Savitribai Phule Pune University
सावित्रीबाई फुले पुणे विद्यापीठ

M.Sc.(Computer Science)

CS-604-MJP

Internet of Things
Semester III

(From Academic Year 2024-25)

Name______________________________________Roll No.__________

College_____________________________________Division _________

Academic Year____________________

BOARD OF STUDIES

EDITOR AND PREPARED BY:

Dr. Reena P. Shinde

(Sinhgad College of Science, Pune)

Mrs. Vaishali Salve

(Sinhgad College of Science, Pune)

Mrs. Shital Jadhav

(Sinhgad College of Science, Pune)

Mr. Prashil D. Deshmukh

(Sinhgad College of Science, Pune)

ABOUT THE WORK BOOK

- **OBJECTIVES OF THIS BOOK**

This lab-book is intended to be used by M.Sc.(Computer Science) students for CS- 604-MJP :Lab Course on CS-603-MJ (Internet of Things)., Semester III.

The objectives of this book are

a. Covers the complete scope of the syllabus.
b. Bringing uniformity in the way course is conducted across different colleges.
c. Continuous assessment of the students.
d. Providing ready references for students while working in the practical.
e. The student should have hands on experience by using various sensors like temperature, humidity, smoke, light, etc. and should be able to use control web camera, network and relays connected to the Pi.

- **How to use this book?**

This book is mandatory for the completion of the CS-604-MJP :Lab Course on CS-603-MJ (Internet of Things). It is a measure of the performance of the student for the entire duration of the course.

- **Instructions to the students**

1. Students should carry this book during practical demonstration sessions.
2. Print outs of source code and outputs is optional
3. Student should read the topics mentioned in reading section of this book before completing the practical assignments.
4. Students should solve those exercises which are selected by subject or practical in-charge as a part of journal activity. However, students are free to solve additional exercises for more practice.
5. Each assignment will be assessed on a scale of 0 to 5 as indicated below.

i)	Not done	0
ii)	Incomplete	1
iii)	Late Complete	2
iv)	Needs improvement	3
v)	Complete	4
vi)	Well Done	5

- **Instruction to the Instructors**

1) Make sure that students follow the instruction as given above.
2) Instructors use programs in workbook for giving practical demonstrations along side theory.
3) After a student completes a specific set, the instructor has to verify the programs and sign in the space provided after the activity.
4) Evaluate each assignment on a scale of 5 as specified above by ticking appropriate box.
5) The value should also be entered on assignment completion page of the respective Lab course.
6) Students should be encouraged to **use any simulator** like proteus, CupCarbon, IoTIFY etc. for their assignments.
   ....
7) College has freedom to **choose the any simulator** for practical demonstrations of Arduino and Raspberry Pi programs.

Assignment Completion Sheet

Name:	
Roll No.	Division:

Sr. No.	Assignment Name	Marks	Signature
1	Sense the available networks using Arduino		
2	Measure the distance using ultrasonic sensor using Arduino.		
3	Detects the vibration of an object with sensor		
4	Sense a finger when it is placed on the board		
5	Connect with the available Wi-Fi using Arduino		
6	Gets temperature notification using Arduino.		
7	Use LDR to vary the light intensity of LED using Arduino.		
8	Start Raspberry Pi and try various Linux commands in command terminal window		
9	Run some python programs on Pi		
10	Run some python programs on Pi		
11	Run some python programs on Pi		

Signature of Incharge:
Date:

Internal Examiner **External Examiner**

- ## What is simulation in IoT?

In the Internet of Things (IoT), simulation is the process of building, testing, and deploying customizable applications that use internet access to exchange information within a network. Simulation involves connecting IoT devices, transmitting data, analyzing information, and more.
Simulation can help you evaluate the effect of process changes, new procedures, and capital investment in equipment. For example, engineers can use simulation to assess the performance of an existing system or predict the performance of a planned system.

Getting Started with Proteus Simulation:

Proteus is a comprehensive and highly regarded software platform widely employed in the field of electronics for its capabilities in electronic design and simulation. This versatile tool caters to a broad spectrum of users, including engineers, students, and electronics enthusiasts, offering a rich set of features that facilitate the development and testing of electronic circuits.

At the core of Proteus lies its robust simulation engine, which is instrumental in enabling users to emulate and test electronic designs with precision and efficiency. Proteus simulation provides users with the means to thoroughly assess the functionality and performance of their circuits, without the need for physical prototypes.

Proteus stands out for its capacity to simulate both digital and analog systems with ease. This means users can create intricate digital logic circuits, microcontroller-based projects, and complex analog circuits, all within a unified environment. This adaptability renders Proteus simulation a valuable tool for a broad spectrum of applications, from elementary LED circuits to advanced communication systems.

Steps To Download Proteus

Step 1: First, we need to Download Proteus. Here we are going to install Proteus 8.13. Let's For Downloading Click on Download.
https://drive.google.com/file/d/18dc8n0lpLu9QRzxbgZzciAwhg6NeqnPp/view
Step 2: In the case of google drive it may show the following interface just click on Download anyway.
Step 3: It will start to download the zip file. Based on your internet speed it will take some time. Wait until the download process is completed.
Step 4: When downloading is completed go to the download folder it will show the zip file.
Now on right-click the file and extract this using whatever software you must extract the zip file and select the destination. Here, browses the location where you want to save.
Step 6: Extracted folder will look like this.
Step 7: Now open Proteus 8.13 SP0 Pro Folder and click on the .exe file. It will ask to run. Click on Run.
Step 8: In case if you don't have an installation wizard then first it will ask the Click an

install wizard. Click next and mark check on Agree. Then it will install the wizard, it is an optional process, if you already have this wizard then it will go to the next step directly.
Step 9: Next it will show the following interface, and we need to select the location where we want to install this software. Click on Next.
Step 10: Now it will show the following interface. Simply click on Next.
Step 11: Now it will start to install. It will take some time.
Step 12: After Completion of Installation, it will show the following interface. Simply Click on Finish. Our Proteus 8.13 is installed in our Windows 10.

Step 13: In the desktop icon of this software will show like this. For checking whether it is installed perfectly or not, click on Proteus Professional Icon.
Step 14: It is opening correctly and now we can start to work with this Proteus.

Download Link For Proteus
https://www.geeksforgeeks.org/how-to-download-and-install-proteus-software-on-windows/
Download Libraries For Arduino

First of all, you have got to download the Arduino library for the Proteus software. For this, you will be able to search on Google or download from here.

https://drive.google.com/file/d/0B5vddbSlyKoZeXlJNUV5QmNlSlE/edit?resourcekey=0-8AXs8GQUFYSsTdBAIv70pA

After downloading the file extract the rar file and copy the two file name as **Arduino.idx** and **Arduino.lib**

1. Now paste the copied file in the subsequent path
Drive C >> Program files >> Labcentre Electronics>>Proteus7 Professional>>Data>>Library

- Extract this zip file and open the folder named "Proteus Library Files".

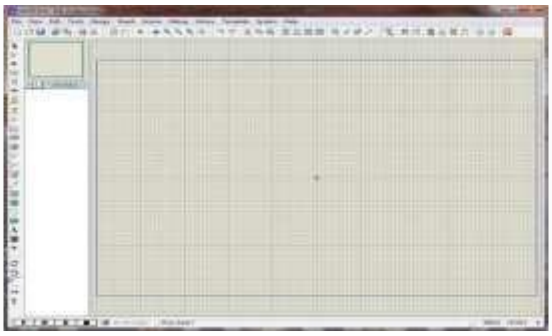
- Inside this folder, you will find these two files:

  - ArduinoUNO2TEP.dll

  - AruinoUNO2TEP.idx

Place these files in the libraries folder of your Proteus software.

- After installing Proteus, run Proteus Professional; the following window interface will appear:

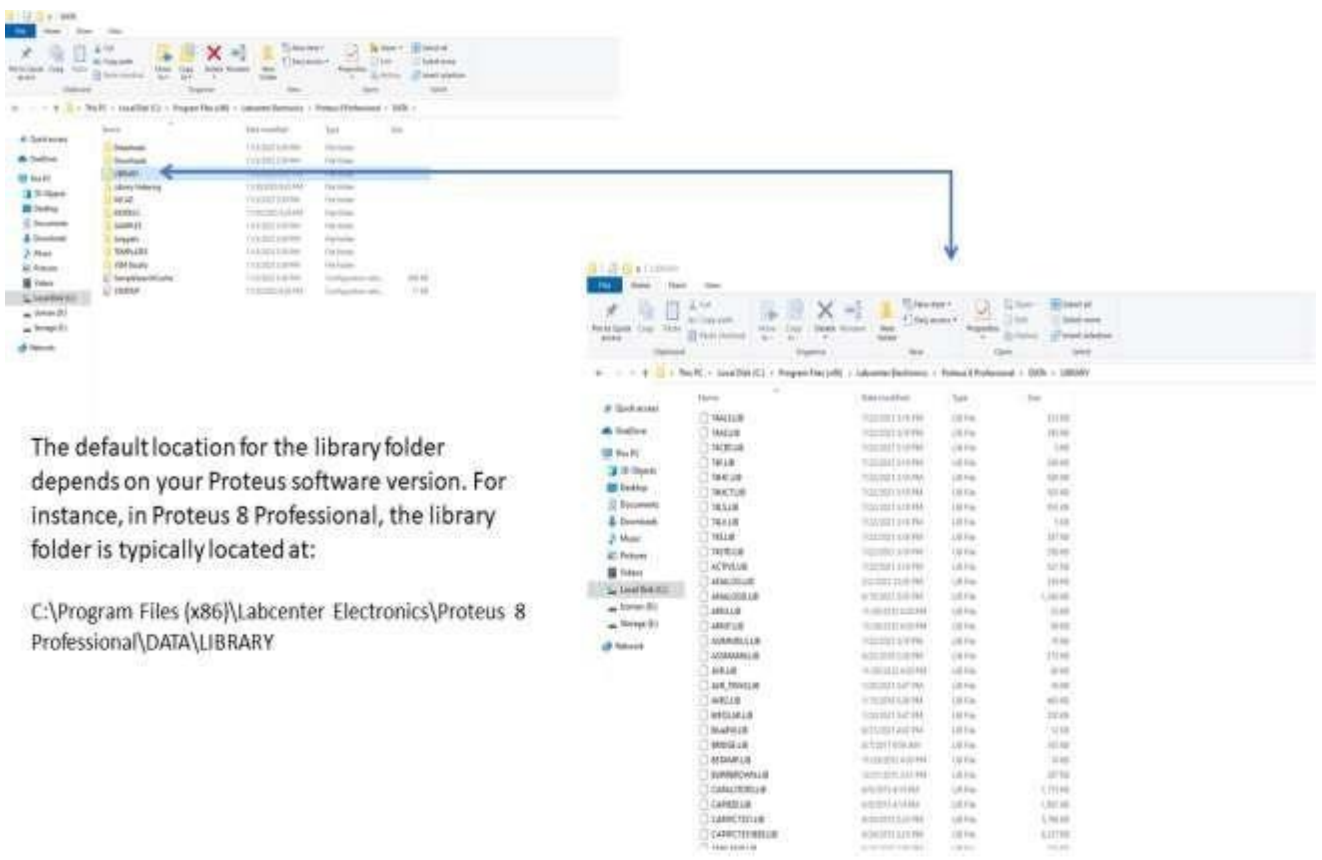


Advantages of Proteus

1. It can be used to simulate and test physical devices.
2. It provides an interface to make circuits in Two -D and how we want to make it also provides this functionality.
3. It takes very less time to test any specific type of circuit compared to physical wire and devices.
4. We can test many tools, and circuits that are very expensive in the physical world.
5. There is no possibility of damaging any physical thing if the circuit is not designed perfectly.

How to Simulate Arduino in Proteus

https://www.instructables.com/How-to-Simulate-Arduino-in-Proteus/
https://maker.pro/arduino/projects/how-to-simulate-arduino-projects-using-proteus
Demo Video For Proteus Project LED Blinking
https://www.google.com/search?sca_esv=7e57735cd28b5eff&sca_upv=1&rlz=1C1CHBD_en-GBIN1069IN1069&q=Led+blinking+using+raspberry+pi+project+in+proteus+8+example&sa=X&ved=2ahUKEwj37bqm6siHAxUB3TgGHY27DTEQ1QJ6BAglEAE#fpstate=ive&vld=cid:2557e6d2,vid:xy4VsU2RGMA,st:0
Extract Library Files:
• Open the .zip file containing the model library.
• Extract the contents, typically finding folders like LIB and MODELS.
Locate Proteus Library Folder:
• Identify your Proteus library folder on your computer. The default location may vary based on your Proteus software version.
• For Proteus 8 Professional, it's often found at:
C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\LIBRARY



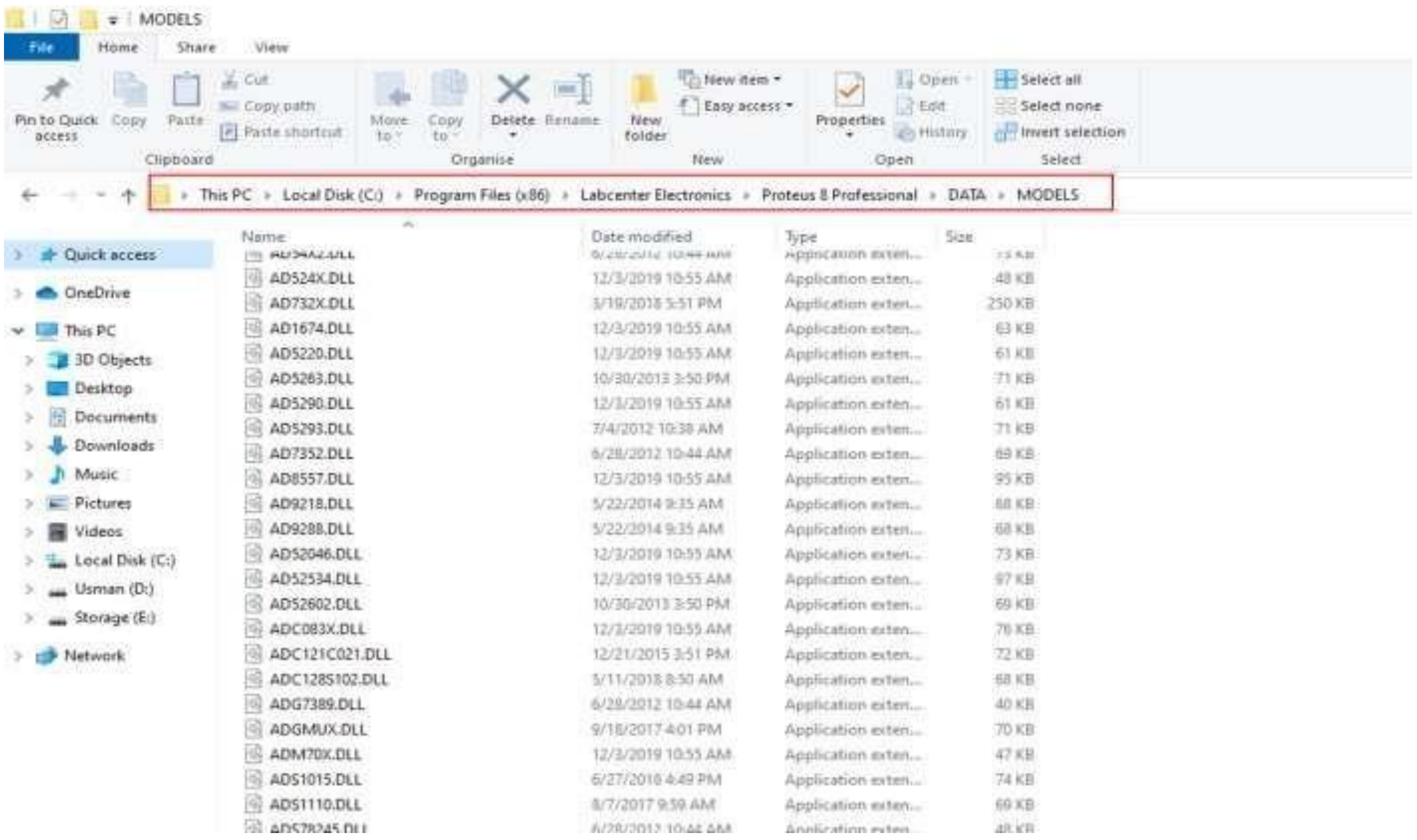
The default location for the library folder depends on your Proteus software version. For instance, in Proteus 8 Professional, the library folder is typically located at:

C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\LIBRARY

Copy LIB Files:

- Open the LIB folder extracted from the model library.
- Copy the files within the LIB folder.
- Paste these files into your Proteus Library
  Folder. Locate Proteus Model Folder:
- Find the Proteus Model folder, usually located alongside the Library Folder.
- For Proteus 8 Professional, the path might be:

C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\MODELS



This PC › Local Disk (C:) › Program Files (x86) › Labcenter Electronics › Proteus 8 Professional › DATA › MODELS

Name	Date modified	Type	Size
[illegible]	[illegible]	Application exten...	[illegible]
AD524X.DLL	12/3/2019 10:55 AM	Application exten...	48 KB
AD732X.DLL	3/19/2018 5:51 PM	Application exten...	250 KB
AD1674.DLL	12/3/2019 10:55 AM	Application exten...	63 KB
AD5220.DLL	12/3/2019 10:55 AM	Application exten...	61 KB
AD5263.DLL	10/30/2013 3:50 PM	Application exten...	71 KB
AD5290.DLL	12/3/2019 10:55 AM	Application exten...	61 KB
AD5293.DLL	7/4/2012 10:38 AM	Application exten...	71 KB
AD7352.DLL	6/28/2012 10:44 AM	Application exten...	69 KB
AD8557.DLL	12/3/2019 10:55 AM	Application exten...	95 KB
AD9218.DLL	5/22/2014 9:35 AM	Application exten...	88 KB
AD9288.DLL	5/22/2014 9:35 AM	Application exten...	68 KB
AD52046.DLL	12/3/2019 10:55 AM	Application exten...	73 KB
AD52534.DLL	12/3/2019 10:55 AM	Application exten...	97 KB
AD52602.DLL	10/30/2013 3:50 PM	Application exten...	69 KB
ADC083X.DLL	12/3/2019 10:55 AM	Application exten...	76 KB
ADC121C021.DLL	12/21/2015 3:51 PM	Application exten...	72 KB
ADC128S102.DLL	5/11/2018 8:50 AM	Application exten...	68 KB
ADG7389.DLL	6/28/2012 10:44 AM	Application exten...	40 KB
ADGMUX.DLL	9/18/2017 4:01 PM	Application exten...	70 KB
ADM70X.DLL	12/3/2019 10:55 AM	Application exten...	47 KB
ADS1015.DLL	6/27/2018 4:49 PM	Application exten...	74 KB
ADS1110.DLL	8/7/2017 9:59 AM	Application exten...	69 KB
ADS78245.DLL	6/28/2012 10:44 AM	Application exten...	48 KB

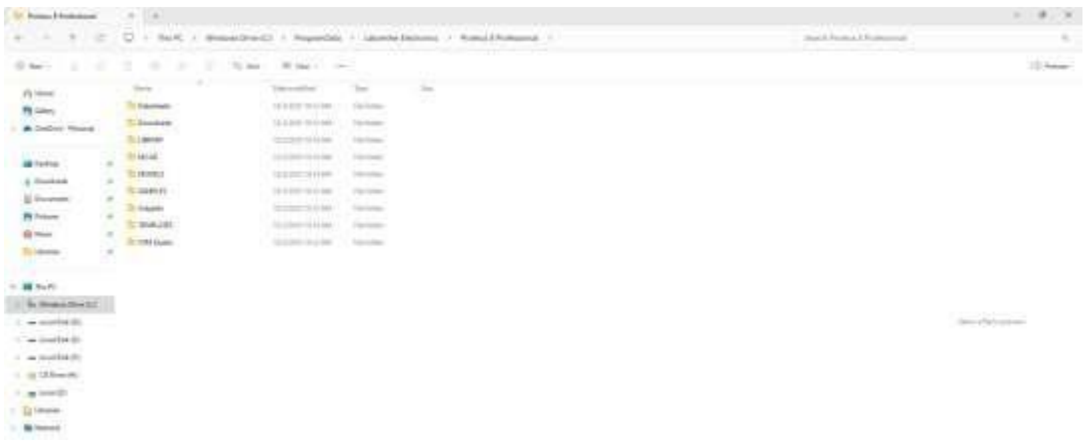
Copy MODELS Files:

- Open the MODELS folder extracted from the model library.
- Copy the files within the MODELS folder.
- Paste these files into your Proteus Model Folder.

Alternative DATA Folder Location:

- For certain Proteus versions, you might locate the DATA folder in a different path, such as:

C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\DATA



Note: The Program Data folder could be hidden, so unhide it if needed.

Restart Proteus:
After copying the library and model files, restart Proteus to apply the changes.

Verify Installation:

- Open Proteus and navigate to the Pick Device Window.
- Confirm that the added model(s) are available for easy selection and integration into your projects.

<u>Note: -</u>

All IoT practical Assignments can be perform on simulator OR using hardware (like Raspberry Pi, Arduino board, sensors and related components) as per convenience.

Internet of Things (IoT)

IOT stands for “Internet of Things”. The IOT is a name for the vast collection of “things” that are being networked together in the home and workplace (up to 20 billion by 2020 according to Gardner, a technology consulting firm).

Characteristics of the IoT

Networking

These IoT devices talk to one another (M2M communication) or to servers located in the local network or on the Internet. Being on the network allows the device the common ability to consume and produce data.

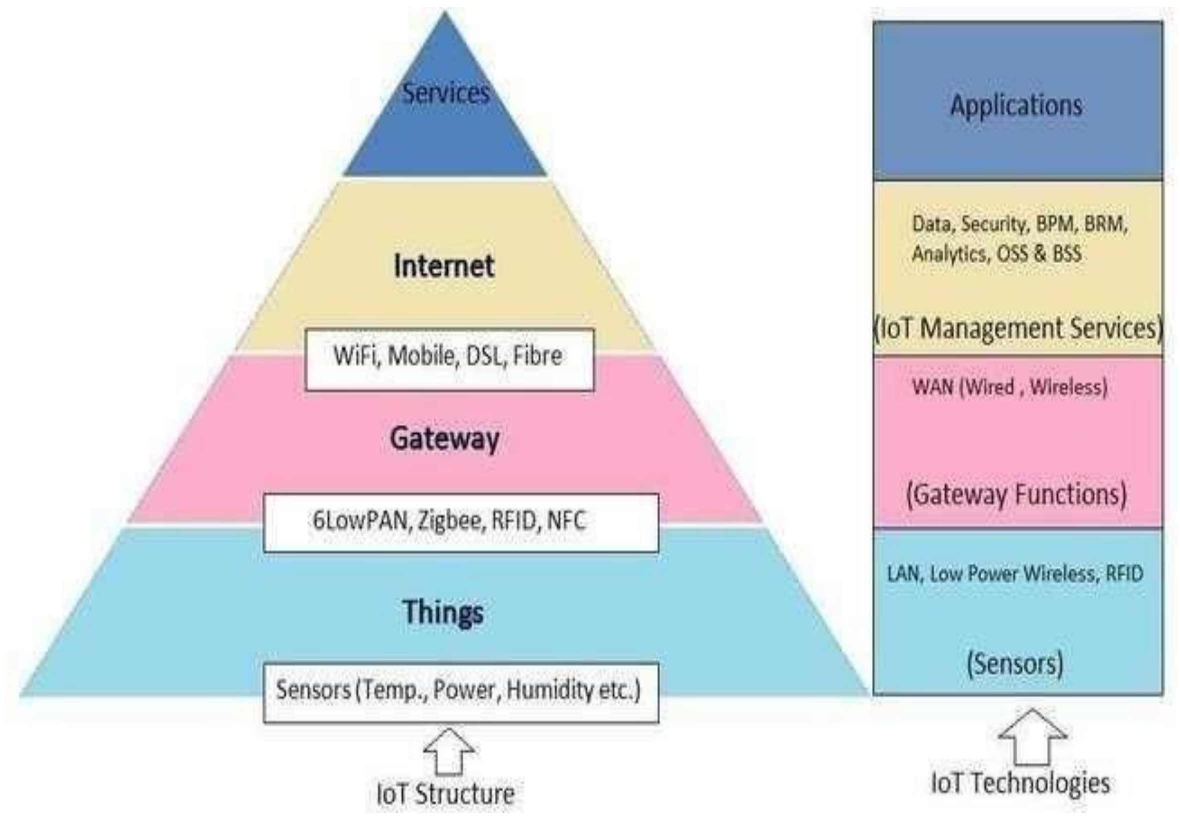
Sensing

IoT devices sense something about their environment.

Actuators

IoT devices that do something. Lockdoors, beep, turn lights on, or turn the TV on

Communications in IoT


Services
Internet
WiFi, Mobile, DSL, Fibre
Gateway
6LowPAN, Zigbee, RFID, NFC
Things
Sensors (Temp., Power, Humidity etc.)
IoT Structure
Applications
Data, Security, BPM, BRM, Analytics, OSS & BSS
(IoT Management Services)
WAN (Wired , Wireless)
(Gateway Functions)
LAN, Low Power Wireless, RFID
(Sensors)
IoT Technologies

Communications are important to IoT projects. In fact, communications are core to the whole genre. There is a trade-off for IoT devices. The more complex the protocols and higher the data rates, them or powerful processor needed and the more electrical power the IoT device will consume.

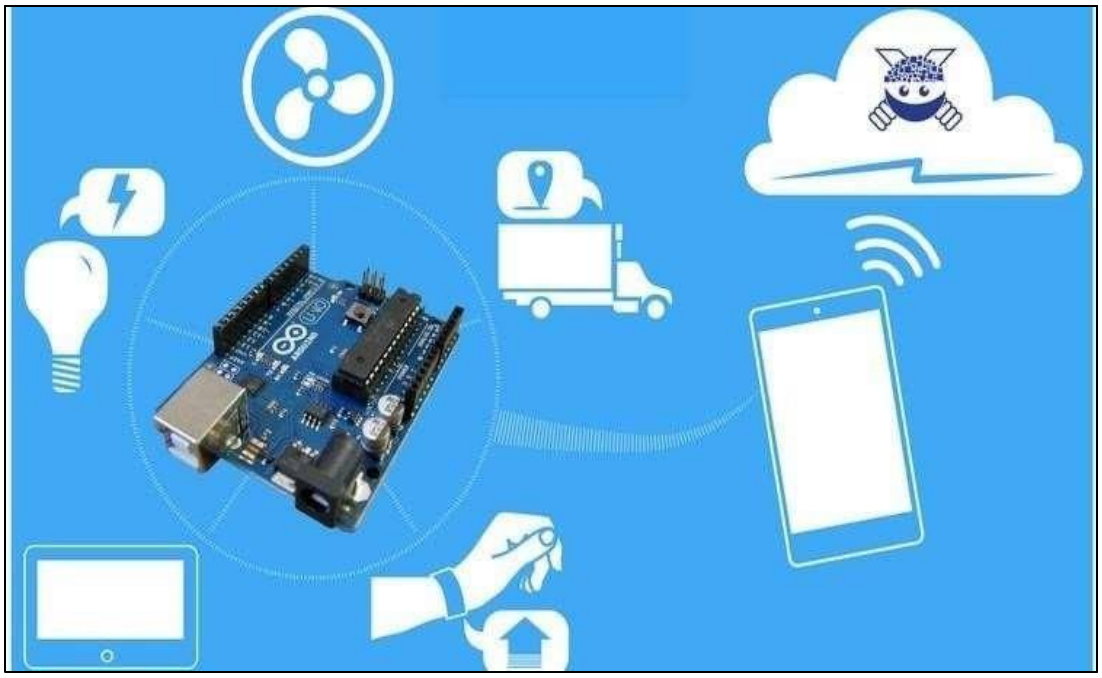
TCP/IP base communications (think web servers; HTTP-based commutation (like REST servers); streams of data; UDP) provide the most flexibility and functionality at a cost of processor and electrical power.

Low-power Bluetooth and Zigbee types of connections allow much lower power for connections with the corresponding decrease in bandwidth and

Functionality. IoT projects can be all over them with requirements for communication flexibility and data bandwidth requirements.

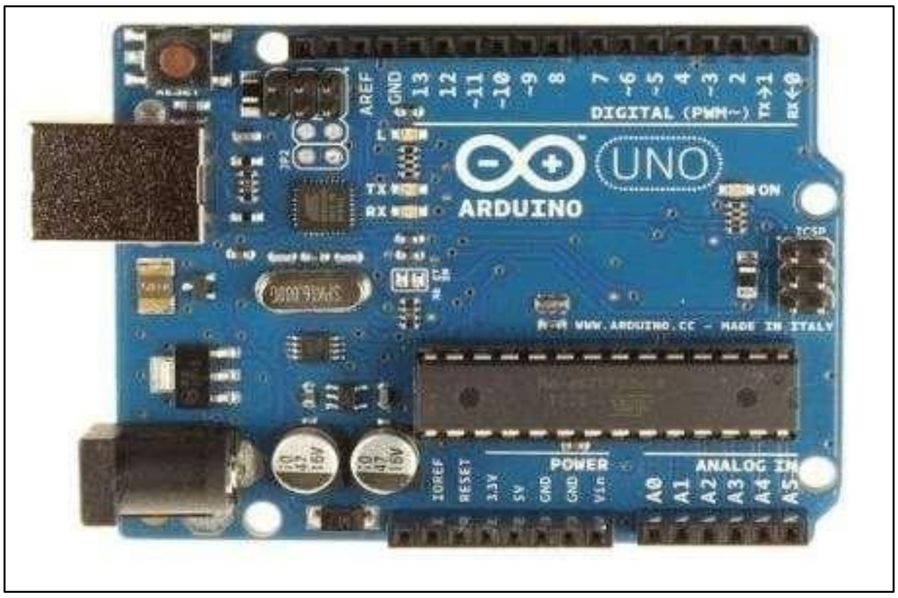
Arduino in IoT

In IoT applications the Arduino is used to collect the data from the sensors/devices to send it to the internet and receives data for purpose of control of actuators.



Arduino Uno

Introduction: The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc.Theboardisequippedwithsetsofdigitalandanaloginput/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type BUSB cable. It can be powered by the USB cable or by an external9-volt battery, though it accepts voltages between 7 and 20volts.The word “uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software.



