Lab Report

Experiment No: 1

Experiment Name: Simulation of Series and Parallel Circuit (DC & AC) by using Proteus Software

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

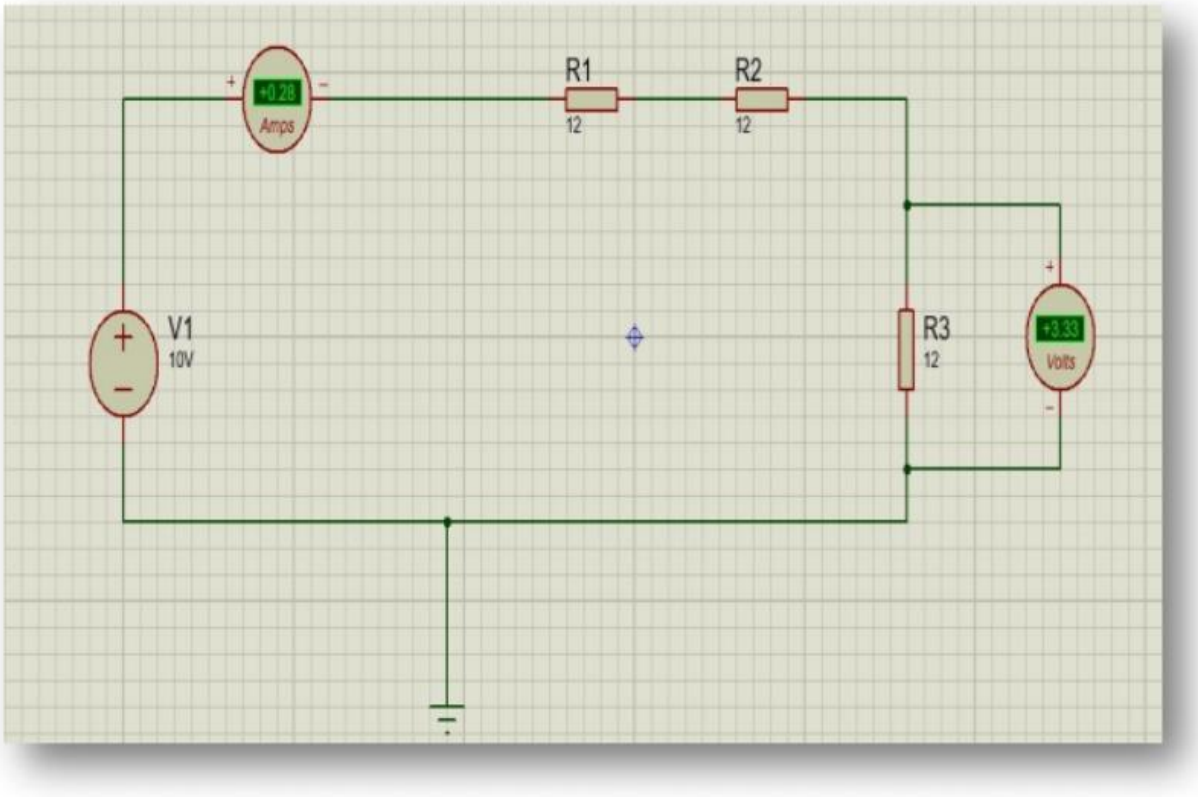
Proteus Design Suite : It's a software which is widely used by electronic design engineers and technicians to create electronic prints and schematic for producing printed circuit boards. It's primarily used for electronic design automation.

Apparatus:

1. Vsource
2. DC Ammeter
3. DC Voltmeter
4. Resistance
5. Wire
6. Ground

Circuits :

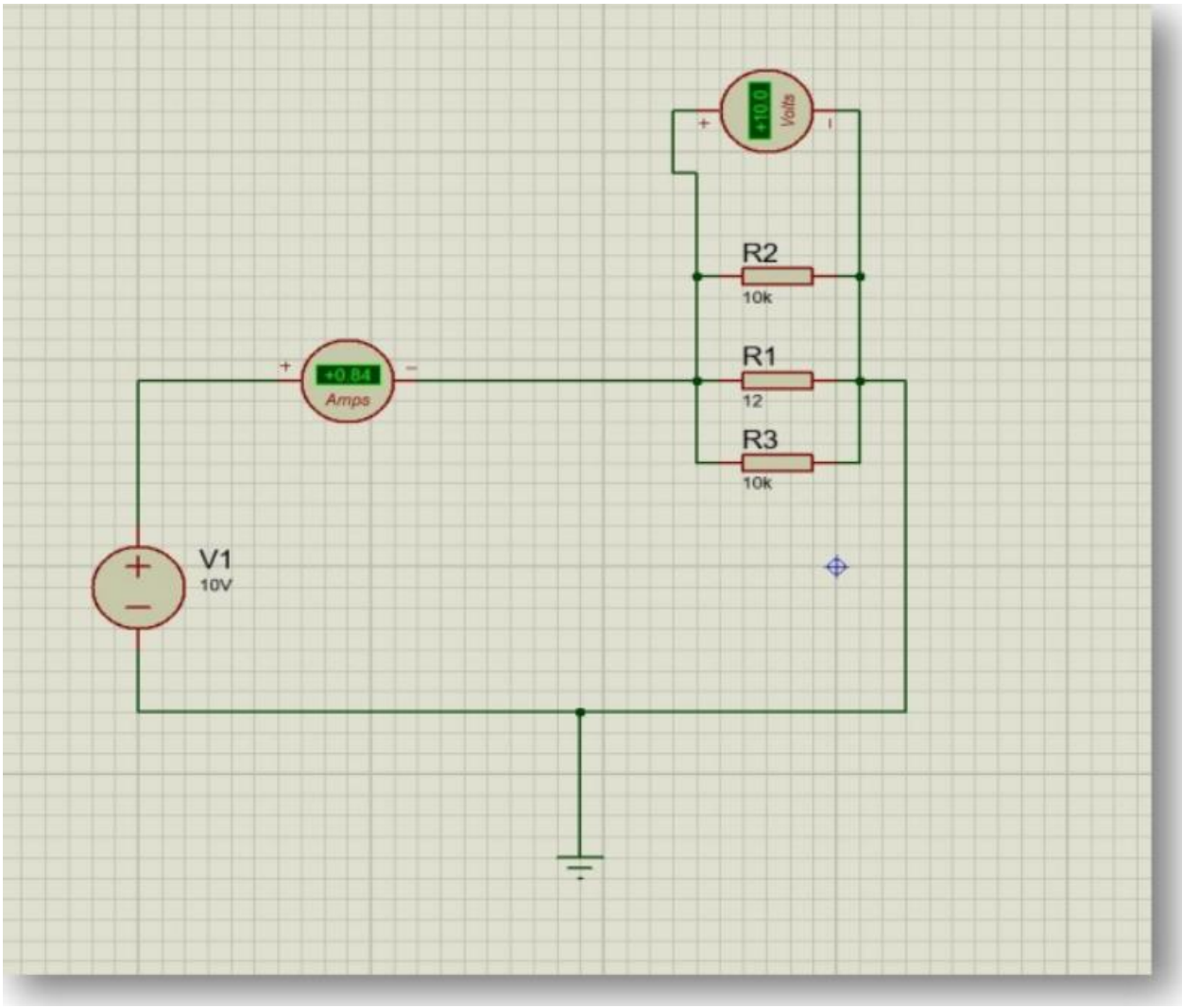
Series DC Circuit:



Discussion:

i)Here R1,R2 and R3 resistors are in series.DC Ammeter is connected is series and
DC Volt-meter is connected in parallel with R3 otherwise it will not work and the
source is DC.
ii)To connect new components or to access the current components make sure the simulation is off else the components can't be accessed.

Parallel DC Circuit:

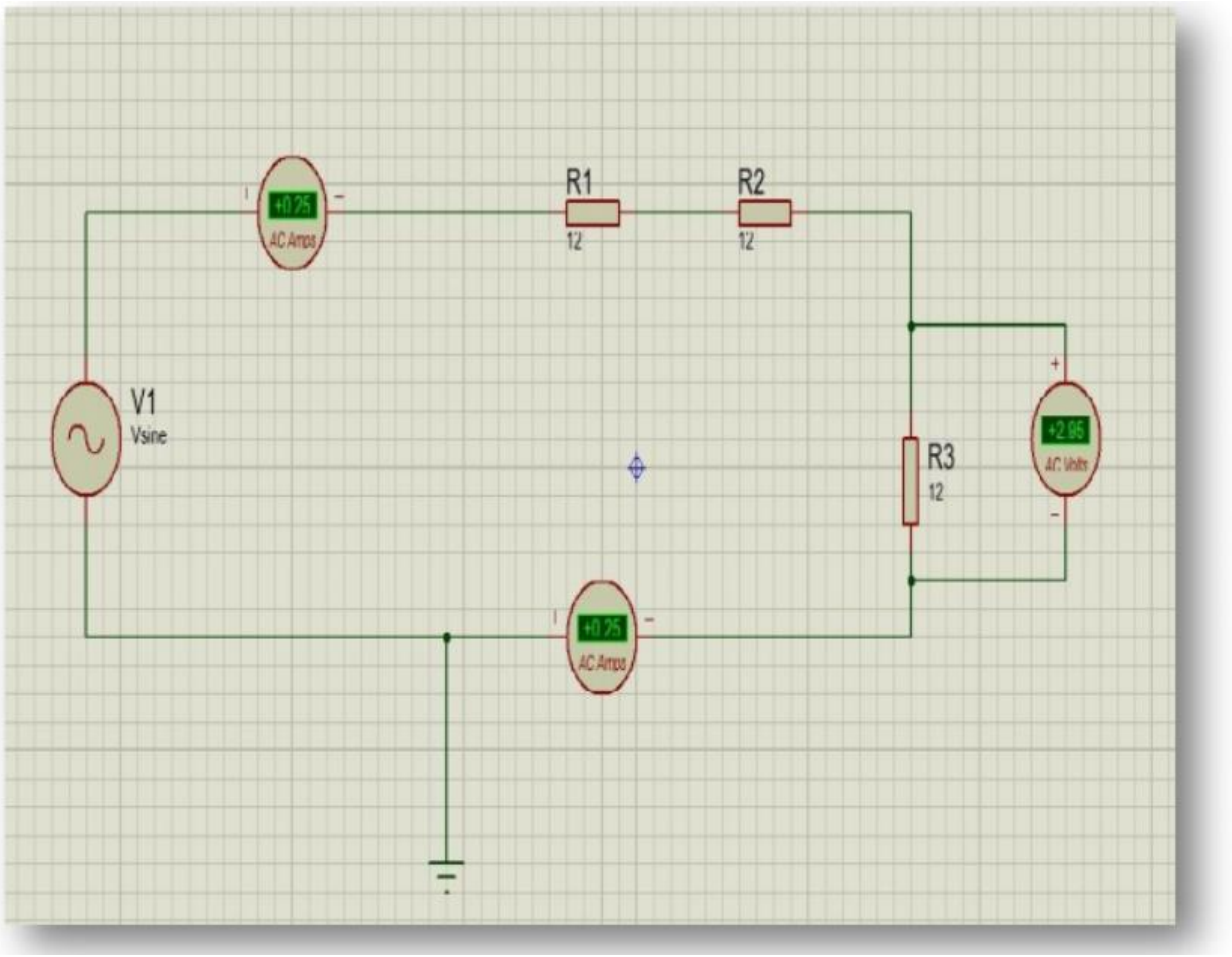


Discussion:

i)Here R1,R2 and R3 resistors are in Parallel.DC Ammeter is connected is series and DC Volt-meter is connected in parallel with R2 otherwise it will not work and
the source is DC.

ii)To connect new components or to access the current components make sure the simulation is off else the components can't be accessed.

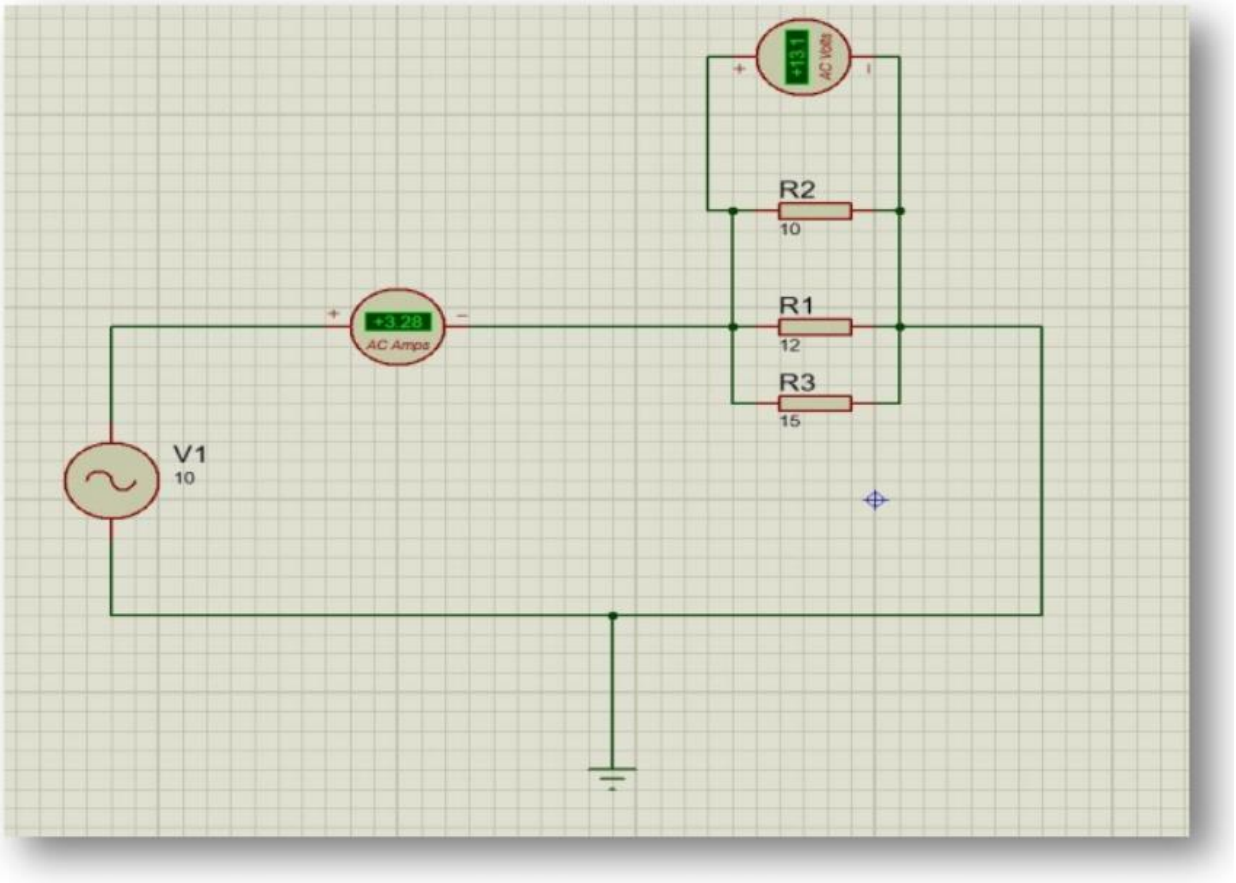
Series AC Circuit:



Discussion:

i)Here R1,R2 and R3 resistors are in Series.AC Ammeter is connected is series and
AC Volt-meter is connected in parallel with R3 otherwise it will not work and the
source is AC.
ii)To connect new components or to access the current components make sure the simulation is off else the components can't be accessed.

