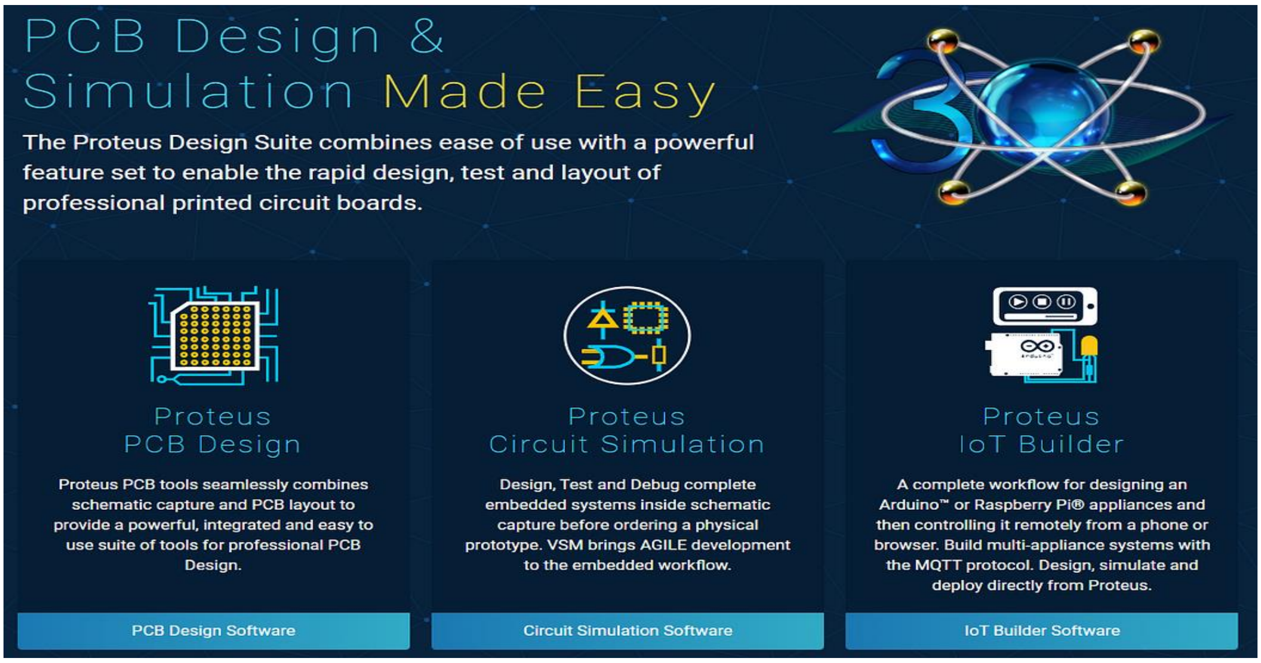
Lab 1 : INTRODUCTION ARDUINO AND LED

1. **Objectives:** After this lab, the learner will be able to
   - Using proteus to simulate an simple IoT project
   - Using Arduino IDE to program Arduino board
2. **Facilities**
   - Arduino Software: Download from here
   - Proteus Simulator
3. **Prerequest**
   - Basis electronic
   - C programming language
4. **Introduction to Proteus**

4.1. What is proteus

Proteus Design Suite is *one of the best* **circuit simulation programs** in its field. It also has Proteus Design Suite, **the PCB design**, and **IoT builder software**.


PCB Design & Simulation Made Easy

The Proteus Design Suite combines ease of use with a powerful feature set to enable the rapid design, test and layout of professional printed circuit boards.

Proteus PCB Design

Proteus PCB tools seamlessly combines schematic capture and PCB layout to provide a powerful, integrated and easy to use suite of tools for professional PCB Design.

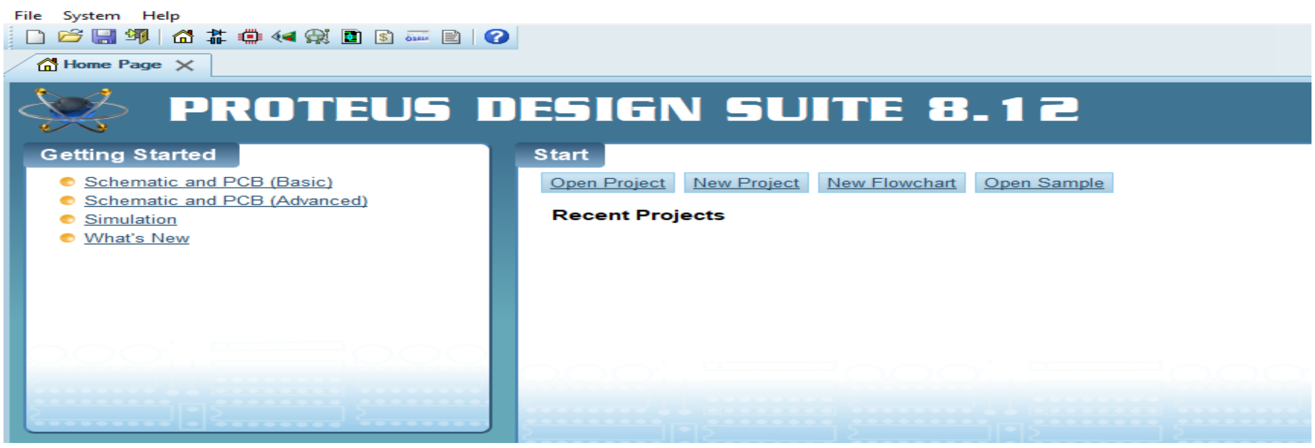
Proteus Circuit Simulation

Design, Test and Debug complete embedded systems inside schematic capture before ordering a physical prototype. VSM brings AGILE development to the embedded workflow.

Proteus IoT Builder

A complete workflow for designing an Arduino™ or Raspberry Pi® appliances and then controlling it remotely from a phone or browser. Build multi-appliance systems with the MQTT protocol. Design, simulate and deploy directly from Proteus.

4.2. How can we use the program?



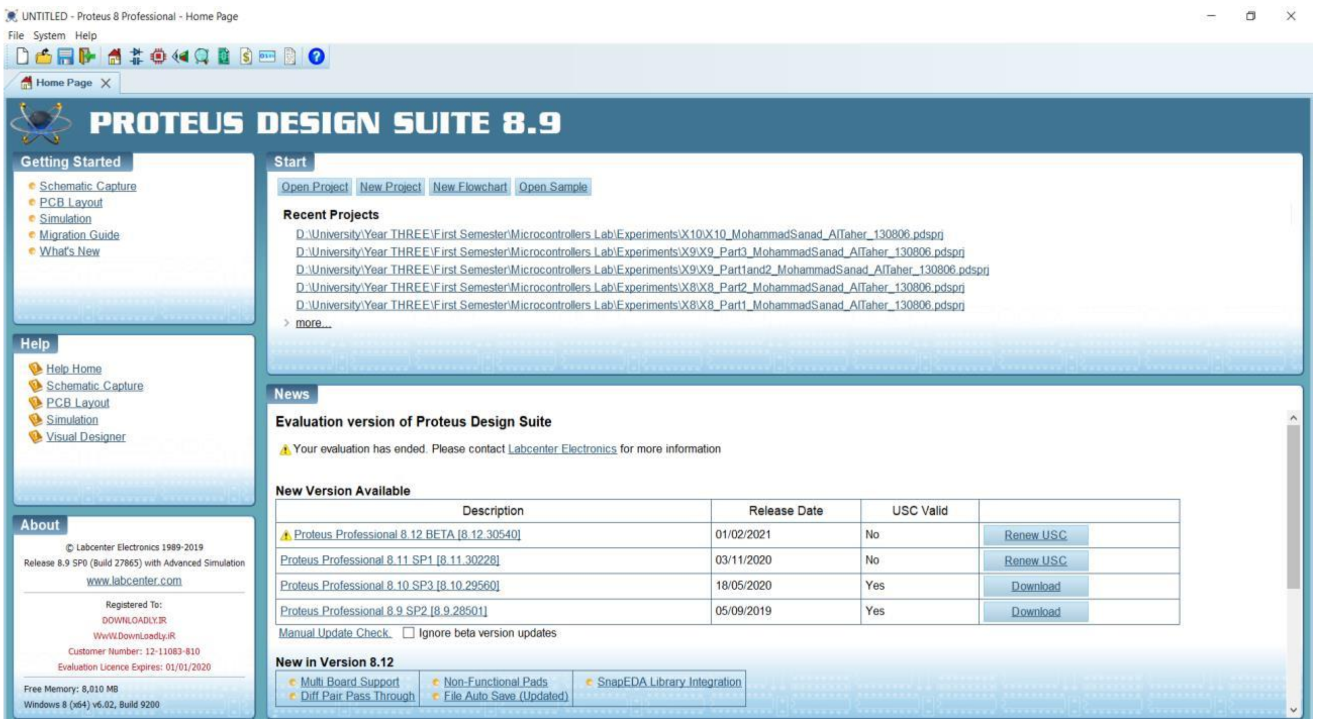
PROTEUS DESIGN SUITE 8.12

Getting Started
- Schematic and PCB (Basic)
- Schematic and PCB (Advanced)
- Simulation
- What's New

Start

Recent Projects

1. **Creating a File:**
   After installing Proteus and activating your license, you will be met with a page similar to that shown below:


UNTITLED - Proteus 8 Professional - Home Page

File System Help

Home Page

PROTEUS DESIGN SUITE 8.9

Getting Started
- Schematic Capture
- PCB Layout
- Simulation
- Migration Guide
- What's New

Help
- Help Home
- Schematic Capture
- PCB Layout
- Simulation
- Visual Designer

About

© Labcenter Electronics 1989-2019
Release 8.9 SP0 (Build 27865) with Advanced Simulation
www.labcenter.com

Registered To:
DOWNLOADLY.IR
WwW.DownLoadLy.iR
Customer Number: 12-11083-810
Evaluation Licence Expires: 01/01/2020

Free Memory: 8,010 MB
Windows 8 (x64) v6.02, Build 9200

Start

Open Project | New Project | New Flowchart | Open Sample

Recent Projects
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X10\X10_MohammadSanad_AlTaher_130806.pdsprj
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X9\X9_Part3_MohammadSanad_AlTaher_130806.pdsprj
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X9\X9_Part1and2_MohammadSanad_AlTaher_130806.pdsprj
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X8\X8_Part2_MohammadSanad_AlTaher_130806.pdsprj
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X8\X8_Part1_MohammadSanad_AlTaher_130806.pdsprj
- more...

News

Evaluation version of Proteus Design Suite

⚠ Your evaluation has ended. Please contact Labcenter Electronics for more information

New Version Available

Description	Release Date	USC Valid	
⚠ Proteus Professional 8.12 BETA [8.12.30540]	01/02/2021	No	Renew USC
Proteus Professional 8.11 SP1 [8.11.30228]	03/11/2020	No	Renew USC
Proteus Professional 8.10 SP3 [8.10.29560]	18/05/2020	Yes	Download
Proteus Professional 8.9 SP2 [8.9.28501]	05/09/2019	Yes	Download

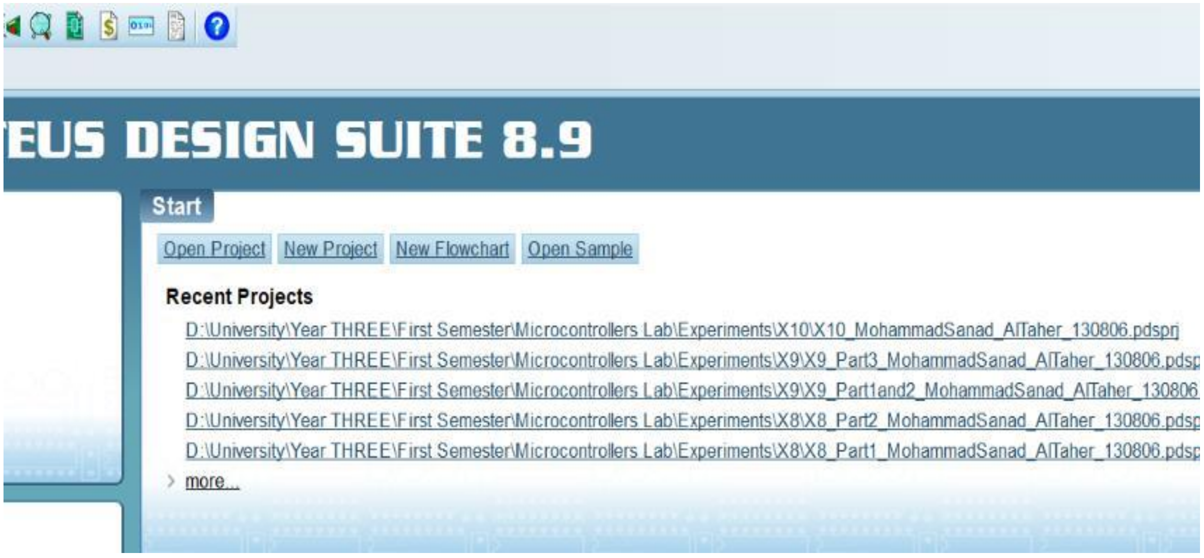
Manual Update Check. ☐ Ignore beta version updates

New in Version 8.12

Multi Board Support Diff Pair Pass Through	Non-Functional Pads File Auto Save (Updated)	SnapEDA Library Integration

"Getting Started" & "Help": Some useful guides can be found here, but they tend to be wordy and filled with jargon. It is advised to avoid them.
"Start" \ "Recent Projects": Here you will find the projects that you opened last. This section will be empty if you just installed Proteus.
"News": This section will contain news about your current license, recent versions of the software, and **links to basic video tutorials** that you might find helpful.
To create a new project, follow the steps below:

- Step 1: Click on "New Project" under the "Start" header in the home page:

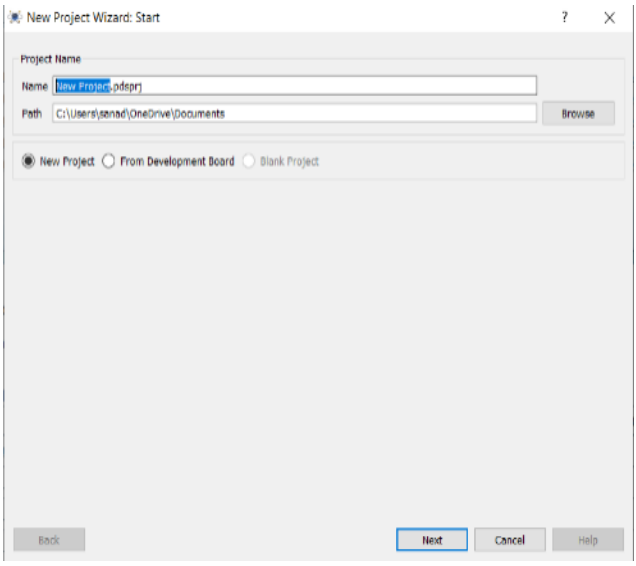

EUS DESIGN SUITE 8.9

Start

Open Project | New Project | New Flowchart | Open Sample

Recent Projects
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X10\X10_MohammadSanad_AlTaher_130806.pdsprj
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X9\X9_Part3_MohammadSanad_AlTaher_130806.pdsp
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X9\X9_Part1and2_MohammadSanad_AlTaher_130806
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X8\X8_Part2_MohammadSanad_AlTaher_130806.pdsp
- D:\University\Year THREE\First Semester\Microcontrollers Lab\Experiments\X8\X8_Part1_MohammadSanad_AlTaher_130806.pdsp
- more...

