

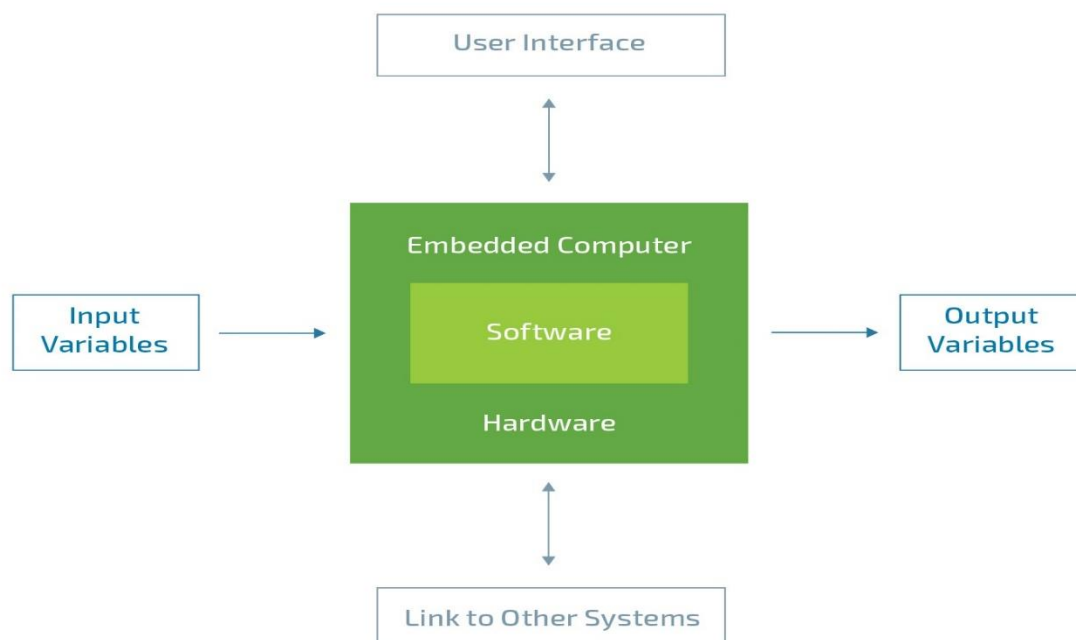
REPORT: ARDUINO UNO-R3

ABSTRACT-

An embedded system is a system that is a combination of hardware peripherals and software languages, that is designed or developed to do a particular task.

An embedded system as basic elements - User interface (Display), Input/Output peripherals (Sensors, LEDs, actuators etc) and a software architecture that creates a logic on which the system should work (Embedded C and Software to program that into the controller) and heart of the system- Microcontroller. This also can connect with other systems, clouds etc. During these recent time, human life have become more precious for us.

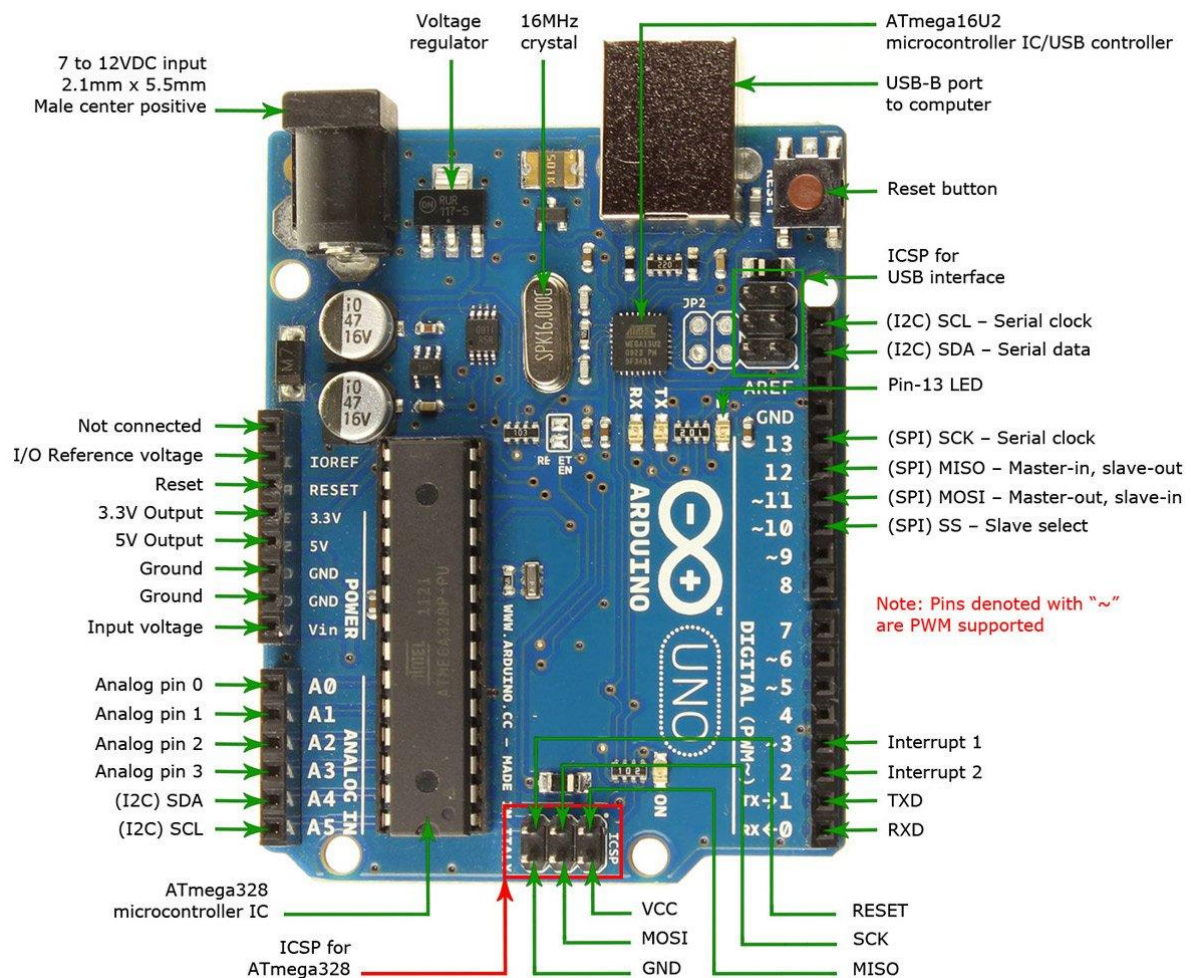
Human involvement in medical field needs replacement from automated technology. We can achieve our target by having a combination of IOT and embedded systems.