Parallel AC Circuit:



Discussion:

i)Here R1,R2 and R3 resistors are in Parallel. AC Ammeter is connected is series and AC Volt-meter is connected in parallel with R2 otherwise it will not work and the source is AC.

ii)To connect new components or to access the current components make sure the simulation is off else the components can't be accessed.

Description :

PCB Layout: it is a high-level engineering tool for board design featuring smart manual routing of high-speed and differential signals, shape-based autorouter, advanced verification, and wide import/export capabilities.

Schematic Capture: Schematic Capture is an advanced circuit design tool with support of multi-sheet and multi-level hierarchical schematics.

Volt-Meter: It is used to measure the DC or AC voltage across any two points of electric circuit. It has to be connected in parallel with the component.

Ammeter: it is used to measure either direct or alternating electric current, in amperes. It has to be connected in parallel with the component.

DC Source: it's a power supply that supplies a constant DC voltage to its load.

AC Source: it is a device that is capable of supplying variable power and frequency to a load.

Resistor: Resistor is an electrical component that reduces the electric current. The resistor's ability to reduce the current is called resistance and is measured in units of ohms (symbol: Ω)

Ground Connection: The purpose of a ground wire is to give excess electrical charges a safe place to go.

Lab Report

Experiment No: 2

Experiment Name: Basic transformer simulation using Proteus Software (OC test and SC test).

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

Objective:

1. To measure the voltage and current of primary and secondary side of a transformer during open circuit and short circuit test.
2. Graph analysis according to open circuit and short circuit test of a transformer.

Apparatus:

1. Proteus 8 Professional Software
2. Simple Transformer
3. AC Voltmeter
4. AC Ammeter
5. AC Voltage Source
6. Voltage Probes

Description:

Transformer:

The transformer is a device that transfers electrical energy from one electrical circuit to another electrical circuit. The two circuits may be operating at different voltage levels but always work at the same frequency. Basically transformer is an electro-magnetic energy conversion device. It is commonly used in electrical power system and distribution systems. It can change the magnitude of alternating voltage or current from one value to another.
So we can say, transformer is a device that in used for increase and decrease the value of voltage without change in frequency.
transformer work only on ac supply because A transformer needs an alternating current that will create a changing magnetic field. A changing magnetic field also induces a changing voltage in a coil. This is the basis of how a transformer works. The primary coil is connected to an AC supply.

Open Circuit Test:

The purpose of the open-circuit test is to determine the no-load current and no-load loss or core loss of the transformer because of which their no-load parameters are determined. This test is performed on the primary winding of the transformer. The wattmeter, ammeter and the voltage are connected to their primary winding. The nominal rated voltage is supplied to their primary winding with the help of the ac source. The secondary winding of the transformer is kept open, and the voltmeter is connected to their terminal. This voltmeter measures the secondary induced voltage. As the secondary of the transformer is open, thus no-load current flows through the primary winding.

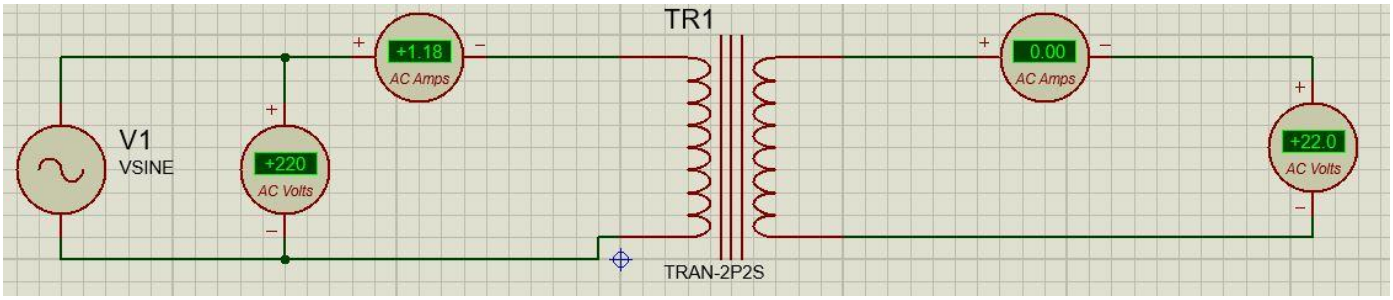
Short Circuit Test:

The short circuit test is performed for determining the copper loss occur on the full load and the equivalent resistance, impedance, and leakage reactance are known by the short circuit test.
The short circuit test is performed on the secondary or high voltage winding of the transformer. The measuring instrument like wattmeter, voltmeter and ammeter are connected to the high voltage winding of the transformer.
Their primary winding is short-circuited by the help of thick strip or ammeter which is connected to its terminal. The low voltage source is connected across the secondary winding because of which the full load current flows from both the secondary and the primary winding of the transformer. The full load current is measured by the ammeter connected across their secondary winding.

In the open circuit test current is the minimum and voltage is the maximum where in the short circuit test current is the maximum and voltage is the minimum.

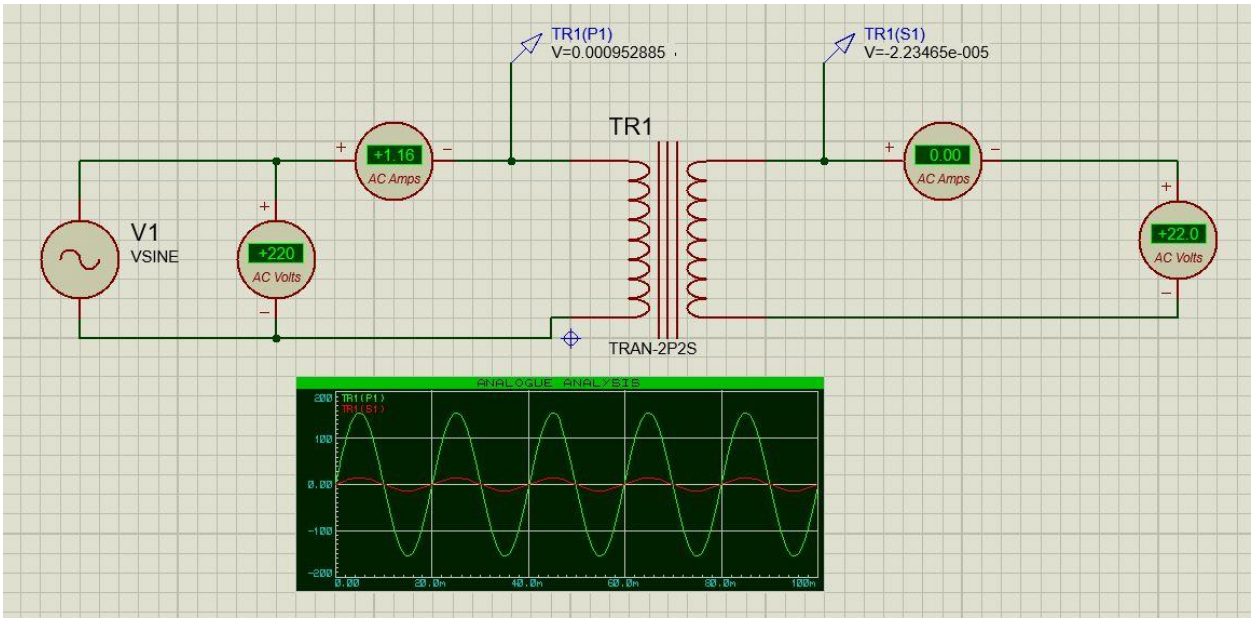
Circuit Diagram:

Open Circuit test on step down transformer:



This is a simple transformer circuit and we are doing open circuit test here.
We gave 311 amplitude and 50Hz frequency in the ac voltage voltmeter in the primary side of the transformer. We know Voltmeters used for AC voltages always show RMS value which means the AC voltmeter we used in the circuit will show the RMS value.
The RMS value = A/√2; Where A is the amplitude, So the RMS value which will show in the ac voltmeter is 311/√2 or 220 volts. And we can see the reading of the voltmeter is 220 volts and the current in the primary side of the transformer is 1.18amp. V1 is 220 volts and coupling factor k=0.1 so the voltmeter reading in the secondary side should be, 220*0.1= 22.0 volts and we can see the voltmeter readings in the secondary side is 22.0 volts and current in the secondary side is 0 amp because in the open circuit current is the minimum (0 amp) and voltage is the maximum. Since the circuit is open so the current will be minimum means 0 Amp.

Graphical analysis of open circuit test:

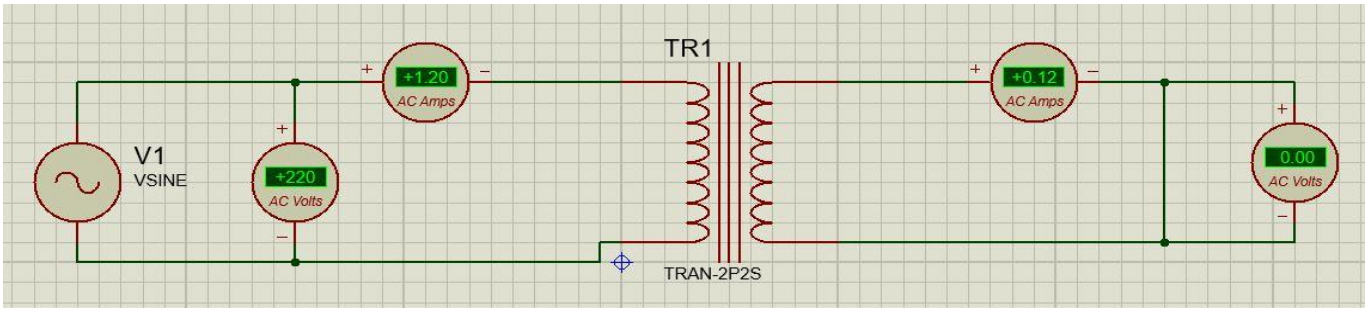


The time taken for an AC Waveform to complete one full pattern from its positive half to its negative half and back to its zero baseline again is called a Cycle and one complete cycle contains both a positive half-cycle and a negative half-cycle. The time taken by the waveform to complete one full cycle is called the Periodic Time of the waveform and the number of complete cycles that are produced within one second (cycles/second) is called the Frequency. We know in transformer frequency is constant and in the graph we can see a constant frequency. Another important parameter of the AC waveform is Amplitude, better known as its Maximum or Peak value represented by the terms, Vmax for voltage or Imax for current. The peak value is the greatest value of either voltage or current that the waveform reaches during each half cycle measured from the zero baseline. For pure sinusoidal waveforms this peak value will always be the same for both half cycles.

The Graph represents two types of wave form green one is the sine wave of primary side and the red one is the sine wave of secondary side. The Horizontal indicates time and the vertical line indicates voltage.

We get the max voltage in the primary side of the transformer is 220 volts and the secondary side is 22.0 volts. That's why we can see in the graph max voltage of the green sine wave is greater than the red sine wave through this we get to know the difference very easily.

Short Circuit test on step down transformer:

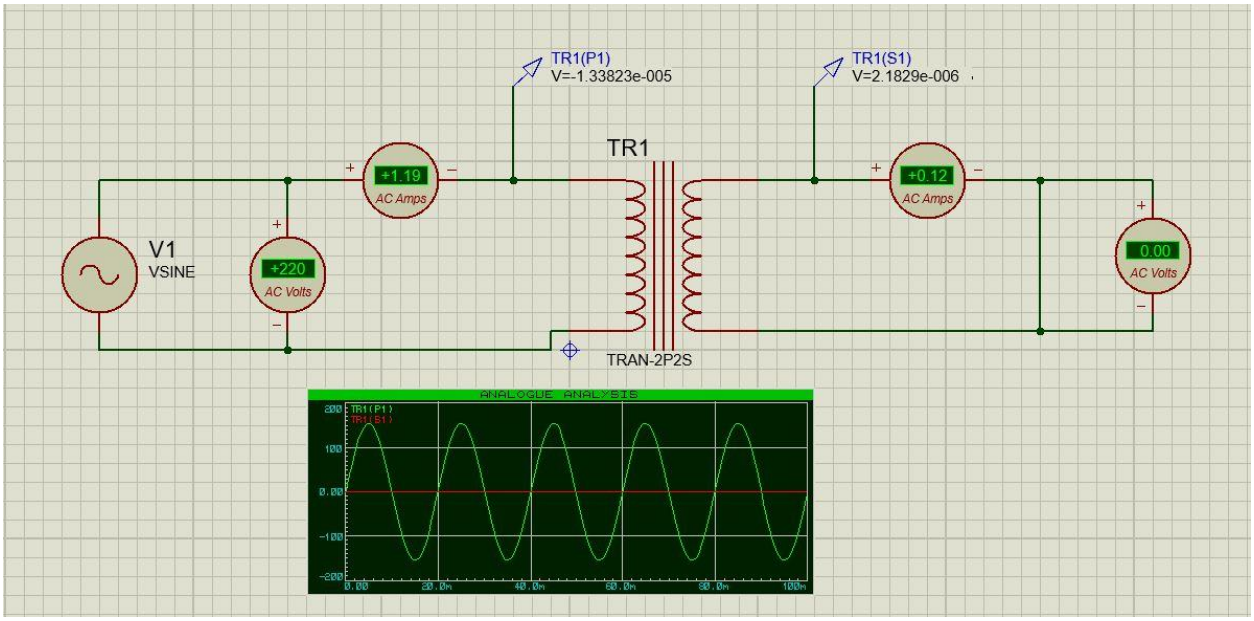


Now we are doing short circuit test in the transformer. We have short circuited by a wire in the secondary side. We gave 311 amplitude and 50hz frequency in the ac voltage voltmeter in the primary side of the transformer. As Voltmeters used for AC voltages always show RMS value which means the RMS value which will show in the ac voltmeter is 311/√2 or 220 volts. And we can see the reading of the voltmeter is 220 volts and the current in the primary side of the transformer is 1.20amp.