The "New Project Wizard" window will now open
- Step 2: Choose a name and file-path for your project:

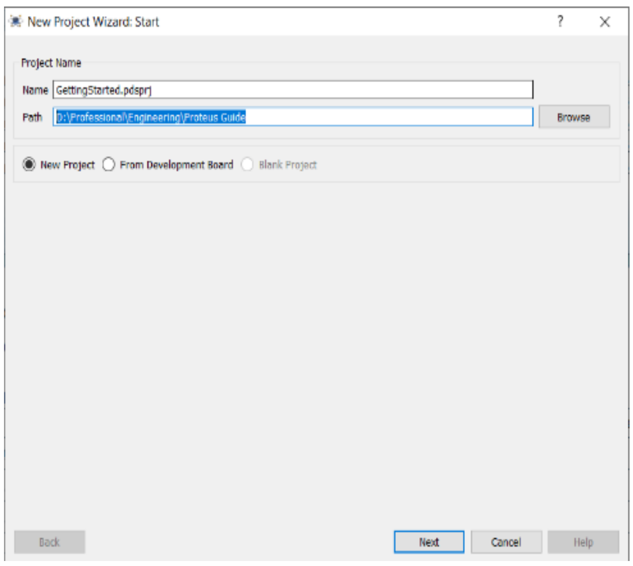

New Project Wizard: Start

Project Name

Name: New Project.pdsprj

Path: C:\Users\senad\OneDrive\Documents

[✓ New Project] From Development Board Blank Project


New Project Wizard: Start

Project Name

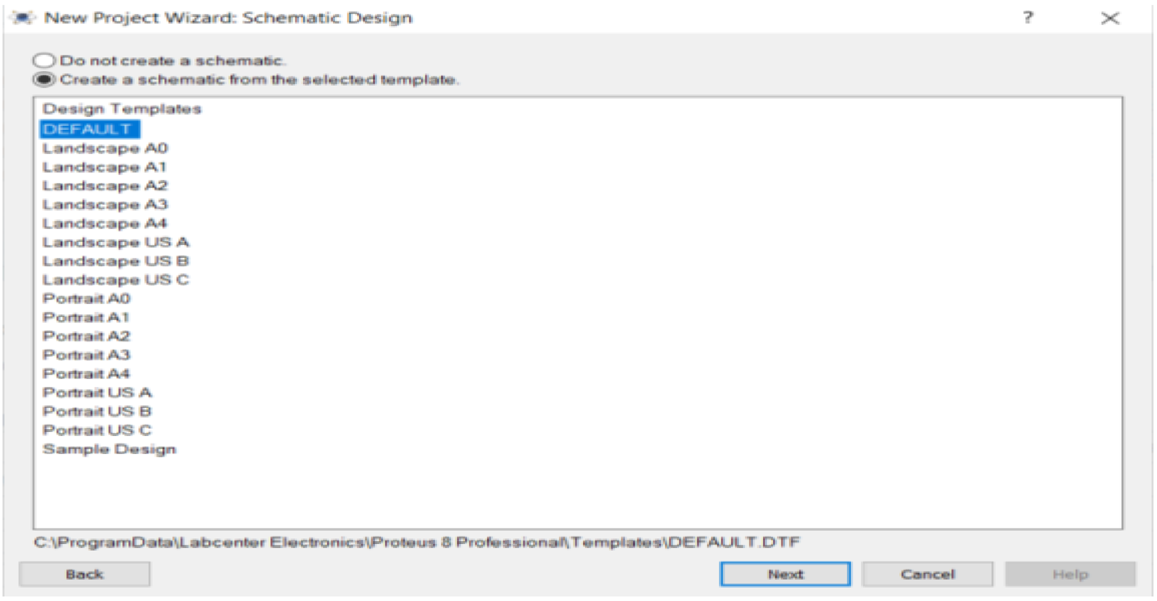
Name: GettingStarted.pdsprj

Path: D:\Professional\Engineering\Proteus Guide

[✓ New Project] From Development Board Blank Project

Then click “Next”.

- Step 3: Select the Schematic Design:


New Project Wizard: Schematic Design

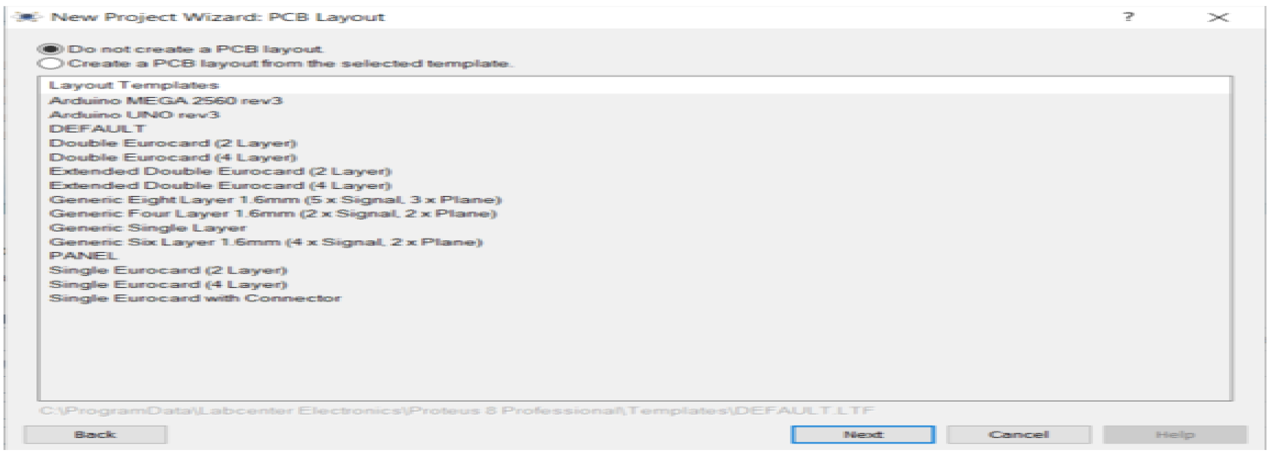
Do not create a schematic.
[✓ Create a schematic from the selected template.]

Design Templates
DEFAULT
Landscape A0
Landscape A1
Landscape A2
Landscape A3
Landscape A4
Landscape US A
Landscape US B
Landscape US C
Portrait A0
Portrait A1
Portrait A2
Portrait A3
Portrait A4
Portrait US A
Portrait US B
Portrait US C
Sample Design

C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\Templates\DEFAULT.DTF

Unless certain dimensions are required (for printing or otherwise), “DEFAULT” will suffice. Click “Next”.

- Step 3: Select the PCB Layout:


New Project Wizard: PCB Layout

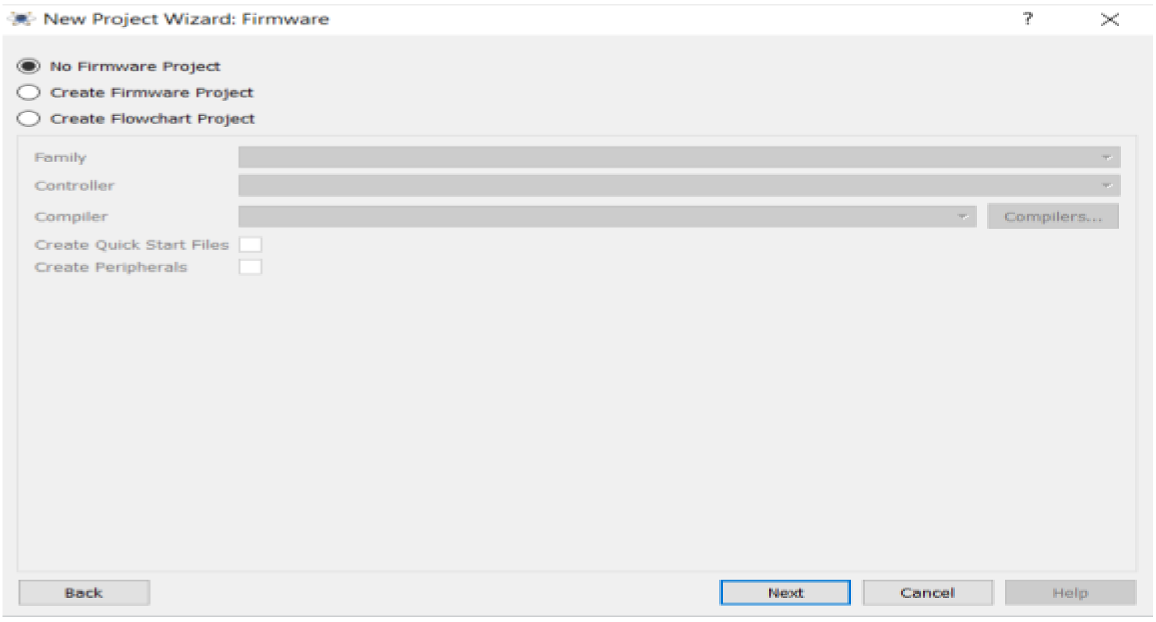
[✓ Do not create a PCB layout.]
Create a PCB layout from the selected template.

Layout Templates
Arduino MEGA 2560 rev3
Arduino UNO rev3
DEFAULT
Double Eurocard (2 Layer)
Double Eurocard (4 Layer)
Extended Double Eurocard (2 Layer)
Extended Double Eurocard (4 Layer)
Generic Eight Layer 1.6mm (5 x Signal, 3 x Plane)
Generic Four Layer 1.6mm (2 x Signal, 2 x Plane)
Generic Single Layer
Generic Six Layer 1.6mm (4 x Signal, 2 x Plane)
PANEL
Single Eurocard (2 Layer)
Single Eurocard (4 Layer)
Single Eurocard with Connector

C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\Templates\DEFAULT.LTF

PCB design will not be included in this guide. Select “Do not create a PCB layout” and click “Next

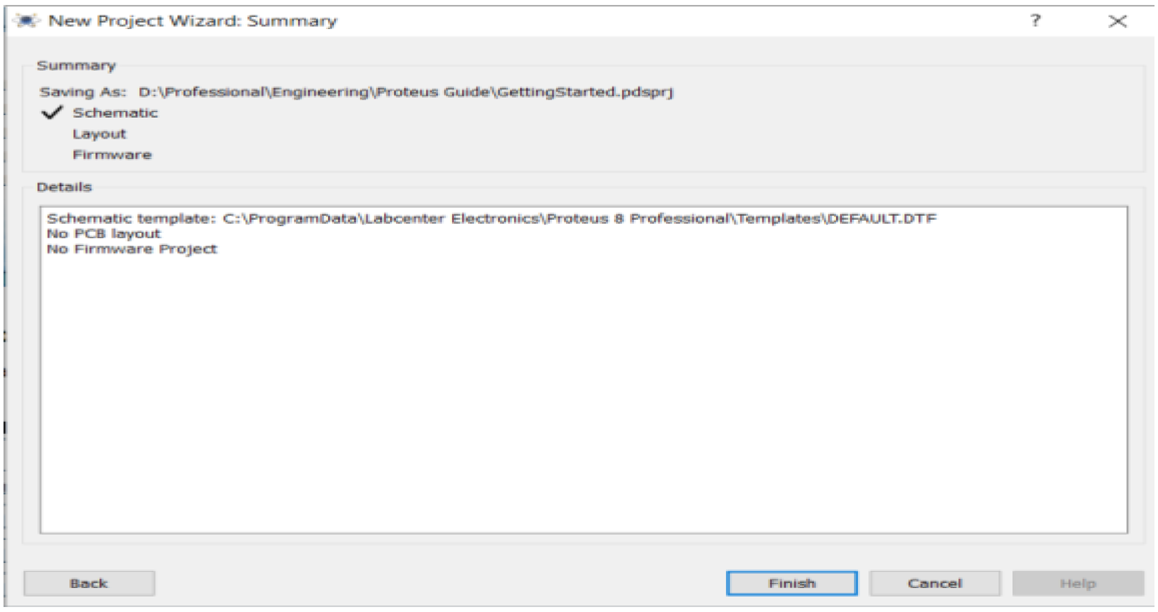
- Step 4: Select Firmware:



● No Firmware Project
○ Create Firmware Project
○ Create Flowchart Project

Editing source code is beyond the scope of this guide, select “No Firmware Project” and click “Next”.