Features of the Arduino

1. Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

2. The board functions can be controlled by sending a set of instructions to the microcontroller on the board via Arduino IDE.
3. Arduino IDE uses a simplified version of C++, making it easier to learn to program.
4. Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Arduino IDE (Integrated Development Environment)

Introduction: The Arduino Software (IDE) is easy-to-use and is based on the Processing programming environment. The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

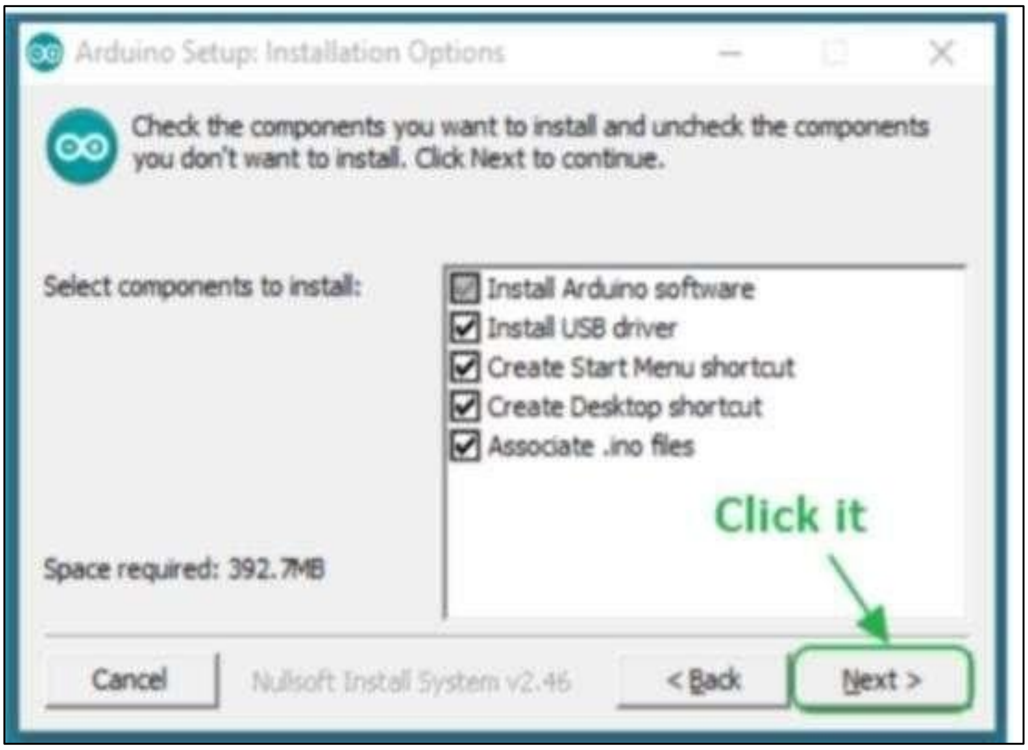
The Arduino Software (IDE)–contains:

- A text editor for writing code
- A message area
- A text consoles
- A toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Installation of Arduino Software (IDE)

Step 1:Downloading

- To install the Arduino software, download this page: Arduino IDE - Download (softonic.com) and proceed with the installation by allowing the driver installation process.


Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

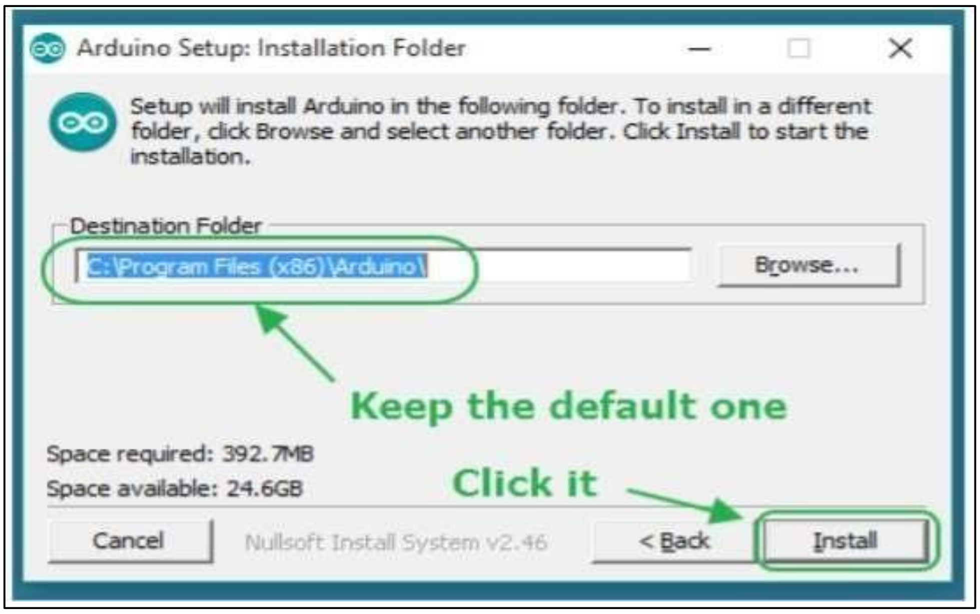
Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 392.7MB

Step 2:Directory Installation

- Choose the installation directory.



Arduino Setup: Installation Folder

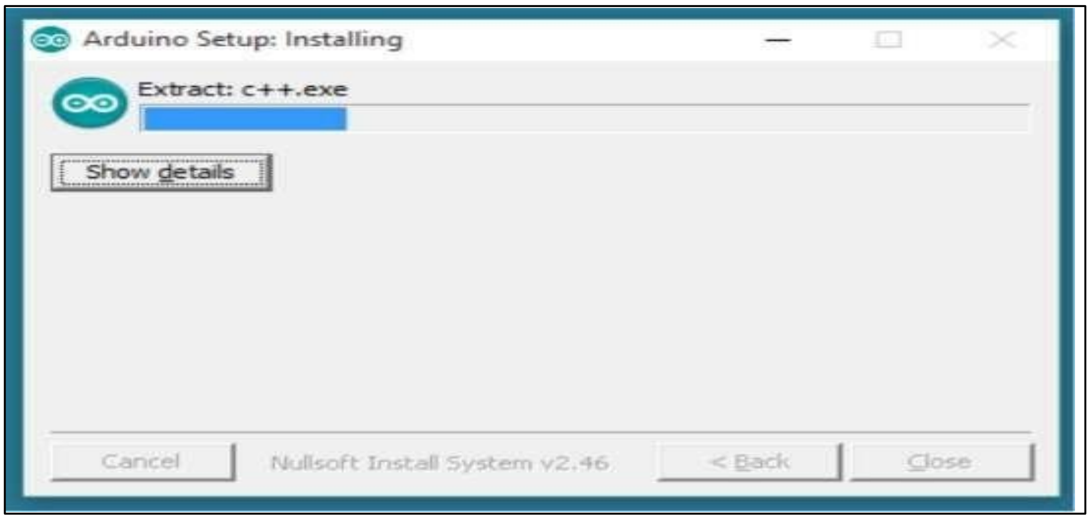
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

Space required: 392.7MB
Space available: 24.6GB

Step 3:Extraction of Files

- The process will extract and install all the required files to execute properly the Arduino Software (IDE)



Arduino Setup: Installing

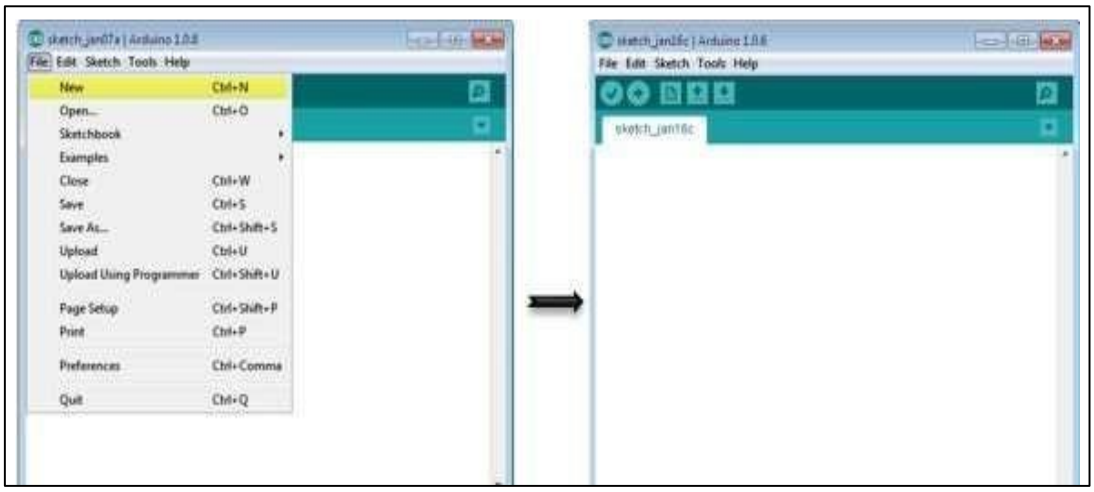
Extract: c++.exe

Step 4: Connecting the board

- The USB connection with the PC is necessary to program the board and not just to power it up. The Uno and Mega automatically draw power from either the USB or an external power supply. Connect the board to the computer using the USB cable. The green power LED (labelled PWR) should go on.

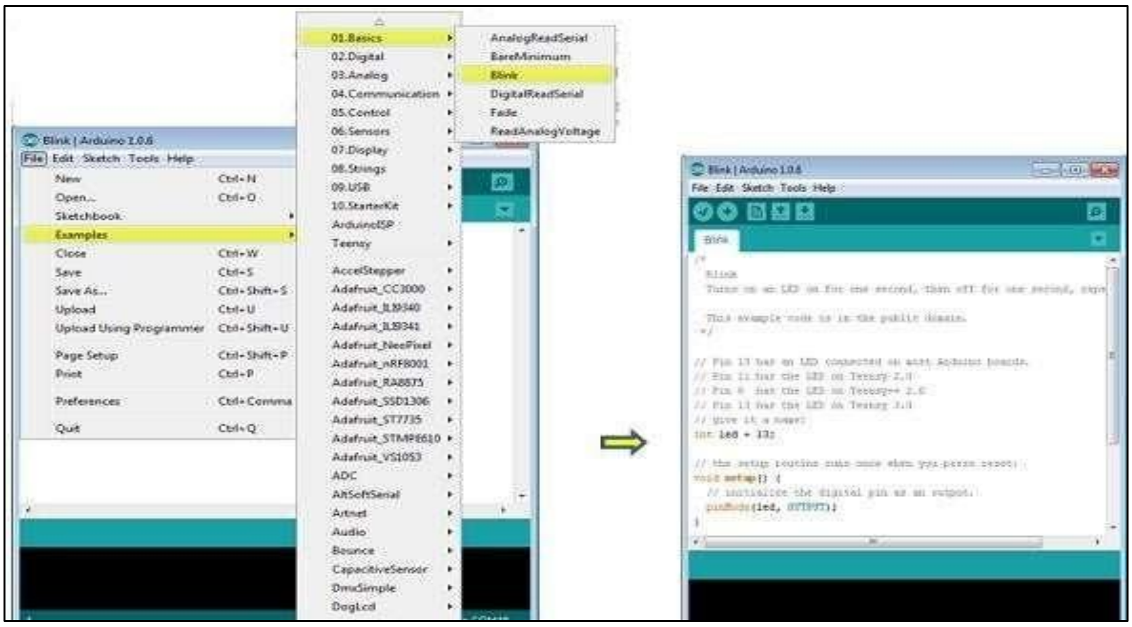
Step 5: Working on the new project

- Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit.
- Open a sketch File by clicking on New.



Step 6: Working on an existing project

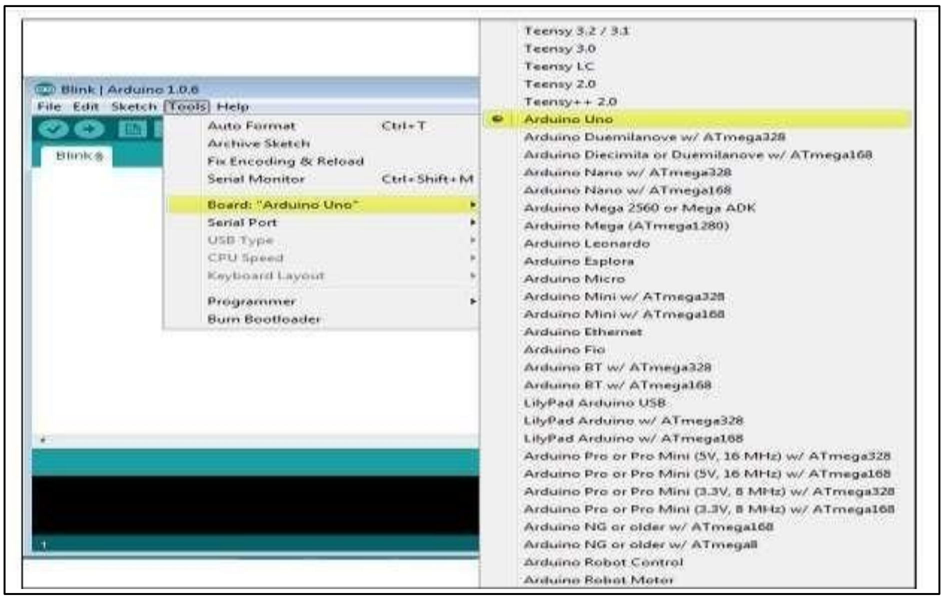
- To open an existing project example, select File →Example →Basics→ Blink.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

Step 7: Select your Arduino board.

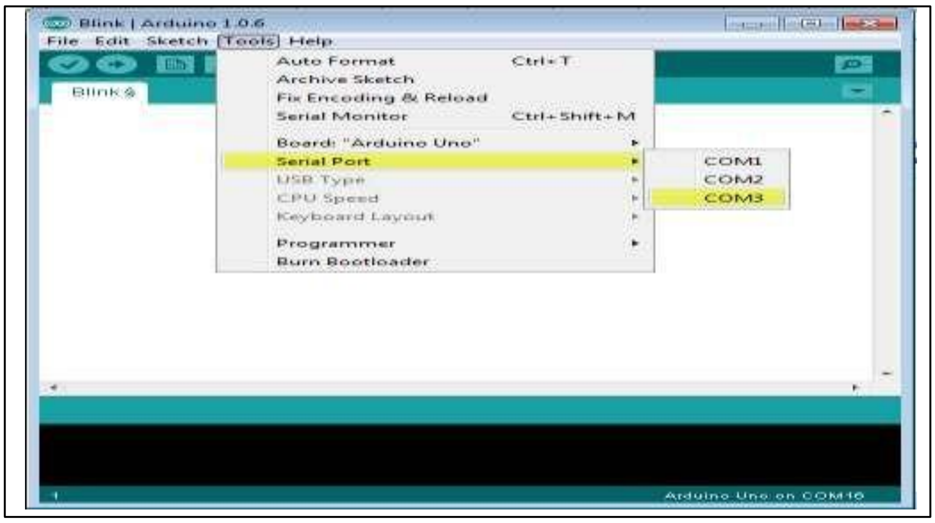
- To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

- Go to Tools →Board and select your board.