Here the coupling factor k on the transformer is 1/10 means 0.1. V1 is 220 volts and coupling factor k=0.1 so the voltmeter reading in the secondary side should be, 220*0.1= 22.0 volts but we can see the voltmeter readings in the secondary side is 0 volts and current in the secondary side is 0.12 amp because in the short circuit current is the maximum and voltage is the minimum (0 volts). Since the circuit is short circuited so the voltage will be minimum means 0 volts.

Graphical analysis of short circuit test:

The Graph represents two types of wave form green one is the sine wave of primary side and the red one is the sine wave of secondary side. The Horizontal indicates time and the vertical line indicates maximum voltage. We get the max voltage in the primary side of the transformer is 220 volts and the secondary side is 0 volts. That's why we can see in the graph max voltage of the green sine wave but where the red line graph is the constant 0 which indicates the 0 volts of short circuit test.



Discussion:

1. We can't change the value of any component when we run the simulation we can change it only when we stop the simulation.
2. Voltmeter must be connected in parallel and ammeter must be in series otherwise we won't get any readings.
3. In open-circuit test, the final value of the input voltage should be the rated voltage, while for short-circuit test the input voltage is increased until the rated currents flow in the primary and secondary of the transformer.
4. The connection of the wires must be done properly in the circuit.

Conclusion:

i) We get to know how to use components in a software to measure the voltage and current of a transformer.
ii) We have to check the connection properly before circuit simulation to avoid errors.
iii) This experiment is very necessary because through this we get to know the accurate result of voltage and current during open circuit test and the short circuit test of a transformer.
iv) We get to know how a transformer simulate during open circuit test
and short circuit test.

Lab Report

Experiment No: 3

Experiment Name: Simulation of DC motor using Proteus software.

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

Objective:

1. To measure the DC Motor rating.
2. To measure the current and voltage corresponding of a motor.

Apparatus:

1. Proteus 8 Professional Software.
2. DC Motor
3. DC Voltmeter
4. DC Ammeter
5. DC Voltage Source
6. Ground

Description:

DC Motor:

An Electrical motor is a machine which converts electrical energy into mechanical energy. Its action is based on the principle that when a current carrying conductor is placed in a magnetic field, it experiences a mechanical force whose direction is given by Fleming's Left-Hand rule and whose magnitudes given by Newton.
DC motors have the advantage of higher starting torque, quick starting and stopping, reversing, variable speeds with voltage input and they are easier and cheaper to control than AC.
DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings.

Torque and Speed of a DC motor:

The term speed regulation refers to the change in speed of a motor with change in applied load torque, other conditions remaining constant. It is defined as the change in speed when the load on the motor is reduced from rated value to zero, expressed as the percent load of the rated load speed. It will be proved that though torque of a motor is admittedly a function of flux and armature current,

yet it is independent of speed. In fact, it is the speed which depends on torque and not vice-versa.
We know, Ta $\propto$ Φla. Here increase in flux would decrease the speed but increase the armature torque. If torque increases, motor speed must be increase rather than decrease.

DC Ammeter:

Measures the amount of current flow through the circuit. DC Ammeter use to measure direct current. Ammeter should always be connected in Series Load.

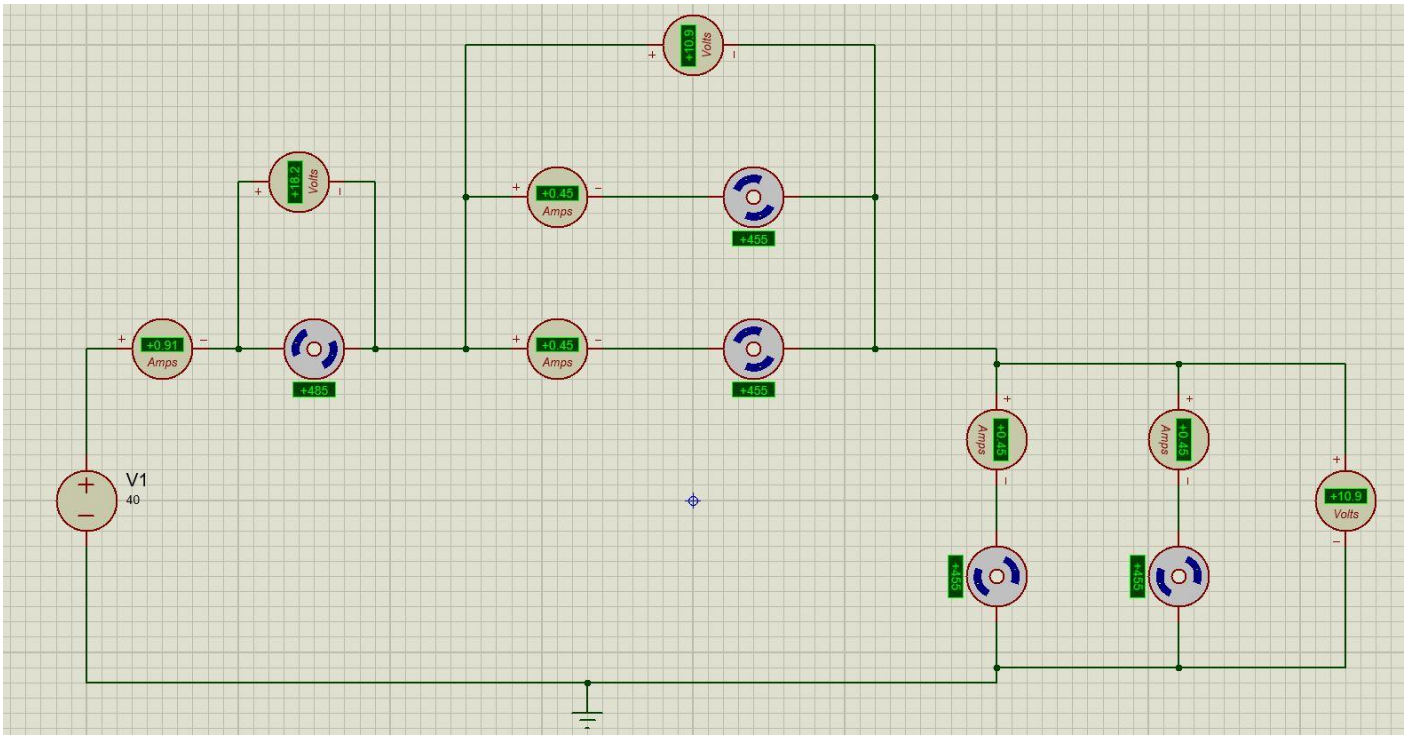
DC Voltemeter:

Use to measuer Electrical potential difference between two points in an electric circuit. DC Voltmeter we use to measure DC voltage. Voltmeter must be connected in parallel with load.

RPM of DC Motor:

The most conventional unit of measuring the speed of a motor is in RPM.
This stands for Revolutions Per Minute or the number if times the motor's shaft will rotate in one minute.

Circuit Diagram:



Here in this circuit we are using 5 DC motors and DC voltage source is 40V and finding the current and voltage according the motors. There are two branches of two motors whose are in parallel and one motor is in series with the circuit.
In DC Motor, the speed(RPM) is proportional to applied voltage. The current that the motor will take from the supply is proportional to load. A motor with no load will take very little current, a motor which is loaded and doing some work will take more current. The load on the motor appears as a torque on the output shaft, the more load, the more torque.
When we simulate the circuit in the proteus software the motor starts rotating and we can notice that the more it is rotating, the more current is decreasing. The RPM of the motor depends on how much the motors are rotating per minute. The more increase in rotation, the more increase in RPM of the motor and this will increase the load. After setting the motor speed by setting the supply voltage, if we increase the load(torque) on the motor, the current taken from the supply will increase.

Here for the first motor which is in the series with the circuit we can see the motor RPM is 485 and the nominal voltage is 15V and load torque is 60Nm.

Here maximum voltage according to the first motor is 18.2V and current across the motor is 0.91 Amp.

We also see there are two motors whose are in parallel with the circuit. The RPM of the parallel motors is 455 and nominal voltage is 12V and load torque is 50Nm. Here maximum voltage according to the motors is 10.9V and current is 0.45 Amp.

Again there are two motors whose are in parallel with the circuit. The RPM of the parallel motors is 455 and nominal voltage is 12V and load torque is 50Nm. Here maximum voltage according to the motors is 10.9V and current is 0.45 Amp.

So, the total voltage according to the 5 motors = 18.2 + 10.9 + 10.9 = 40V
Which is equal to the voltage with the DC Voltage source.

We can see the motors are still rotating but there is no more changes in the RPM so we can say the motors has reached to its rated RPM with rated Voltage so that we get the maximum voltage and current across the motors in the circuit.

Discussion:

i) We must connect the circuit with ground connection
ii) We've to be careful when connecting the devices.
iii) We can't change the value of any component when we run the simulation we can change it only when we stop the simulation.
iv) Voltmeter must be connect in parallel and ammeter must be in series otherwise we won't get any readings

Conclusion:

1. We get to know how to use components in a software to measure the voltage and current of a DC Motor as well as RPM of a motor.
2. We have to check the connection properly before circuit simulation to avoid errors.
3. This experiment is very necessary because through this we get to know the accurate result of voltage and current of DC motor as well as RPM.
4. We get to know how a transformer simulate while DC motor is running.

Lab Report

Experiment No: 4

Experiment Name: Familiarization with Arduino Software and simple LED blinking

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

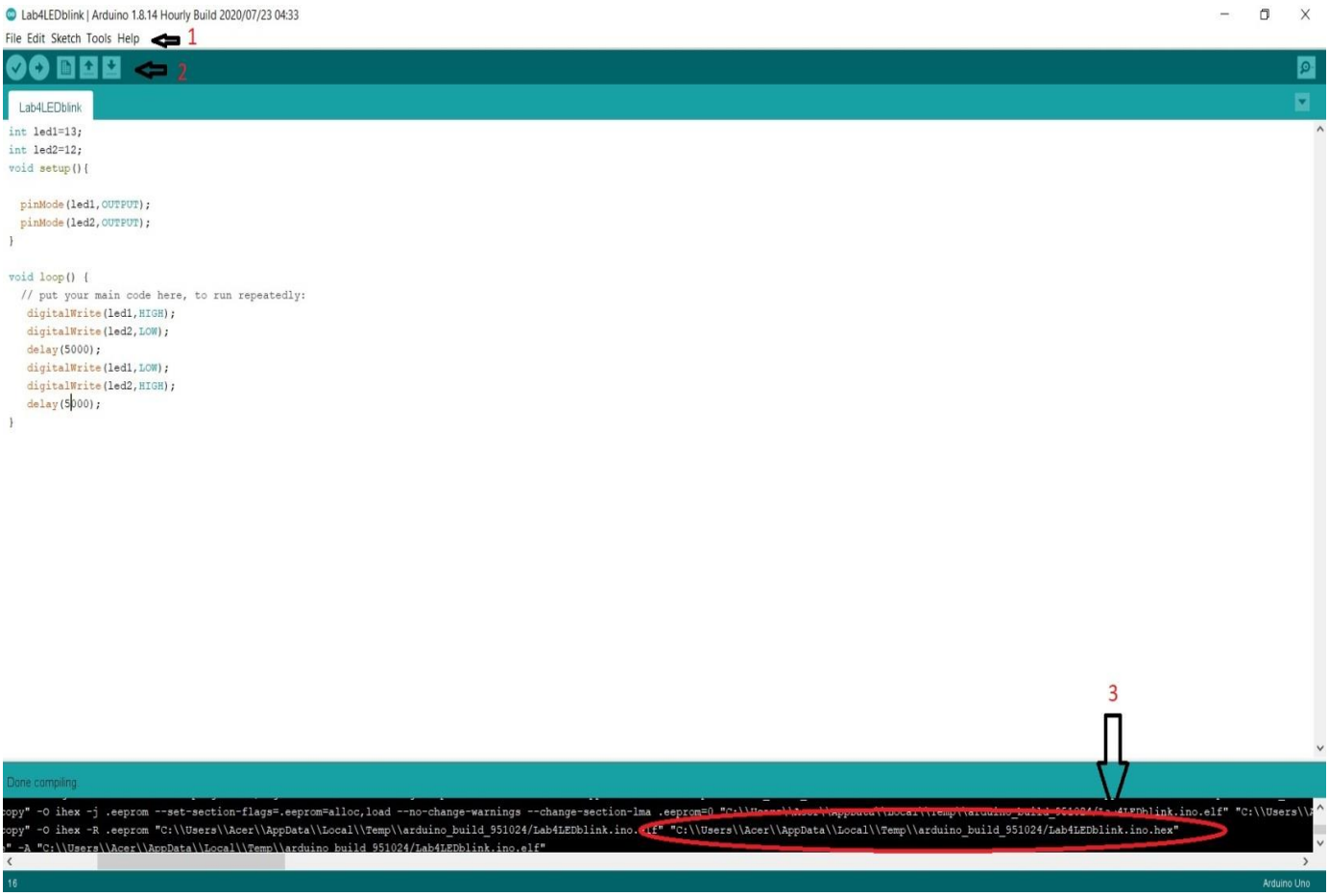
Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

Description:

Arduino IDE - is an open source software that is mainly used for writing and compiling the code into the Arduino Module. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board. The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. This environment supports both C and C++ languages.



In section 1 there’s File, Edit, Sketch, Tool, Help.