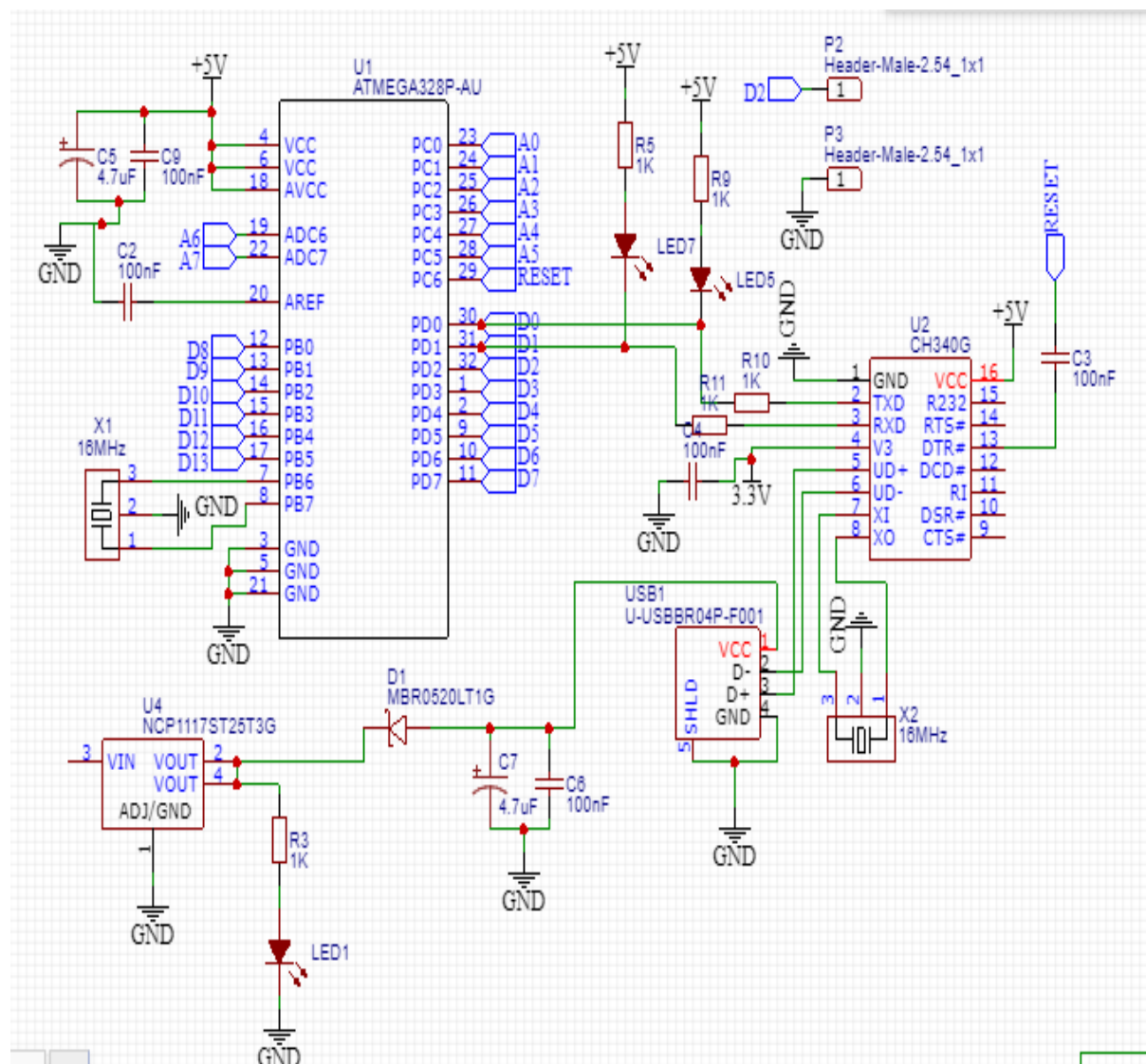
INTRODUCTION -

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. we can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. It is based on ATMEL ATmega328p microcontroller.

CIRCUIT -



SCHEMATIC LAYOUT-



Arduino Uno Technical Specifications

Microcontroller	ATmega328P – 8-bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Arduino Uno Pinout Configuration

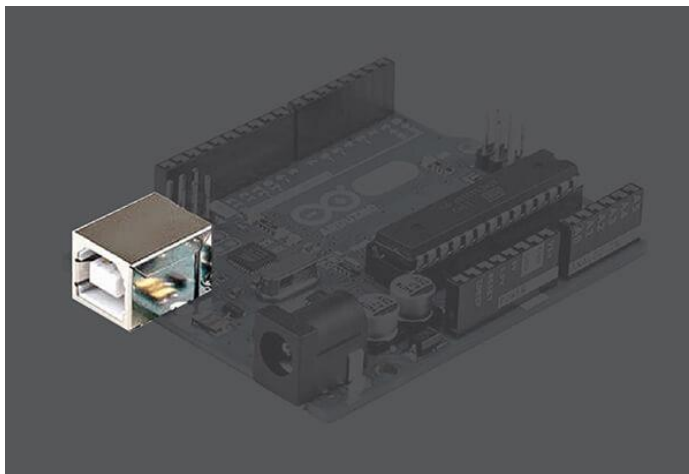
Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	<p>Vin: Input voltage to Arduino when using an external power source.</p> <p>5V: Regulated power supply used to power microcontroller and other components on the board.</p> <p>3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.</p> <p>GND: ground pins.</p>
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

