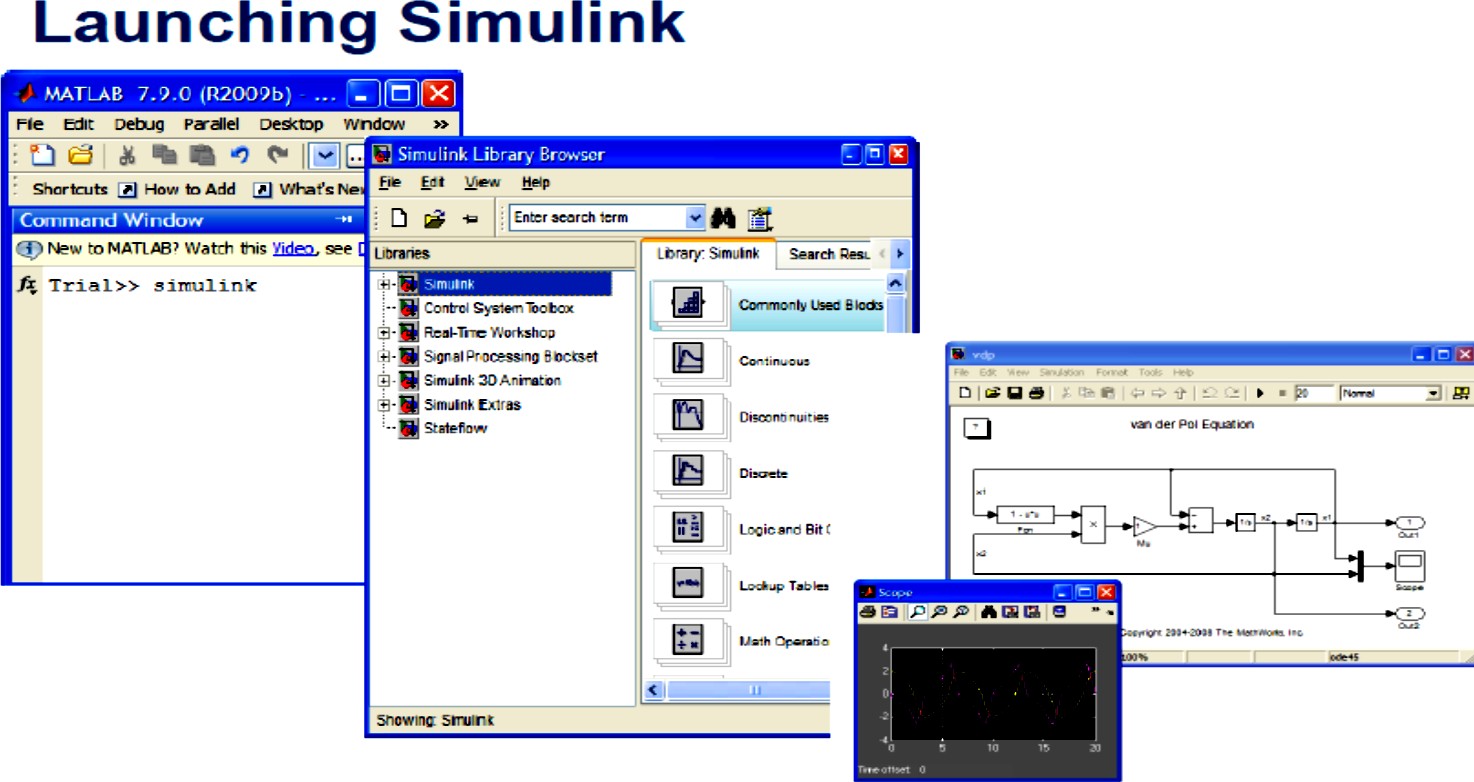
Simulink is a software package for modeling, simulating, and analyzing dynamical systems