File

- *New*
  Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- *Open*
  Allows to load a sketch file browsing through the computer drives and folders.
- *Open Recent*
  Provides a short list of the most recent sketches, ready to be opened.
- *Sketchbook*
  Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- *Examples*
  Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- *Close*
  Closes the instance of the Arduino Software from which it is clicked.
- *Save*
  Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- *Save as...*
  Allows to save the current sketch with a different name.
- *Page Setup*
  It shows the Page Setup window for printing.
- *Print*
  Sends the current sketch to the printer according to the settings defined in Page Setup.
- *Preferences*
  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- *Quit*
  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

Edit

- *Undo/Redo*
  Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- *Cut*
  Removes the selected text from the editor and places it into the clipboard.
- *Copy*
  Duplicates the selected text in the editor and places it into the clipboard.
- *Copy for Forum*
  Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- *Copy as HTML*
  Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- *Paste*
  Puts the contents of the clipboard at the cursor position, in the editor.
- *Select All*
  Selects and highlights the whole content of the editor.
- *Comment/Uncomment*
  Puts or removes the // comment marker at the beginning of each selected line.
- *Increase/Decrease Indent*
  Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- *Find*
  Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- *Find Next*
  Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- *Find Previous*
  Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

- *Verify/Compile*
  Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- *Upload*
  Compiles and loads the binary file onto the configured board through the configured Port.
- *Upload Using Programmer*
  This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command must be executed.
- *Export Compiled Binary*
  Saves a .hex file that may be kept as archive or sent to the board using other tools.
- *Show Sketch Folder*
  Opens the current sketch folder.
- *Include Library*
  Adds a library to your sketch by inserting #include statements at the start of your code. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- *Add File...*
  Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

Tools

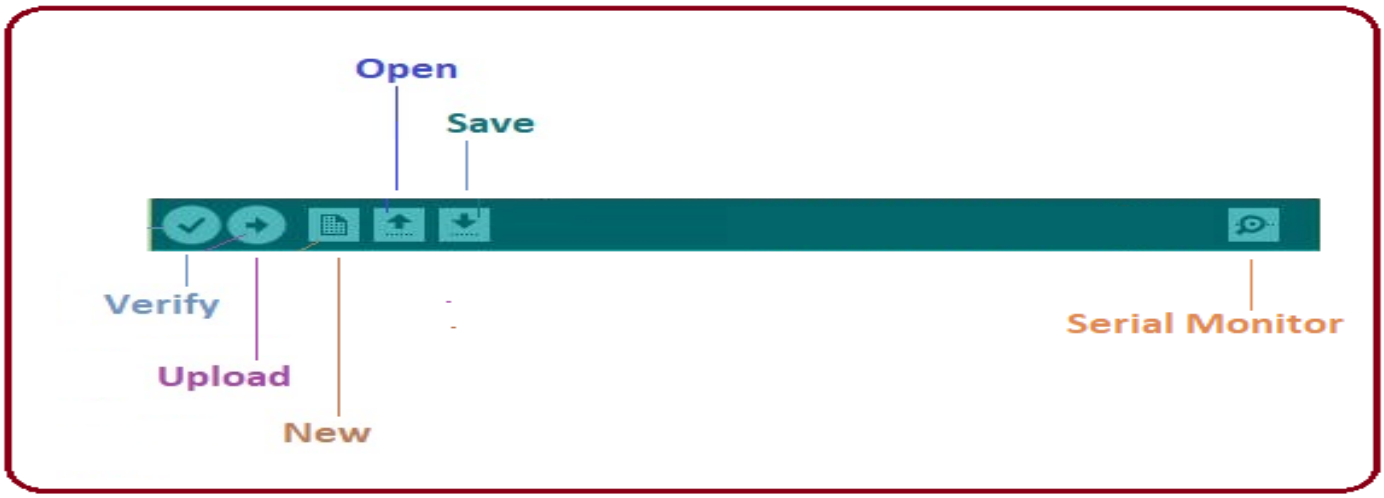
- *Auto Format*
  This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- *Archive Sketch*
  Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- *Fix Encoding & Reload*
  Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- *Serial Monitor*
  Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.
- *Board*
  Select the board that you're using. See below for descriptions of the various boards.
- *Port*
  This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- *Programmer*
  For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.
- *Burn Bootloader*
  The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- *Find in Reference*
  This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Section 2 contains 6 buttons:



>>The check mark appearing in the circular button is used to verify the code. Click this once you have written your code.
>>The arrow key will upload and transfer the required code to the Arduino board.
>>The dotted paper is used for creating a new file.
>>The upward arrow is reserved for opening an existing Arduino project.
>>The downward arrow us used to save the current running code.
>>The button appearing on the top right corner is a **Serial Monitor** - A separate pop-up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data.The Serial Monitor will actually help to

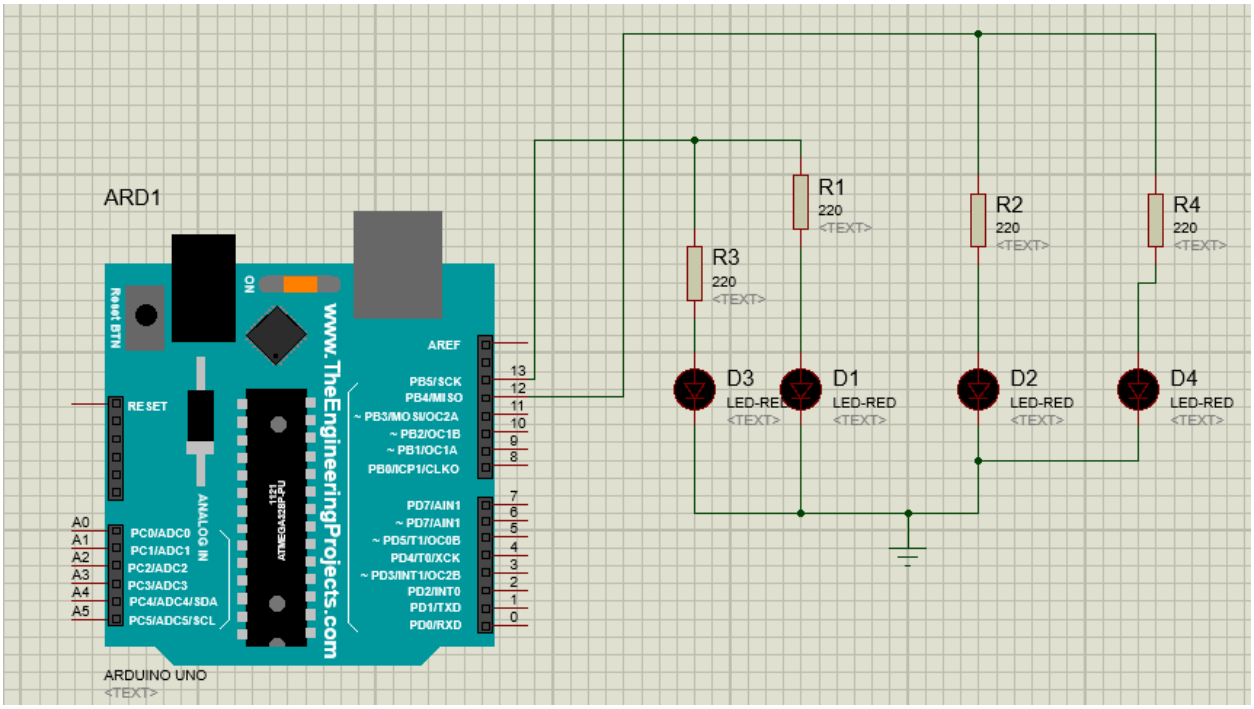
debug the written Sketches where you can get a hold of how your program is operating.

And section 3 is the path for the hex code that you will need to copy-paste on Proteus Arduino-uno simulation version.

Implementation of simple LED Blinking on Proteus Software:

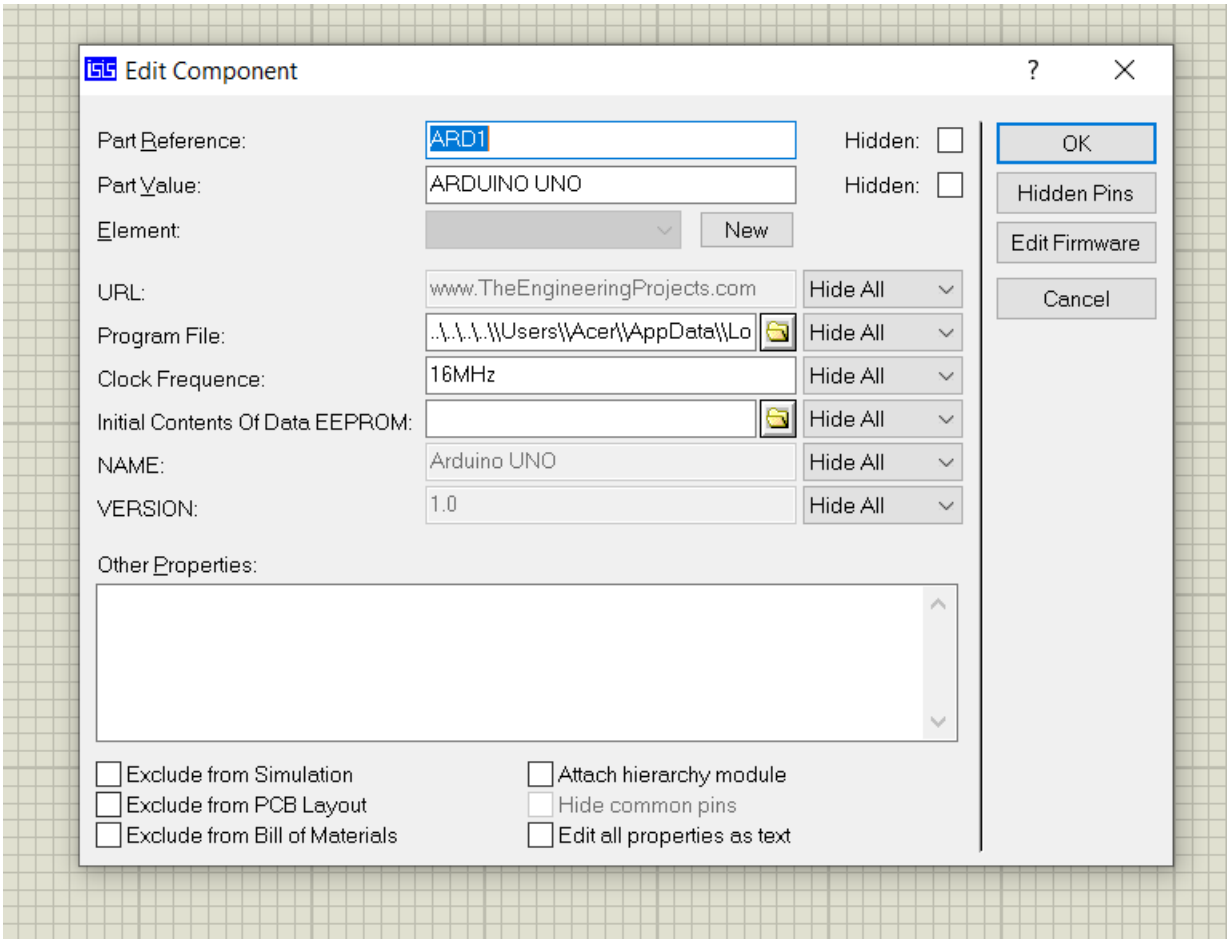
Apparatus:

1. Arduino UNO
2. LED-RED(D1,D2,D3,D4)
3. Resistor(R1,R2,R3,R4)
4. Ground
5. Wire



Procedure:

Here, LED-RED D1,D2,D3,D4 is connected with Resistances R1,R2,R3 and R4 with 220 Ohm each. Both R1, R3 and R2, R4 is connected to Arduino UNO on pin 13 and 12 respectively. D1, D2,D3 and D4 are connected to ground.
The arduino Code hex file path is put on the program file path. Then we see light blinking hence light is getting turned on and off after every 1 second.

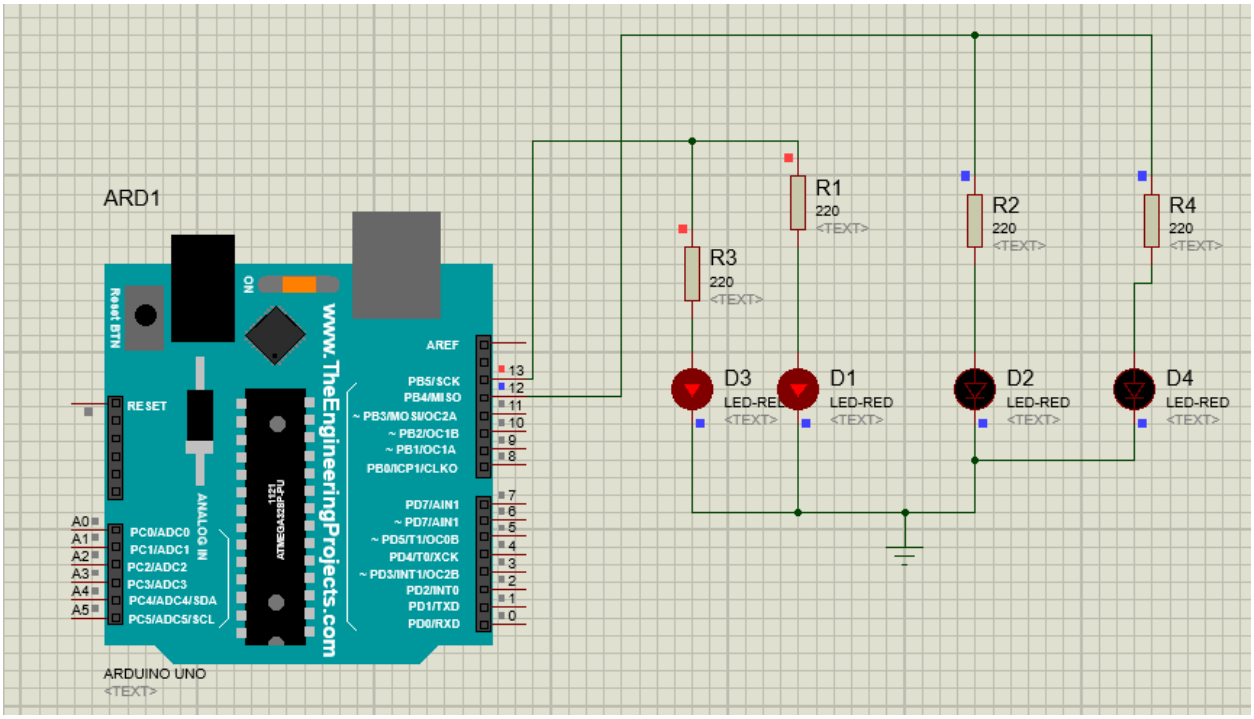


The code for simple LED Blinking after 1sec is:

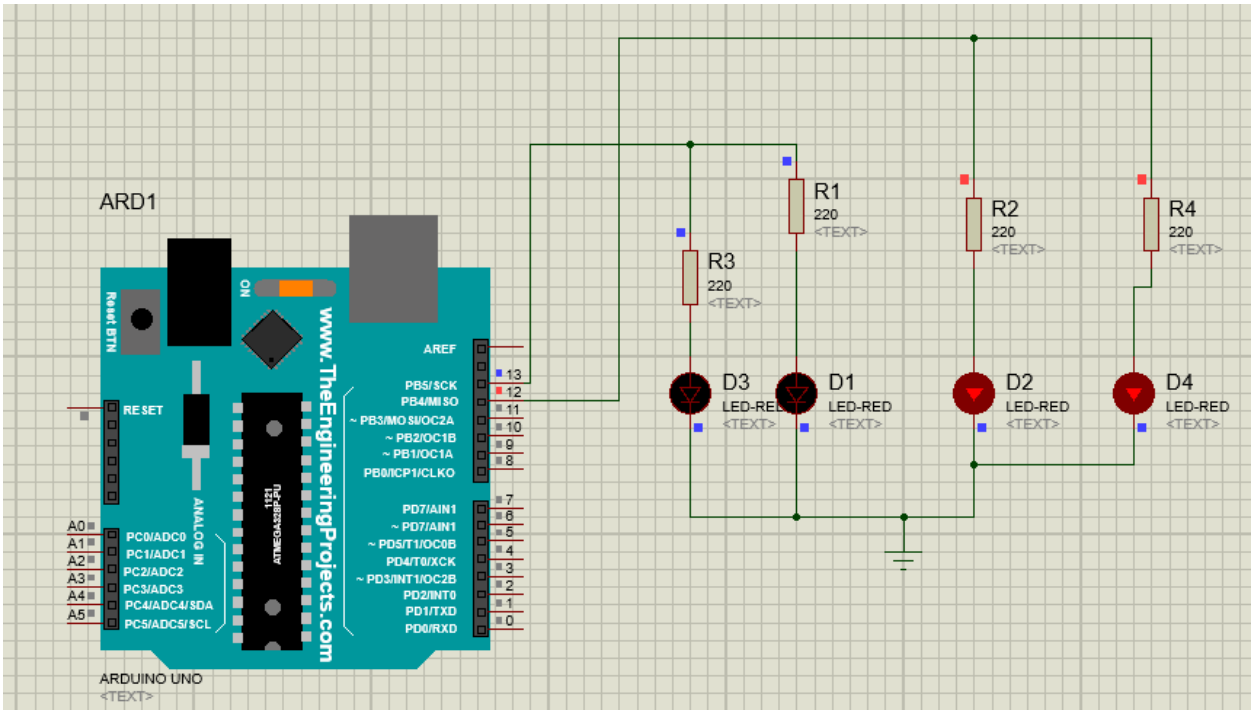
```
int led13=13;
int led24=12;
void setup(){

  pinMode(led13,OUTPUT);
  pinMode(led24,OUTPUT);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
   digitalWrite(led13,HIGH);
   digitalWrite(led24,LOW);
   delay(1000);
   digitalWrite(led13,LOW);
   digitalWrite(led24,HIGH);
   delay(1000);
}
```

Initially after pressing the start button:



After 1 second:



This is how a simple LED-RED light blinking experiment can be done.

Discussion:

1. We can't change the value of any component when we run the simulation we can change it only when we stop the simulation.
2. We have to connect output pin 12 and 13 of Arduino carefully.

Conclusion:

i) We get to know how to use LED-RED light in a software to turn a light ON/OFF.
ii) We have to check the connection properly before circuit simulation to avoid errors.
iii) We get to know how LED-RED light simulate.

Lab Report

Experiment No: 5

Experiment Name: Simulation of Light Dependent Resistor (LDR) using Arduino and Proteus Software.

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

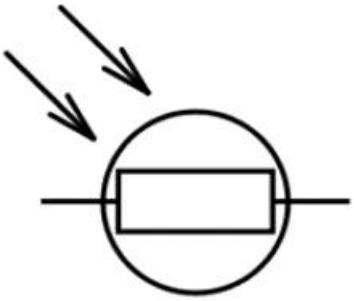
Submitted by:

Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

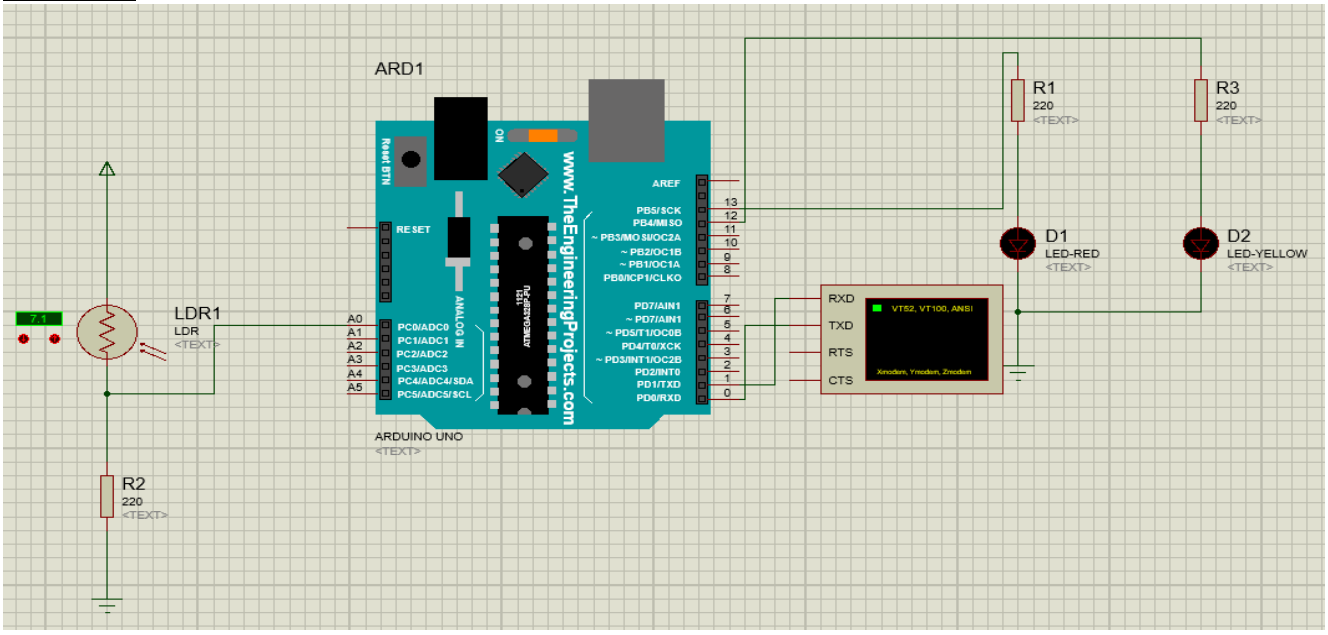
LDR: LDR is a light-controlled variable resistor. The resistance of a photo resistor decreases with increasing incident light intensity; in other words it exhibits photo conductivity. There are many symbol used to indicate LDR, one of the most common used symbol is shown below:



Apparatus:

1. Arduino UNO(ARD1)
2. LDR(LDR1)
3. LED-RED(D1)
4. LED-YELLOW(D2)
5. Resistor(R1,R2,R3 with 220 Ohm each)
6. Ground
7. Wire

Initially :

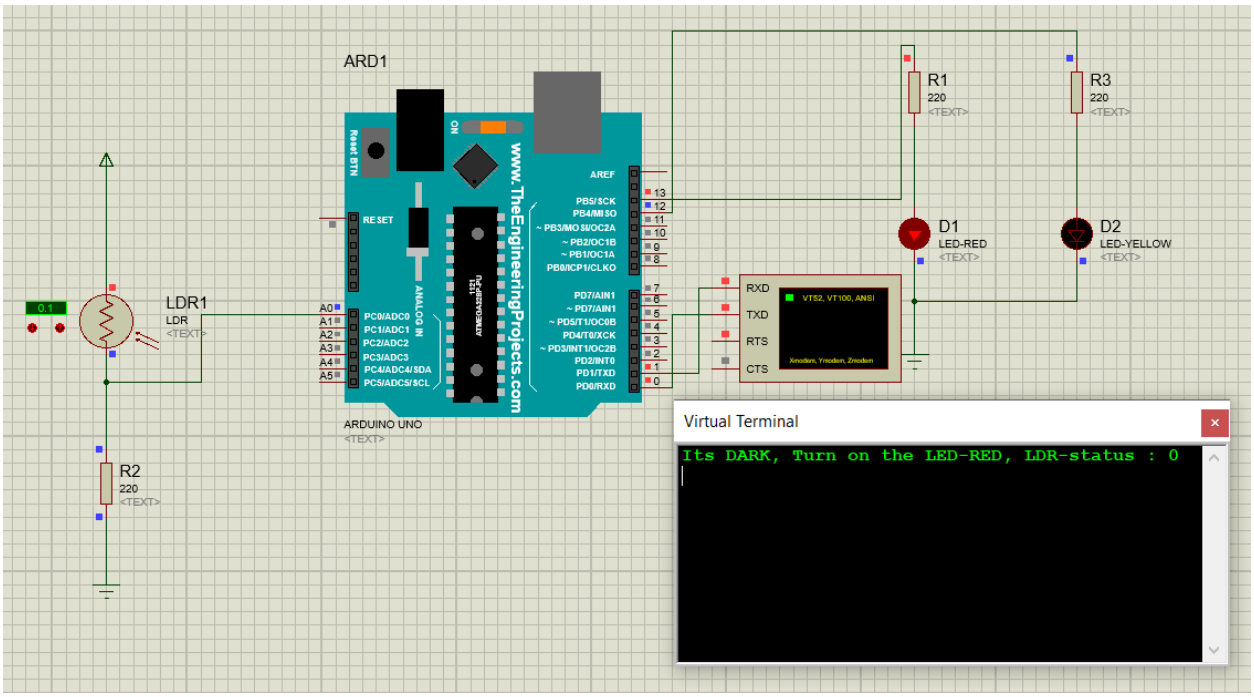


Code:

```
const int ledRed = 13;
const int ledYel = 12;
const int ldrPin = A0;

void setup() {
  Serial.begin(9600);
  pinMode(ledRed, OUTPUT);
  pinMode(ledYel, OUTPUT);
  pinMode(ldrPin, INPUT);
}

void loop() {
  int ldrStatus = analogRead(ldrPin);
  if (ldrStatus <= 1) {
    digitalWrite(ledRed, HIGH);
    digitalWrite(ledYel, LOW);
    Serial.print("Its DARK, Turn on the LED-RED : ");
    Serial.println(ldrStatus);
    delay(3000);
  }
  else if (ldrStatus > 1 and ldrStatus < 6)
  { digitalWrite(ledRed, LOW);
    digitalWrite(ledYel, HIGH);
       Serial.print("Not too Dark, Turn on the LED-YELLOW : ");
    Serial.println(ldrStatus);
    delay(3000);
  }
  else {
    digitalWrite(ledRed, LOW);
    digitalWrite(ledYel, LOW);
    Serial.print("Its BRIGHT, Turn off the LED : ");
    Serial.println(ldrStatus);
    delay(3000);
  }
}
```

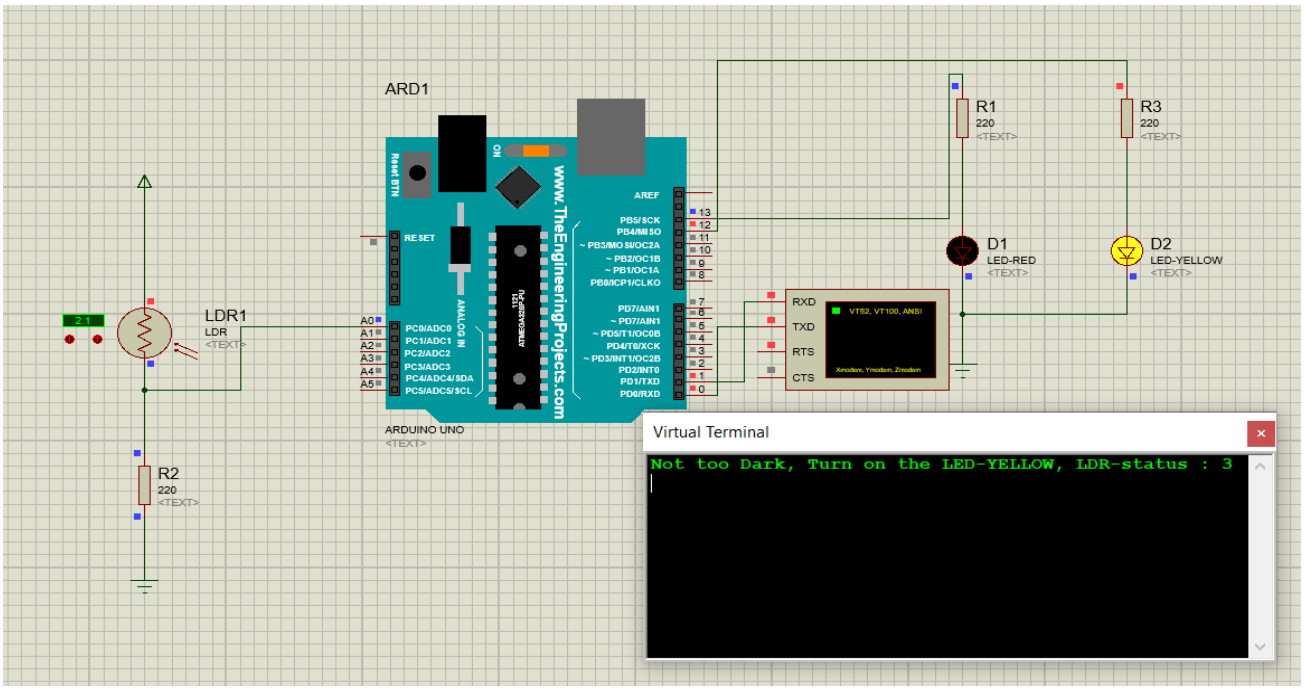
Procedure: Here, R1, R3 and R2 is connected to LED-RED(D1), LED-YELLOW(D2) and LDR(LDR1) respectively. D1,D2 and R2 is connected to ground, LDR(LDR1) is connected to A0 pin of Arduino Uno, A0 pin will get the input from LDR. LED-RED(D1) is connected to output pin 13 of Arduino UNO. If we put the code path on program file path of Arduino simulation and run the simulation then we see when the light intensity is less than 1 the LED-RED(D1) will be turned on and when the light intensity greater than 1 and less 5 LED-YELLOW(D2) will be turned on and when the light intensity is greater than 5 LED will be turned off.