COMPONENTS DETAILS-

The major components of Arduino UNO board are as follows:

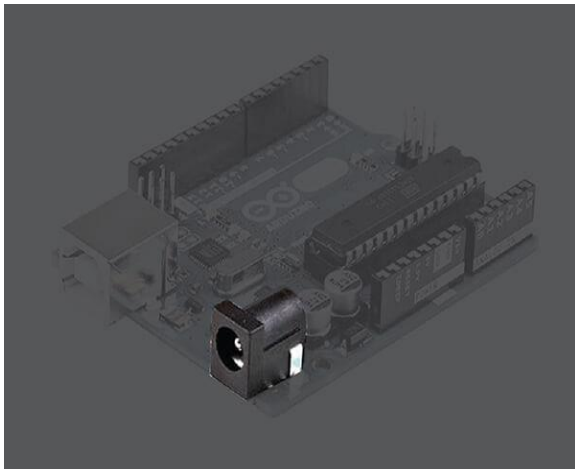
- USB connector
- Power port
- Microcontroller
- Analog input pins
- Digital pins
- Reset switch
- Crystal oscillator
- USB interface chip
- TX RX LEDs

➤ **USB connector:**



This is a printer USB port used to load a program from the Arduino IDE onto the Arduino board. The board can also be powered through this port.

➤ **Power port:**



Power port

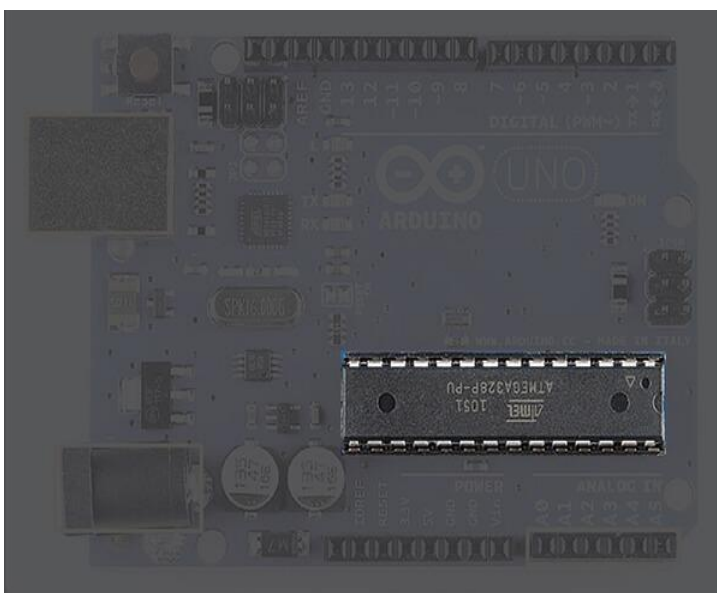


2.1mm center-positive plug

The Arduino board can be powered through an AC-to-DC adapter or a battery. The power source can be connected by plugging in a 2.1mm centre-positive plug into the power jack of the board.

The Arduino UNO board operates at a voltage of 5 volts, but it can withstand a maximum voltage of 20 volts. If the board is supplied with a higher voltage, there is a voltage regulator (it sits between the power port and USB connector) that protects the board from burning out.

➤ **Microcontroller:**

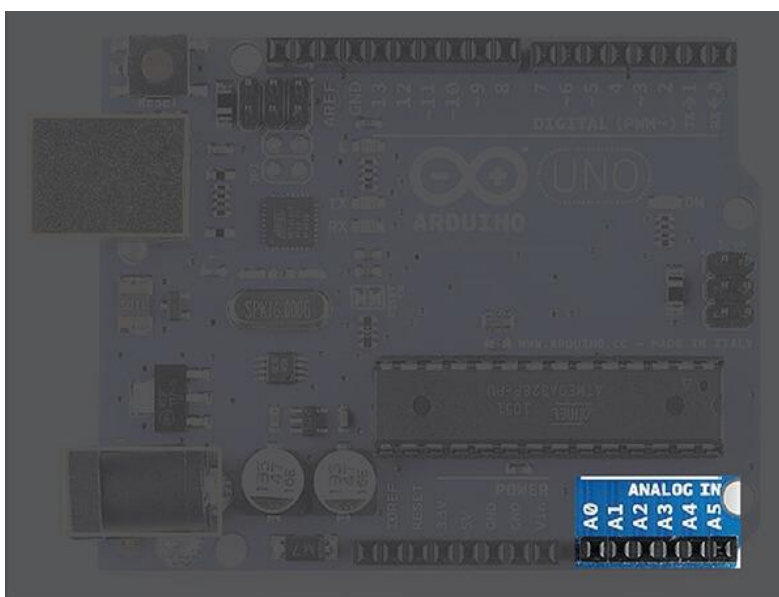


Atmega328P microcontroller

It is the most prominent black rectangular chip with 28 pins. Think of it as the brains of your Arduino. The microcontroller used on the UNO board is Atmega328P by Atmel (a major microcontroller manufacturer). Atmega328P has the following components in it:

- **Flash memory** of 32KB. The program loaded from Arduino IDE is stored here.
- **RAM** of 2KB. This is a runtime memory.
- **CPU**: It controls everything that goes on within the device. It fetches the program instructions from flash memory and runs them with the help of RAM.
- **Electrically Erasable Programmable Read Only Memory (EEPROM)** of 1KB. This is a type of non-volatile memory, and it keeps the data even after device restart and reset.
- Atmega328P is pre-programmed with bootloader. This allows you to directly upload a new Arduino program into the device, without using any external hardware programmer, making the Arduino UNO board easy to use.