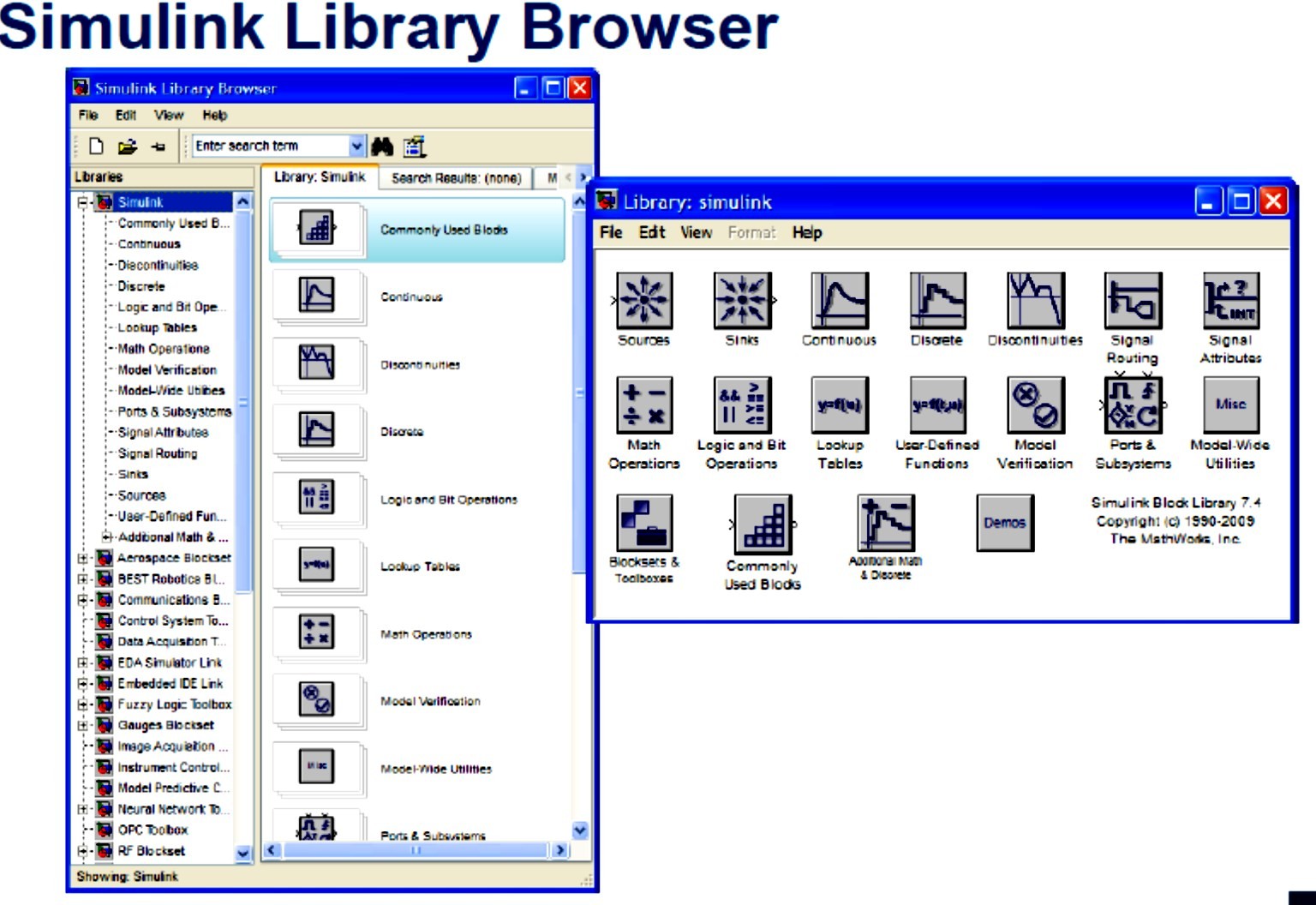
* Block diagram editing
* Nonlinear simulation
* Hybrid (continuous and discrete) models
* Asynchronous (non-uniform sampling) simulation
* Fully integrated with MATLAB, MATLAB toolboxes and blocksets.



Simulink accurately designs, implements, and tests:

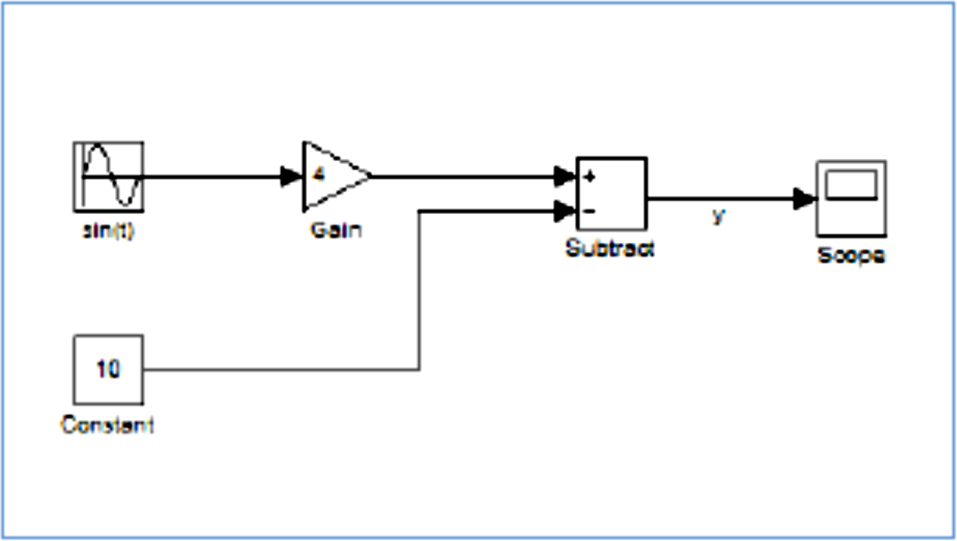
* + Control systems
  + Signal Processing systems
  + Communications systems
  + Embedded systems
  + Physical systems
  + other Dynamical systems