- **Summary page:**



Summary

Saving As: D:\Professional\Engineering\Proteus Guide\GettingStarted.pdsprj

✓ Schematic
Layout
Firmware

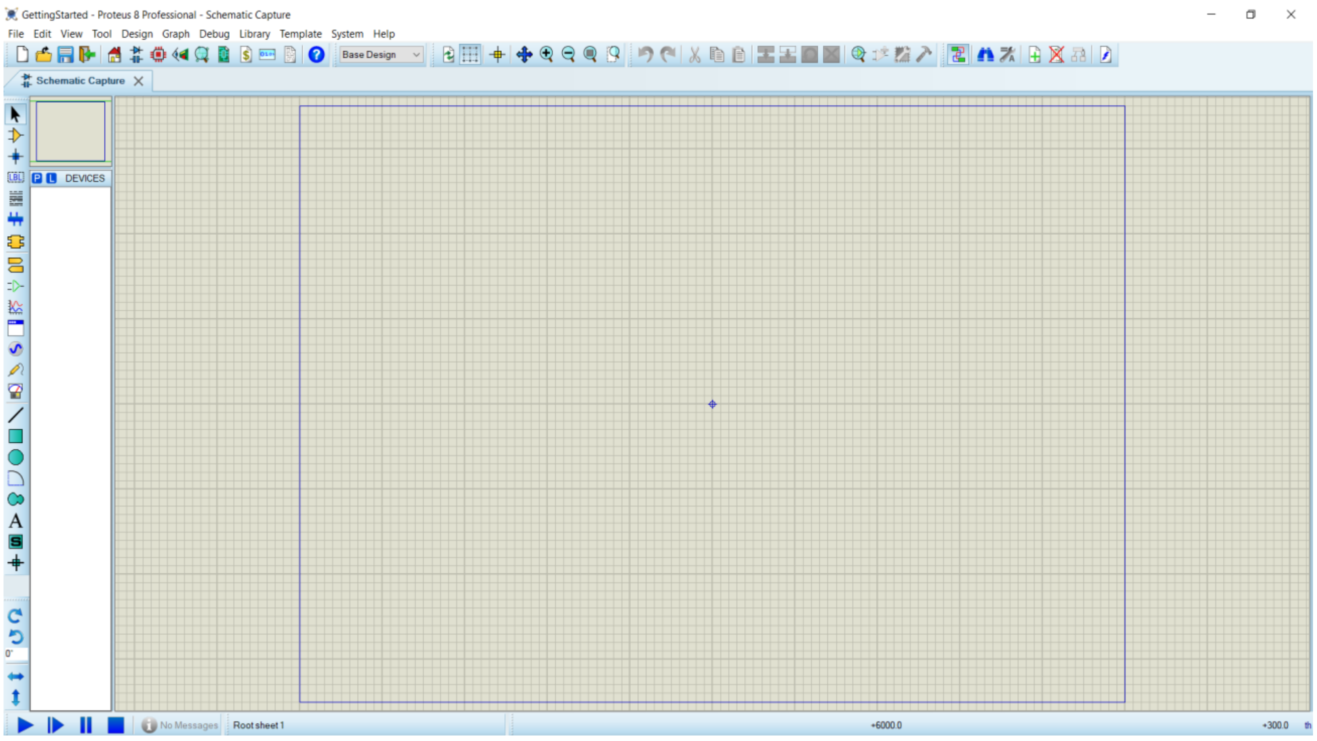
Details

Schematic template: C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\Templates\DEFAULT.DTF
No PCB layout
No Firmware Project

You should now see a summary of all your previous selections. If any mistakes were made, you can click “Back” and correct them.

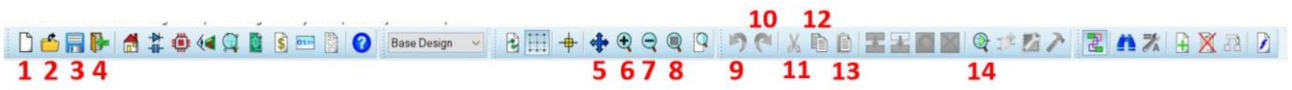
- Click “Finish” to open the Schematic Capture module and begin working on your project.

2. Interface:


GettingStarted - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
DEVICES
No Messages
Root sheet 1
+6000.0
+300.0

The Schematic Capture interface is split into four main parts, the Menu and Module bars on top, the Sidebar on the left-hand side, the Root sheet in the center, and the Simulation Control bar at the bottom.
Our circuits will be shown on the Root sheet, whereas most of our work will be done using the Sidebar.

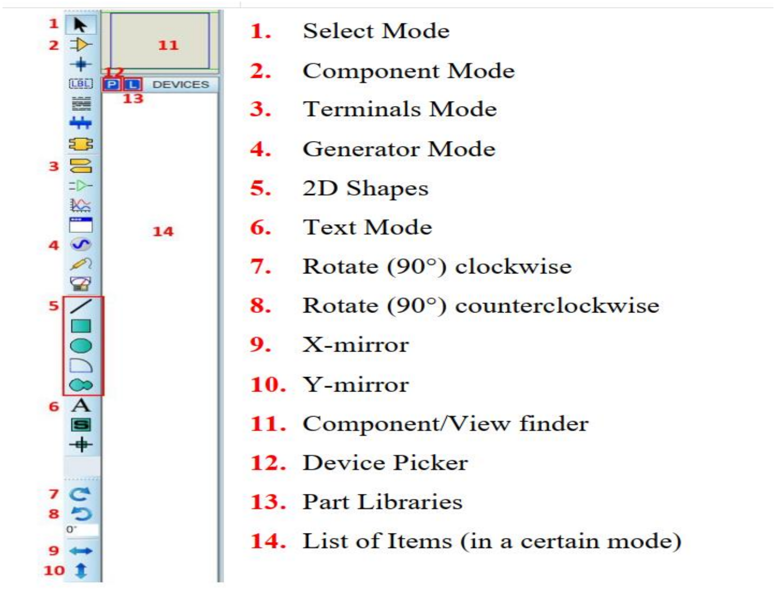
Module Bar:



1. New Project
2. Open Project
3. Save Project
4. Close Project
5. Center at Cursor
6. Zoom in
7. Zoom out
8. Zoom to View Sheet
9. Undo
10. Redo
11. Cut
12. Copy
13. Paste
14. Pick Parts

Note: letter "M" before a number will indicate a button on the Module bar. Thus, M12 refers to the "Copy" button on the Module bar.

Sidebar:



1. Select Mode
2. Component Mode
3. Terminals Mode
4. Generator Mode
5. 2D Shapes
6. Text Mode
7. Rotate (90°) clockwise
8. Rotate (90°) counterclockwise
9. X-mirror
10. Y-mirror
11. Component/View finder
12. Device Picker
13. Part Libraries
14. List of Items (in a certain mode)

Note: letter "S" before a number will indicate a button on the Sidebar. Thus, S12 refers to the "Device Picker" button on the Sidebar.

3. Collecting Parts:

In this part, we will prepare all the parts needed for our experiment. Experiment two will be used to demonstrate the steps

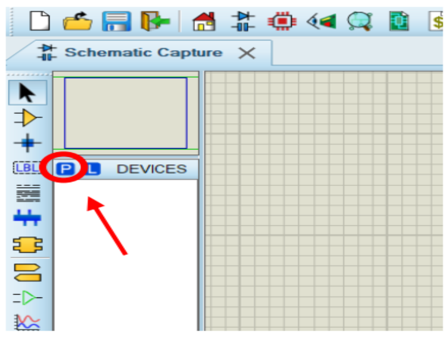
0. Identify the needed parts:
   Usually, all needed chips for a certain experiment are listed in its document.
   The chips needed for experiment two are in the excerpt below:

"

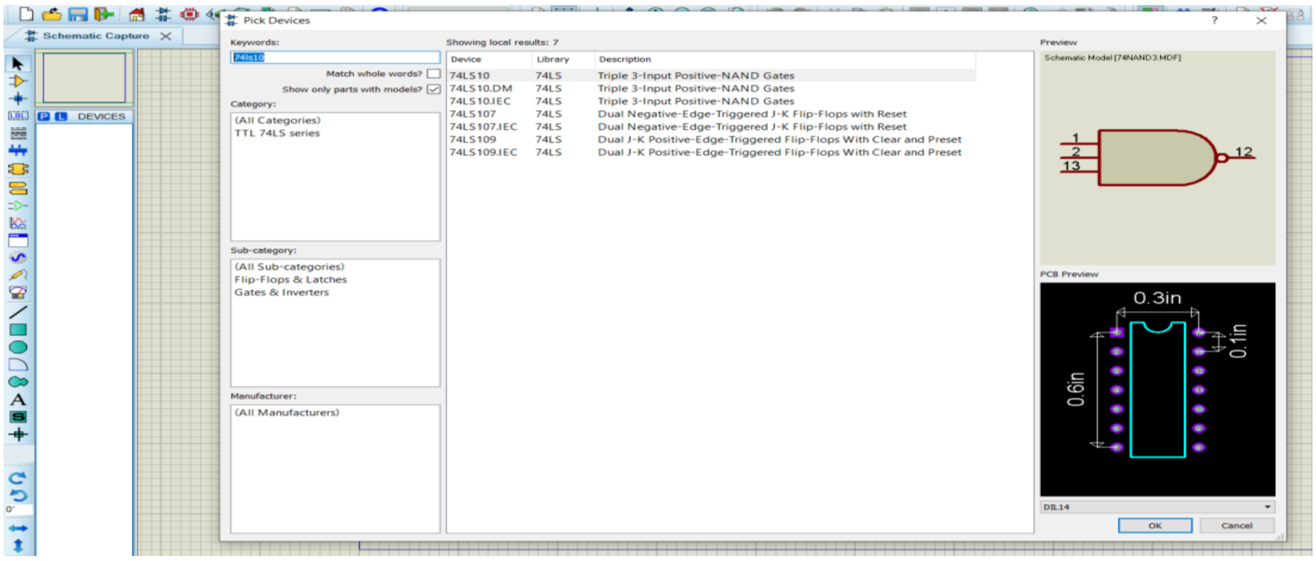
7410 TRIPLE 3-INPUT NAND GATE, QTY = 2 Chips
7427 TRIPLE 3-INPUT NOR GATE, QTY = 2 Chips
7404 HEX INVERTER, QTY = 1 needed in order to make the variables and their complements available.
7400 QUADRUPLE 2-INPUT NAND GATE, QTY = 1 Chip
7402 QUADRABLE 2-INPUT NOR GATE, QTY = 1 Chip
74283 4-BIT BIANRY ADDER

"

- Step 1: Open the Part Picker (S12 or M14):



- Step 2: After the "Pick Devices" window has appeared, begin writing keywords for the part you are looking for:


Pick Devices

Keywords: 74ls10

Match whole words? ☐

Show only parts with models? ☑

Category:
(All Categories)
TTL 74LS series

Sub-category:
(All Sub-categories)
Flip-Flops & Latches
Gates & Inverters

Manufacturer:
(All Manufacturers)

Showing local results: 7

Device	Library	Description
74LS10	74LS	Triple 3-Input Positive-NAND Gates
74LS10.DM	74LS	Triple 3-Input Positive-NAND Gates
74LS10.IEC	74LS	Triple 3-Input Positive-NAND Gates
74LS107	74LS	Dual Negative-Edge-Triggered J-K Flip-Flops with Reset
74LS107.IEC	74LS	Dual Negative-Edge-Triggered J-K Flip-Flops with Reset
74LS109	74LS	Dual J-K Positive-Edge-Triggered Flip-Flops With Clear and Preset
74LS109.IEC	74LS	Dual J-K Positive-Edge-Triggered Flip-Flops With Clear and Preset

Preview

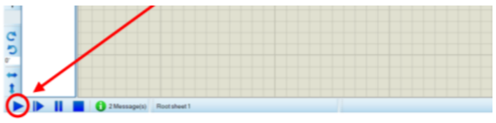
Schematic Model [74NAND3.MDF]

PCB Preview

DIL14

- Step 3: Click "OK" to add the part to your list of components.
- Repeat step 2 for all other components while making sure that all components have "Simulator Model(s)". If "No Simulator Model" is **not** written, then it is assumed that there is a simulator model for the selected component.
- Step 4: Now we add other required components such as: Switches, LEDs, Clock, etc. Inputs: The two most common input components are "LOGICTOGGLE" and "SW-SPDT" (Single Pole Double Throw) and can be added from "Device Picker". There are no practical differences between them.

5. **Connecting a Circuit:**
   Once we have all the needed parts listed (in S**14**) in the Component Mode (S**2**), we are ready to begin constructing a circuit. In this part, a half-adder will be constructed using NOT gates and NOR gates (from experiment two)
6. **Running the Simulator:**
   Perhaps the easiest part of this guide is running a simulation. Once all components are connected, simply click the "Play" button at the bottom-left corner of the page.



7. Introduction to arduino

7.1. What is Arduino

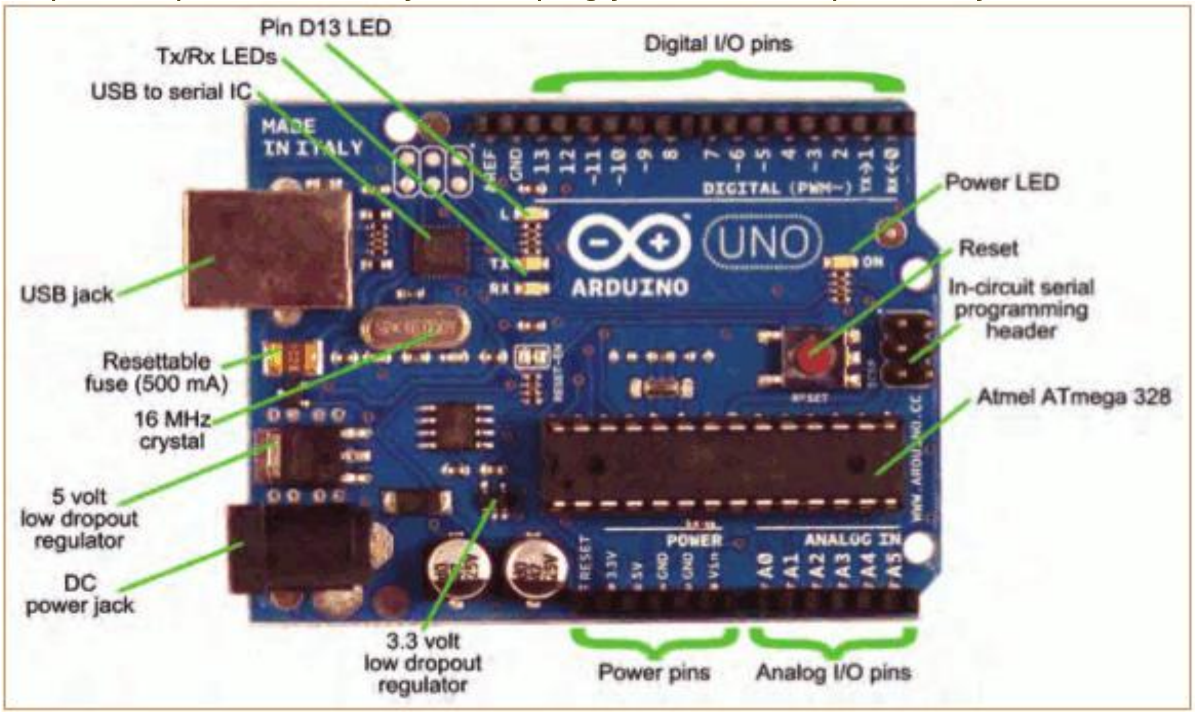
- Arduino is nothing but a simple microcontroller board which is normally used in engineering projects where there's a need to automate something.
- You can interface sensors with this board, can drive motors with this board, can plug switches in it etc.
- In old ages (not old enough 😛), people used simple switches for turning ON a bulb so like you click the switch and the bulb is ON, it was quite a simple circuit, after that relays are invented and then engineers used 555 timer circuits in order to turn ON lights on some specific time. But the 555 timer circuits are quite big

in size, so finally engineers discovered Microcontrollers in which there are simple OUTPUT and INPUT pins, so now if you want to turn on light at some certain time then you just simply plug the blub on output pin of microcontroller and then do some programming and add a timer to automatically turn on the bulb.