Light intensity less than 1:



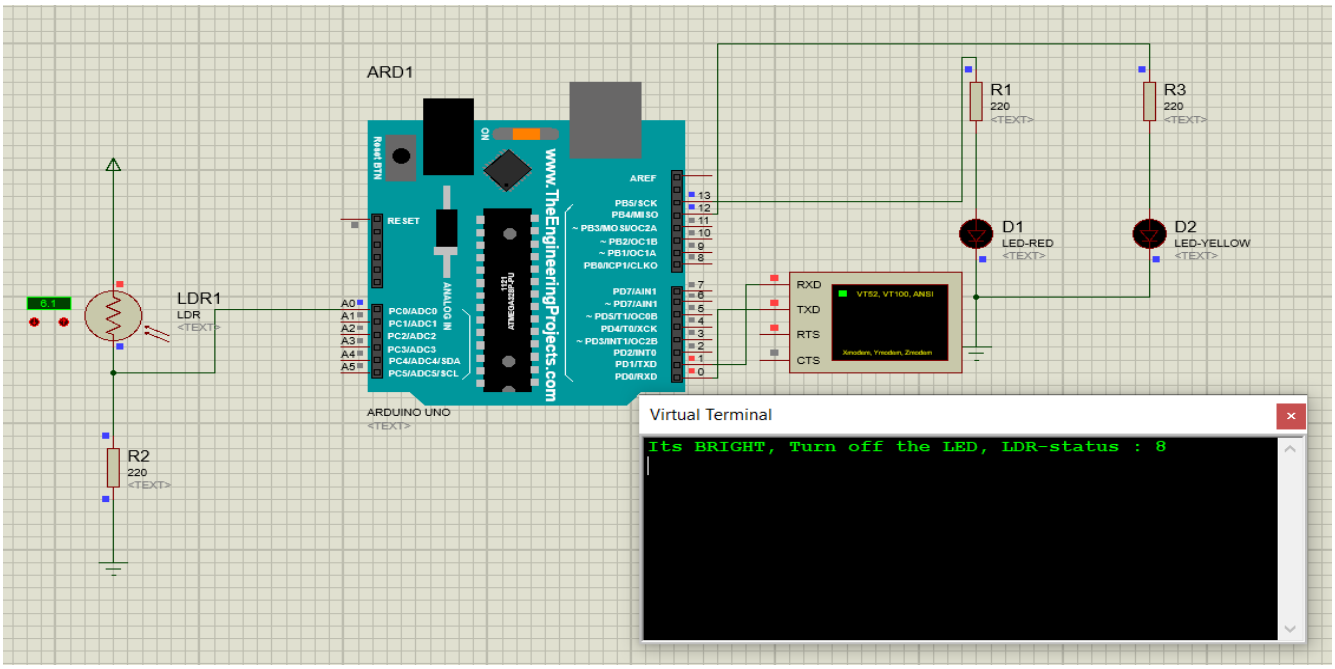
Here, we can see the light intensity was 0 on LDR, hence the LED-RED light got turned on.

Light intensity greater than 1 and less than 5:



Here, we can see that when the light intensity was between 1 to 5 LED-YELLOW light got turned on.

Light intensity greater than 5:



Here LED was turned off because it's too bright & LDR-status was greater than 5.

Discussion:

1. We can't change the value of any component when we run the simulation we can change it only when we stop the simulation.
2. We have to connect input pin A0 and output pin 13 and 12 of Arduino carefully.

Conclusion:

i) We get to know how to use LDR in a software to turn a light ON/OFF with respect to its light intensity.
ii) We have to check the connection properly before circuit simulation to avoid errors.
iii) We get to know how LDR simulate.

Lab Report

Experiment No: 6

Experiment Name: Simulation of Temperature Sensing using LM-35 Temperature Sensor in Proteus Software.

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

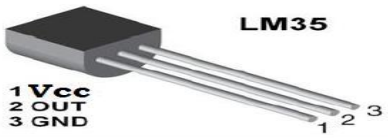
Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

LM-35: It is an integrated-circuit temperature sensor with an output voltage linearly proportional to the Centigrade Temparature. It doesn't require any external calibration to provide typical accuracy over full -55°C to 150°C range.It makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies or with plus and minus supplies. The temperature sensing element is comprised of a delta-V BE architecture.
The temperature sensing element is then buffered by an amplifier and provided to the VOUT pin. The amplifier has a simple class. A output stage with typical 0.5 ohm output impedance as shown in the Functional block diagram:

Pin Description:



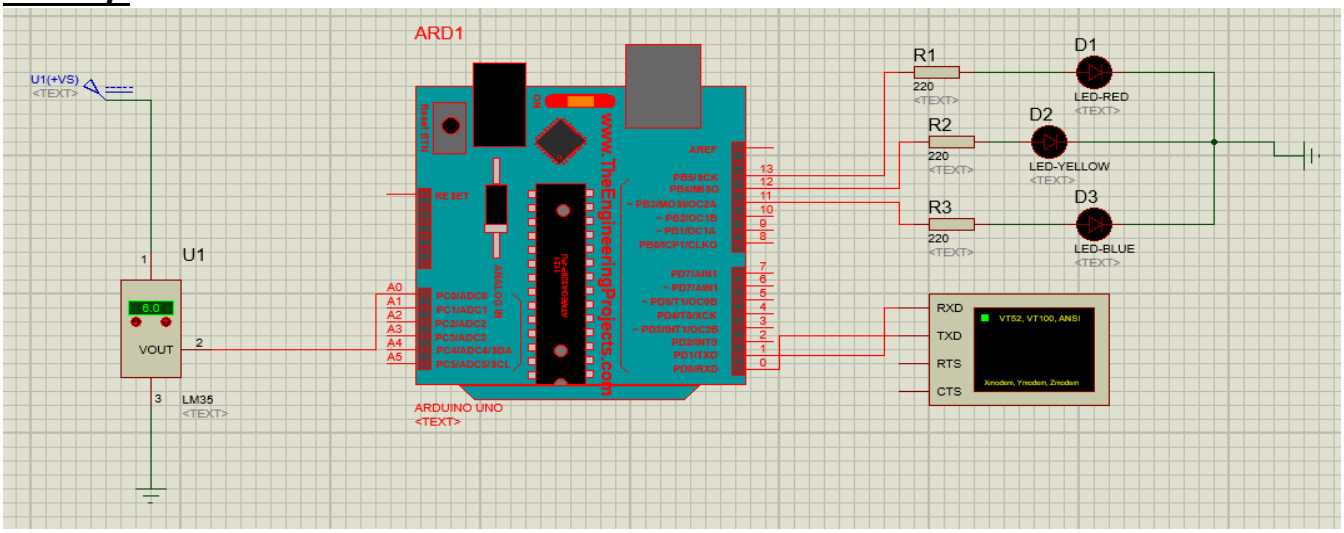
Pin No	Function	Name
1	Supply voltage; 5V (+35V to -2V)	Vcc
2	Output voltage (+6V to -1V)	Output
3	Ground (0V)	Ground

Therefore, the LM-35 can only source current.

Apparatus:

1. Arduino UNO(ARD1)
2. LM-35
3. LED-RED(D1), LED-YELLOW(D2),LED-BLUE(D3)
4. Resistance(R1,R2,R3 each 220 Ohm)
5. Virtual Terminal
6. DC Pulse
7. Ground

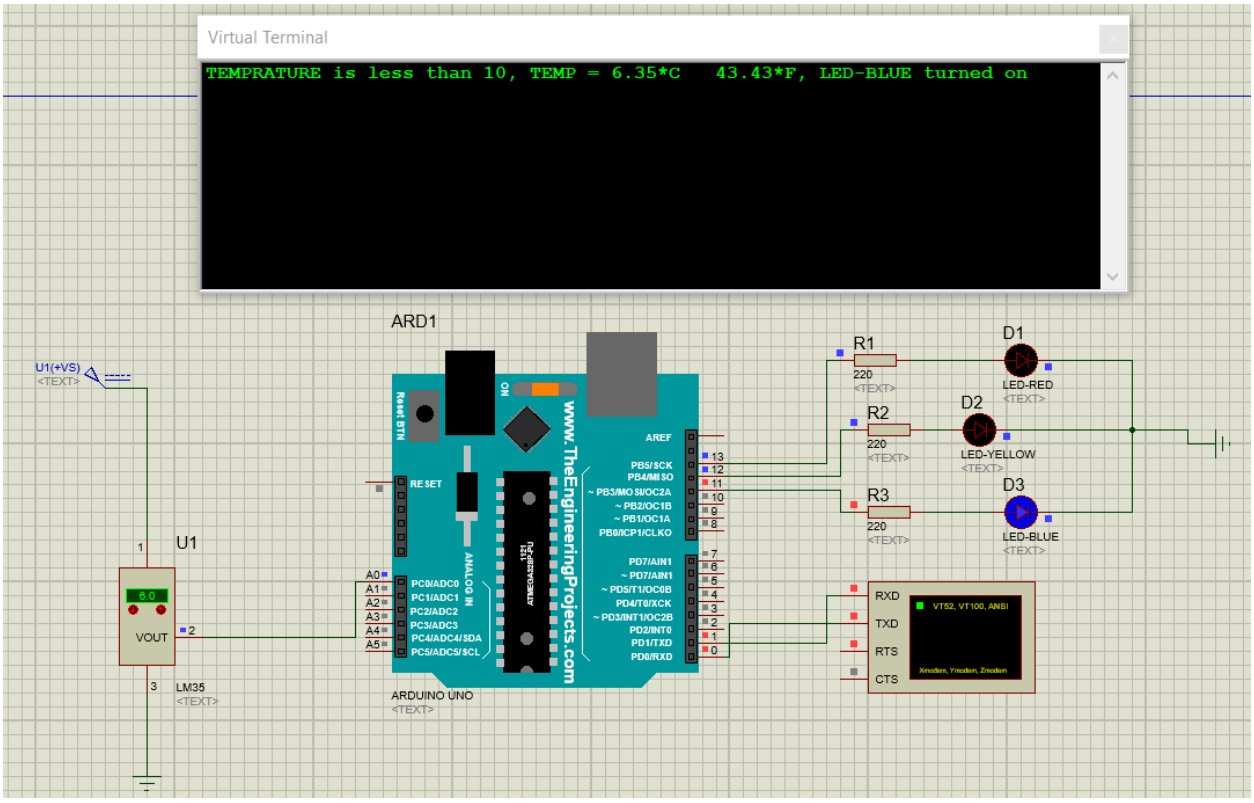
Initially:



Procedure:

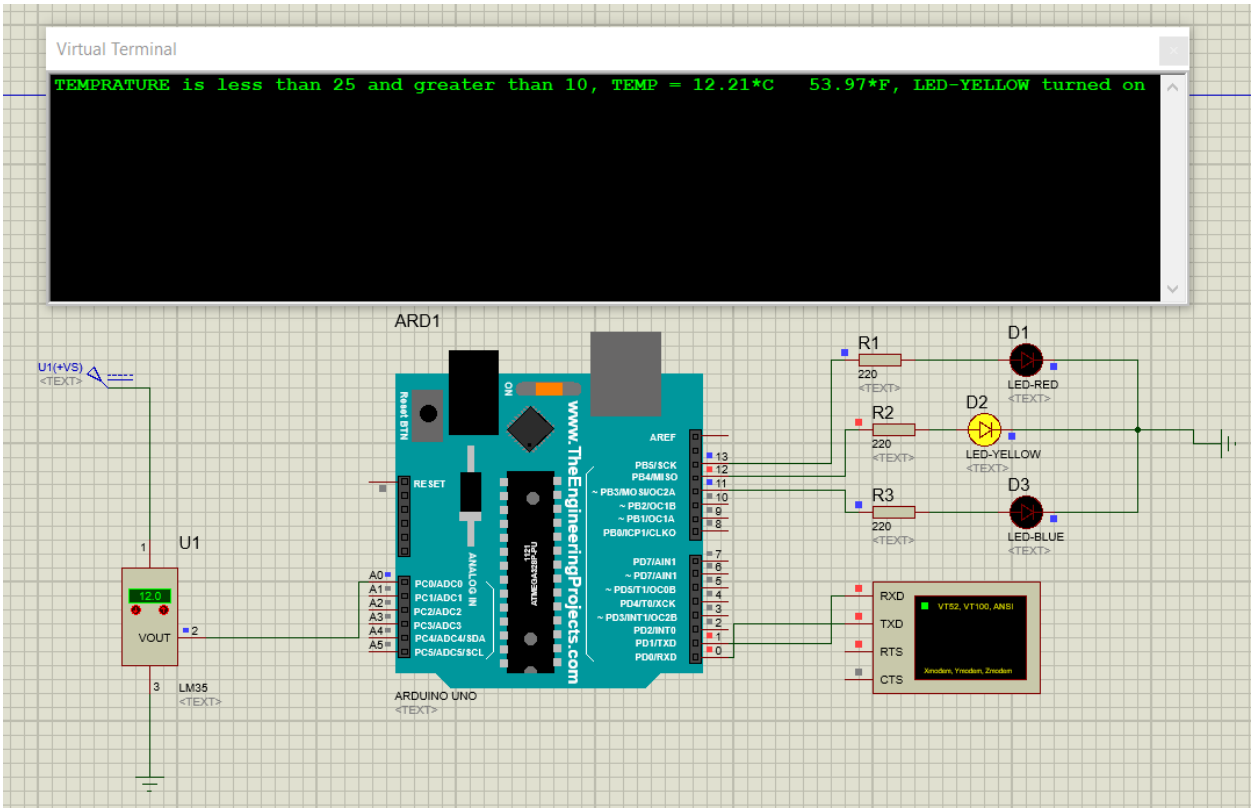
The LM-35 is connected to DC-Pulse, ground and A0 pin of Arduino. Arduino will get input through A0 pin from LM-35 and output will be shown in Virtual Terminal. Virtual Terminal is connected to arduino, the RXD and TXD of virtual terminal is connected to TXD and RXD of arduino respectively. D1,D2 and D3 is connected to ground as well as with resistance R1,R2 and R3 of 220 Ohm each. R1,R2 and R3 Is connected to arduino output pin 13, 12 and 11 respectively. The arduino code path is put on program file path of arduino. If we put the code path on program file path of Arduino simulation and run the simulation then we get to see temperature of an environment and it updates after every 3 seconds, when the temperature is less than 10 the LED-Blue light will be turned on, when the temperature is between 10 to 25 LED-Yellow light will be turned on and finally when the temperature is greater than 25 LED-Red will be turned on.