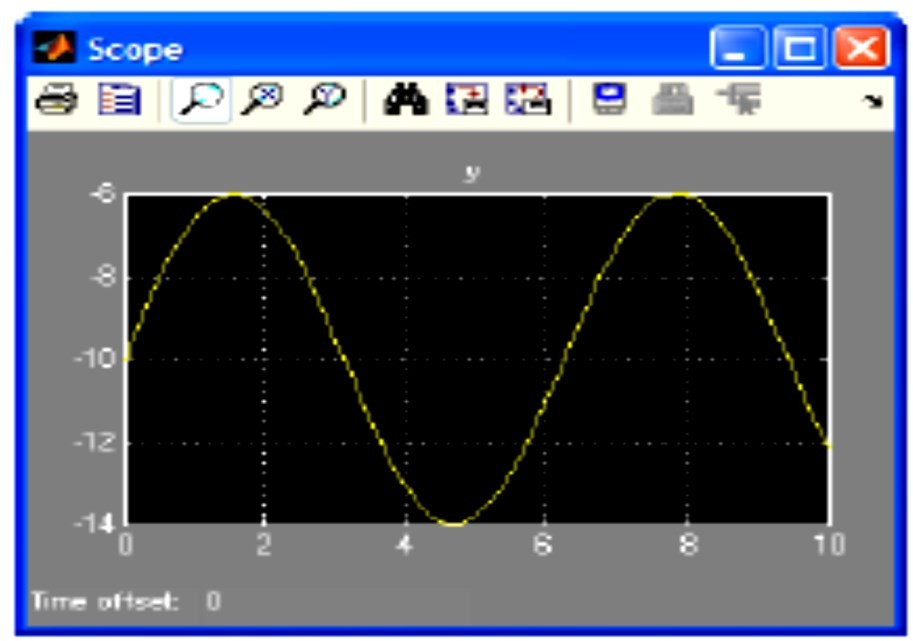


## Demonstration: Working with a simple model

Create a Simulink model to generate a sinusoidal waveform given by:  Simulink Block Diagram:



Output:



Conclusion:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Attempt the following and attach at the end of the experiment:

* 1. Explain any 5 commands/syntax used in MATLAB (other than already mentioned in the report) and give a simple example how to use them. (Do not use basic syntax such as clc, clear all, etc). You may create a small code also to explain. (5 marks)
  2. Explain any three components/block from the Simulink library. You may also add the screenshot of the properties of the block. (5 marks)

Experiment No:-04

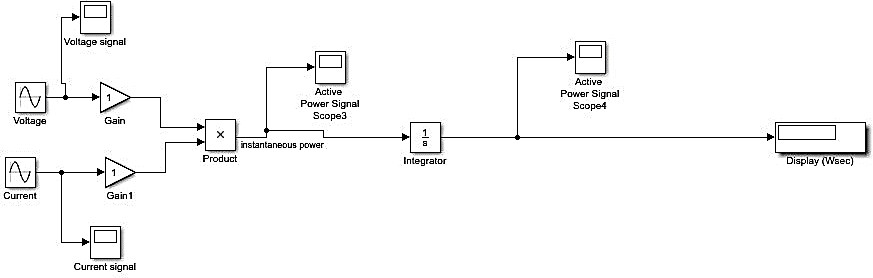
Design of smart meter for recording the electricity consumption

Aim: To create a simulink model of smart energy meter to understand its functioning.

Theory: Students must write a half page concept about what are smart energy meters, its

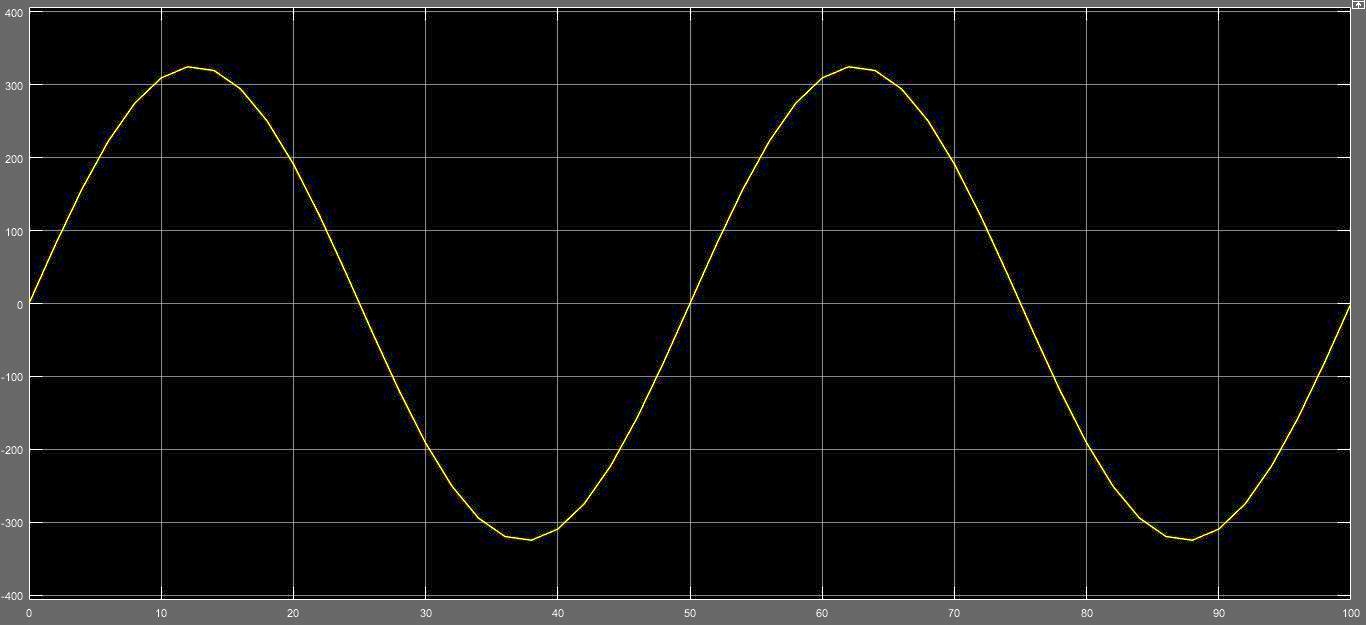
working (3 marks)