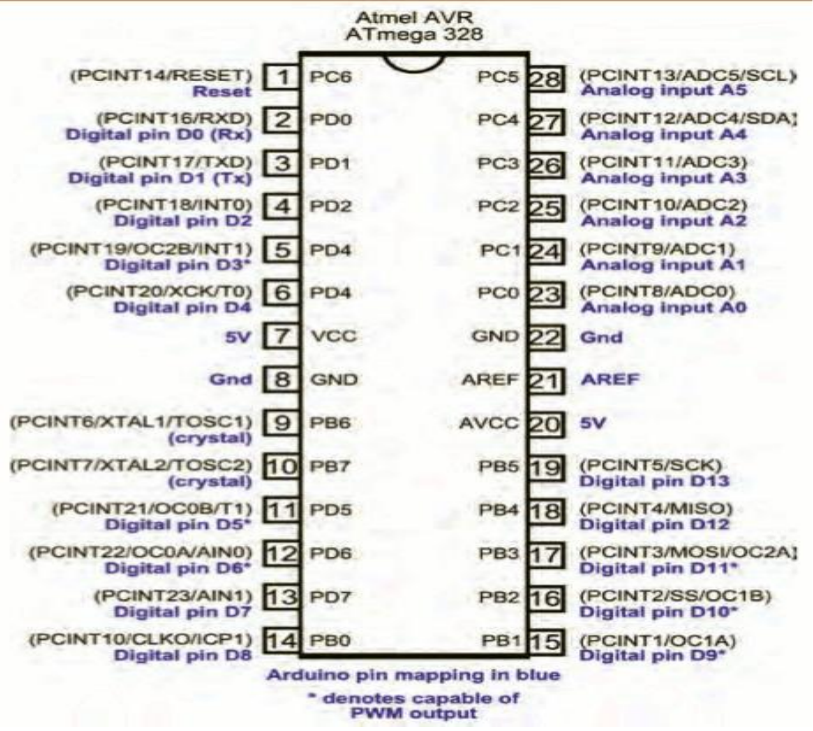
- So, the benefit of microcontroller is the circuit is quite simple and small in size.Moreover, its flexible, suppose you want to change the time of turning ON bulb then what you need to do is simply change the coding and it will be changed, but in 555 timer circuits you need to change the components in order to do so.
- Now, we know the use of microcontroller and also their benefit but thing is what is Arduino ??? In microcontrollers like PIC or Atmel, there’s a small drawback.
- Suppose you want to work on PIC then you have to first design its basic circuit also need to design a power circuit to supply power to it and after that in order to upload the code in it, you have to buy a programmer/ burner as well. So, first of all you need to write the code for PIC Microcontroller and after that you need to upload code in it using a programmer and then plce PIC microcontroller back into the circuit and test, which is quite lengthy plus also got hectic when you are working on some project because you have to test code again and again.
- By the way, now advance programmers like PICkit2 and PICkit3 can be plugged on board but still you have to design the basic circuit so coming to bottom line, in order to do project with PIC or Atmel microcontroller you have to do soldering etc.
- But that’s not the case with Arduino Board, Arduino has built in programmer and the basic circuit in it. So what you need to do is simply plug in Arduino with your computer via usb cable, get its software and start uploading code and also start testing.
- So, you don’t need to plug unplug or do anything, simply upload the code and test. Moreover, it also has some very efficient tools using which you can test your output as well quite easily. Arduino board also has the pins on which you can simply plug your devices and can turn them ON or OFF. So, hats off to Arduino team for providing us a simple board which has everything on it.

7.2. Types of Arduino Boards

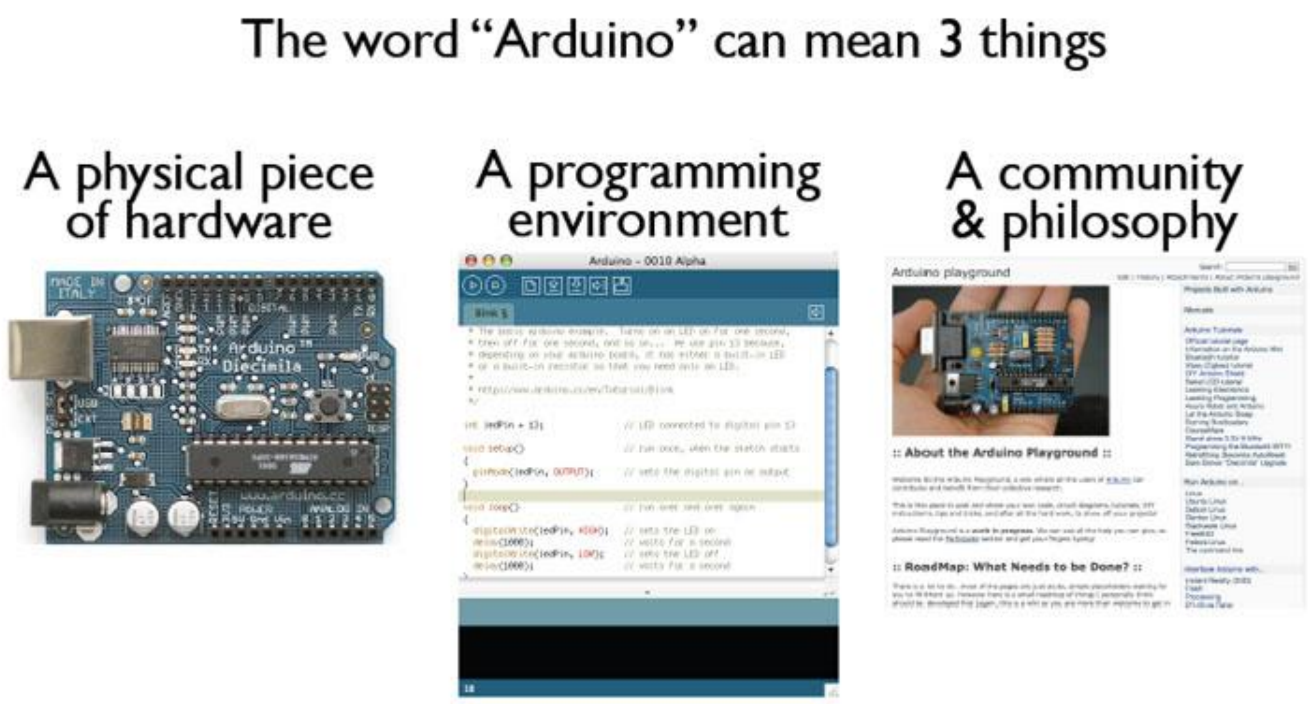
- There's a long range of Arduino boards available online, the basic Arduino board is named as Arduino UNO which is most widely used in projects.
- Arduino UNO has total 13 digital pins and 6 analog pins which are used for connecting sensors with them.
- Suppose you have a project in which you want to interface 30 sensors, then what you need to do ?? Now you need to buy another Arduino board named as Arduino Mega 2560. This board has around 70 pins on it which can be used as output or input and hence you can plug your sensors quite easily.



7.3. The Arduino Microcontroller: Atmel ARV Atmega 328


Atmel AVR ATmega 328
(PCINT14/RESET) Reset 1 PC6
(PCINT16/RXD) Digital pin D0 (Rx) 2 PD0
(PCINT17/TXD) Digital pin D1 (Tx) 3 PD1
(PCINT18/INT0) Digital pin D2 4 PD2
(PCINT19/OC2B/INT1) Digital pin D3* 5 PD4
(PCINT20/XCK/T0) Digital pin D4 6 PD4
5V 7 VCC
Gnd 8 GND
(PCINT6/XTAL1/TOSC1) (crystal) 9 PB6
(PCINT7/XTAL2/TOSC2) (crystal) 10 PB7
(PCINT21/OC0B/T1) Digital pin D5* 11 PD5
(PCINT22/OC0A/AIN0) Digital pin D6* 12 PD6
(PCINT23/AIN1) Digital pin D7 13 PD7
(PCINT10/CLKO/ICP1) Digital pin D8 14 PB0
PC5 28 (PCINT13/ADC5/SCL) Analog input A5
PC4 27 (PCINT12/ADC4/SDA) Analog input A4
PC3 26 (PCINT11/ADC3) Analog input A3
PC2 25 (PCINT10/ADC2) Analog input A2
PC1 24 (PCINT9/ADC1) Analog input A1
PC0 23 (PCINT8/ADC0) Analog input A0
GND 22 Gnd
AREF 21 AREF
AVCC 20 5V
PB5 19 (PCINT5/SCK) Digital pin D13
PB4 18 (PCINT4/MISO) Digital pin D12
PB3 17 (PCINT3/MOSI/OC2A) Digital pin D11*
PB2 16 (PCINT2/SS/OC1B) Digital pin D10*
PB1 15 (PCINT1/OC1A) Digital pin D9*
Arduino pin mapping in blue
* denotes capable of PWM output

7.4. What is the Arduino


The word "Arduino" can mean 3 things
A physical piece of hardware
A programming environment
A community & philosophy
Arduino - 0010 Alpha
Blink §
Arduino playground
:: About the Arduino Playground ::
:: RoadMap: What Needs to be Done? ::

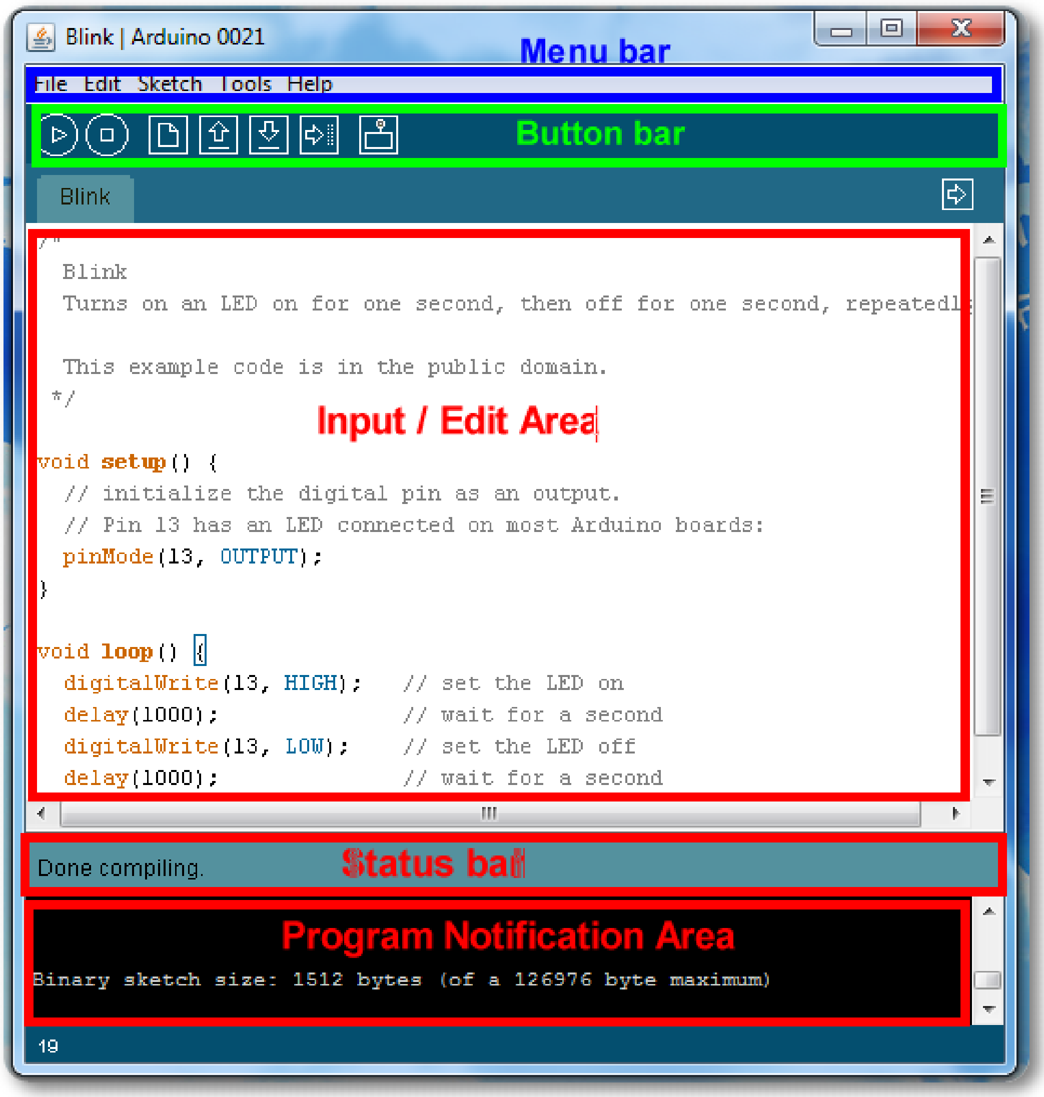
```
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13;                // LED connected to digital pin 13

void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

7.5. How to use Arduino

- Now I think you have got the basic idea of what is Arduino ? and why is it so popular ? So now lets have a look at how to use Arduino.
- When you order for your Arduino board, you will get a package similar to the image below:
-

- Along with this box, you will also have the USB cable, now take your Arduino board out of this box and plug the cable in it, connect the Arduino with your computer and you are ready to start working on it.
- In order to connect Arduino with your computer you have to install the Arduino drivers in Windows.
- That's all for today, hope I have conveyed some knowledge and you now know the basics of Arduino i.e. what is Arduino ? and why to use Arduino.

7.6. Arduino IDE

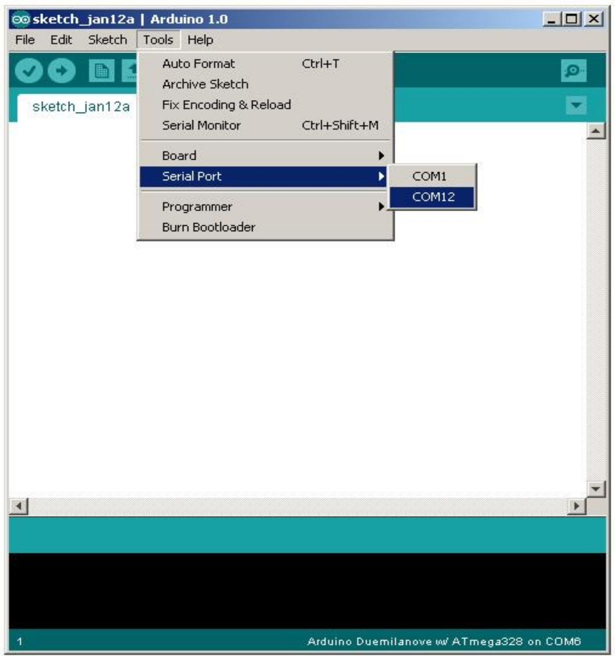


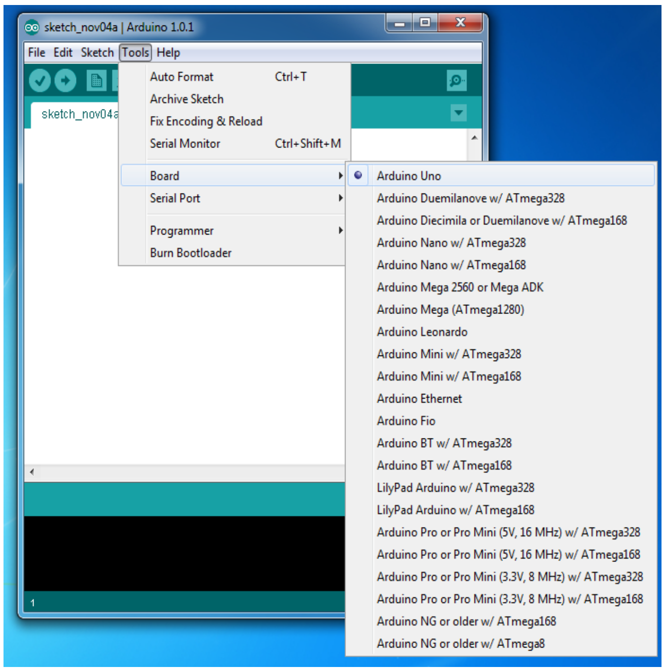
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedl

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
```

```
Done compiling.
```