Step 8: Select your serial port

- Select the serial device of the Arduino board.
- Go to Tools →Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).
- To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

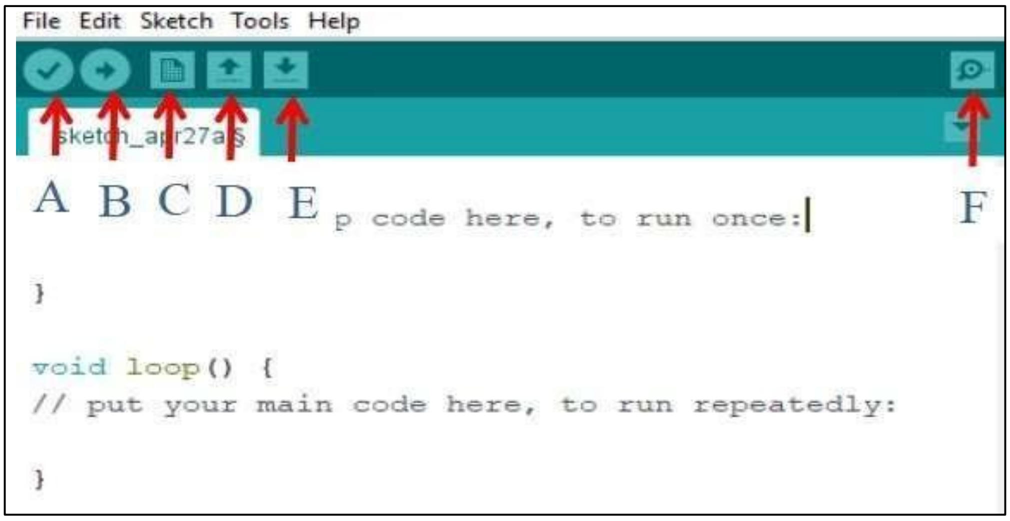


Step 9: Upload the program to your board.

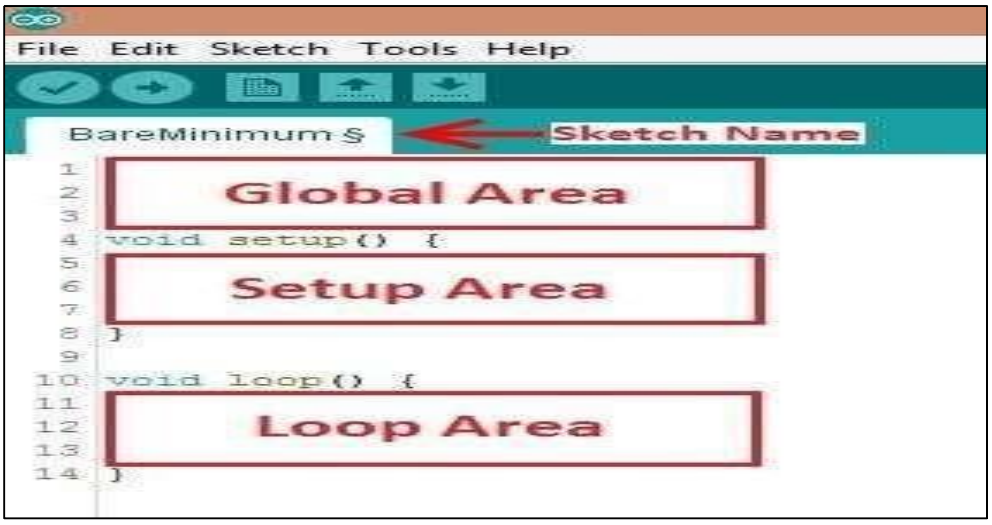
- Click the "Upload" button in the environment.
- Wait a few seconds; you will see the RX and TXLED son the board, flashing.
- If the upload is successful, the message "Done uploading" will appear in the status bar.

A	Verify
B	Upload
C	New
D	Open

E	Save
F	Serial Motor



```
p code here, to run once:

}

void loop() {
// put your main code here, to run repeatedly:

}
```



```
void setup() {

}

void loop() {

}
```

- **AIM:**

  To write a program to sense the available networks using Arduino.
- **Practical Objectives:**

  Sense the available networks using Arduino.
- **Components Required:**

  1. WiFi Module or ESP 8266 Module.
  2. Connecting cable or USB cable.
- **Algorithm:**

  STEP 1: Start the process.

  STEP 2: Start ->Arduino IDE -1.8.8

  STEP3: Then enter the coding in Arduino Software.

  STEP4: Compile the coding in Arduino Software.

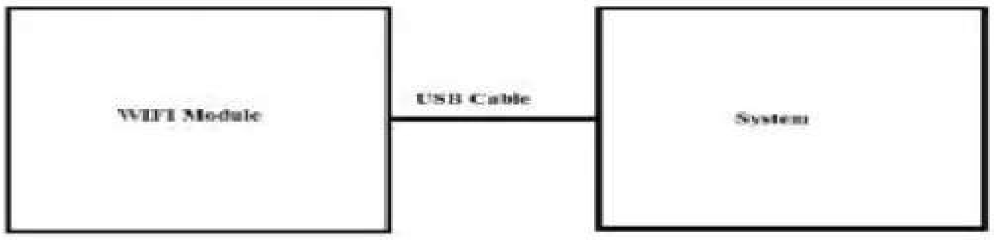
  STEP5: Connect the USB cable to WiFi module.

  STEP 6: Select tools -> select board -> Module node Mch.0.9CE ESP1.2 modules -> select port.

  STEP 7: Upload the coding in ESP Module node Mch.0.9CE and open serial monitor to view the available networks.

  STEP 8: Stop the process.

- **BLOCK MODULE**



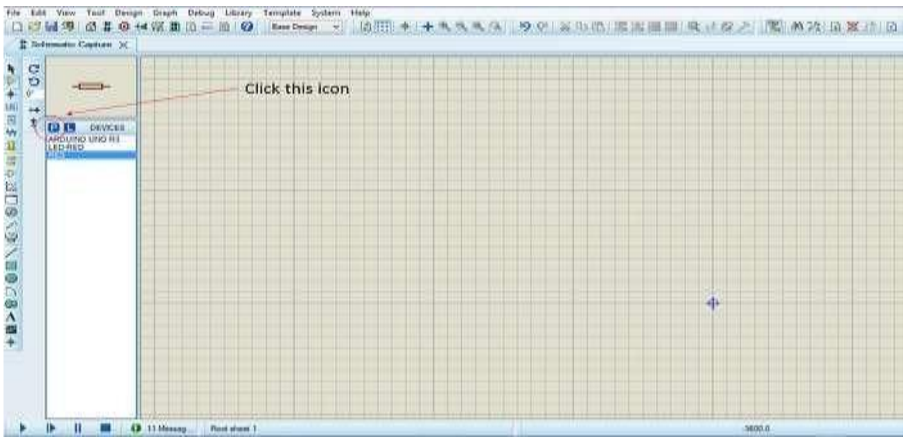
CODING:

```
#include<ESP8266WiFi.h>
Void setup()
 {
Serial.begin(115200);
WiFi.mode(WIFI_STA);
WiFi.disconnect();
delay(100);
Serial.println("Setup done");
}
void loop()
{
 Serial.println("scan start");
intn = WiFi.scanNetworks();
Serial.println( " s c a n  d o n e " )
; if(n  ==  0)
 {
Serial.println("no networks found");
}
 Else
{
Serial.print(n);
Serial.println(" networks found");
for(i=0;i<n;++i)

{
Serial.print(i +1);
Serial.print(": ");
Serial.print(WiFi.SSID(i));
Serial.print(" (");
Serial.print(WiFi.RSSI(i));
Serial.print(")");
Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " :"*");
delay(10);
}
}
Serial.println(""); delay(5000);
}
```

Arduino LED Blink Simulation on Proteus

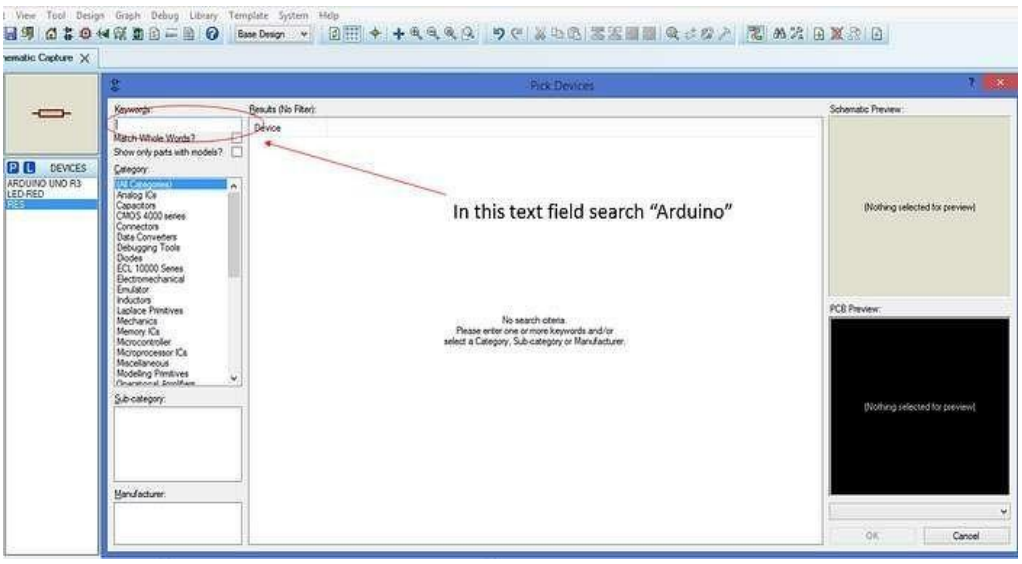
Now open the Proteus software and follow the below steps.

Step-1 Click on New project>>click on No create framework



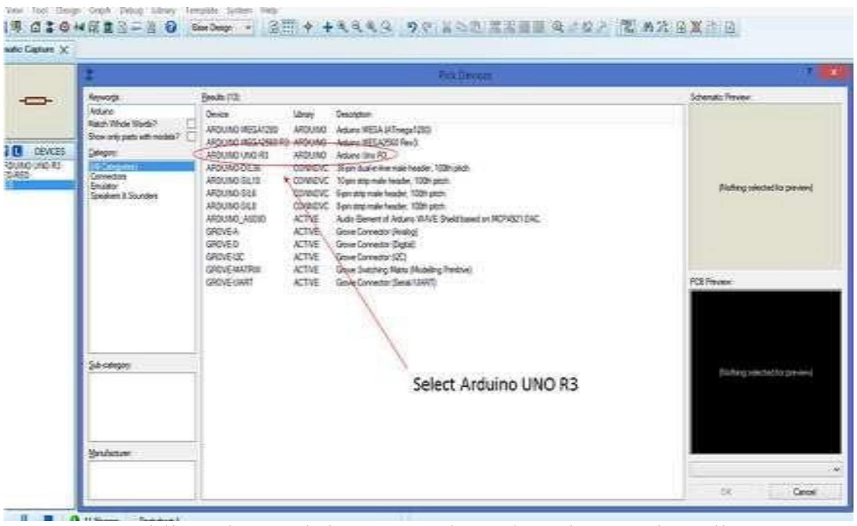
Adding devices figure 1

Step-2 After that, you can see the **Pick Devices** Window. in this window, there is a search bar, in that search bar search the keyword “Arduino”.


Pick Devices
Keywords:
Match Whole Words?
Show only parts with models?
Category:
Results (No Filter):
Device
In this text field search “Arduino”
No search criteria.
Please enter one or more keywords and/or select a Category, Sub-category or Manufacturer.
Schematic Preview:
(Nothing selected for preview)
PCB Preview:
Sub-category:
Manufacturer:
OK
Cancel

Adding devices figure 2

Step-3 Now you can see the **ARDUINO UNO R3** device. add it to the DEVICES list by clicking it and OK.

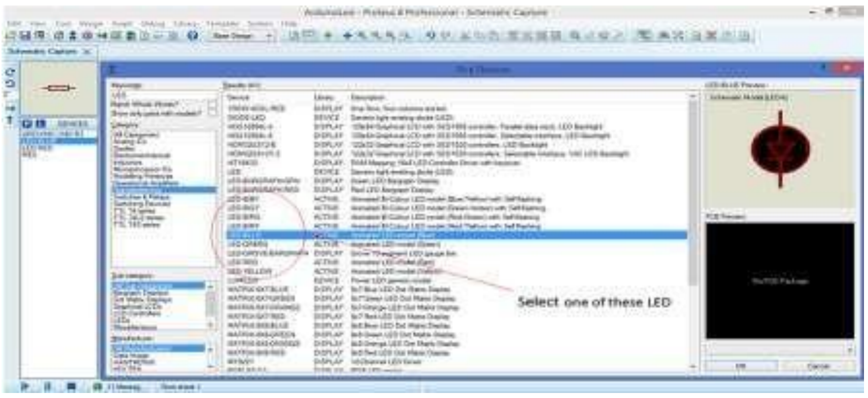

Pick Devices

Results (13):

Device	Library	Description
ARDUINO MEGA1280	ARDUINO	Arduino MEGA (ATmega1280)
ARDUINO MEGA2560 R3	ARDUINO	Arduino MEGA2560 Rev3
ARDUINO UNO R3	ARDUINO	Arduino Uno R3
ARDUINO DIL36	CONNDVC	36pin dual-in-line male header, 100th pitch
ARDUINO SIL10	CONNDVC	10pin strip male header, 100th pitch
ARDUINO SIL6	CONNDVC	6pin strip male header, 100th pitch
ARDUINO SIL8	CONNDVC	8pin strip male header, 100th pitch
ARDUINO_AS030	ACTIVE	Audio Element of Arduino WAVE Shield based on MCP4921 DAC
GROVE-A	ACTIVE	Grove Connector (Analog)
GROVE-D	ACTIVE	Grove Connector (Digital)
GROVE-I2C	ACTIVE	Grove Connector (I2C)
GROVE-MATRIX	ACTIVE	Grove Switching Matrix (Modelling Primitive)
GROVE-UART	ACTIVE	Grove Connector (Serial UART)

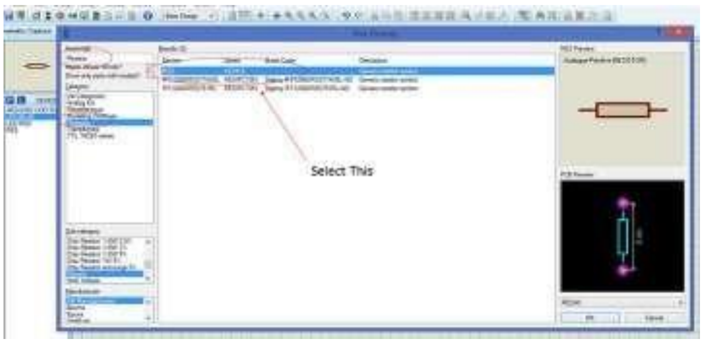
Adding the Arduino UNO board to the Devices list

Like that you can LED bulb and the Resistor.



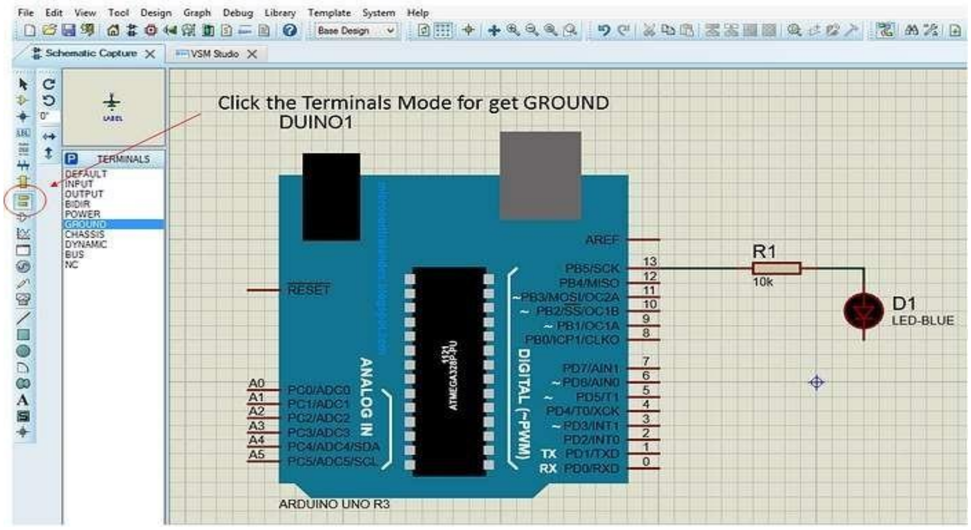
Adding the LED bulb to the Devices list

Adding a Resister.