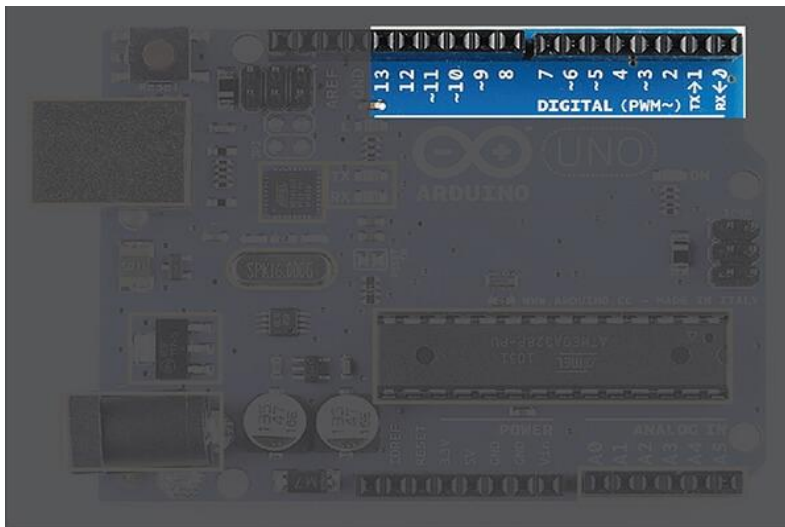
➤ **Analog input pins:**



Analog input pins

- The Arduino UNO board has 6 analog input pins, labelled “Analog 0 to 5.” These pins can read the signal from an analog sensor like a temperature sensor and convert it into a digital value so that the system understands. These pins just measure voltage and not the current because they have very high internal resistance. Hence, only a small amount of current flows through these pins.
- Although these pins are labelled analog and are analog input by default, these pins can also be used for digital input or output.

➤ Digital pins:

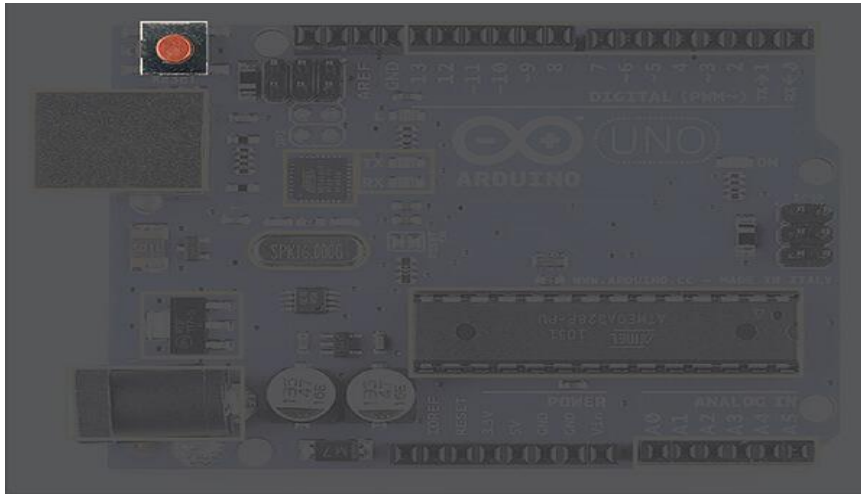


Digital pins

- We can find these pins labelled “Digital 0 to 13.” These pins can be used as either input or output pins. When used as output, these pins act as a power supply source for the components connected to them. When used as input pins, they read the signals from the component connected to them.
- When digital pins are used as output pins, they supply 40 milliamps of current at 5 volts, which is more than enough to light an LED.
- Some of the digital pins are labelled with tilde (~) symbol next to the pin numbers (pin numbers 3, 5, 6, 9, 10, and 11). These pins act as normal

digital pins but can also be used for Pulse-Width Modulation (PWM), which simulates analog output like fading an LED in and out.

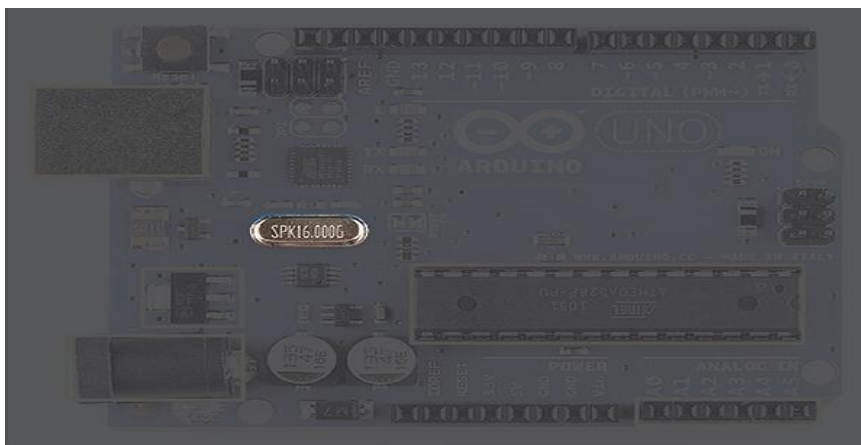
➤ **Reset switch:**



Reset switch

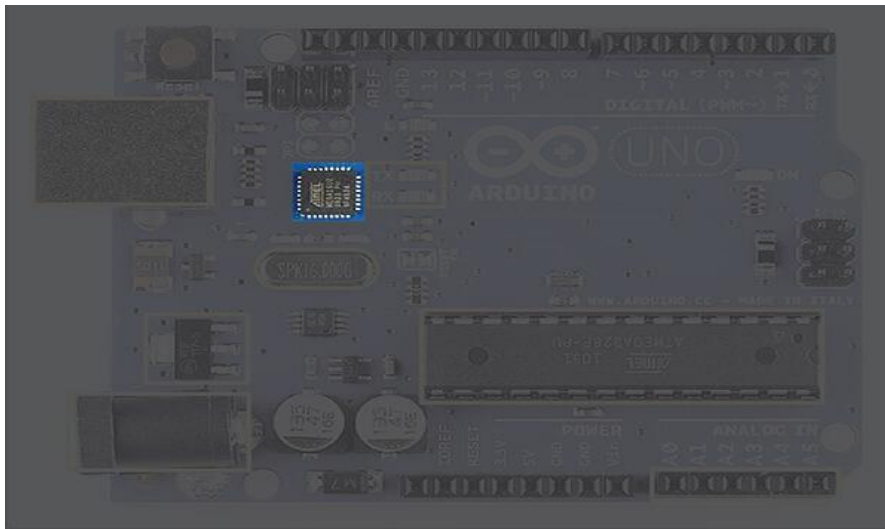
When this switch is clicked, it sends a logical pulse to the reset pin of the Microcontroller, and now runs the program again from the start. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

➤ **Crystal oscillator:**



This is a quartz crystal oscillator which ticks 16 million times a second. On each tick, the microcontroller performs one operation, for example, addition, subtraction, etc.