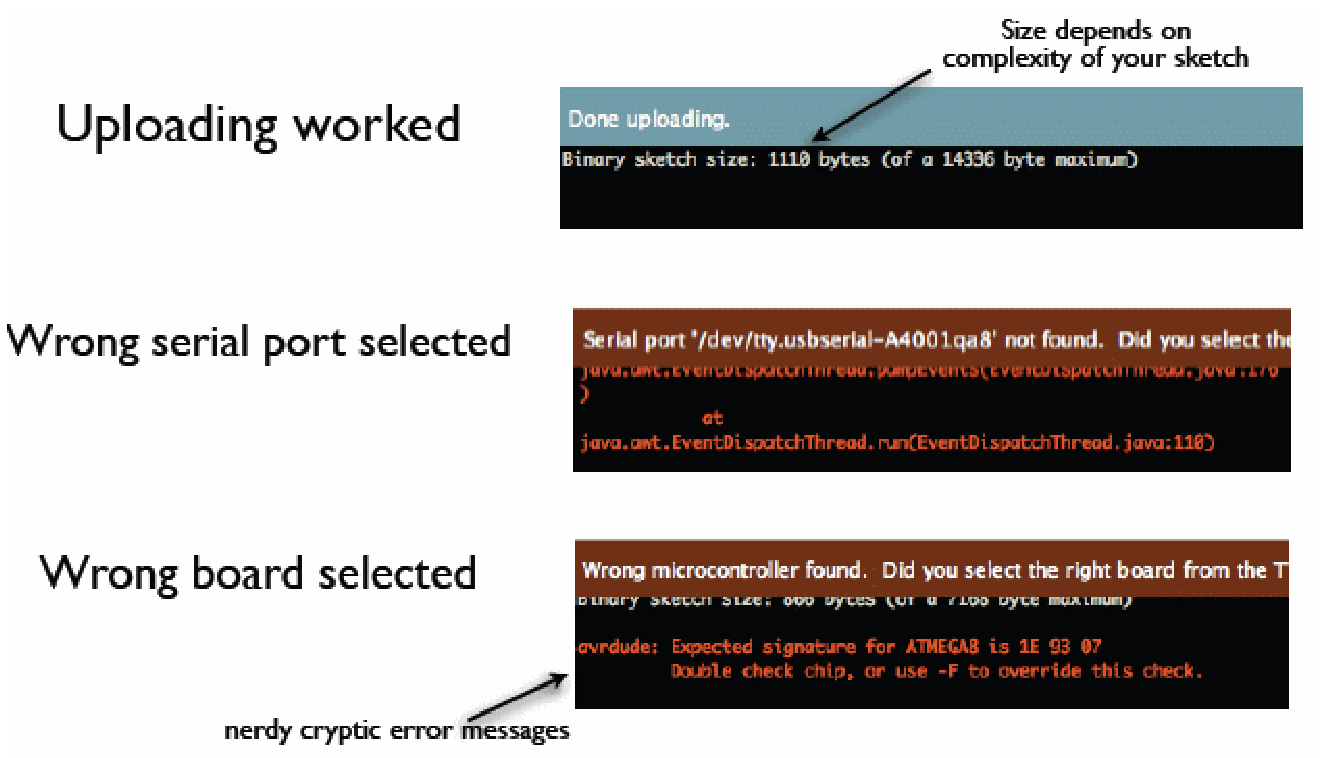
```
Binary sketch size: 1512 bytes (of a 126976 byte maximum)
```

- Select Serial Port and Board


sketch_jan12a | Arduino 1.0
File Edit Sketch Tools Help
sketch_jan12a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
COM1
COM12
1
Arduino Duemilanove w/ ATmega328 on COM6


sketch_nov04a | Arduino 1.0.1
File Edit Sketch Tools Help
sketch_nov04a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
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 Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
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
1

- Status Messages



Uploading worked

Size depends on complexity of your sketch

```
Done uploading.
Binary sketch size: 1110 bytes (of a 14336 byte maximum)
```

Wrong serial port selected

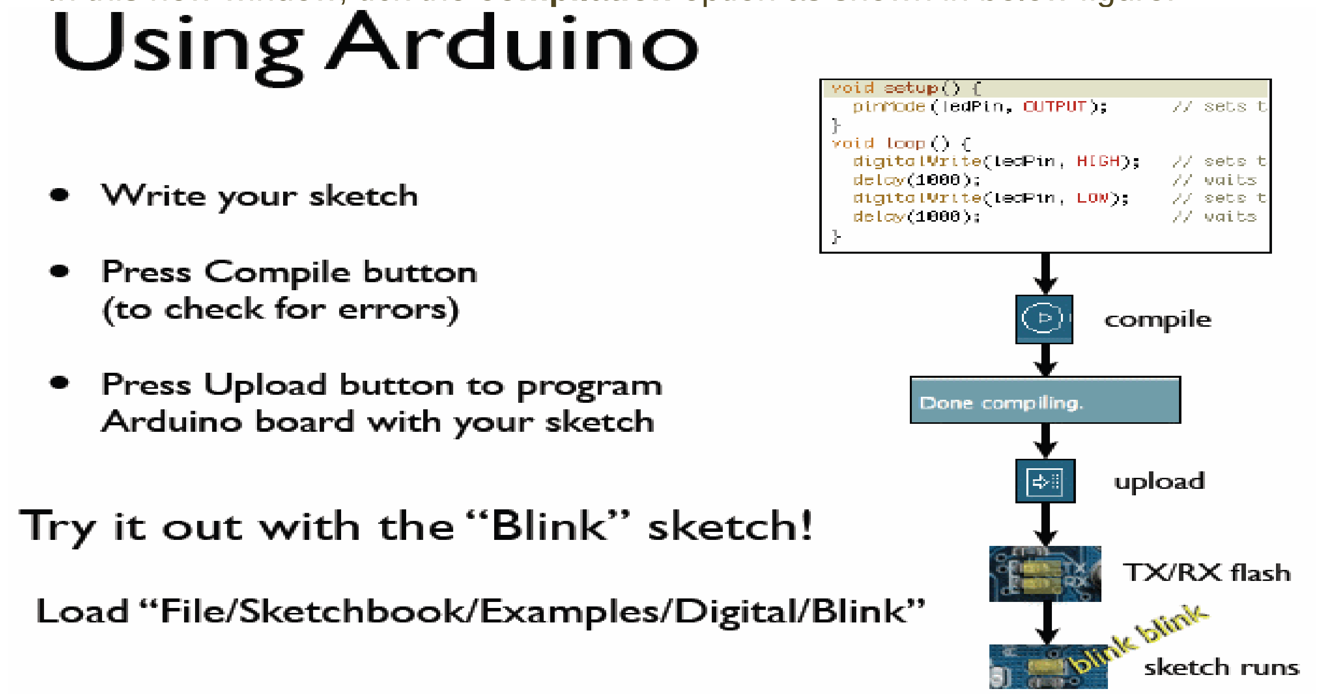
```
Serial port '/dev/tty.usbserial-A4001qa8' not found.  Did you select the
)
            at
java.awt.EventDispatchThread.run(EventDispatchThread.java:110)
```

Wrong board selected

```
Wrong microcontroller found.  Did you select the right board from the T

avrdude: Expected signature for ATMEGA8 is 1E 93 07
         Double check chip, or use -F to override this check.
```

nerdy cryptic error messages

- In the above figure, I have used the simple blink example and I am gonna generate its hex file.
- Now when you click on the Preferences, a new window will pop up.
- In this new window, tick the **compilation** option as shown in below figure:



Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

Try it out with the "Blink" sketch!

Load "File/Sketchbook/Examples/Digital/Blink"

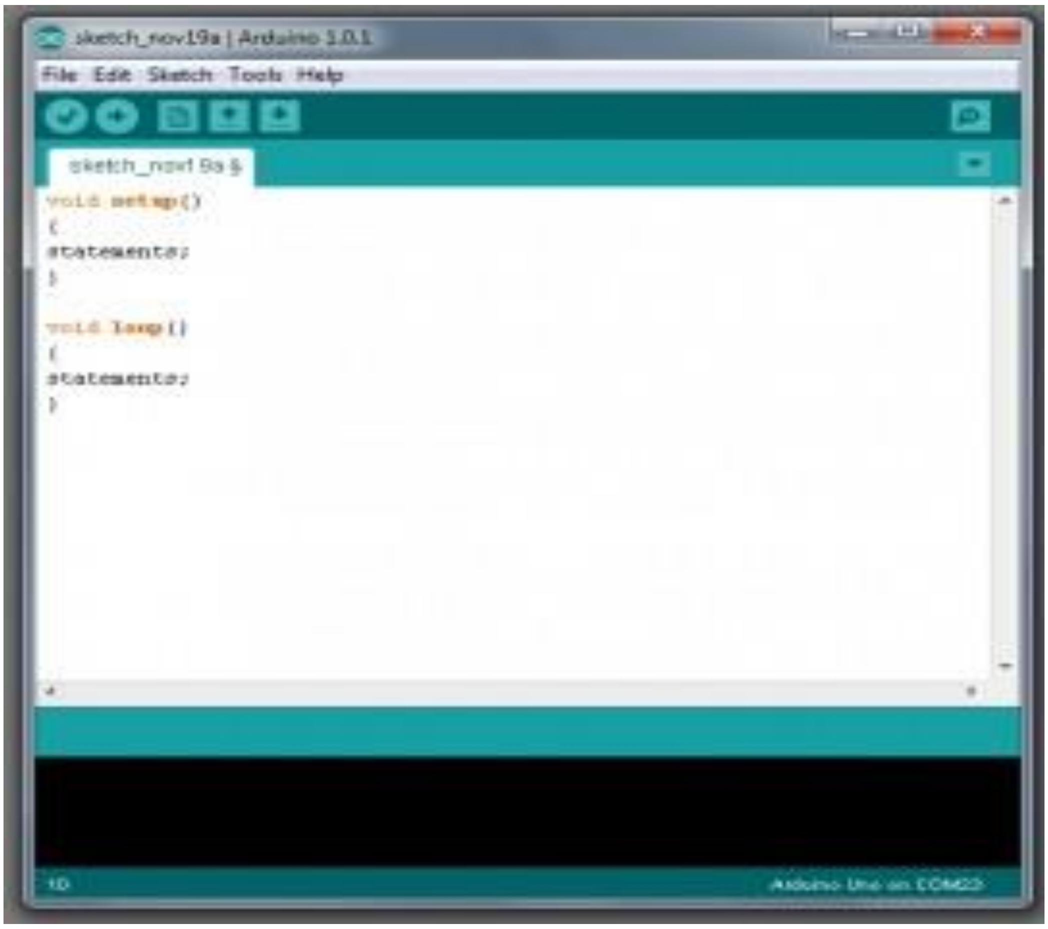
```
void setup() {
  pinMode(ledPin, OUTPUT);        // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);     // sets t
  delay(1000);                    // waits
  digitalWrite(ledPin, LOW);      // sets t
  delay(1000);                    // waits
}
```

```
Done compiling.
```

7.7. A Little Bit About Programming

- Code is case sensitive
- Statements are commands and must end with a semi-colon

- Comments follow a // or begin with /* and end with */



```
void setup()
{
statements;
}

void loop()
{
statements;
}
```

7.8. Terminology

"*sketch*" – a program you write to run on an Arduino board

"*pin*" – an input or output connected to something.

e.g. output to an LED, input from a knob.