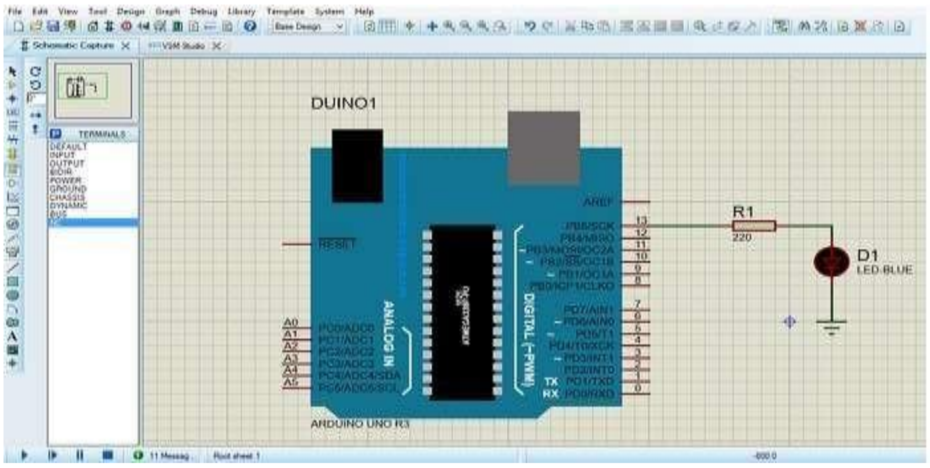
Adding a resistor to the Devices list

Step-4 Now you have to add those things in your Schematic Capture window devices list. Simply select those ones by one by one and drop the Schematic workspace. After adding the components, it will show like this. Note that we are code for **LED_BUILTIN** pin. It's pin **13**.
But where's the GROUND? to get the GROUND to click the Terminals Mode icon.



showing terminals mode

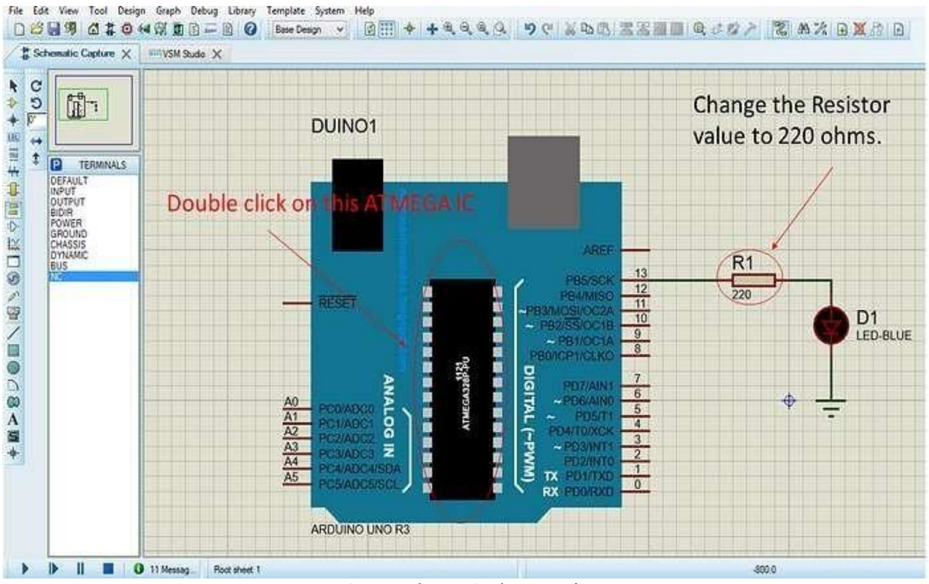
Ok, now you can see the GROUND and add to the Schematic.



After adding the ground

There is a little bit problem, isn't it? how we change resistor value? you can change it by double-clicking on the resistor and change the value to 220 ohms.
Now you need to add the hex file to the ATMEGA 328P IC. Let's do it.

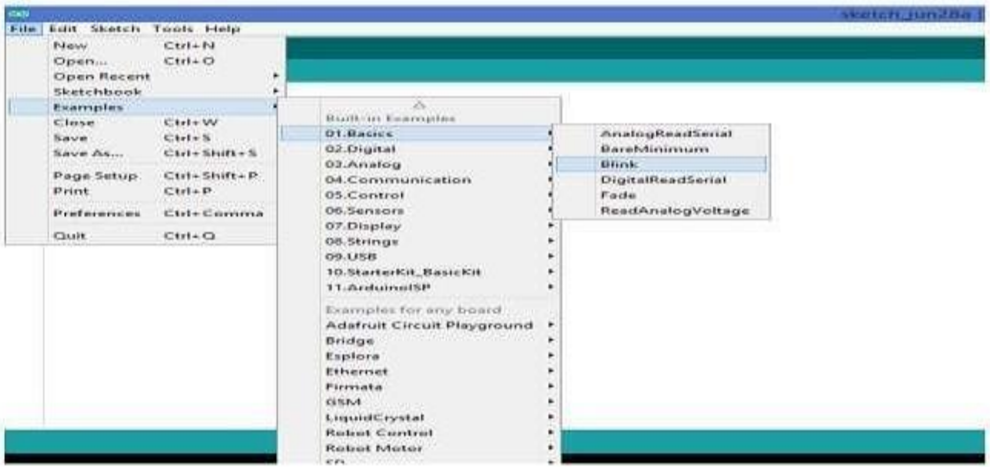
Step-5 Click on the Arduino board, **ATMEGA 328P IC.**


DUINO1
Double click on this ATMEGA IC
Change the Resistor value to 220 ohms.
R1
220
D1
LED-BLUE
ARDUINO UNO R3
Complete Schematic

Oh!.. before clicking it, we have work to do.

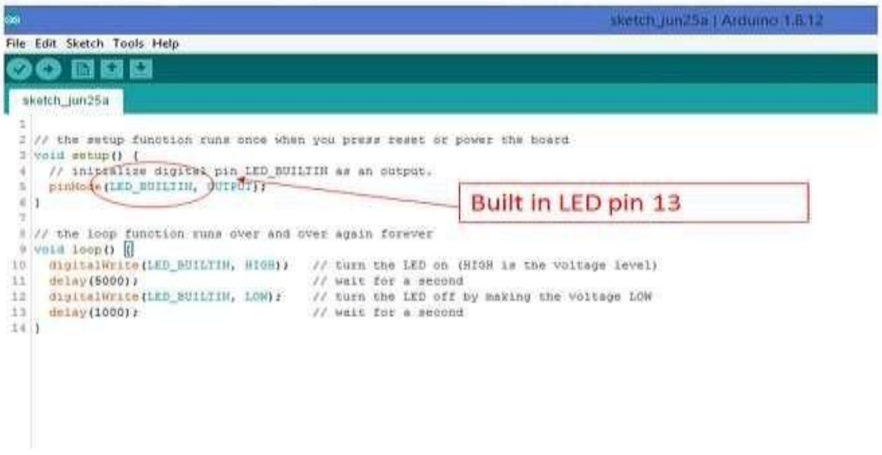
Step-6
Open the Arduino sample Blink program and follow the steps.

a) Open a new blink program...



Selecting the program

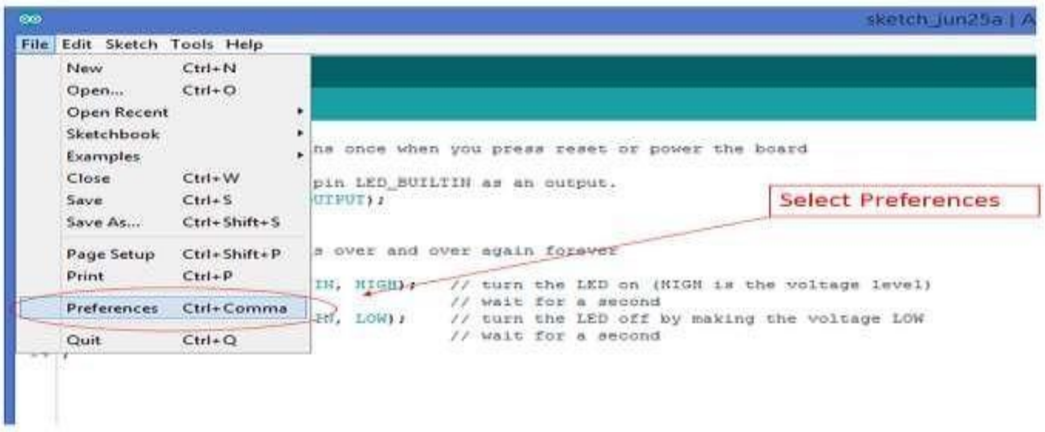
b) Blink Example code...



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(5000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

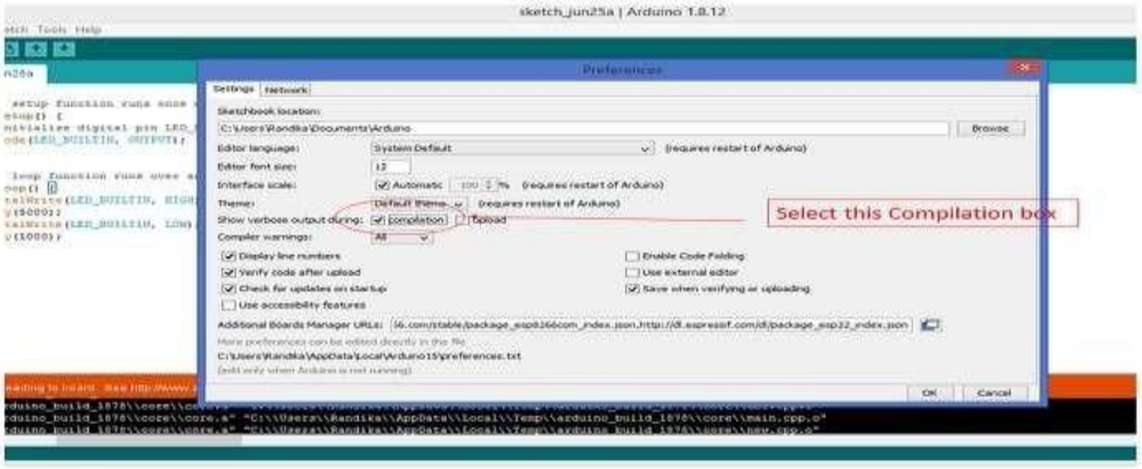
Step-7
Before running this example, we need to change one thing. We need to tik the compilation method.

a) Go to the Preferences window.



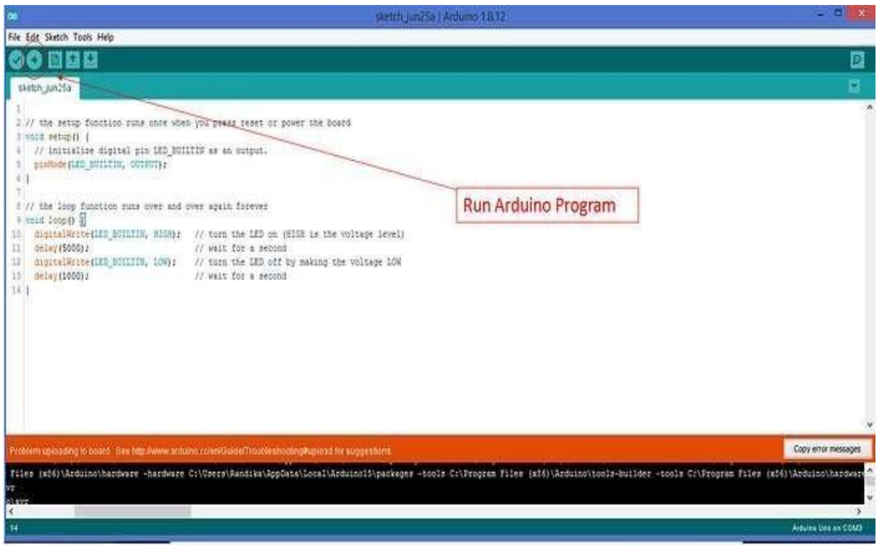
```
ns once when you press reset or power the board

pin LED_BUILTIN as an output.
UTPUT);

s over and over again forever
IN, HIGH);   // turn the LED on (HIGH is the voltage level)
             // wait for a second
IN, LOW);    // turn the LED off by making the voltage LOW
             // wait for a second
```

b) Tik on the compilation box, then save it...



Sketchbook location: C:\Users\Randika\Documents\Arduino
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: [✓] compilation [ ] upload
Compiler warnings: All
[✓] Display line numbers
[✓] Verify code after upload
[✓] Check for updates on startup
[ ] Use accessibility features
[ ] Enable Code Folding
[ ] Use external editor
[✓] Save when verifying or uploading
Additional Boards Manager URLs: l6.com/stable/package_esp8266com_index.json,http://dl.espressif.com/dl/package_esp32_index.json
More preferences can be edited directly in the file
C:\Users\Randika\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

c) Run the program….



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(5000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

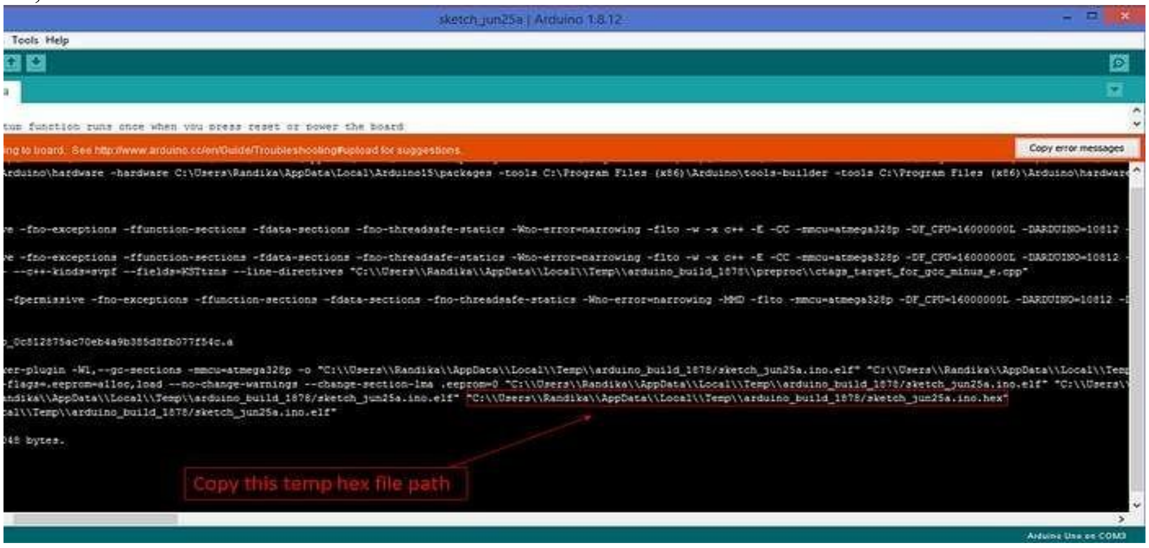
```
Problem uploading to board.  See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
Files (x86)\Arduino\hardware -hardware C:\Users\Randika\AppData\Local\Arduino15\packages -tools C:\Program Files (x86)\Arduino\tools-builder -tools C:\Program Files (x86)\Arduino\hardware
vr
pavr
```

d) Selection of Board->File>Preferences->Show verbose output during: click on compilation box

e) Go to Tool menu>Board->select Arduino uno board.

f) After the Successful compiling of the program, you need to copy the hex file path.