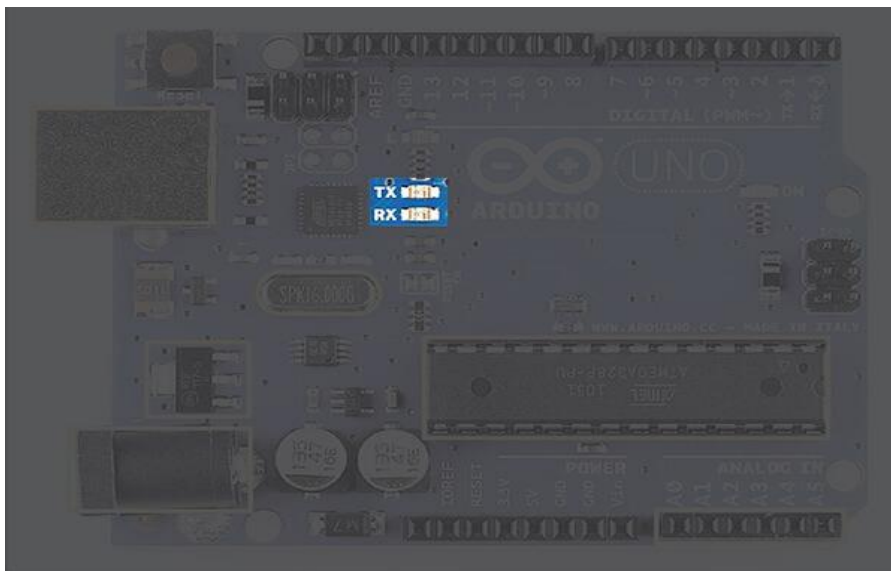
➤ **USB interface chip:**



USB interface chip

Think of this as a signal translator. It converts signals in the USB level to a level that an Arduino UNO board understands.

➤ **TX – RX LEDs:**



TX – RX indicator

TX stands for transmit, and RX for receive. These are indicator LEDs which blink whenever the UNO board is transmitting or receiving data.

Now that you have explored the Arduino UNO board, you have started your journey toward building your first IoT prototype. In the next article, we will discuss Arduino programming and do a few experiments with Arduino and LEDs.

ARDUINO SOFTWARE (IDE)-

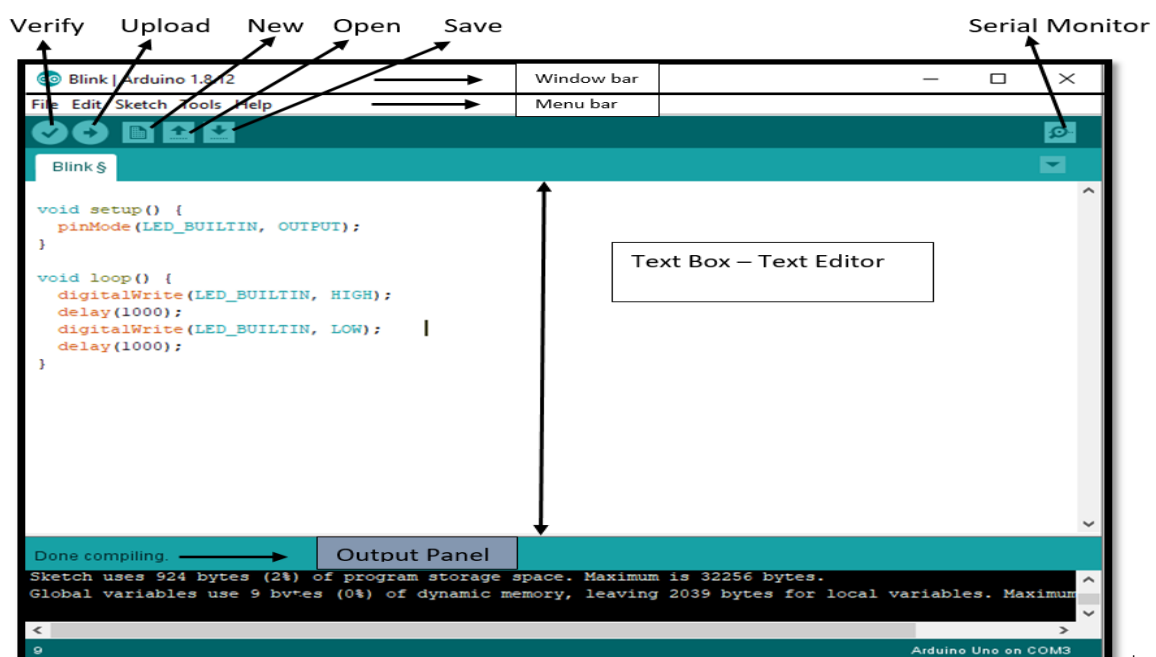
- Arduino IDE (Integrated Development Environment) is the software for Arduino.
- It is a text editor like a notepad with different features.
- It is used for writing code, compiling the code to check if any errors are there and uploading the code to the Arduino.
- It is a cross-platform software which is available for every Operating System like Windows, Linux, macOS.
- It supports C/C++ language
- It is open-source software, where the user can use the software as they want it to. They can also make their own modules/functions and add them to the software
- It supports every available Arduino board including Arduino mega, Arduino Leonardo, Arduino Ethernet and more
- Word file is called a Document similarly, Arduino file is called a **Sketch** where the user writes code.
- The format of Arduino is saved as. **ino**

FUNCTIONS OF ARDUINO IDE-

When a user writes code and compiles, the IDE will generate a Hex file for the code. (**Hex file are Hexa-Decimal files which are understood by Arduino**) and then sent to the board using a USB cable. Every Arduino board is integrated with a microcontroller, the microcontroller will receive the hex file and runs as per the code written.