"*digital*" – value is either HIGH or LOW.

(aka on/off, one/zero) e.g. switch state

"*analog*" – value ranges, usually from 0-255.

e.g. LED brightness, motor speed, etc.

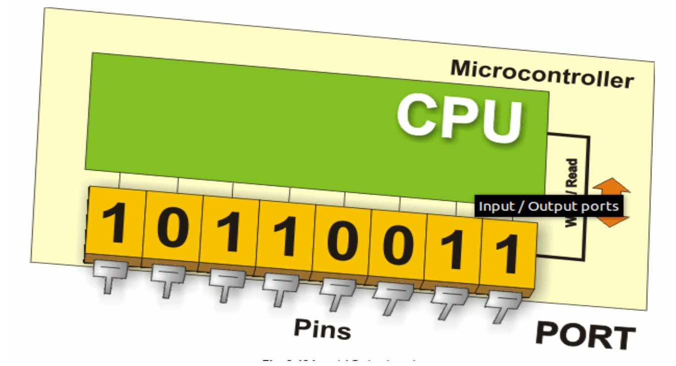
7.9. Variables

- **boolean** (8 bit) – Chứa logic đơn giản là true/false
- **byte** (8 bit) – Chứa các số không dấu (chỉ chứa số dương không chứa số âm) từ 0-255
- **char** (8 bit) – Số có dấu từ -128 đến 127. IDE sẽ mặc định dùng biến này để chỉ ký tự tương ứng trong bảng mã ASCII khi dùng các lệnh, hàm liên quan đến ký tự.
- **unsigned char** (8 bit) – Tương tự như Byte. Thông thường bạn nên sử dụng Byte để tránh nhầm lẫn
- **word** (16 bit) – Số không dấu từ 0-65535

- **unsigned int** (16 bit)- Tương tự như Word. Bạn nên sử dụng Word để tránh nhầm lẫn. Tuy nhiên, một số bạn vẫn tiện tay dùng.
- **int** (16 bit) – Số có dấu từ -32768 đến 32767. Đây là biến thông dụng
- **unsigned long** (32 bit) – Số không dấu từ 0-4,294,967,295. thường sử dụng để lưu giá trị trong hàm millis(), đây là hàm trả về giá trị milli giây kể từ khi Arduino bắt đầu hoạt động.
- **long** (32 bit) – Số có dấu từ -2,147,483,648 to 2,147,483,647
- **float** (32 bit) – Số có dấu từ -3.4028235E38 to 3.4028235E38. Dấu chấm động thường ít được sự dụng trong Arduino vì trình biên dịch rất khó khăn để chuyển đổi thực thi các lệnh liên quan. Nếu thật sự không quá cần thiết, bạn không nên sử dụng đến nó.

- **String –** Biến chứa một chuỗi các ký tự (char).
- **Void –** Chỉ sử dụng để khai báo hàm khi không cần phải trả về bất cứ giá trị nào.

7.10. Digital I/0



- pinMode(*pin*, *mode*): Sets pin to either INPUT or OUTPUT
- digitalRead(*pin*): Reads HIGH or LOW from a pin
- digitalWrite(*pin*, *value*): Writes HIGH or LOW to a pin
- Electronic stuff : Output pins can provide 40 mA of current, Writing HIGH to an input pin installs a 20KΩ pullup

7.11. Arduino Timing

- delay(*ms*): Pauses for a few milliseconds
- delayMicroseconds(*us*): Pauses for a few microseconds

7.12. Digital? Analog?

- Digital has two values: **on** and **off**
- Analog has many (infinite) values
- Computers don't really do analog, they ***quantize***
- Remember the 6 analog input pins---here's how they work

7.13. Important functions

- Serial.println(value);
  - Prints the value to the Serial Monitor on your computer
- pinMode(pin, mode);
  - Configures a digital pin to read (input) or write (output) a digital value
- digitalRead(pin);
  - Reads a digital value (HIGH or LOW) on a pin set for input
- digitalWrite(pin, value);
  - Writes the digital value (HIGH or LOW) to a pin set for output

8. Simple Steps to of Arduino code simulator using Proteus

8.1. Write Arduino Codes in Arduino Software IDE

8.2. *Configure Arduino Prote*us Library Files

We are turning the Proteus as a **Simulator for Arduino** with the help of **Arduino Preoteus library files.**[Download Arduino Protius library from here](), then copy and paste the 2 library files (**ARDUINO.IDX, ARDUINO.LIB**) in to LIBRARY folder of Proteus.
LIBRARY location of Proteus may be different according to your Proteus version.

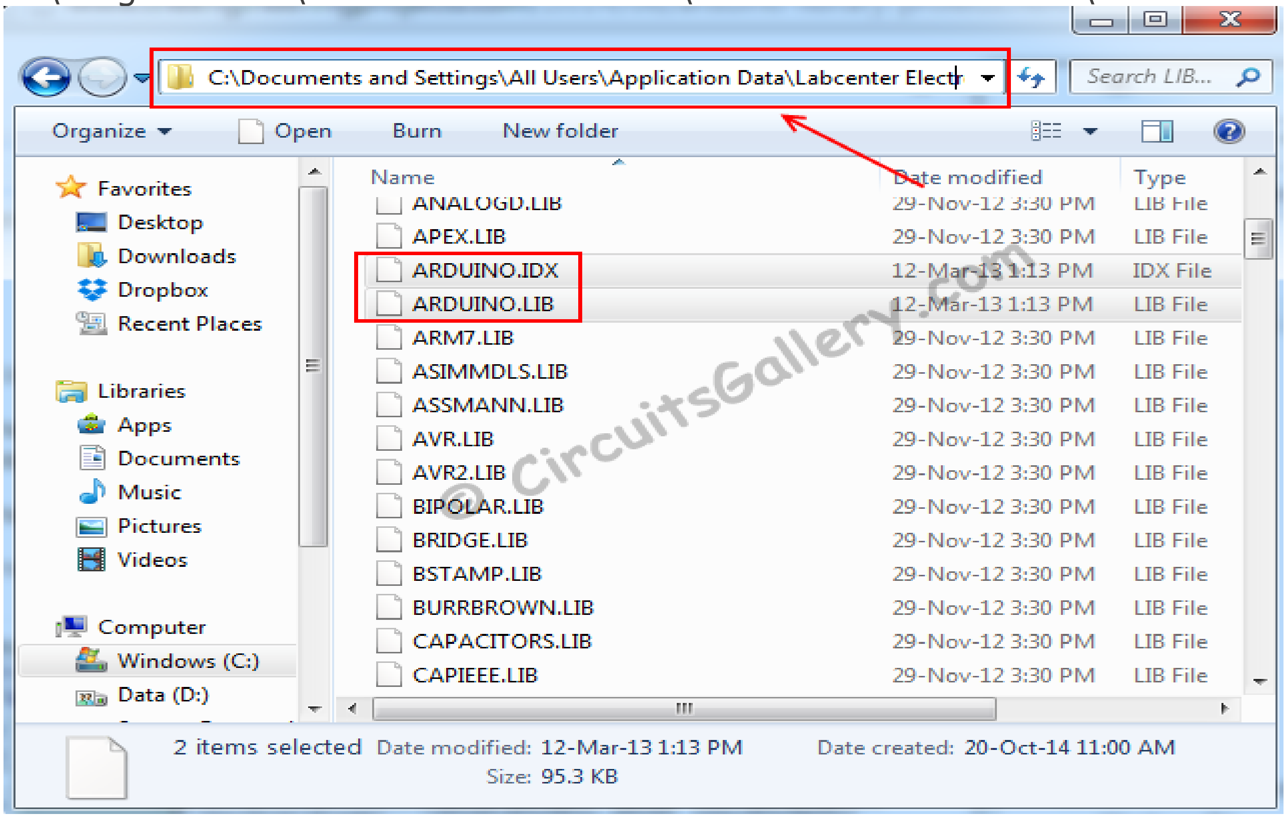
Proteus 8 Users

C:\Documents and Settings\All Users\Application Data\Labcenter Electronics\Proteus 8 Professional\LIBRARY

Proteus 7 Users

C:\Labcenter Electronics\Proteus 7 Professional\LIBRARY

Notes: if you use windows-10, path of following is used:
C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\LIBRARY



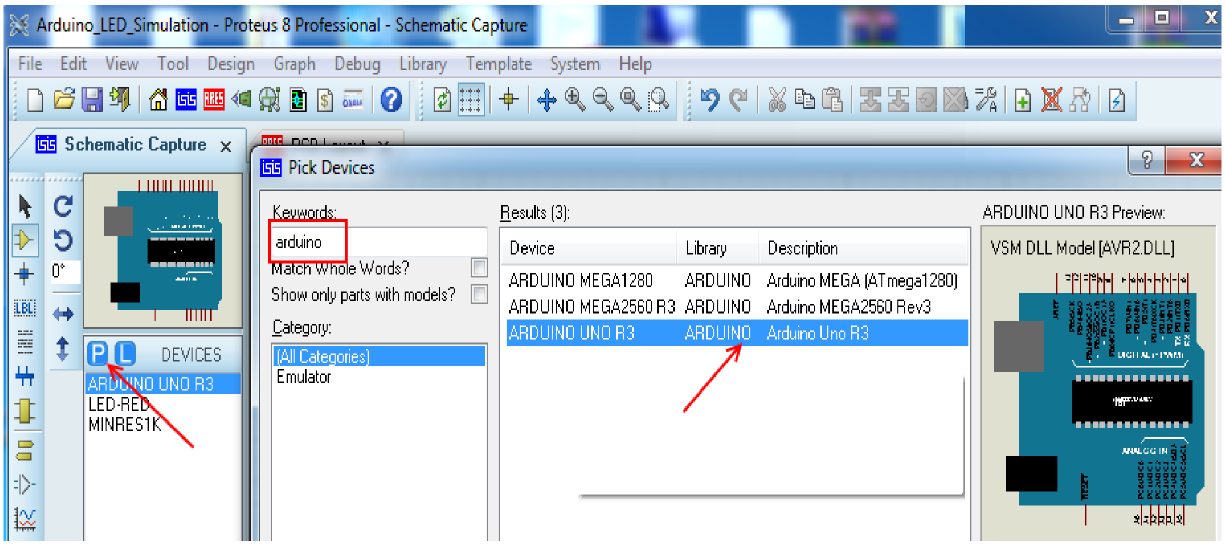
C:\Documents and Settings\All Users\Application Data\Labcenter Electr

Name	Date modified	Type
ANALOGD.LIB	29-Nov-12 3:30 PM	LIB File
APEX.LIB	29-Nov-12 3:30 PM	LIB File
ARDUINO.IDX	12-Mar-13 1:13 PM	IDX File
ARDUINO.LIB	12-Mar-13 1:13 PM	LIB File
ARM7.LIB	29-Nov-12 3:30 PM	LIB File
ASIMMDLS.LIB	29-Nov-12 3:30 PM	LIB File
ASSMANN.LIB	29-Nov-12 3:30 PM	LIB File
AVR.LIB	29-Nov-12 3:30 PM	LIB File
AVR2.LIB	29-Nov-12 3:30 PM	LIB File
BIPOLAR.LIB	29-Nov-12 3:30 PM	LIB File
BRIDGE.LIB	29-Nov-12 3:30 PM	LIB File
BSTAMP.LIB	29-Nov-12 3:30 PM	LIB File
BURRBROWN.LIB	29-Nov-12 3:30 PM	LIB File
CAPACITORS.LIB	29-Nov-12 3:30 PM	LIB File
CAPIEEE.LIB	29-Nov-12 3:30 PM	LIB File

2 items selected Date modified: 12-Mar-13 1:13 PM Date created: 20-Oct-14 11:00 AM
Size: 95.3 KB

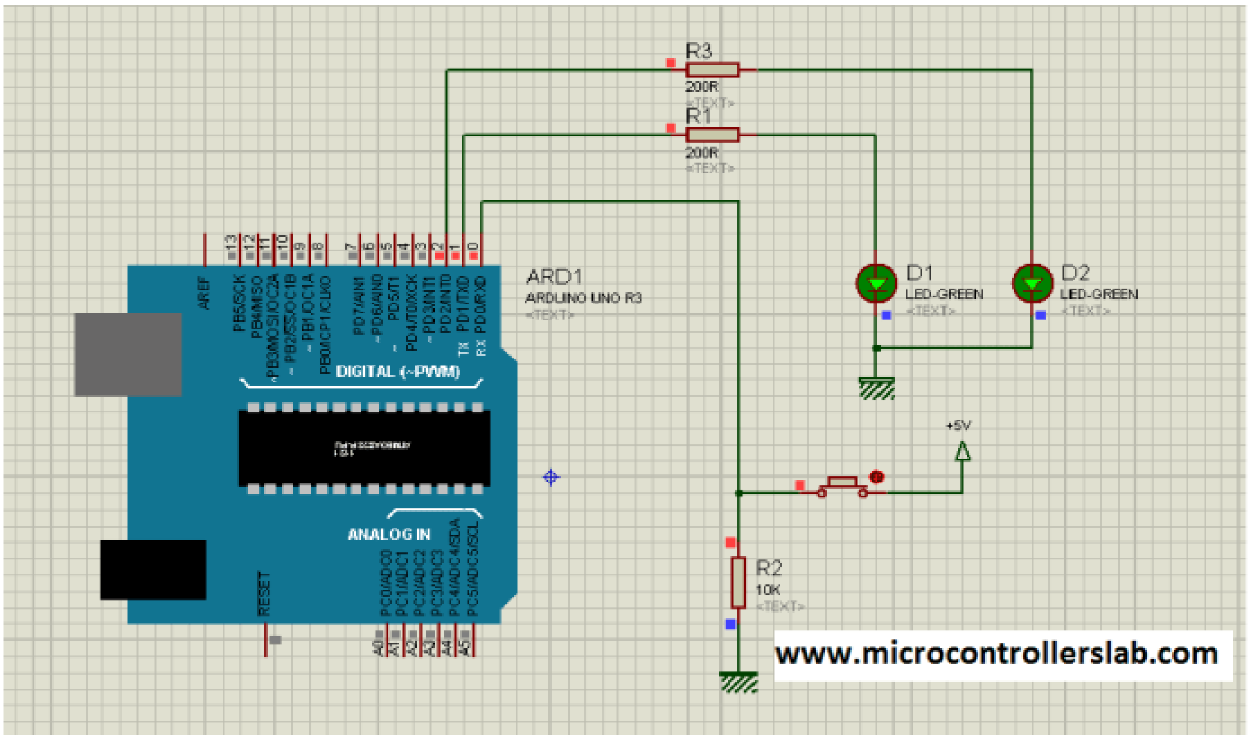
8.3. Draw schematics in Proteus

Draw the required schematic diagram of Proteus that you would like to Simulate, you can choose the **Arduino board** from the menu.