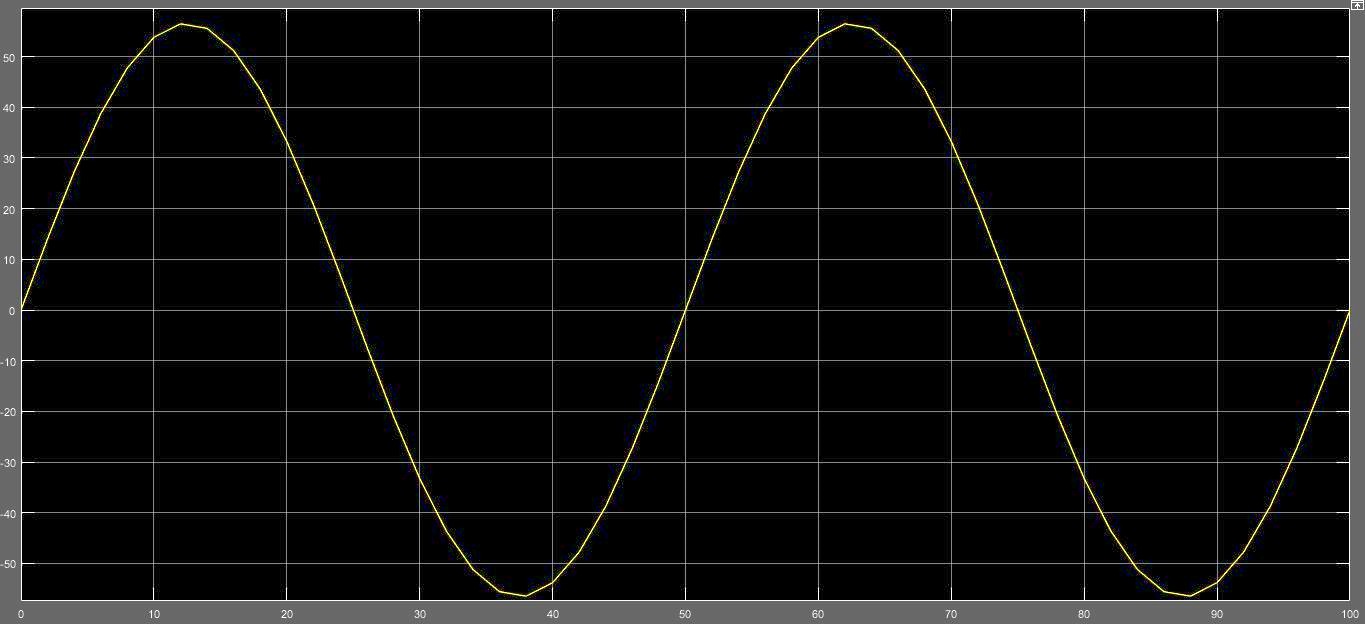
Simulink diagram:



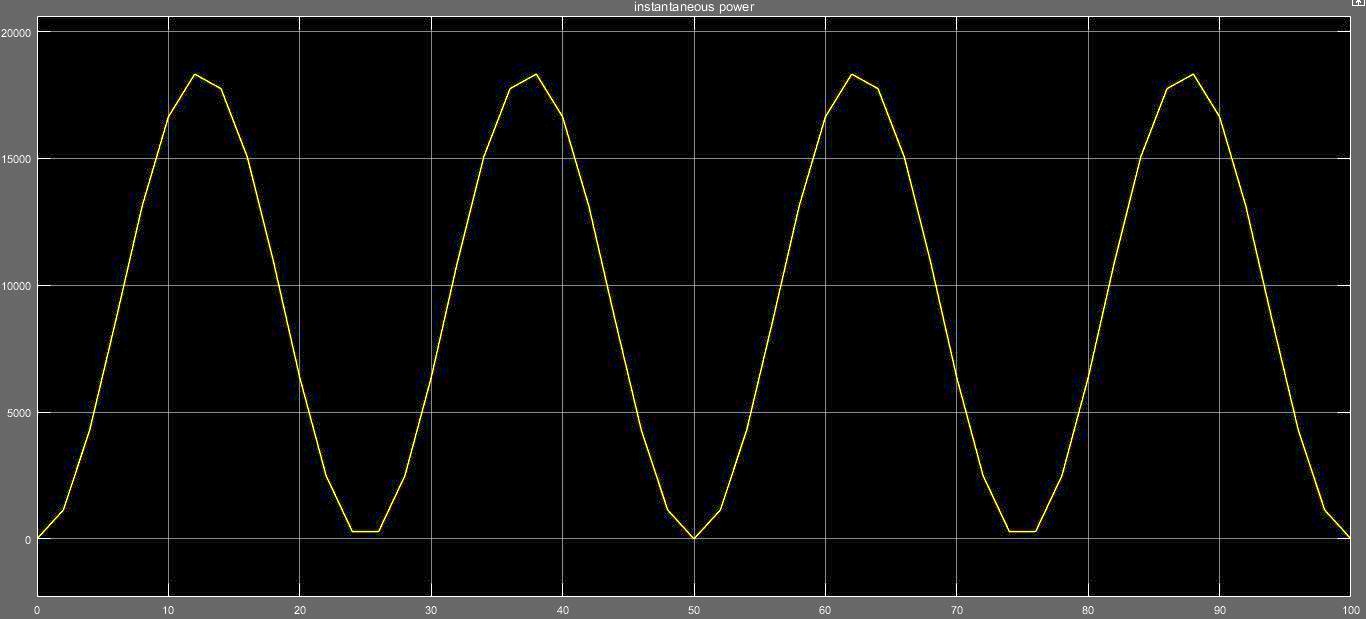
### Input parameters:

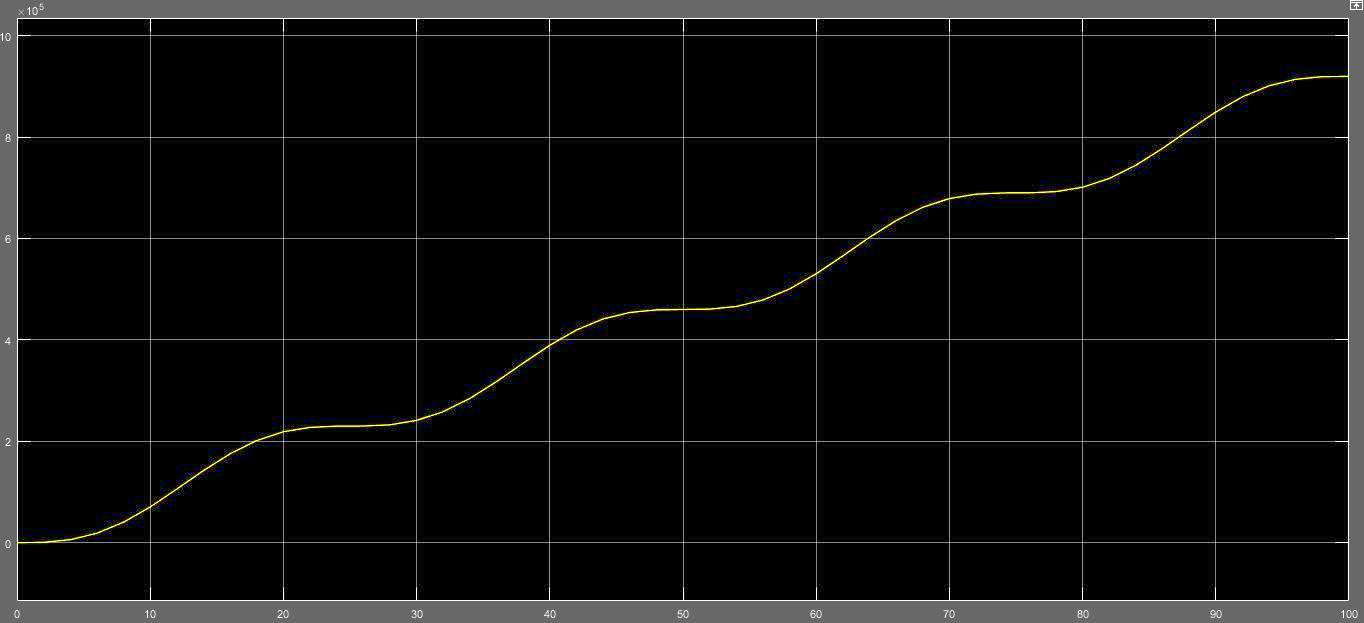
Voltage = 230 V single phase Current = 56.56 A

Voltage waveform from scope:

Current waveform from scope:

Active power waveform from scope3:



Active power waveform from scope4:

### Working Principle:

--------Students must explain the logic behind the model used in simulation. You may also explain some of the key blocks/components in the model (2 marks)