Temperature less than 10:



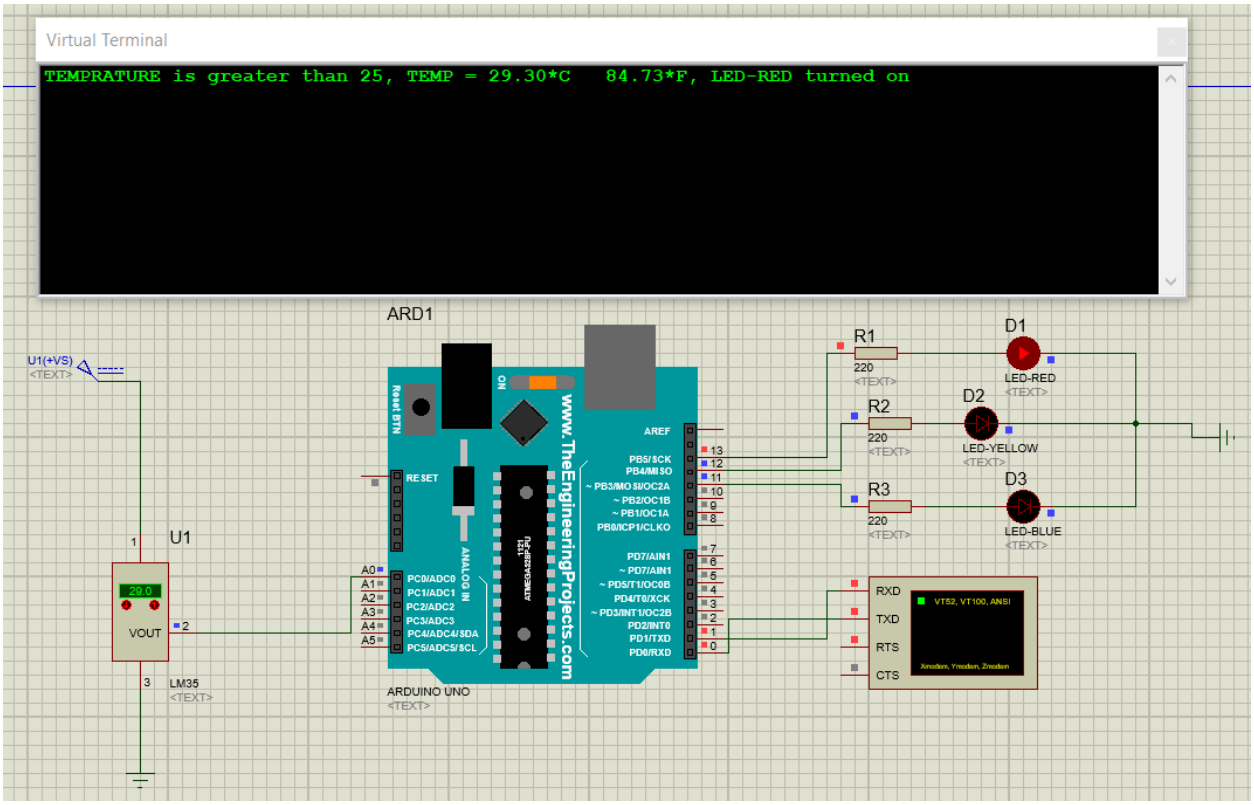
Temperature is less than 10 hence we can see the LED-Blue turned on.

Temperature between 10 to 25:



Temperature between 10 to 25 , hence we can see LED-Yellow turned on.

Temperature greater than 25:



Temperature is greater than 25 hence LED-Red turned on.

Code:

```
int val;
int tempPin = A0;
int ledRed = 13;
int ledYellow = 12;
int ledBlue = 11;


void setup()
{
  Serial.begin(9600);
  pinMode(ledRed, OUTPUT);
```

```
  pinMode(ledYellow, OUTPUT);
  pinMode(ledBlue, OUTPUT);
}
void loop()
{
  val = analogRead(tempPin);
  float mv = ( val / 1024.0) * 5000;
  float cel = mv / 10;
  float farh = (cel * 9) / 5 + 32;
  if (cel <= 10)
  {
    digitalWrite(ledBlue, HIGH);
    digitalWrite(ledRed, LOW);
    digitalWrite(ledYellow, LOW);
    Serial.print("TEMPRATURE is less than 10, TEMP = ");
    Serial.print(cel);
    Serial.print("*C   ");
    Serial.print(farh);
    Serial.print("*F, ");
    Serial.print("LED-BLUE turned on");
    Serial.println();
    delay(3000);
  }
  else if (cel > 10 and cel < 25)
  { digitalWrite(ledBlue, LOW);
    digitalWrite(ledRed, LOW);
    digitalWrite(ledYellow, HIGH);
    Serial.print("TEMPRATURE is less than 25 and greater than 10, TEMP = ");
    Serial.print(cel);
    Serial.print("*C   ");
    Serial.print(farh);
    Serial.print("*F, ");
    Serial.print("LED-YELLOW turned on");
    Serial.println();
    delay(3000);
  }
  else
```

```
 { digitalWrite(ledRed, HIGH);
   digitalWrite(ledBlue, LOW);
   digitalWrite(ledYellow, LOW);
   Serial.print("TEMPRATURE is greater than 25, TEMP = ");
   Serial.print(cel);
   Serial.print("*C   ");
   Serial.print(farh);
   Serial.print("*F, ");
   Serial.print("LED-RED turned on");
   Serial.println();
   delay(3000);
 }

}
```

Discussion:

1. We can't change the value of any component when we run the simulation we can change it only when we stop the simulation.
2. We have to connect the TXD and RXD connection carefully.
3. LEDs should be connected to ground.

Conclusion:

i) We get to know how to use LM-35 in a software to measure the temperature of any environment.
ii) We have to check the connection properly before circuit simulation to avoid errors.
iii) This experiment is very necessary because through this we get to know the accurate result of temperature.
iv) We get to know how a LM-35 works and how to trigger LED light according to temperature.

Lab Report

Experiment No: 7

Experiment Name: Distance Measurement Simulation in Proteus Software using Ultrasonic Sensor.

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

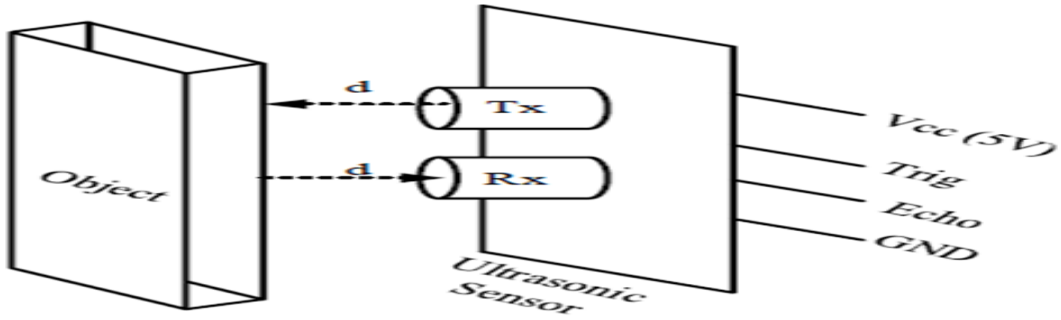
Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

Ultrasonic Sensor:

It is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that is distance= velocity * time. The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below:

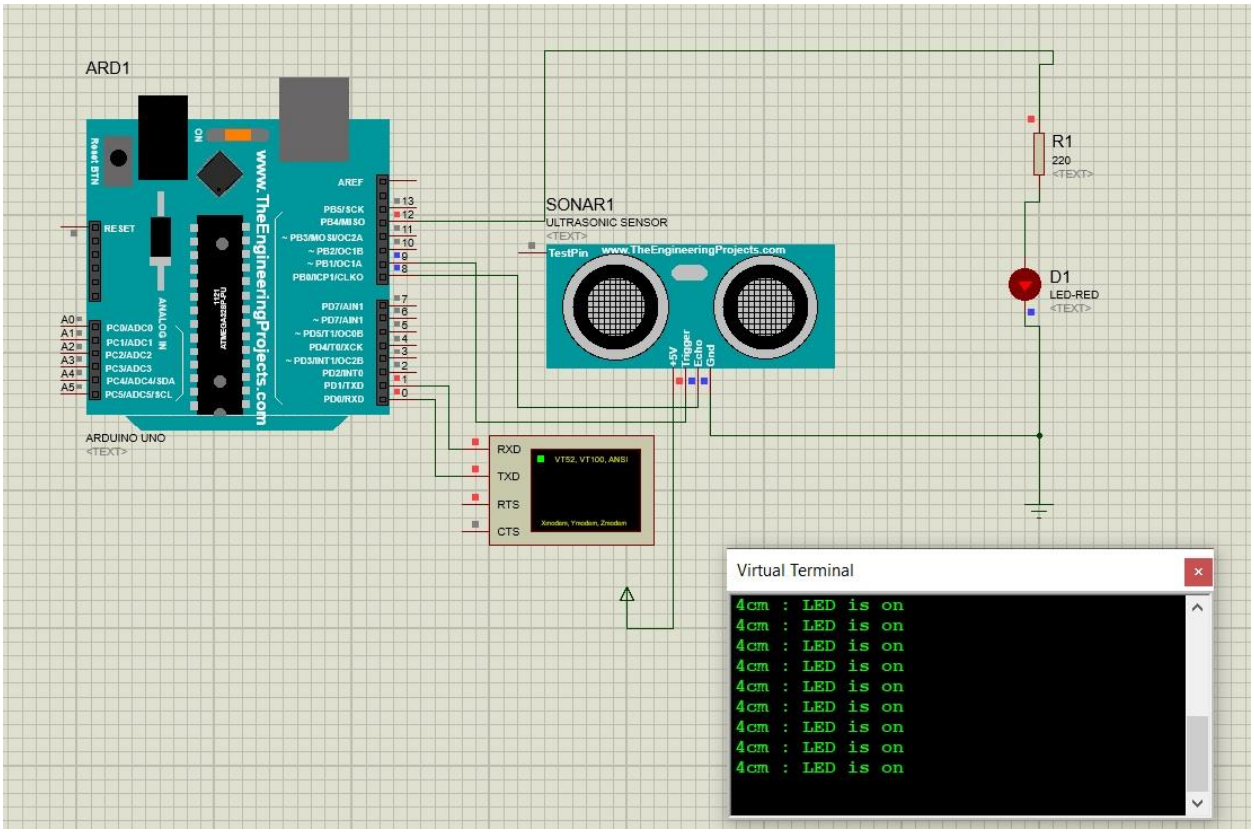


Apparatus:

1. Arduino UNO(ARD1)
2. Ultrasonic Sensor(SONAR1)
3. LED-RED(D1)
4. Virtual Terminal
5. Resistance(R1)
6. Power Supply(+5V)
7. Ground
8. Wire

Procedure:

Here, we connect the RXD and TXD of virtual terminal to pin 0 and 1 respectively of Arduino. Vcc(+5V) of Ultrasonic sensor is connected to a power supply, Trigger and Echo is connected to Arduino UNO pin 9 and 8 respectively. Grounding is done to the Ultrasonic Sensor. LED-RED(D1) is connected to resistor(R1) and to ground. If we put the code path on program file path of Arduino simulation and libaray hex file on ultrasonic sensor path and run the simulation then if any object is placed in front of the sensor, the sensor will measure the distance and will be shown in virtual terminal and if it's within 10cm the LED will blink. Here we see that the virtual terminal shows the object is 4cm ahead hence the LED blinked.



CODE:

```
#define trigPin 9
#define echoPin 8
int led=12;
void setup(){
```

```
Serial.begin (9600);
pinMode(trigPin,OUTPUT);
pinMode(echoPin,INPUT);
pinMode(led,OUTPUT);
}
void loop()
{ long duration, distance;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
if (distance <= 10)
{
Serial.print(distance);
  digitalWrite(led,HIGH);
   Serial.println("cm : LED is on ");
   delay(500);
   }
else { Serial.print(distance);
digitalWrite(led,LOW);
Serial.println(" cm : LED is off ");
delay(500);
    }
}
```

Discussion:

i) We can't change the value of any component when we run the simulation, We can change it only when we stop the simulation.

ii) RXD and TXD of Virtual terminal must be connected properly on Arduino.

iii) The connection of the wires must be done properly in the circuit.

iv)Trigger and Echo pin must be connected to Arduino according to the code.

Conclusion:

i) We get to know how to use Ultrasonic Sensor in a software to measure the distance if any object is ahead.

ii) We have to check the connection properly before circuit simulation to avoid errors.

iii) We get to know how Ultrasonic Sensor simulate.

Lab Report

Experiment No: 8

Experiment Name: Showing Alphabet and Decimal Number by using 7-segment display in Proteus Software.