Device	Library	Description
ARDUINO MEGA1280	ARDUINO	Arduino MEGA (ATmega1280)
ARDUINO MEGA2560 R3	ARDUINO	Arduino MEGA2560 Rev3
ARDUINO UNO R3	ARDUINO	Arduino Uno R3

In our example I'm using LED blinking program and my schematics is shown below.



8.4. Programming For Push Button With Arduino

To understand the programming of push button use with Arduino, First, you need to know the input function and its use. Explanation of the input function is given below :

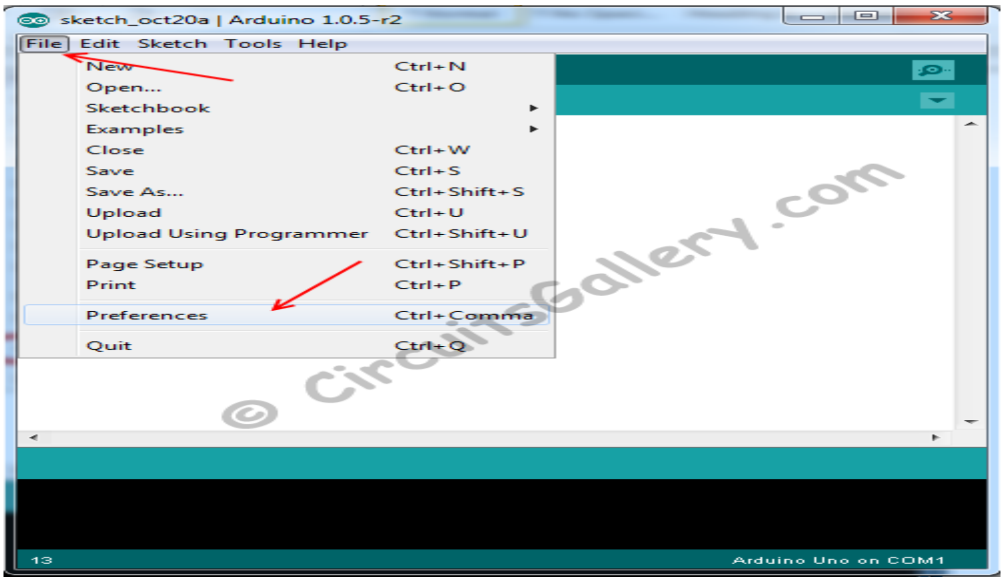
pinMode(pin number, INPUT): I have already discussed pinMode function use in **LED blinking with Arduino UNO R3**. **The same** function is used to declare pin as an input pin. Where **INPUT** argument is used to declare the specified pin number as an input pin. For example pinMode(0, INPUT) function declare pin 0 as a input pin.
digitalRead(pin number): digitalRead function is used to read status of input pin either pin is at logic level high or logic level low. The output of this function is stored in another variable which can be used to perform other functions according to your application.
digitalWrite(pin number, HIGH/LOW): digitalWrite function is used to write or output at specified pin number either output is logic high or logic low.

The code of the above circuit is given below. Copy this code to Arduino software.

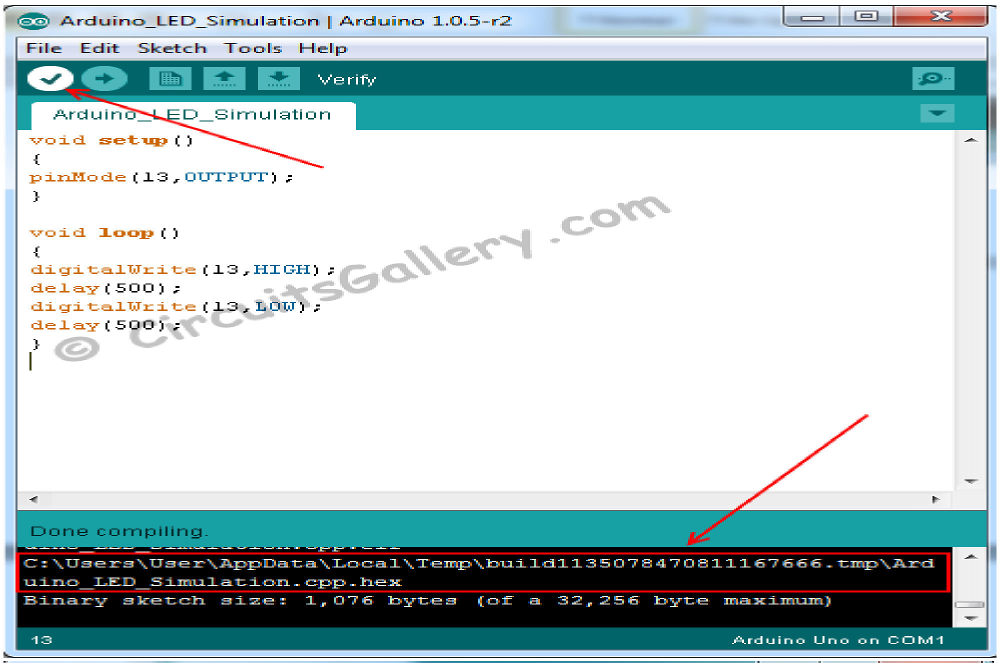
```
int buttonStatus; // Variable declare to store status of digitalWrite
 void setup() {
            pinMode(1, OUTPUT);
            pinMode(2, OUTPUT);
            pinMode(0, INPUT);
            }
void loop()
 {
            buttonStatus = digitalRead(0);
            if (buttonStatus == HIGH)
            {
            digitalWrite(1, HIGH);
            digitalWrite(2, HIGH);
            }
            else
            {
            digitalWrite(1, LOW);
            digitalWrite(2, LOW);
            }
 }
```

8.5. **Build .hex file from Arduino Software**

Go to **File** → **Preferences**

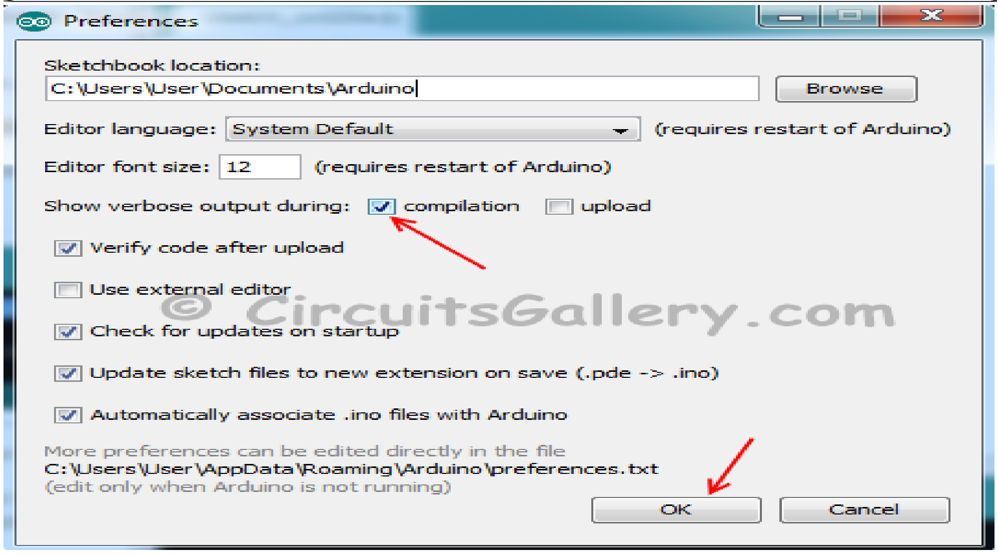


There will be one 2 check box near to '**Show verbose output during**', please check the first one.


Arduino_LED_Simulation | Arduino 1.0.5-r2
File Edit Sketch Tools Help
Verify
Arduino_LED_Simulation

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

void loop()
{
digitalWrite(13,HIGH);
delay(500);
digitalWrite(13,LOW);
delay(500);
}
```