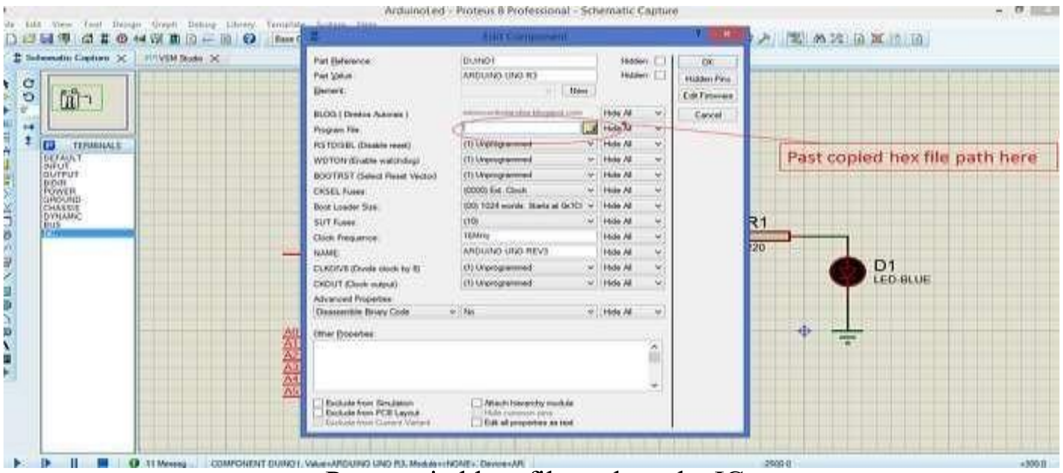
In my case, I've got some errors on the port. It is not a serious matter for Arduino Uno, MEGA, Nano, etc.



```
tup function runs once when you press reset or power the board
ng to board.  See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
rduino\hardware -hardware C:\Users\Randika\AppData\Local\Arduino15\packages -tools C:\Program Files (x86)\Arduino\tools-builder -tools C:\Program Files (x86)\Arduino\hardware

ve -fno-exceptions -ffunction-sections -fdata-sections -fno-threadsafe-statics -Wno-error=narrowing -flto -w -x c++ -E -CC -mmcu=atmega328p -DF_CPU=16000000L -DARDUINO=10812 -

ve -fno-exceptions -ffunction-sections -fdata-sections -fno-threadsafe-statics -Wno-error=narrowing -flto -w -x c++ -E -CC -mmcu=atmega328p -DF_CPU=16000000L -DARDUINO=10812 -
- --c++-kinds=svpf --fields=KSTtzns --line-directives "C:\\Users\\Randika\\AppData\\Local\\Temp\\arduino_build_1878\\preproc\\ctags_target_for_gcc_minus_e.cpp"

 -fpermissive -fno-exceptions -ffunction-sections -fdata-sections -fno-threadsafe-statics -Wno-error=narrowing -MMD -flto -mmcu=atmega328p -DF_CPU=16000000L -DARDUINO=10812 -

_0c812875ac70eb4a9b385d8fb077f54c.a

er-plugin -Wl,--gc-sections -mmcu=atmega328p -o "C:\\Users\\Randika\\AppData\\Local\\Temp\\arduino_build_1878/sketch_jun25a.ino.elf" "C:\\Users\\Randika\\AppData\\Local\\Temp
-flags=.eeprom=alloc,load --no-change-warnings --change-section-lma .eeprom=0 "C:\\Users\\Randika\\AppData\\Local\\Temp\\arduino_build_1878/sketch_jun25a.ino.elf" "C:\\Users\\
ndika\\AppData\\Local\\Temp\\arduino_build_1878/sketch_jun25a.ino.elf" "C:\\Users\\Randika\\AppData\\Local\\Temp\\arduino_build_1878/sketch_jun25a.ino.hex"
cal\\Temp\\arduino_build_1878/sketch_jun25a.ino.elf"

48 bytes.
```

Copying the hex file path
OR
Save Arduino IDE program on desktop it create folder then under that ->Build folder>arduino.avr.uno->.hex file ->copy and paste on following path

Step 8



ArduinoLed - Proteus 8 Professional - Schematic Capture

Edit Component

Part Reference:	DUINO1	Hidden:
Part Value:	ARDUINO UNO R3	Hidden:
Element:		New
BLOG (Diseños Autorais)	[illegible]	Hide All
Program File:		Hide All
RSTDISBL (Disable reset)	(1) Unprogrammed	Hide All
WDTON (Enable watchdog)	(1) Unprogrammed	Hide All
BOOTRST (Select Reset Vector)	(1) Unprogrammed	Hide All
CKSEL Fuses:	(0000) Ext. Clock	Hide All
Boot Loader Size:	(00) 1024 words. Starts at 0x1C0	Hide All
SUT Fuses:	(10)	Hide All
Clock Frequency:	16MHz	Hide All
NAME:	ARDUINO UNO REV3	Hide All
CLKDIV8 (Divide clock by 8)	(1) Unprogrammed	Hide All
CKOUT (Clock output)	(1) Unprogrammed	Hide All
Advanced Properties:		
Disassemble Binary Code	No	Hide All

Other Properties:

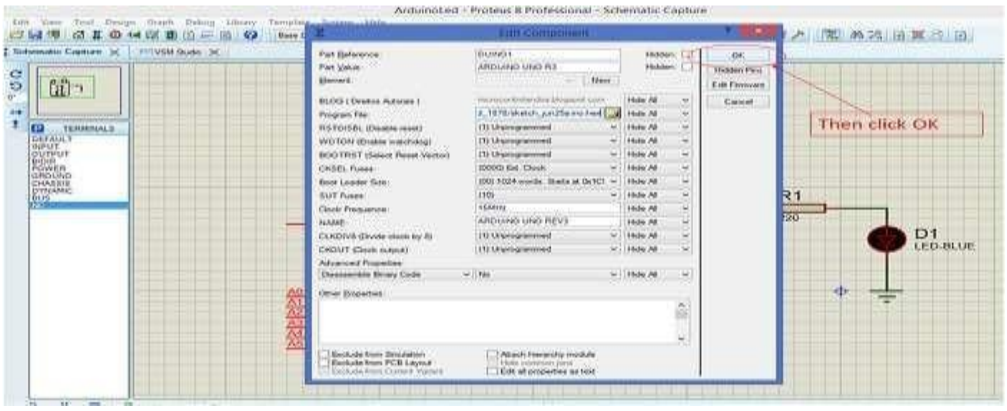
Exclude from Simulation
Exclude from PCB Layout
Exclude from Current Variant
Attach hierarchy module
Hide common pins
Edit all properties as text

OK
Hidden Pins
Edit Firmware
Cancel

COMPONENT DUINO1, Value=ARDUINO UNO R3, Module=<NONE>, Device=AR

Past copied hex file path to the IC

Step 9



ArduinoLed - Proteus 8 Professional - Schematic Capture

Edit Component

Part Reference:	DUINO1	Hidden:
Part Value:	ARDUINO UNO R3	Hidden:
Element:		New
BLOG (Diseños Autorais)	[illegible]	Hide All
Program File:	2_1978/sketch_jun25a.ino.hex	Hide All
RSTDISBL (Disable reset)	(1) Unprogrammed	Hide All
WDTON (Enable watchdog)	(1) Unprogrammed	Hide All
BOOTRST (Select Reset Vector)	(1) Unprogrammed	Hide All
CKSEL Fuses:	(0000) Ext. Clock	Hide All
Boot Loader Size:	(00) 1024 words. Starts at 0x1C0	Hide All
SUT Fuses:	(10)	Hide All
Clock Frequency:	16MHz	Hide All
NAME:	ARDUINO UNO REV3	Hide All
CLKDIV8 (Divide clock by 8)	(1) Unprogrammed	Hide All
CKOUT (Clock output)	(1) Unprogrammed	Hide All
Advanced Properties:		
Disassemble Binary Code	No	Hide All

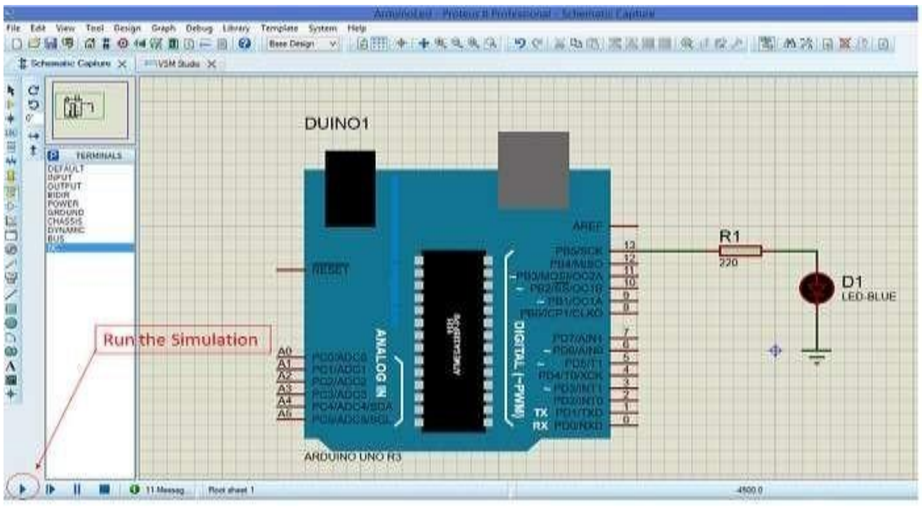
Other Properties:

Exclude from Simulation
Exclude from PCB Layout
Exclude from Current Variant
Attach hierarchy module
Hide common pins
Edit all properties as text

OK
Hidden Pins
Edit Firmware
Cancel

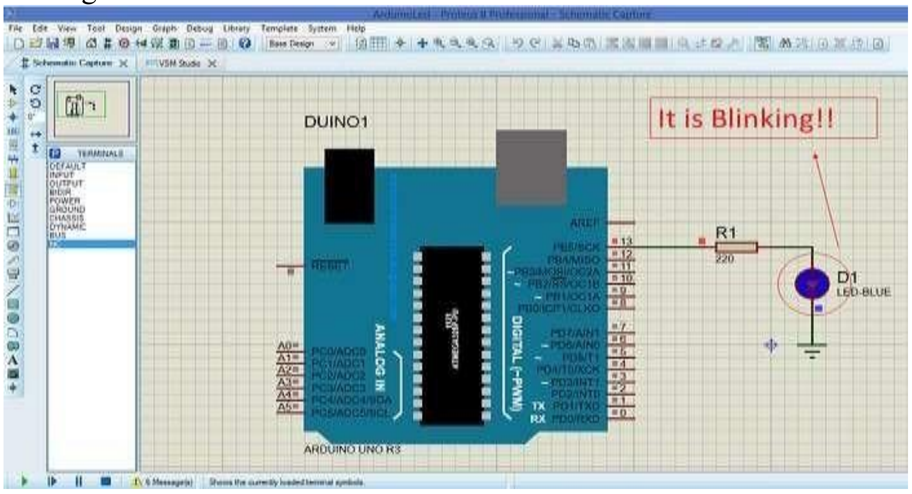
Save the configuration

Step 10



Run the simulation

Finally, it's running… .



Lab Assignments

1. **To write a program to sense the available networks using Arduino**
2. **To write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.**
3. **To write a program to detects the vibration of an object with sensor using Arduino.**
4. **To write a program to sense a finger when it is placed on the board Arduino.**
5. **To write a program to connect with the available Wi-Fi using Arduino.**
6. **To write a program to get temperature notification using Arduino.**
7. **To write a program for LDR to vary the light intensity of LED using Arduino.**

Signature of the instructor   **Date**

0: Not done Assignment   **2: Late Complete**   **4: Complete**

1:Incomplete   **3:Needs improvement**   **5:Well Done**

Raspberry Pi in IoT

What is raspberry pi in IoT?

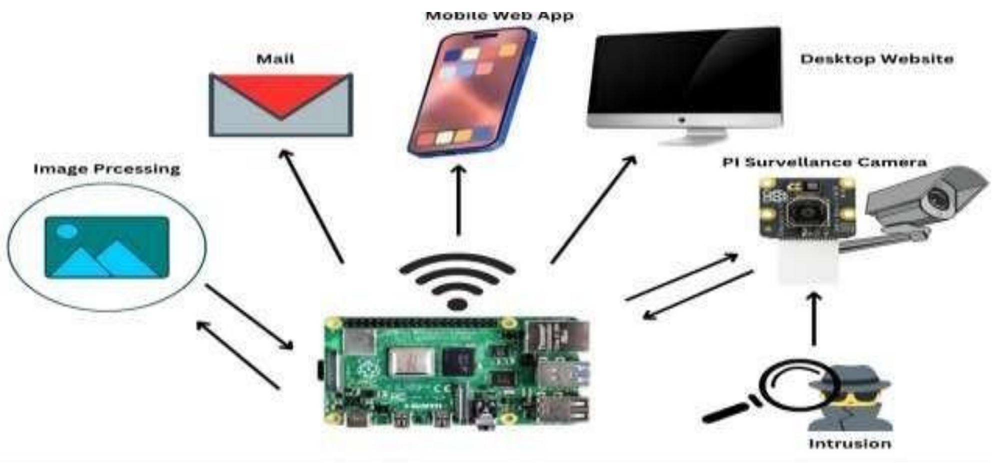
One crucial component of the Internet of Things (IoT) is the Raspberry Pi. Users can construct their own smart devices and link them to other loT devices thanks to its compact, potent design. It's easy to build connected household products or even sophisticated industrial systems with a few basic components.
Because of its inexpensive price, enthusiasts who wish to start tinkering without going over budget can use it. Most significantly, though, is that because it is open-source, developers have flexible access to create apps that quickly and inexpensively integrate IOT functionality across several platforms and hardware layers. This is because programming activities need little energy.

What type of IoT device is the raspberry pi?
The Raspberry Pi is a versatile device with a range of digital media, communications, and computer applications. Initially developed as a low-cost alternative to smartphones and tablets, it has evolved into a powerful tool for robotics development. Its affordability allows users to create custom programs in various languages, including HTML5, CSS3, Python, Ruby, and C++, with a special interface called Raspbian OS for secure data import and export

Why is Raspberry pi used for IoT devices?

The ideal platform for Internet of Things (IoT) device powering is the Raspberry Pi. It's compact, affordable, and incredibly functional. It can be simply integrated into practically any project or device with built-in Bluetooth and Wi-Fi communication choices, negating the need for large overhead infrastructure.

.



•

Benefit of IoT in Raspberry Pi

1. **Cost savings:** Automated processes allow machines to operate more quickly and with less need for human intervention, which results in more economical operations all around.

2. **Enhanced privacy and security:** By combining the most recent technological advancements,

like blockchain, artificial intelligence, and machine learning, with the secure architecture of the Raspberry Pi, you can add an extra line of defence against potential threats and bad actors attempting to access your system through the internet or other digital channels.

3. **Real-time sensor tracking**: You have direct visibility over all actions occurring on site thanks to the installation of sensors inside a networked system, enabling you to take prompt action when necessary.

4. **Data visualization insights**: When making decisions based on the myriad crucial bits of information gathered from their surroundings, managers can swiftly spot patterns that might point to areas that require attention thanks to the real-time monitoring that Raspberry Pi's loT systems enables.