Conclusion:

Attempt the following and attach at the end of the experiment:

* 1. Explain the advantages of smart energy meters over conventional meters. (3 marks)
  2. What are the challenges faced in implementing/installing smart energy meters on a wide scale (2 marks)

Experiment No:-05

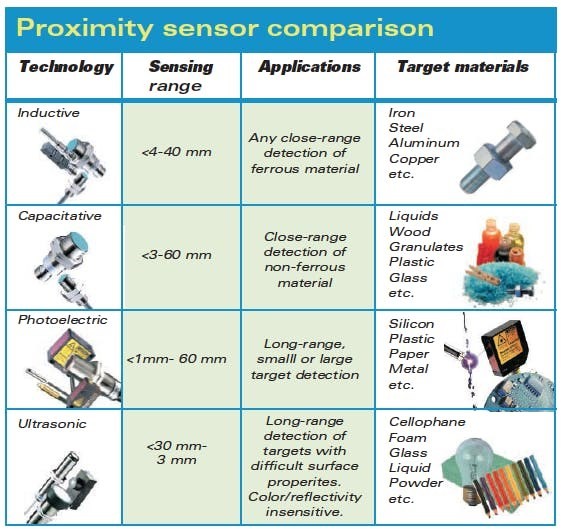
Design of Ultrasonic proximity sensors with Arduino

Aim: To design an ultrasonic proximity sensor using Arduino.

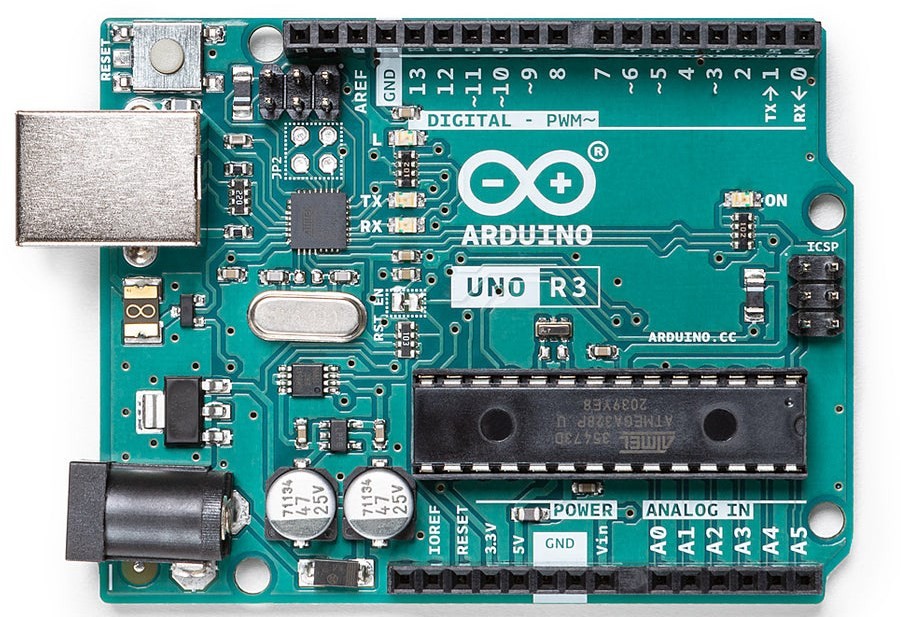
Components: Arduino Uno, HC-SR04 proximity sensor, breadboard, jumper wires, small object for demonstration (pen, ball or disk)

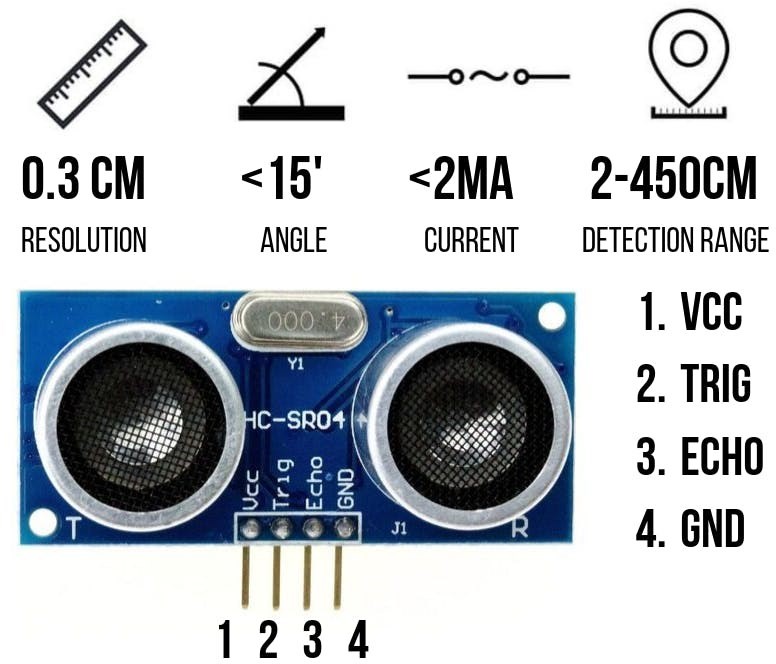
Theory: Students must write a half page concept about what are proximity sensors, their

working and types (2 marks)