```
Done compiling.
C:\Users\User\AppData\Local\Temp\build1135078470811167666.tmp\Ard
uino_LED_Simulation.cpp.hex
Binary sketch size: 1,076 bytes (of a 32,256 byte maximum)
```



Sketchbook location: C:\Users\User\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12 (requires restart of Arduino)

Show verbose output during: [✓ compilation] upload

[✓ Verify code after upload]

Use external editor

[✓ Check for updates on startup]

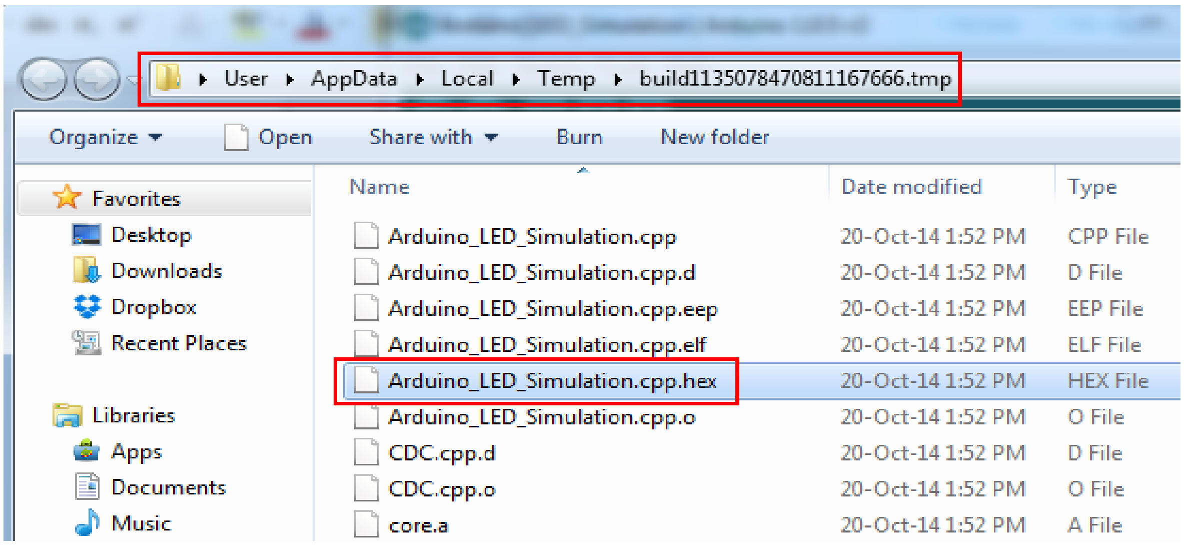
[✓ Update sketch files to new extension on save (.pde -> .ino)]

[✓ Automatically associate .ino files with Arduino]

More preferences can be edited directly in the file
C:\Users\User\AppData\Roaming\Arduino\preferences.txt
(edit only when Arduino is not running)

Now click **Compile button.**

Compilation yields .hex file, go to the directory location marked above (may be different location in your case), this is the **location of your .hex file.** Copy the file and keep it in a safe location.

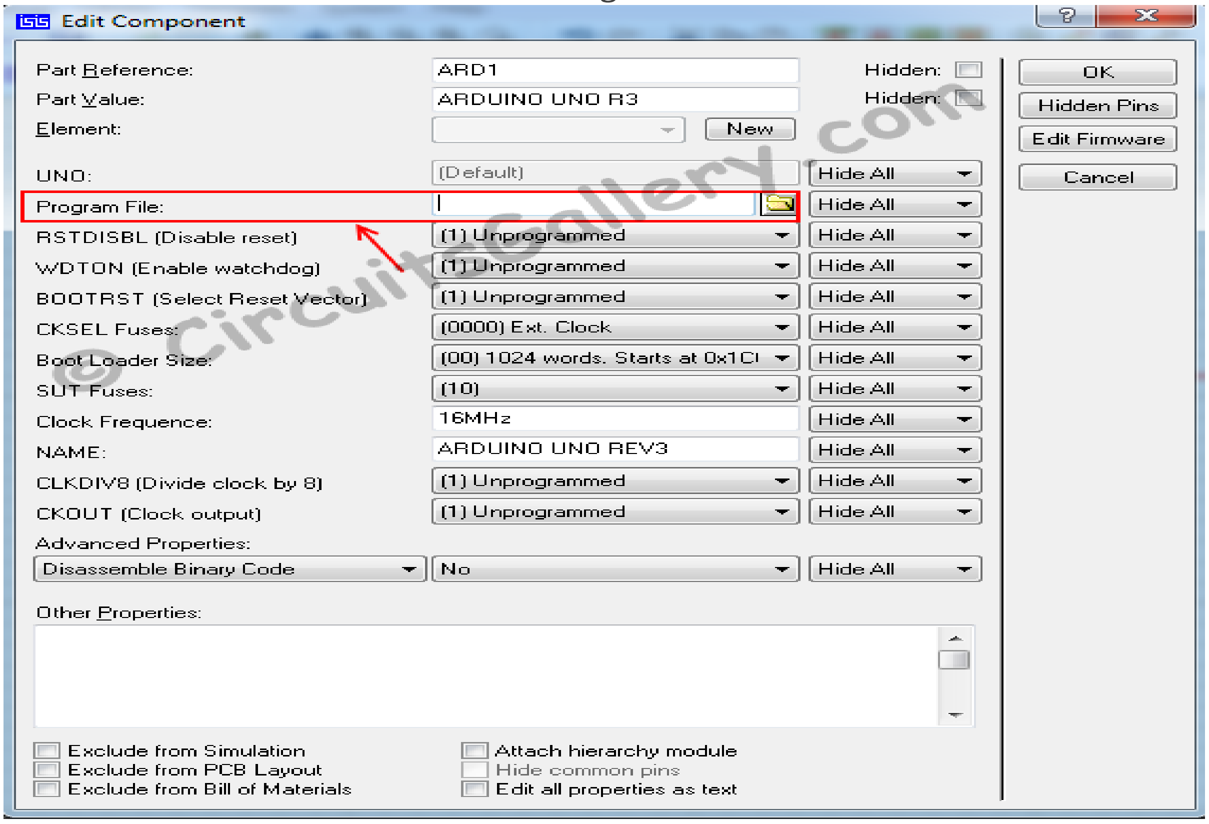


User ▸ AppData ▸ Local ▸ Temp ▸ build1135078470811167666.tmp

Name	Date modified	Type
Arduino_LED_Simulation.cpp	20-Oct-14 1:52 PM	CPP File
Arduino_LED_Simulation.cpp.d	20-Oct-14 1:52 PM	D File
Arduino_LED_Simulation.cpp.eep	20-Oct-14 1:52 PM	EEP File
Arduino_LED_Simulation.cpp.elf	20-Oct-14 1:52 PM	ELF File
Arduino_LED_Simulation.cpp.hex	20-Oct-14 1:52 PM	HEX File
Arduino_LED_Simulation.cpp.o	20-Oct-14 1:52 PM	O File
CDC.cpp.d	20-Oct-14 1:52 PM	D File
CDC.cpp.o	20-Oct-14 1:52 PM	O File
core.a	20-Oct-14 1:52 PM	A File

8.6. Load the .hex file to Proteus

To load .hex file to the Arduino board, just double click on the board and browse the .hex file in the **Program File** section.



Edit Component

Part Reference:	ARD1	Hidden: ☐
Part Value:	ARDUINO UNO R3	Hidden: ☐
Element:		New
UNO:	(Default)	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 0x1C(	Hide All
SUT Fuses:	(10)	Hide All
Clock Frequence:	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

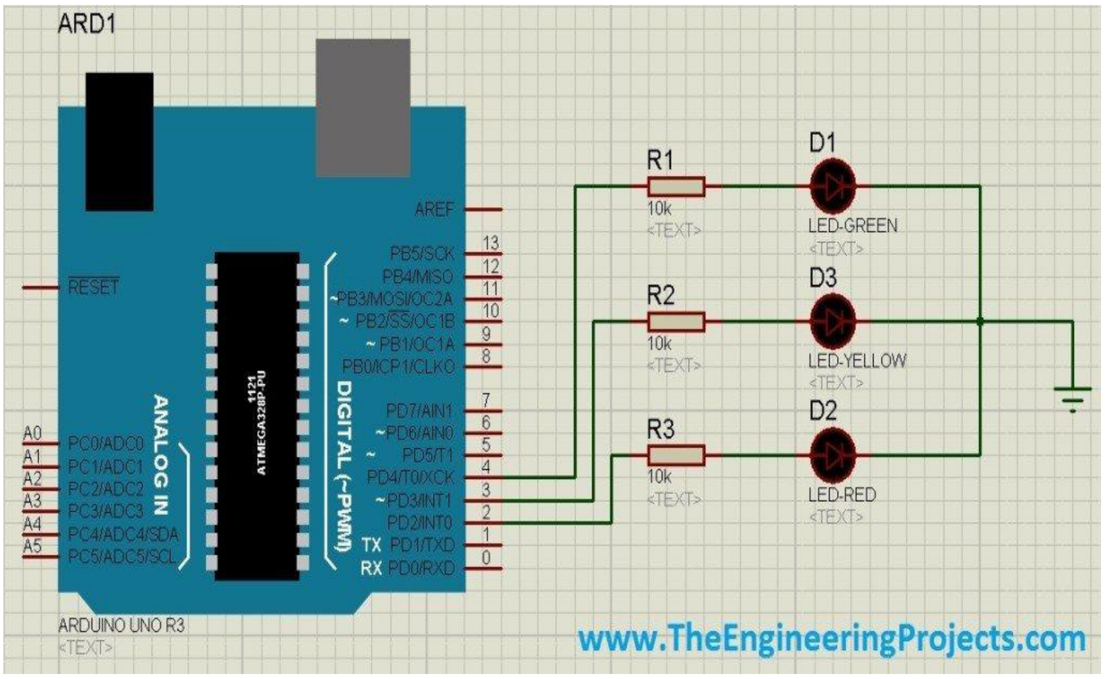
Other Properties:

☐ Exclude from Simulation
☐ Exclude from PCB Layout
☐ Exclude from Bill of Materials
☐ Attach hierarchy module
☐ Hide common pins
☐ Edit all properties as text

Everything is ready to go! **Run the Simulation** now. You can see the simulated output of the Arduino Program, watch the video for my simulation output.

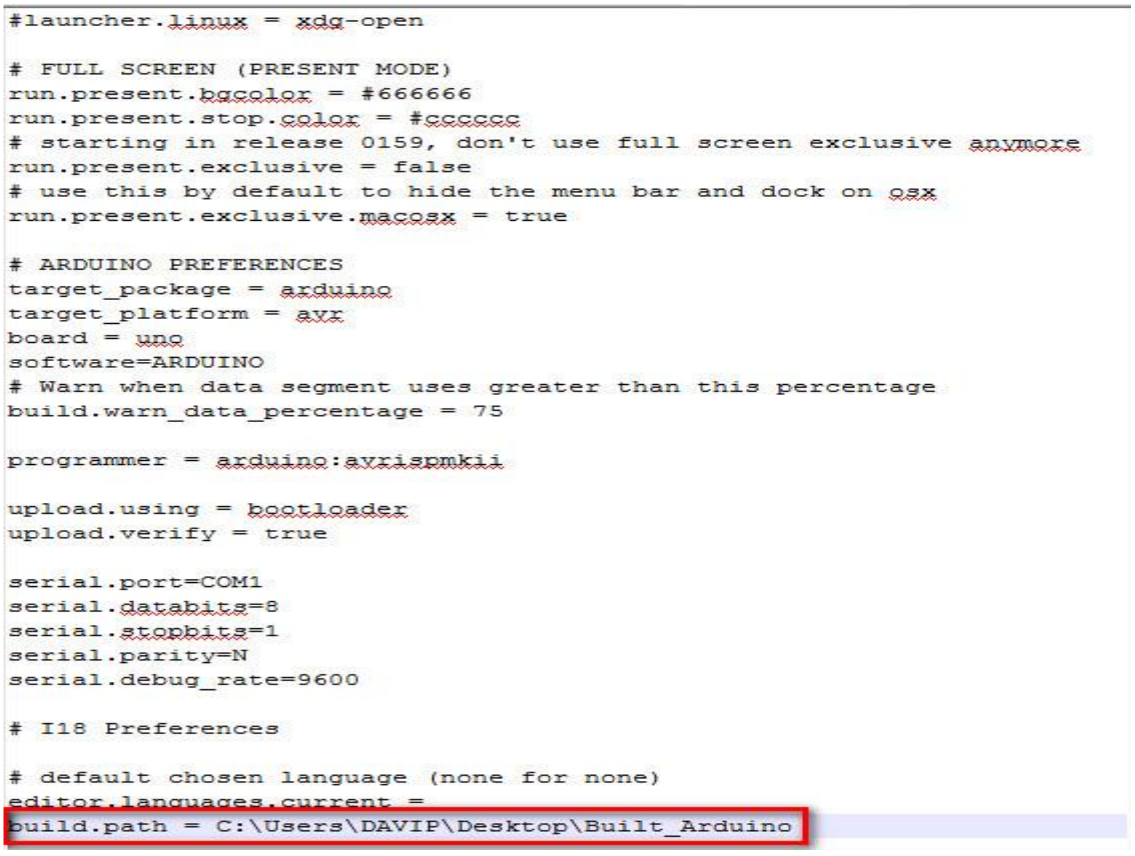
9. Traffic signal control

9.1. circuit in Proteus

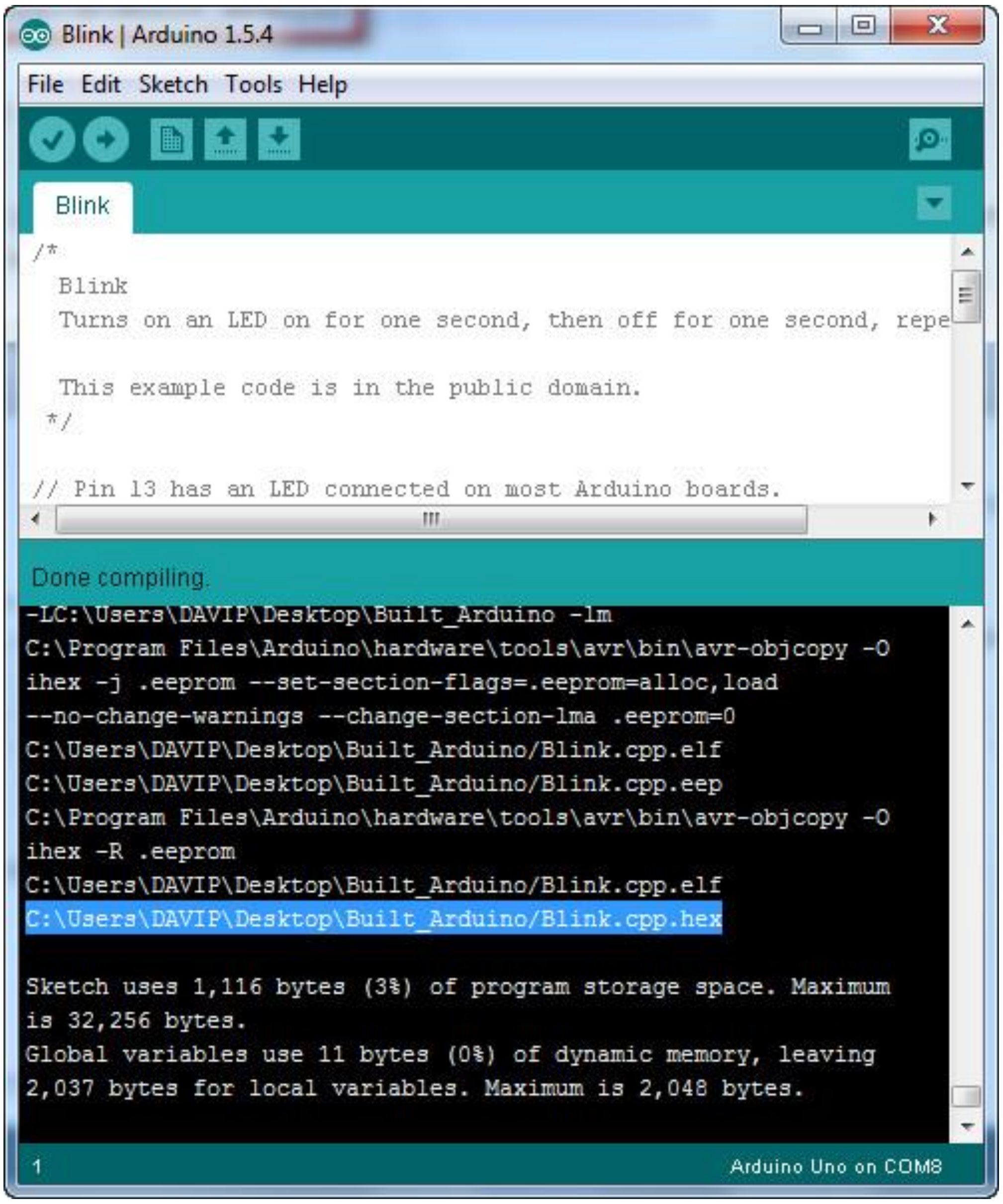


10. Lưu file Hex vào vị trí mong muốn

- Bước 1: Mở tập tin **C:\Program Files\Arduino\lib\preferences.txt** bằng Notepad hoặc Notepad++
- Bước 2: Thêm dòng lệnh sau vào cuối file preferences.txt: **build.path = <path>** (<path> là đường dẫn đến thư mục mà bạn muốn chứa file hex sau khi Arduino hoàn tất quá trình biên dịch)
- Ví dụ: Để lưu vào thư mục **Built_Arduino** (mình vừa tạo) trên desktop của mình, mình viết như sau:


```
#launcher.linux = xdg-open

# FULL SCREEN (PRESENT MODE)
run.present.bgcolor = #666666
run.present.stop.color = #cccccc
# starting in release 0159, don't use full screen exclusive anymore
run.present.exclusive = false
# use this by default to hide the menu bar and dock on osx
run.present.exclusive.macosx = true

# ARDUINO PREFERENCES
target_package = arduino
target_platform = avr
board = uno
software=ARDUINO
# Warn when data segment uses greater than this percentage
build.warn_data_percentage = 75

programmer = arduino:avrispmkii

upload.using = bootloader
upload.verify = true

serial.port=COM1
serial.databits=8
serial.stopbits=1
serial.parity=N
serial.debug_rate=9600

# I18 Preferences

# default chosen language (none for none)
editor.languages.current =
build.path = C:\Users\DAVIP\Desktop\Built_Arduino
```

- Sau đó bạn nhớ ấn **Ctr + S** để **Save** lại nhé :D
- Tiếp theo bạn **tắt IDE đi và mở lại** (nhất thiết phải vậy nhé), mở **CODE Blink** led nhấp nháy như trên mình trình bày và **Verify** ✓
- Bạn sẽ thấy file .hex đã không còn ở thư mục temp khó tìm nữa mà vào đúng thư mục mình mong muốn.



```
/*
  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.
```

```
-LC:\Users\DAVIP\Desktop\Built_Arduino -lm
C:\Program Files\Arduino\hardware\tools\avr\bin\avr-objcopy -O
ihex -j .eeprom --set-section-flags=.eeprom=alloc,load
--no-change-warnings --change-section-lma .eeprom=0
C:\Users\DAVIP\Desktop\Built_Arduino/Blink.cpp.elf
C:\Users\DAVIP\Desktop\Built_Arduino/Blink.cpp.eep
C:\Program Files\Arduino\hardware\tools\avr\bin\avr-objcopy -O
ihex -R .eeprom
C:\Users\DAVIP\Desktop\Built_Arduino/Blink.cpp.elf
C:\Users\DAVIP\Desktop\Built_Arduino/Blink.cpp.hex

Sketch uses 1,116 bytes (3%) of program storage space. Maximum
is 32,256 bytes.
Global variables use 11 bytes (0%) of dynamic memory, leaving
2,037 bytes for local variables. Maximum is 2,048 bytes.
```

11.