Course Code : EEE-2422

Course Title : Electrical Drives Sessional

Submitted by:

Name : Iftehaz Newaz
ID : *******
Section : 5AM
Semester : 5th

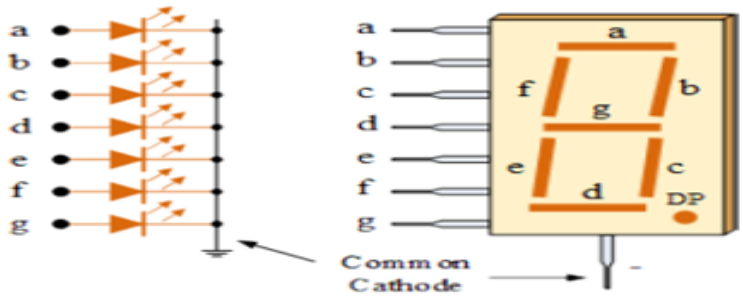
Submitted to :

Jobair Al Rafi
Assistant Lecturer
CSE,IIUC

7-Segment Common Cathode Display:

7-segment display consists of seven LEDs arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit(both decimal and hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point(DP) when two or more 7 segment displays are connected together to display numbers greater than ten.

In the **Common Cathode** display all the cathode connections of the LED segments are joined together to logic '0' or ground. The individual segments are illuminated by application of a 'HIGH' or logic '1' signal via a current limiting resistor to forward bias the individual Anode terminal(a-g).



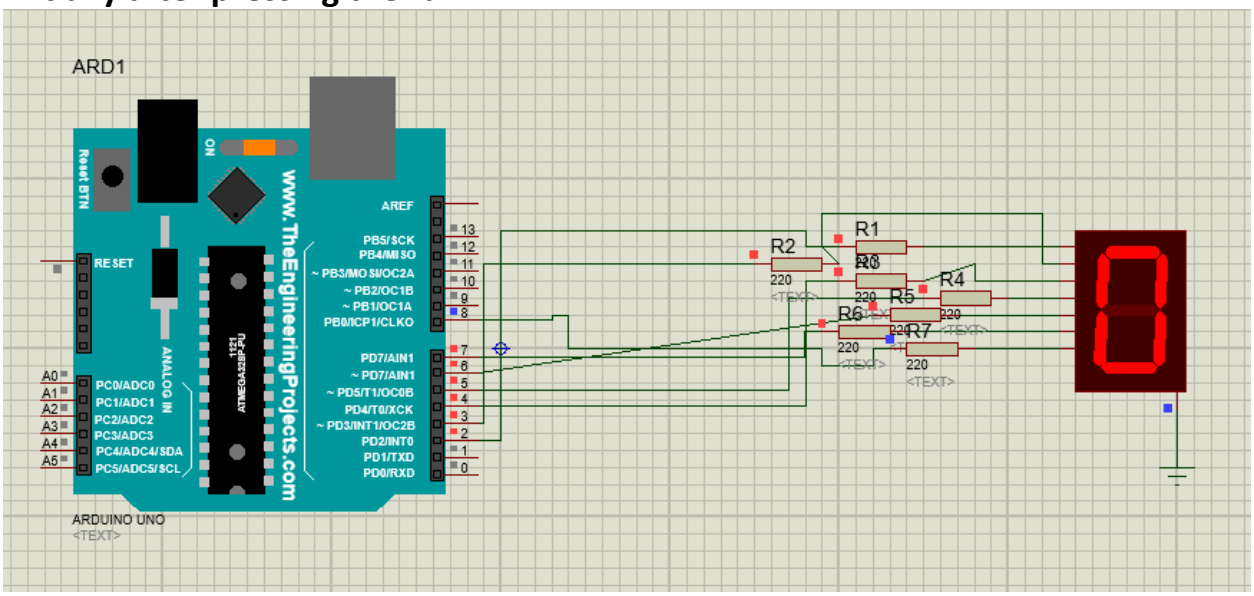
Apparatus:

1. Arduino UNO(ARD1)
2. 7-Segment Common Cathode Display.
3. Resistor(R1,R2,R3,R4,R5,R6,R7)
4. Ground
5. Wire

Procedure:

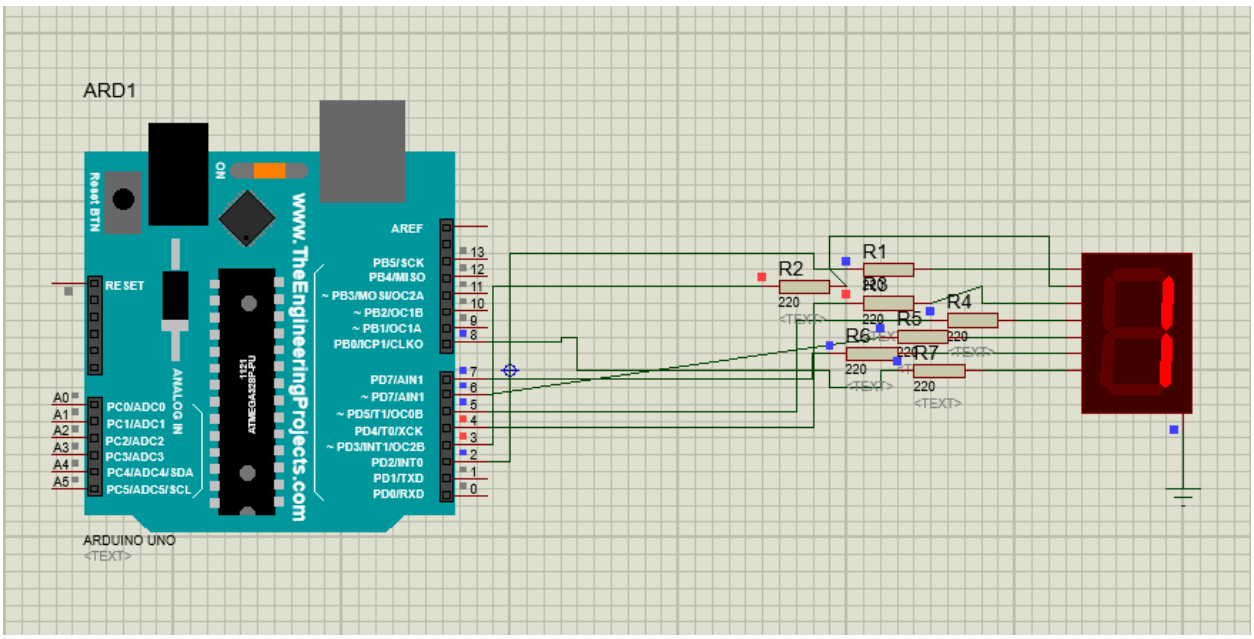
Here, every segment(a-g) of display is connected to resistor(R1-R7) having 220 Ohm each. Arduino Pin from 2 to pin 8 is connected serially to every segment(a-g) resistor(R1-R7) respectively. if we put the code on program file path and execute. Then 0 to 9 and alphabets are shown.

Initially after pressing the run:



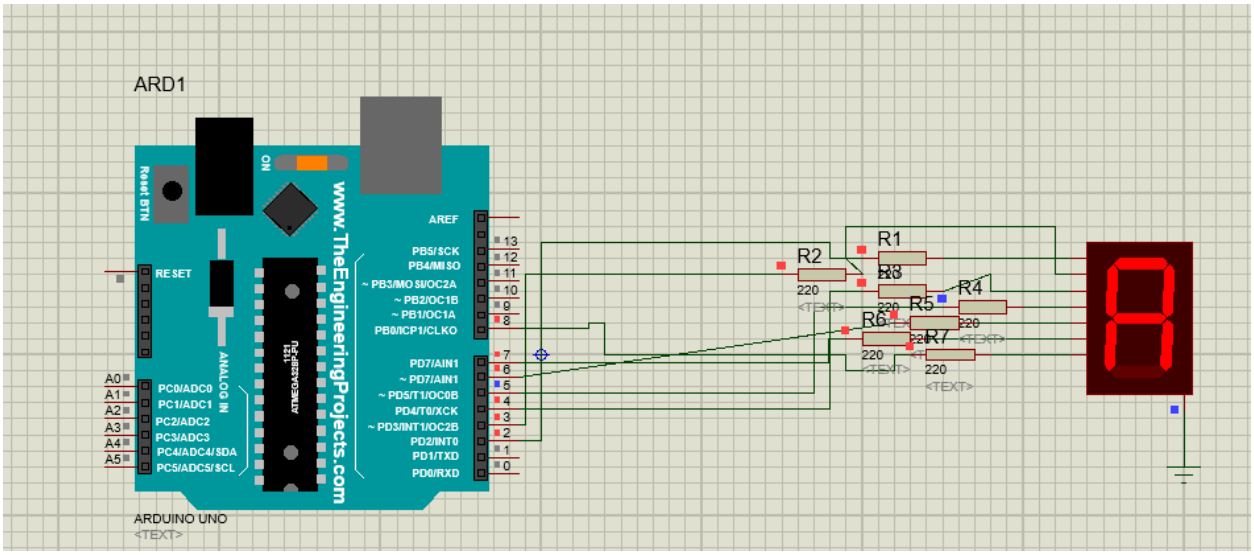
0 is shown.

After a second:



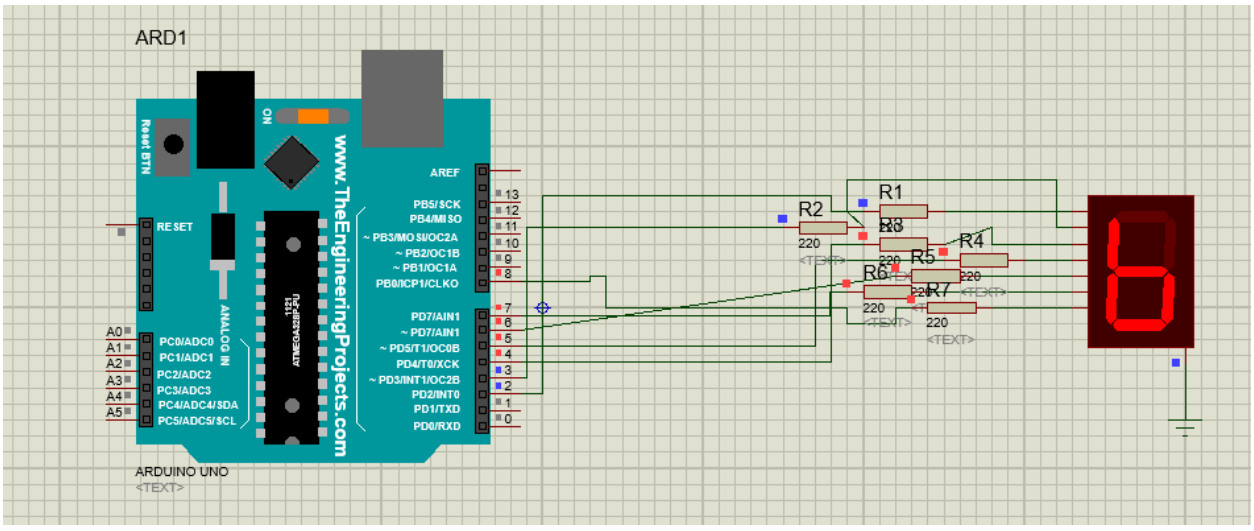
1 is shown.

After 10 seconds alphabets are shown:



A is shown.

Then:



b is shown.

Thus all the alphabets that can be shown in 7-segment display are shown after every 1 second.

Code:

```
void setup() {
  for (int i = 2; i < 9; i++)
  {
    pinMode(i, OUTPUT);
  }

}
void loop() {
  for (int i = 2; i < 9; i++)
  {
    if (i == 8)
    {
      digitalWrite(i, LOW);
    }
    else {
      digitalWrite(i, HIGH);
    }

  }
  delay(1000);
  for (int i = 2; i < 9; i++)
  {
    if (i == 3 || i == 4)
    {
      digitalWrite(i, HIGH);
    }
    else {
      digitalWrite(i, LOW);
    }
  }
  delay(1000);
  for (int i = 2; i < 9; i++)
  {
    if (i == 4 || i == 7)
    {
      digitalWrite(i, LOW);
```

```
  }
  else {
    digitalWrite(i, HIGH);
  }
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 6 || i == 7)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 2 || i == 5 || i == 6)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 3 || i == 6)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }
```

```
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 3)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 4 || i == 3 || i == 2)
  {
    digitalWrite(i, HIGH);
  }
  else {
    digitalWrite(i, LOW);
  }
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  digitalWrite(i, HIGH);
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 6)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
```

```
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 5)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 2 || i == 3)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 8 || i == 3 || i == 4)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }
```

```
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 2 || i == 7)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 4 || i == 3)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 3 || i == 4 || i == 5)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }
```

```
}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 2 || i == 5)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 6 || i == 7)
  {
    digitalWrite(i, HIGH);
  }
  else {
    digitalWrite(i, LOW);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 3 || i == 4 || i == 5)
  {
    digitalWrite(i, HIGH);
  }
  else {
    digitalWrite(i, LOW);
  }

}
```

```
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 5 || i == 6 || i == 7)
  {
    digitalWrite(i, HIGH);
  }
  else {
    digitalWrite(i, LOW);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 6 || i == 4 || i == 8)
  {
    digitalWrite(i, HIGH);
  }
  else {
    digitalWrite(i, LOW);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 3 || i == 2 || i == 7)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
```

```
for (int i = 2; i < 9; i++)
{
  if ( i == 4 || i == 5)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 6 || i == 8)
  {
    digitalWrite(i, HIGH);
  }
  else {
    digitalWrite(i, LOW);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
{
  if (i == 3 || i == 6)
  {
    digitalWrite(i, LOW);
  }
  else {
    digitalWrite(i, HIGH);
  }

}
delay(1000);
for (int i = 2; i < 9; i++)
```

```
  {
    if (i == 2 || i == 8)
    {
      digitalWrite(i, LOW);
    }
    else {
      digitalWrite(i, HIGH);
    }

  }
  delay(1000);
  for (int i = 2; i < 9; i++)
  {
    if (i == 2 || i == 6)
    {
      digitalWrite(i, LOW);
    }
    else {
      digitalWrite(i, HIGH);
    }

  }
  delay(1000);
  for (int i = 2; i < 9; i++)
  {
    if (i == 4 || i == 7)
    {
      digitalWrite(i, LOW);
    }
    else {
      digitalWrite(i, HIGH);
    }

  }
  delay(1000);
}
```

Discussion:

i) We can't change the value of any component when we run the simulation we can change it only when we stop the simulation.
ii) 7-Segment display must be connected to Resistor.
iii) The connection of the wires must be done properly in the circuit.

Conclusion:

i) We get to know how to use 7-segment common cathode display in a software to display any number and alphabets
ii) We have to check the connection properly before circuit simulation to avoid errors.
iii) We get to know how a 7-segment common cathode display simulates.