Arduino IDE consists of different sections

1. Window Bar
2. Menu Bar
3. Shortcut Buttons
4. Text Editor
5. Output Panel



➤ Window Bar:

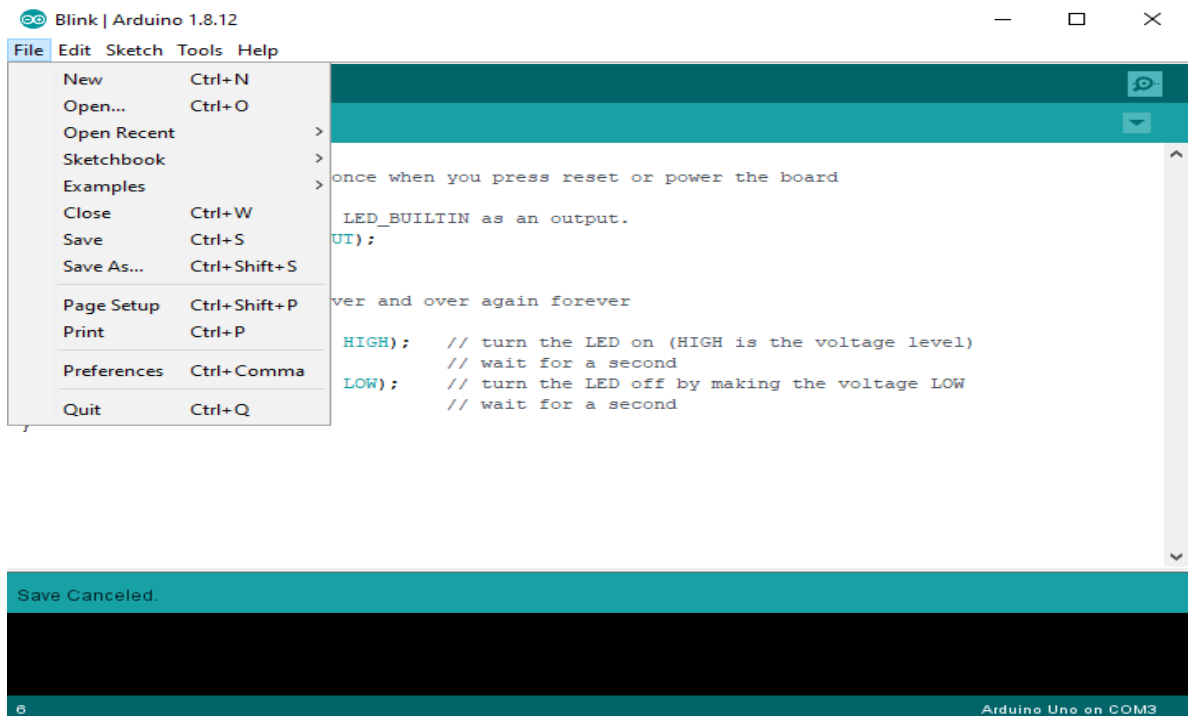
The window bar consists the name of File and the Arduino IDE software version

➤ Menu Bar:

The menu bar consists of

- File
- Edit
- Sketch
- Tools
- Help

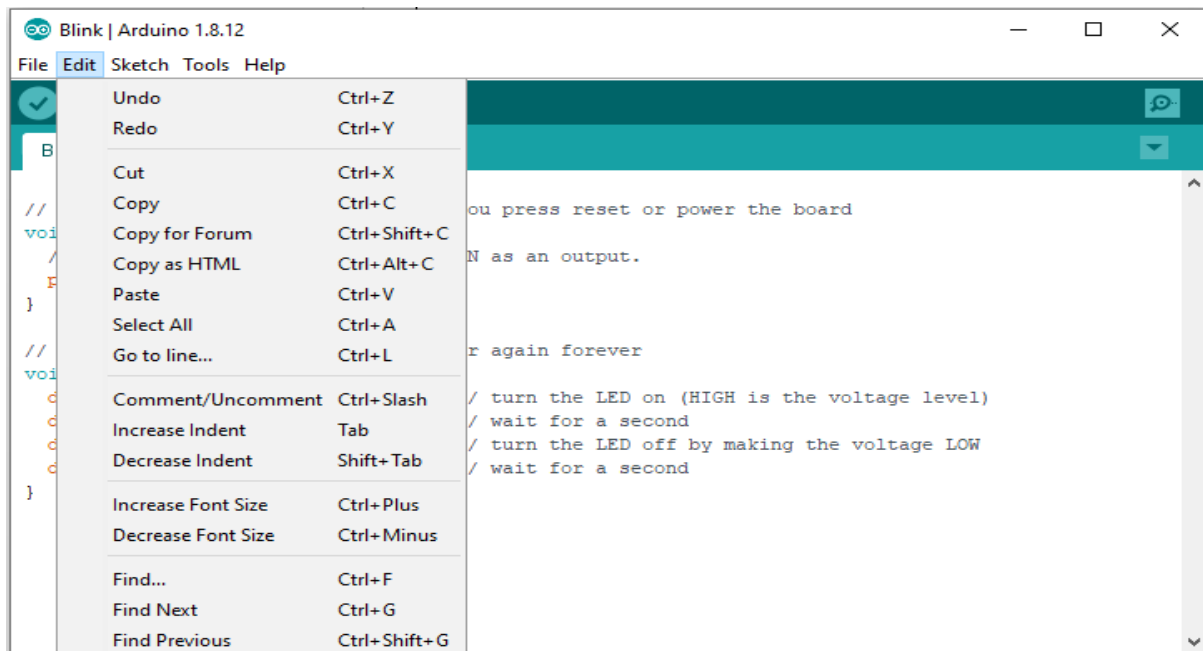
➤ **File:**



- **New**
It creates a new File. (**Ctrl+N**)
- **Open**
It is used to open the file which was saved before. (**Ctrl+O**)
- **Open Recent**
It shows the shortlist of Recently opened programs.
- **Sketchbook**
Shows the current sketches which you have used for your project
- **Examples**
Examples of a few basic problems for reference.
- **Close**
Closes the main screen window. (**Ctrl+W**)
- **Save**
It is used to save the current sketch. (**Ctrl+S**)
- **Save as...**
Allows saving the current sketch with a different name. (**Ctrl+Shift+S**)
- **Page Setup**
Page settings for modifying the page (Text). (**Ctrl+Shift+P**)

- **Print**
Used to print the current program. **(Ctrl+P)**
- **Preferences**
Settings of the IDE software can be changed here. **(Ctrl+,)**
- **Quit**
Closes all IDE windows. **(Ctrl+Q)**

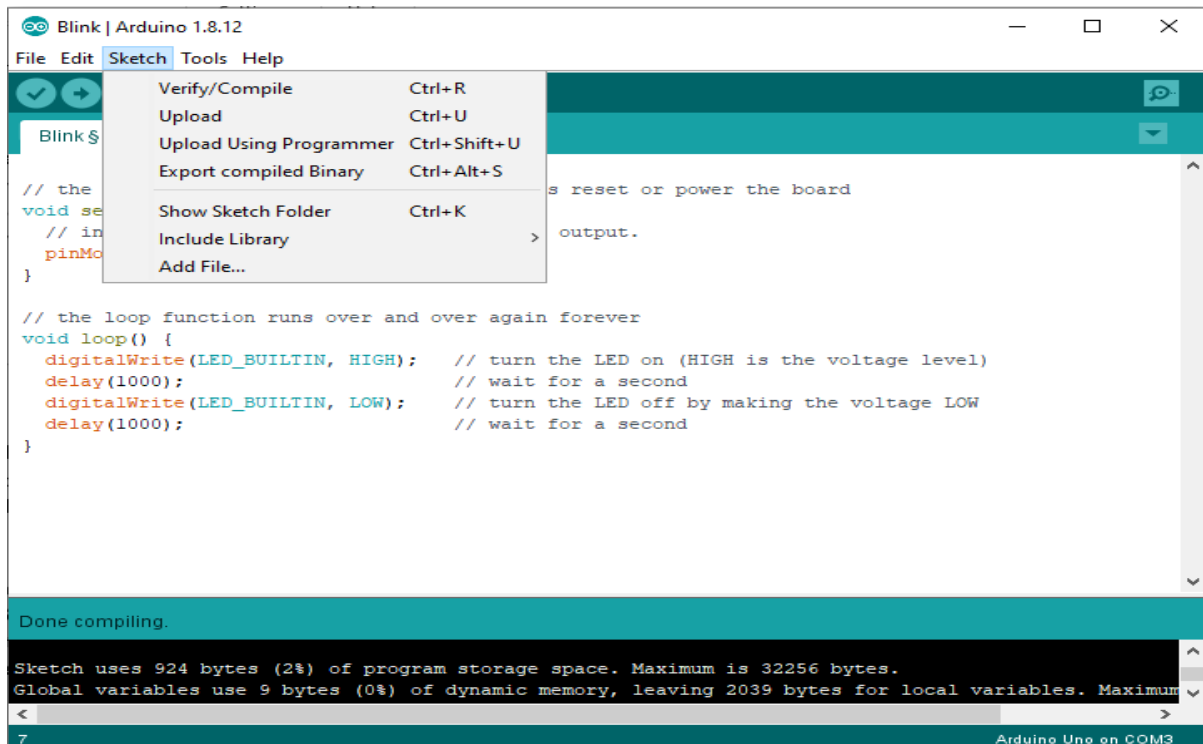
➤ **Edit:**



- **Undo/Redo**
Goes back of one or more steps you did while editing.
- **Cut**
Cuts the selected text from the editor.
- **Copy**
Copies the selected text from the editor
- **Copy for Forum**
It copies and changes the style of code suitable for the forum.
- **Copy as HTML**
It copies and changes the style of code suitable for the Html.
- **Paste**
It pastes the text from the copied text.
- **Select All**
Select's all the content from the editor.
- **Comment/Uncomment**
It is used to comment and uncomment selected lines of code.
- **Increase/Decrease Indent**
Adds or removes a space at the beginning of each selected line

- **Find**
Finds the typed text in the editor
- **Find Next**
Finds the next position of the searching word.
- **Find Previous**
Finds the previous position of the searching word.