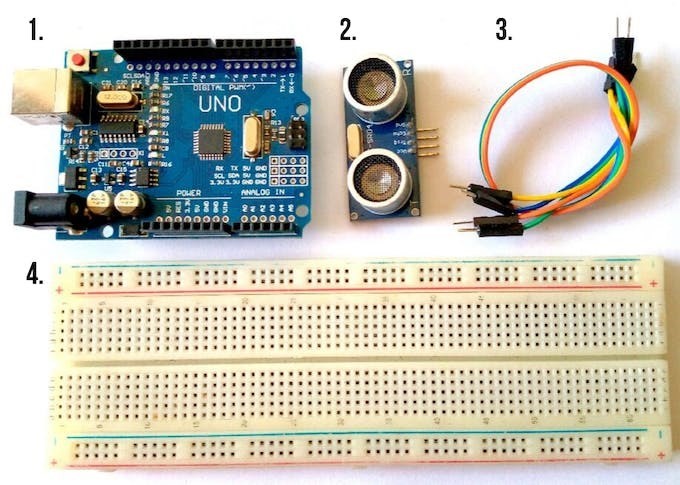
# Components and Supplies



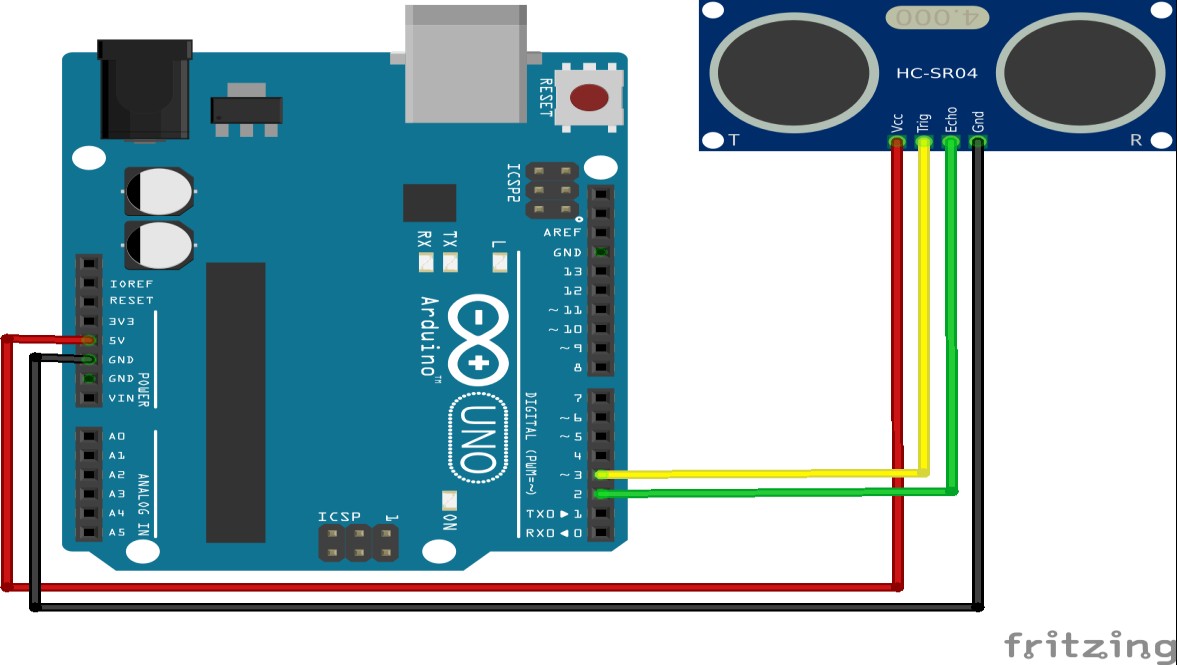
Arduino Uno R3

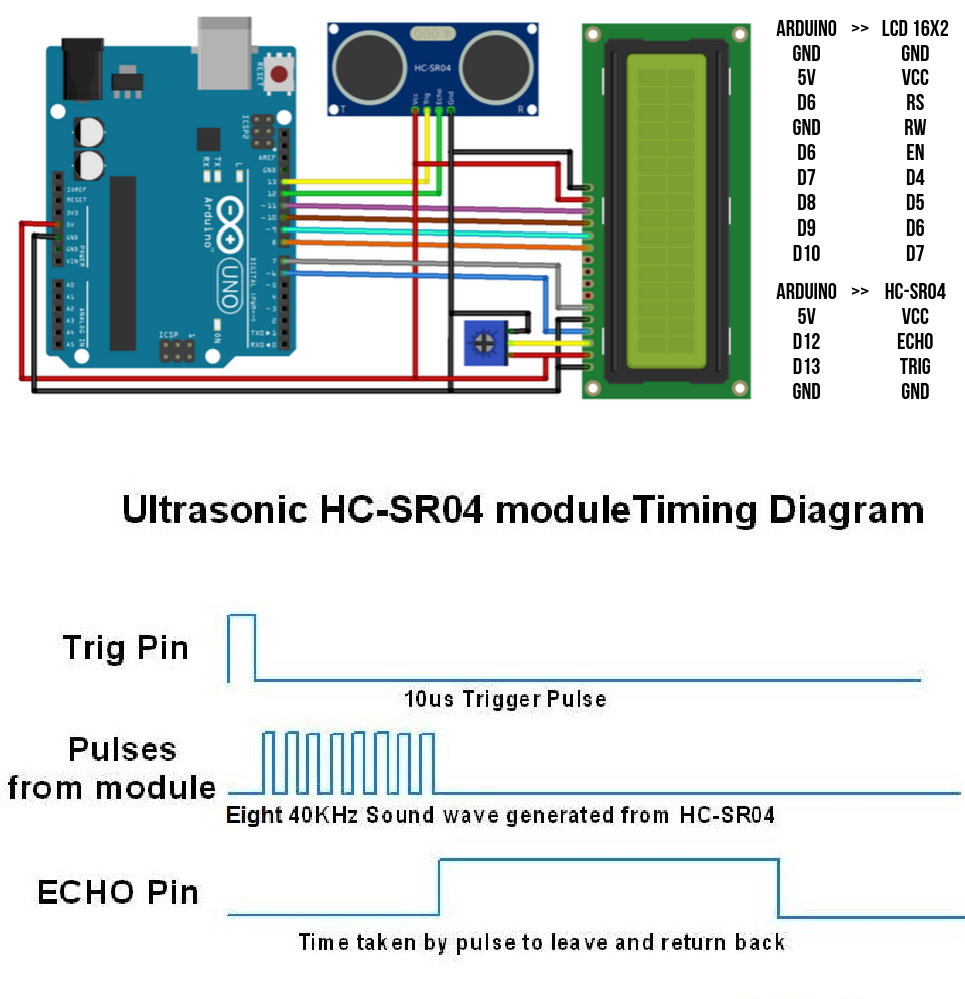
Ultrasonic Distance Sensor - HC-SR04

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40000 Hz (40 kHz) which travels through the air and if there is an object or obstacle on its path, it will bounce back to the module. Considering the **travel time and the speed** of the sound you can calculate the distance.

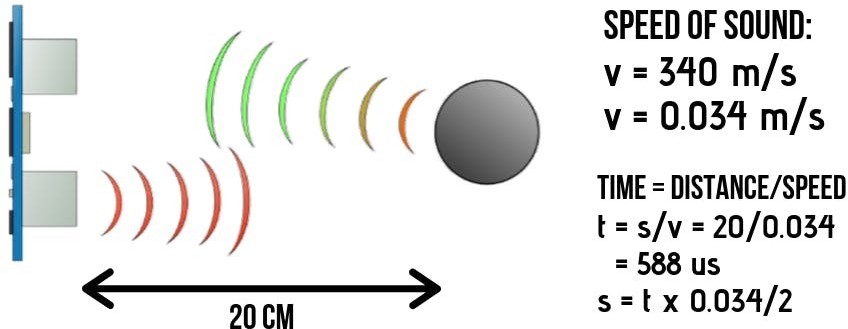


# Connection Diagram:





Distance Calculation



# Code implementation for sensor application:

## Code:

#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04 #define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04

// defines variables

long duration; // variable for the duration of sound wave travel int distance; // variable for the distance measurement

void setup() {

pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor Serial.println("with Arduino UNO R3");

}

void loop() {

// Clears the trigPin condition digitalWrite(trigPin, LOW); delayMicroseconds(2);

// Sets the trigPin HIGH (ACTIVE) for 10 microseconds digitalWrite(trigPin, HIGH);

delayMicroseconds(10); digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance = duration \* 0.034 / 2; // Speed of sound wave divided by 2 (go and back)

// Displays the distance on the Serial Monitor

Serial.print("Distance: "); Serial.print(distance); Serial.println(" cm");

# Steps:

1. First do the wiring as shown in the picture
2. Open Arduino IDE Software and write down your code, or download the code below and open it
3. Choose your own Arduino board (in this case Arduino Uno), by selecting Tools > Board > Arduino/Geniuno Uno
4. Choose your COM Port (usually it appears only one existing port), Tools > Port > COM.. (If there are more than one ports, try it one by one)
5. Upload your code by pressing Ctrl + U or Sketch > Upload
6. To display the measurement data you can use Serial Monitor by pressing Ctrl + Shift + M (make sure that the baudrate speed is 9600)

# Conclusion:

Attempt the following and attach at the end of the experiment:

Q.1 Explain practical application of proximity sensor; one specific for your domain engineering (Electrical, CSE, Chemical appropriately) and one for day to day common usage. (6 marks) (Maximum one page limit)