5. **Improved asset utilization:** By utilizing Internet of Things (IoT) technology, places such as major industrial sites can benefit from sophisticated analytics tools while gathering valuable data about device usage levels, which enables improved deployment strategies throughout Implementation of IoT with Raspberry PI

The implementation of IoT with Raspberry Pi has revolutionized the way businesses operate. This powerful platform is designed for scalability and offers a range of benefits.
Professionals can make it easy by combining inexpensive hardware components like the Raspberry Pi with powerful software tools like Node-RED or MQTT.
The system enables remote creation of complex systems using secure environments and modern technologies like AI and facial recognition, enhancing management and collaboration between stakeholders.

What are the Raspberry Pi Commands?

The Raspberry Pi is an easy, charge card measured PC that connects to a PC screen or TV, and utilizes a standard console and mouse. It is an able little gadget that empowers individuals, all things considered, to investigate processing, and to figure out how to program in dialects like Scratch and Python.

Utilization of Raspberry Pi command line?

Open Raspberry Pi Configuration (Menu > Preferences > Raspberry Pi Configuration). Change the Boot setting ‘To CLI’ and snap OK. Presently when you reboot, you’ll start in the command line (enter startx to boot into the work area).

General Commands

apt-get update: Updates the rundown of bundles on your framework to the rundown in the vaults. Use it before putting in new bundles to ensure you are introducing the most recent form.

apt-get upgrade: Redesigns the entirety of the product bundles you have introduced.

startx: Opens the Graphical User Interface.

clear: Clears recently run orders and text from the terminal screen.

date: Prints the current date.

find / -name text1.txt: Searches the whole system for the file text1.txt and outputs a list of all directories that contain the file.

reboot: To reboot immediately.

nano text.txt: Opens the file text.txt in the Linux text editor Nano.

Power off: To shut down immediately.

raspi-config: Opens the configuration settings menu.

shutdown -h now: To shut down immediately.

shutdown -h 11:11: To shutdown at 11:11 AM.

File And Directory Commands

mv YYY: Moves the file or directory named YYY to a specified location.

rm text.txt: Deletes the file text.txt.

rmdir a_directory: Deletes(if it is empty) the directory a_directory .

cat text.txt: Displays the contents of the file text.txt.

cd /abc/xyz: Changes the current directory to the /abc/xyz directory.

cp XXX: Copies the file or directory XXX and pastes it to a specified location

mkdir text_directory: Creates a new directory named text_directory inside the current directory.

touch text.txt: Creates a new, empty file named text.txt in the current directory.

ls -l: Lists files in the current directory, along with file size, date modified, and permissions.

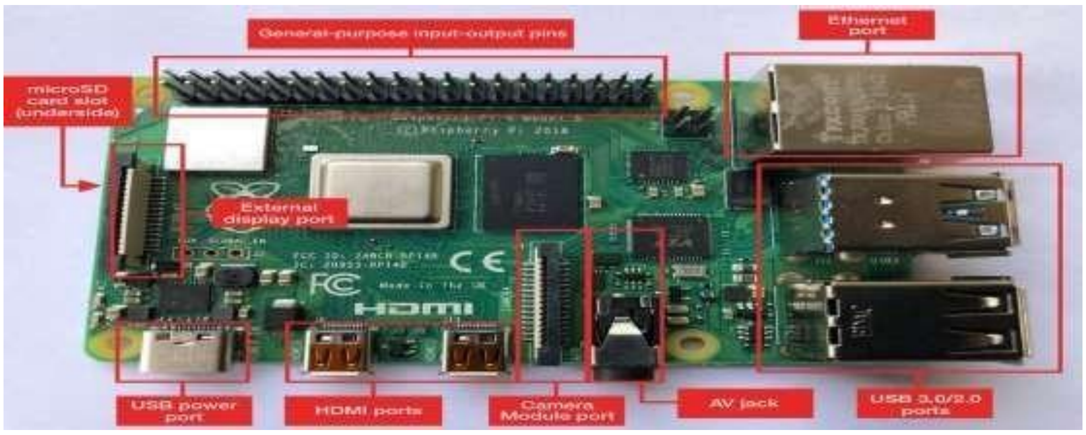
Chgrp: To change the group ownership of a file

Chown: allows users to change the owner of files and directories.

Cron: a *software utility* that is provided by a Linux-like OS that operates the scheduled operation on a predetermined period

Raspberry Pi Board Overview(https://realpython.com/python-raspberry-pi/)

The Raspberry Pi comes in a variety of form factors for different use cases. In this tutorial, you'll be looking at the most recent version, the Raspberry Pi 4. Below is the board layout of the Raspberry Pi 4. While this layout is slightly different from previous models of the Raspberry Pi, most of the connections are the same. The setup described in the next section should be the same for both a Raspberry Pi 3 and a Raspberry Pi 4:



The Raspberry Pi 4 board contains the following components:
General-purpose input–output pins: These pins are used to connect the Raspberry Pi to electronic components.

Ethernet port: This port connects the Raspberry Pi to a wired network. The Raspberry Pi also has Wi-Fi and Bluetooth built in for wireless connections.
Two USB 3.0 and two USB 2.0 ports: These USB ports are used to connect peripherals like a keyboard or mouse. The two black ports are USB 2.0 and the two blue ports are USB 3.0

AV jack: This AV jack allows you to connect speakers or headphones to the Raspberry Pi.
Camera Module port: This port is used to connect the official Raspberry Pi Camera Module, which enables the Raspberry Pi to capture images.

HDMI ports: These HDMI ports connect the Raspberry Pi to external monitors. The Raspberry Pi 4 features two micro HDMI ports, allowing it to drive two separate monitors at the same time.
USB power port: This USB port powers the Raspberry Pi. The Raspberry Pi 4 has a USB Type-C port, while older versions of the Pi have a micro-USB port.

External display port: This port is used to connect the official seven-inch Raspberry Pi touch display for touch-based input on the Raspberry Pi.
microSD card slot (underside of the board): This card slot is for the microSD card that contains the Raspberry Pi operating system and files.

Examples:

Ex-1)Program to perform addition, subtraction, multiplication and division on two input numbers in Python

```
num1 = int(input("Enter First Number: "))
num2 = int(input("Enter Second Number: "))

print("Enter which operation would you like to perform?")
ch = input("Enter any of these char for specific operation +,-,*,/: ")

result =  0
if ch == '+':
   result = num1 + num2
elif ch == '-':
   result = num1 - num2
elif ch == '*':
   result = num1 * num2
elif ch == '/':
   result = num1 / num2
else:
   print("Input character is not recognized!")

print(num1, ch , num2, ":", result)
```

Output 1: Addition

```
Enter First Number: 100
Enter Second Number: 5
Enter which operation would you like to perform?
Enter any of these char for specific operation +,-,*,/: +
100 + 5 : 105
```

Ex-2) Print Current time for 10 times with interval of 10 seconds
import time from datetime import datetime

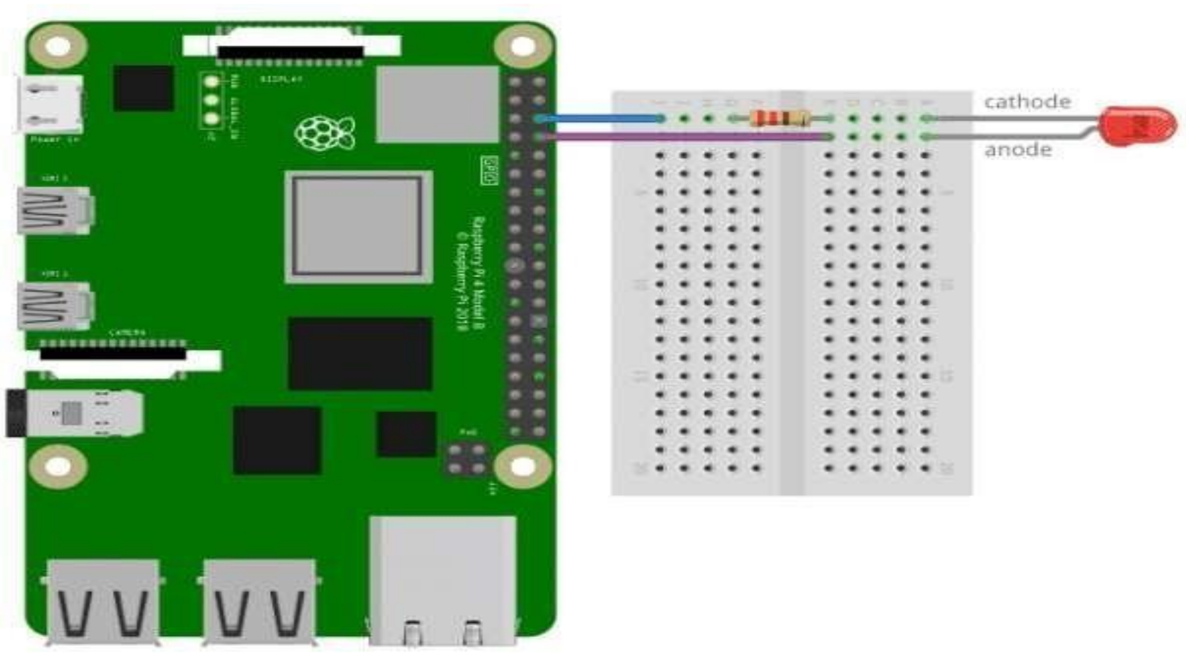
```
#For 10 times
for x in range(10):
  # Get current time
  now = datetime.now()
  # Make a string of it
  current_time = now.strftime("%H:%M:%S")
  # Print it
  print(current_time)

  # Wait for 10 seconds
  time.sleep(10)
```

Output:

```
14:33:33
14:33:43
14:33:53
14:34:03
14:34:13
14:34:23
14:34:33
14:34:43
14:34:53
14:35:03
```

How to Control LEDs With the Raspberry Pi

Using the Raspberry Pi to control an LED might seem like a basic or boring task. However, the same hardware and programming concepts used to control an LED can be used to control a wide variety of sensors and modules. Learning how to control an LED with the Raspberry Pi's GPIO pins will open up a whole new variety of devices you can use with the Raspberry Pi.
Connect the LED to the Raspberry Pi
Let's jump right in by connecting the LED to the Raspberry Pi.
For this project we will need the following:
Raspberry Pi
One LED
One 330 Ohm resistor
Jumper wires
Breadboard
Connect the components as shown in the wiring diagram below.



The 330 Ohm resistor is a current limiting resistor. Current limiting resistors should always be used when connecting LEDs to the GPIO pins. If an LED is connected to a GPIO pin without a resistor, the LED will draw too much current, which can damage the Raspberry Pi or burn out the LED. Here is a nice calculator that will give you the value of a current limiting resistor to use for different LEDs.

Python Code

After connecting the hardware components, the next step is to create a Python program to switch on and off the LED. This program will make the LED turn on and off once every second and output the status of the LED to the terminal.

The first step is to create a Python file. To do this, open the Raspberry Pi terminal and type nano LED.py. Then press Enter.

This will create a file named LED.py and open it in the Nano text editor. Copy and paste the Python code below into Nano and save and close the file.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(14,GPIO.OUT)

# While loop
while True:
        # set GPIO14 pin to HIGH
        GPIO.output(14,GPIO.HIGH)
        # show message to Terminal
        print "LED is ON"
        # pause for one second
        time.sleep(1)

        # set GPIO14 pin to HIGH
        GPIO.output(14,GPIO.LOW)
        # show message to Terminal
        print "LED is OFF"
        # pause for one second
        time.sleep(1)
```

Explanation of the Code

At the top of the program we import the RPi.GPIO and time libraries. The RPi.GPIO library will allow us to control the GPIO pins. The time library contains the sleep() function that we will use to make the LED pause for one second.

Next we initialize the GPIO object with GPIO.setmode(GPIO.BCM). We are using the BCM pin numbering system in this program.

We use GPIO.setwarnings(False) to disable the warnings and GPIO.setup(14,GPIO.OUT) is used to set GPIO14 as an output.

Now we need to change the on/off state of GPIO14 once every second. We do this with the GPIO.output() function. The first parameter of this function is the GPIO pin that will be switched high or low. We have the LED connected to GPIO14 in this circuit, so the first argument is 14.
The second parameter of the GPIO.output() function is the voltage state of the GPIO pin. We can use either GPIO.HIGH or GPIO.LOW as an argument to turn the pin on or off.
Each GPIO.output() function in the code above is followed by a sleep() function that causes the pin to hold its voltage state for the time (in seconds) defined in the parameter of the function. In this program we are switching the LED on and off once every second so the argument is 1. You can change this value to make the LED blink on and off faster or slower.
Run the Program
Run the Python program above by entering the following into the Raspberry Pi's terminal:

```
sudo python LED.py
```

You should see the LED blinking on and off once every second.

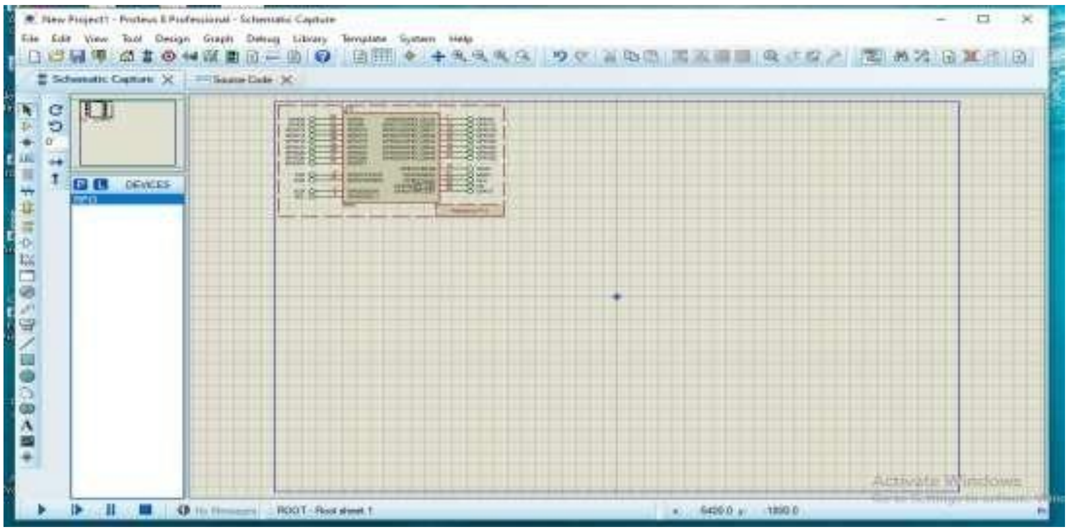
https://www.youtube.com/watch?v=xoaddp1hAXo

Raspberry Pi LED Blink Simulation on Proteus

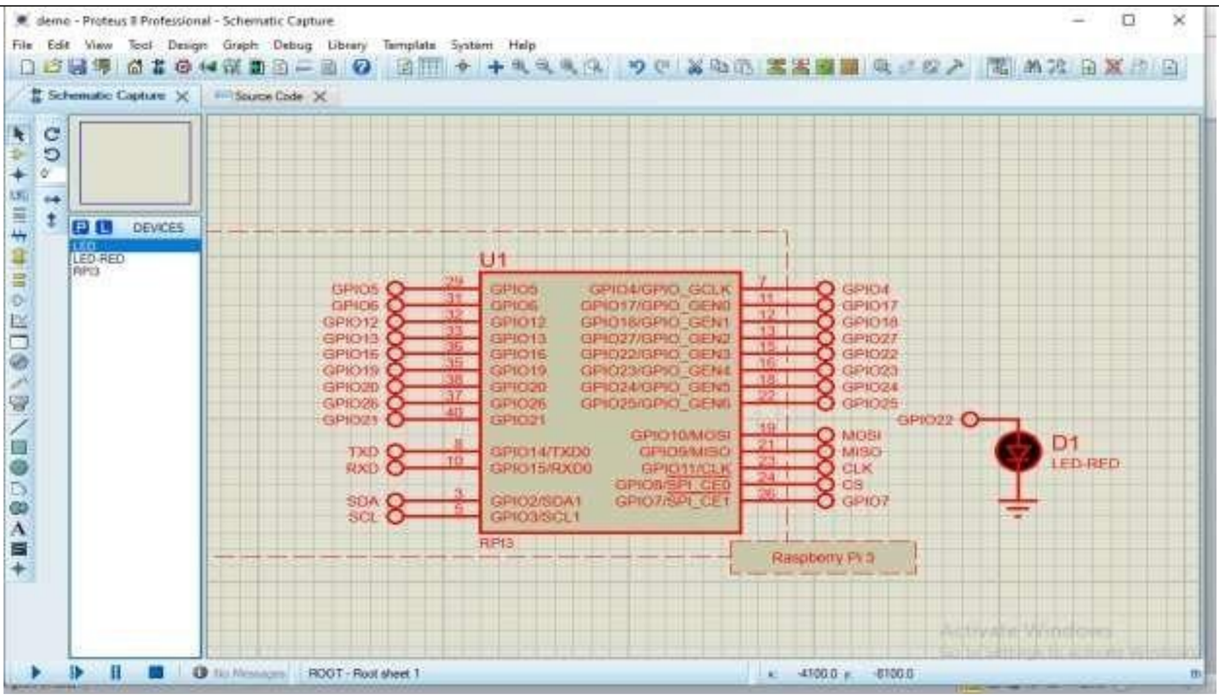
Now open the Proteus software and follow the below steps.

Step-1 Click on New project>>create a schematic from the selected template >>Do not create a PCB layout>>click on create framework project and select family Raspberry Pi >>finish following window opened.

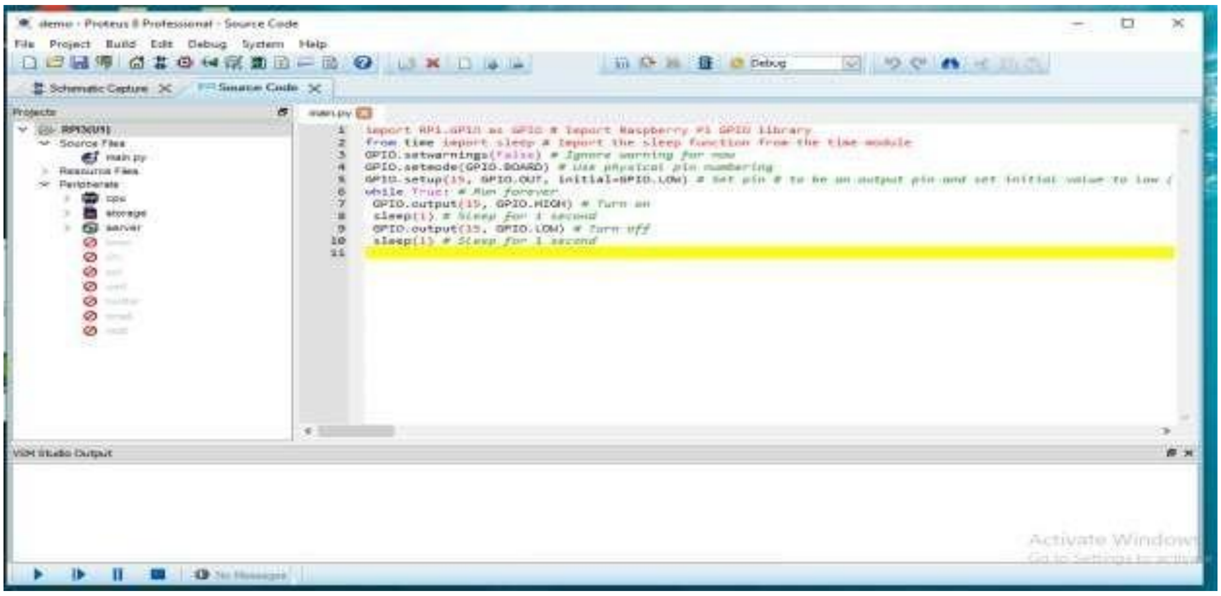
It having two windows one is schematic capture window and another on source code.


New Project1 - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Source Code
DEVICES
Activate Windows
ROOT - Root sheet 1

Step-2 Draw the schematic on schematic window and write program code for the schematic in source code



write program code for the schematic in source code



```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(15, GPIO.OUT, initial=GPIO.LOW) # Set pin # to be an output pin and set initial value to low (
while True: # Run forever
 GPIO.output(15, GPIO.HIGH) # Turn on
 sleep(1) # Sleep for 1 second
 GPIO.output(15, GPIO.LOW) # Turn off
 sleep(1) # Sleep for 1 second
```

How to Simulate Arduino/ Raspberry Pi Projects Using Proteus
For all detailed process for simulate project using arduino and raspberry pi go through given link
https://maker.pro/arduino/projects/how-to-simulate-arduino-projects-using-proteus
https://www.youtube.com/live/MoZCyv8zQHQ

Lab Assignments

1. Start Raspberry Pi and try various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, pingetc.

2. Run some python programs on Pi like: a) Read your name and print Hello message with name b)Read two numbers and print their sum, difference, product and division. c) Word and character count of a given string. d)Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

3. Run some python programs on Pi like: a) Print a name 'n' times, where name and n are read from standard input, using for and while loops. b) Handle Divided by Zero Exception. c) Print current time for 10 times with an interval of10seconds. d) Read a fileline byline and print the word count of each line

4. Run some python programs on Pi like a) Light an LED through Python program b) Get input from two switches and switch on corresponding LEDs c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file

Signature of the instructor   **Date**

Assignment Evaluation

0:Not done	**2:Late Complete**	**4:Complete**
1:Incomplete	**3:Needs improvement**	**5:Well Do**

Case Study: “Raspberry Pi based Home Automation System”

Objective: This Project is developed using Raspberry Pi hardware and Proteus simulation. The project objective is to for user convenience, security, and energy consumption within home.

Requirements: Proteus software, also need to install MQ2, PIR etc. sensor libraries in proteus software.

Features: This system includes features like light on/off system, fire detection, and fan control using temperature sensor.

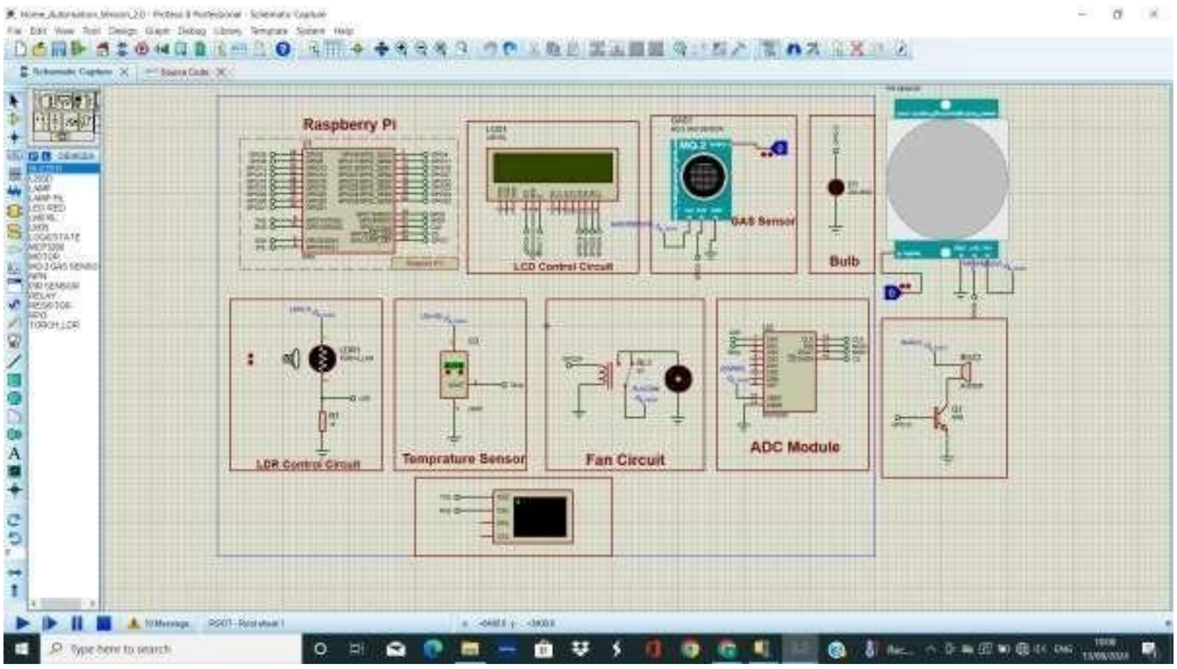
The light on/off system allows users to remotely control and automate the lighting in their homes. This feature improves energy management by effective utilization of lighting resources.

The fire detection component is for safety purpose of home. The presence of fire is detected by smoke and heat sensor. The alarm is turn on and gives alarm and sent alert to central ministering system.

The fan control system, in which a temperature sensor is used to regulate the operation of fans based on the ambient temperature. When the temperature exceeds a predefined value, the system tern on the fans to provide cooling. Once the temperature falls within an acceptable range, the fans are automatically turned off.

It also includes human detection using PIR sensor.

Schematic Diagram:



Conclusion: The integration of Raspberry Pi with Proteus simulation allows us to test and validation of actual functionality of the system prior to actual implementation. We can make necessary modification once we found defect/issue in the system or required new changes/ adjustment.

Proteus simulation gives surety, reliability and efficiency of the system, allowing for whole integration and best performance in actual applications.

ADC Value (approx)	Temp	Volts
0	-50	0.00
78	-25	0.25
155	0	0.50
233	25	0.75
310	50	1.00
465	100	1.50
775	200	2.50
1023	280	3.30

Table 1.1: ADC Values

Source Code:-

```
import spidev
import time
import RPi.GPIO as GPIO
import pio
import Ports

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

pio.uart=Ports.UART () # Define serial port

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Define GPIO to LCD mapping
LCD_RS = 7
LCD_E  = 11
LCD_D4 = 12
LCD_D5 = 13
LCD_D6 = 15
LCD_D7 = 16
bulb_pin =  32
motor_pin =  18
pir_pin = 31
gas_pin = 29
```

```
buzzer_pin =33
# Define sensor channels
ldr_channel = 0
temp_channel = 1
'''
define pin for lcd
'''
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1

GPIO.setup(LCD_E, GPIO.OUT) # E
GPIO.setup(LCD_RS, GPIO.OUT) # RS
GPIO.setup(LCD_D4, GPIO.OUT) # DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7, GPIO.OUT) # DB7
GPIO.setup(bulb_pin, GPIO.OUT) # DB7
GPIO.setup(motor_pin, GPIO.OUT) # DB7
GPIO.setup(buzzer_pin, GPIO.OUT) # DB7
GPIO.setup(gas_pin, GPIO.IN) # DB7
GPIO.setup(pir_pin, GPIO.IN) # DB7
# Define some device constants
LCD_WIDTH = 16     # Maximum characters per line
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line
'''
Function Name :lcd_init()
Function Description : this function is used to initialized lcd by sending the different commands
'''
def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)
'''
Function Name :lcd_byte(bits ,mode)
Fuction Name :the main purpose of this function to convert the byte data into bit and send to lcd port
'''
def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True for character
  #        False for command

  GPIO.output(LCD_RS, mode) # RS
  # High bits

 GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
  GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
  GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
  GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
  GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

  # Low bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
  GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
  GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
  GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
  GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()
'''
Function Name : lcd_toggle_enable()
Function Description:basically this is used to toggle Enable pin
'''
def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
'''
Function Name :lcd_string(message,line)
Function  Description :print the data on lcd
'''
def lcd_string(message,line):
  # Send string to display
  message = message.ljust(LCD_WIDTH," ")

  lcd_byte(line, LCD_CMD)


 for i in range(LCD_WIDTH):
   lcd_byte(ord(message[i]),LCD_CHR)

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data
# Function to calculate temperature from
# TMP36 data, rounded to specified
# number of decimal places.
def ConvertTemp(data,places):
 # ADC value as per table 1.1
  temp = ((data * 330)/float(1023))
  temp  = round(temp,places)
  return temp

# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(1)
while 1:
  gas_data = GPIO.input(gas_pin)
  if(gas_data == True):
   lcd_byte(0x01,LCD_CMD) # 000001 Clear display
   lcd_string("Fire Detected",LCD_LINE_1)
   lcd_string("Buzzer On",LCD_LINE_2)
   GPIO.output(bulb_pin, False)
   GPIO.output(motor_pin, False)
   GPIO.output(buzzer_pin, True)
   time.sleep(0.5)
   while(1):
     lcd_byte(0x01,LCD_CMD) # 000001 Clear display
     lcd_string("Sending Message",LCD_LINE_1)
     pio.uart.println("AT")
     pio.uart.println("AT+CMGF=1")
     pio.uart.println("AT+CMGS=\"+919922512017\"\r")
     pio.uart.println("Fire Detected")
  pir_data = GPIO.input(pir_pin)
  if(pir_data  ==  True):
   light_level = ReadChannel(ldr_channel)
   time.sleep(0.2)
   lcd_byte(0x01,LCD_CMD) # 000001 Clear display
   lcd_string("Person Detected ",LCD_LINE_1)
   time.sleep(0.5)
   lcd_byte(0x01,LCD_CMD) # 000001 Clear display
   lcd_string("Automatic Light and ",LCD_LINE_1)
   lcd_string("Fan System Active ",LCD_LINE_2)
   time.sleep(0.5)
   lcd_byte(0x01,LCD_CMD) # 000001 Clear display
   lcd_string("Light Intencsty ",LCD_LINE_1)
   lcd_string(str(light_level),LCD_LINE_2)
   time.sleep(0.5)
   if(light_level < 100 ):
    lcd_byte(0x01,LCD_CMD) # 000001 Clear display
    lcd_string("Bulb ON",LCD_LINE_2)
    GPIO.output(bulb_pin, True)
    time.sleep(0.5)
   else:
    lcd_byte(0x01,LCD_CMD) # 000001 Clear display
    lcd_string("Bulb OFF",LCD_LINE_2)
    GPIO.output(bulb_pin, False)
    time.sleep(0.5)
   # Print out results
   temp_level = ReadChannel(temp_channel)
   time.sleep(0.5)
   temperature      = ConvertTemp(temp_level,2)
   lcd_byte(0x01,LCD_CMD) # 000001 Clear display
   lcd_string("Temperature ",LCD_LINE_1)
   lcd_string(str(temperature),LCD_LINE_2)
   time.sleep(0.5)
   if(temperature > 30):
    lcd_byte(0x01,LCD_CMD) # 000001 Clear display
    lcd_string("Fan ON ",LCD_LINE_1)
    GPIO.output(motor_pin, True)
    time.sleep(0.5)
   else:
    lcd_byte(0x01,LCD_CMD) # 000001 Clear display
    lcd_string("Fan Off ",LCD_LINE_1)
    GPIO.output(motor_pin, False)
    time.sleep(0.5)
  else:
    lcd_byte(0x01,LCD_CMD) # 000001 Clear display
    lcd_string("Person Not Detected",LCD_LINE_1)
    GPIO.output(motor_pin, False)
    GPIO.output(bulb_pin, False)
    time.sleep(0.5)
```

Assignment on Case Study:

All Students execute any one case study step by step as per syllabus.