➤ Sketch

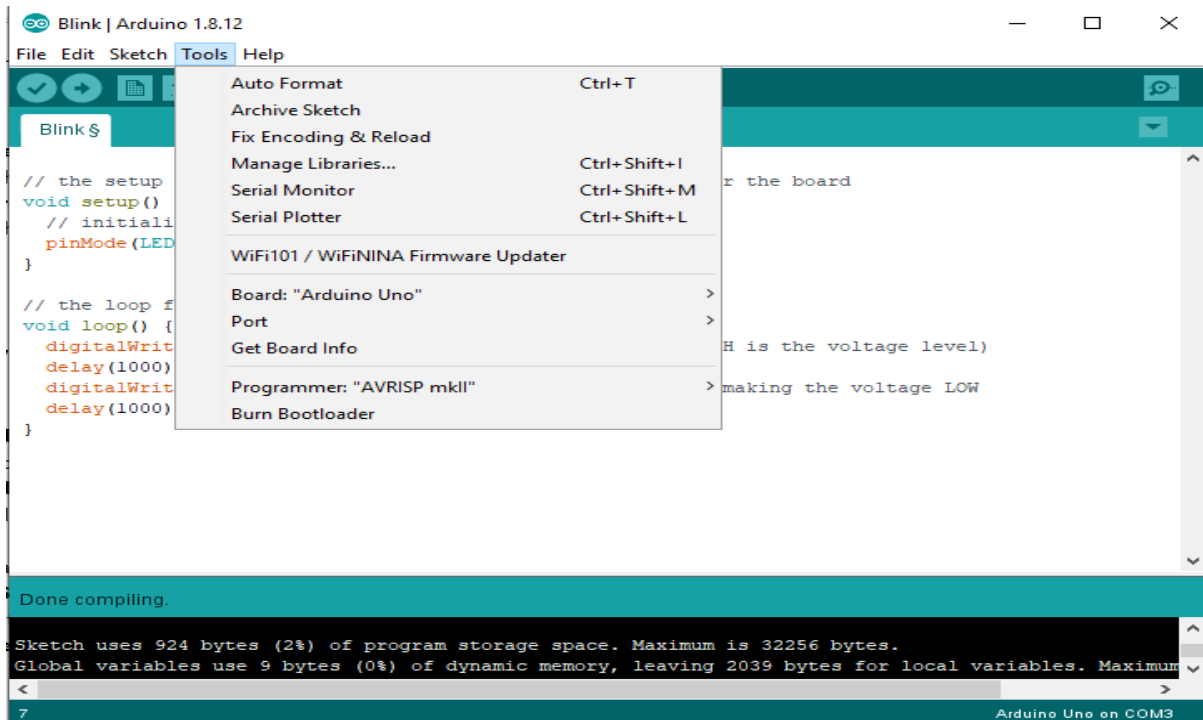


- **Verify/Compile**
Checks or verifies your program if any error is there, and displays in the output panel.
- **Upload**
It compiles and also uploads the code to the Arduino board.
- **Upload Using Programmer**
Uploads code using Programmer which is available in Tools Tab.
- **Export Compiled Binary**
Saves a .hex file in the System
- **Show Sketch Folder**
Opens the current sketch folder.
- **Include Library**
Adds a library to your sketch by inserting #include statements at the start of your code

- **Add File...**

Adds a file to the sketch and the new file appears in a new tab in the window.

➤ Tools



- **Auto Format**

This option formats your code to a nice format so everyone can understand.

- **Archive Sketch**

Copies the code into WinRAR format(.zip)

- **Fix Encoding & Reload**

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

- **Serial Monitor**

Serial monitor shows the visual communication by sending and receiving data

- **Board**

To select the type of Arduino Board

- **Port**

To select the port where you have connected the Arduino

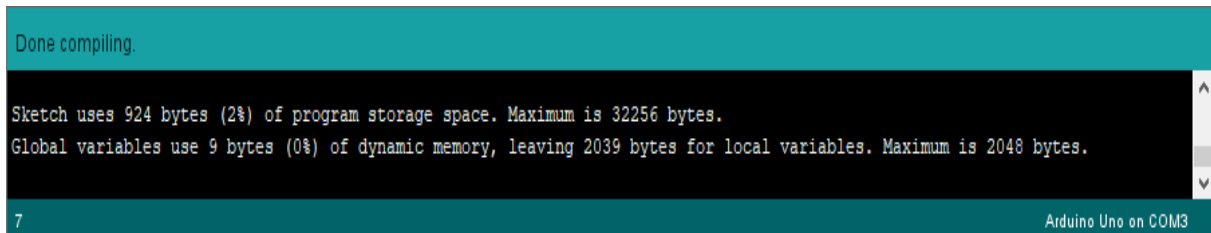
- **Programmer**

For selecting a hardware programmer when programming a board or chip and not using the USB type of communication.

- **Burn Bootloader**

- It is used to burn bootloader to the Arduino board

➤ **Output Panel:**

A screenshot of the Arduino IDE's Output Panel. The panel has a teal header bar with the text "Done compiling." in white. Below the header is a black area with white text providing memory usage statistics: "Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes." and "Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes." On the right side of the black area is a vertical scrollbar. At the bottom of the panel is a teal bar containing the number "7" on the left and "Arduino Uno on COM3" on the right.

```
Done compiling.  
  
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.  
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.  
  
7 Arduino Uno on COM3
```

This output panel is used to give comments about the code

- if the code is successfully compiled or any error occurs.
- If the code has been successfully uploaded to the board.

PROJECT IMPLEMENTATION-

Here we have integrated several sensors, LDR, relays, actuators, servos, motors, LCD display, Ultrasonic sensor, I2C bus with our Arduino board to get out desired output using different output devices.

The key and important role of Arduino uno board here is to processes the data taken by the input pins by different input devices, convert that data into user friendly way, implement the given task provided by the user and show final result to the output devices connected to the output pins of the Arduino board.

When a sensor sends data to the board, board compile that data in the way it was encoded, the implemented output was shown by the respective output device.

Taking an example, a temperature sensor has taken a temperature data of 45°C, then according to our encoded programming if temperature is higher than 40°C a fan connected to the output pin will get started automatically.

ARDUINO UNO: advantages and disadvantages

Advantages

- Not much knowledge required to get started
- Fairly low cost, depending on shields you need
- Lots of sketches and shields available
- No external programmer or power supply needed

Disadvantages

- No understanding of the AVR microcontroller
- Sketches and shields can be difficult to modify
- No debugger included for checking scripts
- You get no experience of C or professional development tools.

REFERENCES-

<https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf>

<https://www.circuito.io/blog/arduino-code/>

<https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html>

<https://www.google.com/search?q=working+principle+of+arduino+uno&oq=working+principle+of+arduino+uno+&aqs=chrome..69i57j0i512j0i390l4.20770j1j15&sourceid=chrome&ie=UTF-8>

<https://www.google.com/search?q=components+details+of+arduino+uno&oq=components+details+of+arduino+uno+&aqs=chrome..69i57j33i160l2.11641j1j15&sourceid=chrome&ie=UTF-8>

