American International University- Bangladesh

Department of Electrical and Electronic Engineering

EEE 4103: Microprocessor and Embedded Systems Laboratory

Title: Familiarization with microcontroller, study of blink test using and implementation of a traffic control system using microcontrollers.

Introduction:

The objective of this experiment is to get familiarized with Microcontroller.

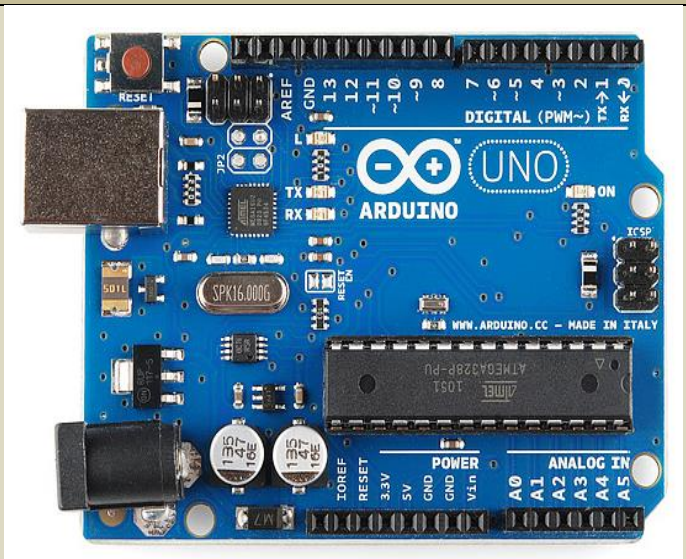
- Learning to make the LED blink using Arduino and the delay functions
- Implementation of a traffic control system using Arduino.

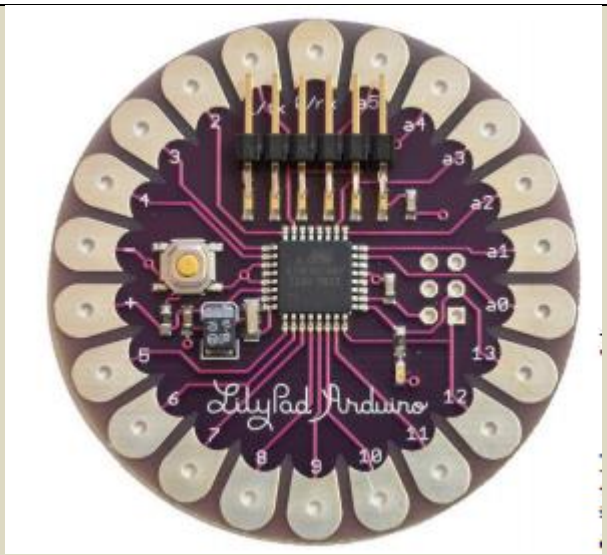
Theory and Methodology:

Arduino is an open-source platform used for creating interactive electronics projects. Arduino consists of both a programmable microcontroller and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the microcontroller board. Arduino Uno also doesn't need a hardware circuit (programmer/ burner) to load a new code into the board. We can easily load a code into the board just using a USB cable and the Arduino IDE (that uses an easier version of C++ to write a code).

Arduino Family:

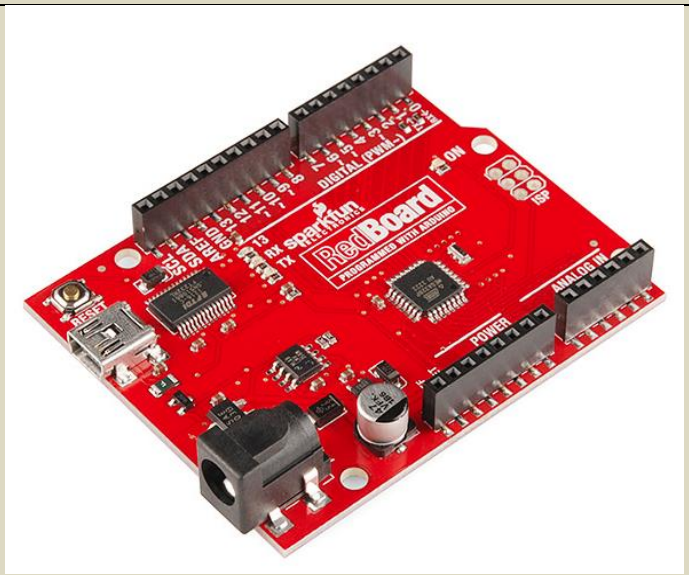
Arduino makes several different boards, each with different capabilities. In addition, part of being open source hardware means that others can modify and produce derivatives of Arduino boards that provide even more form factors and functionality. Here are a few options that are well-suited to someone new to the world of Arduino:

Figures of different types of Arduino boards	Configuration & Features
	**Arduino Uno (R3)** The Uno is a great choice for your first Arduino. It's got everything you need to get started, and nothing you don't. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, a reset button and more. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



Lilypad Arduino

LilyPad is a wearable e-textile technology developed by Leah Buechley and cooperatively designed by Leah and SparkFun. Each LilyPad was creatively designed with large connecting pads and a flat back to allow them to be sewn into clothing with conductive thread. The LilyPad also has its own family of input, output, power, and sensor boards that are also built specifically for e-textiles. They're even washable. The LilyPad is unique because it is designed to be sewn into clothing. Using conductive thread, you can wire it up to sewable sensors, LEDs,and more. To keep size down, it can be programmed by using an FTDI cable.



RedBoard

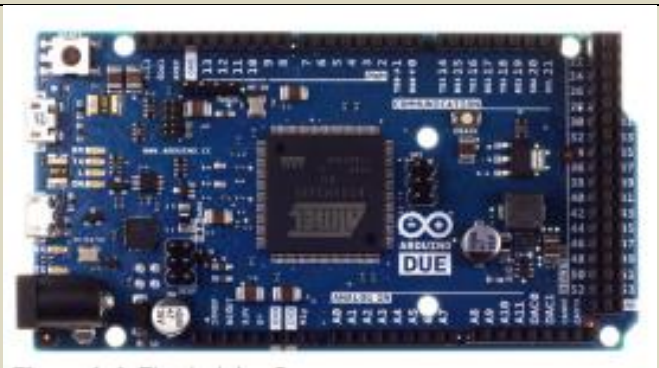
RedBoard can be programmed over a USB Mini-B cable using the Arduino IDE. It'll work on Windows 8 without having to change your security settings (we used signed drivers, unlike the UNO). It's more stable due to the USB/FTDI chip used in it, plus it's completely flat on the back, making it easier to embed in your projects. Just plug in the board, select "Arduino UNO" from the board menu and you're ready to upload code. You can power the RedBoard over USB or through the barrel jack. The on-board power regulator can handle anything from 7 to 15VDC

Arduino Mega (R3)

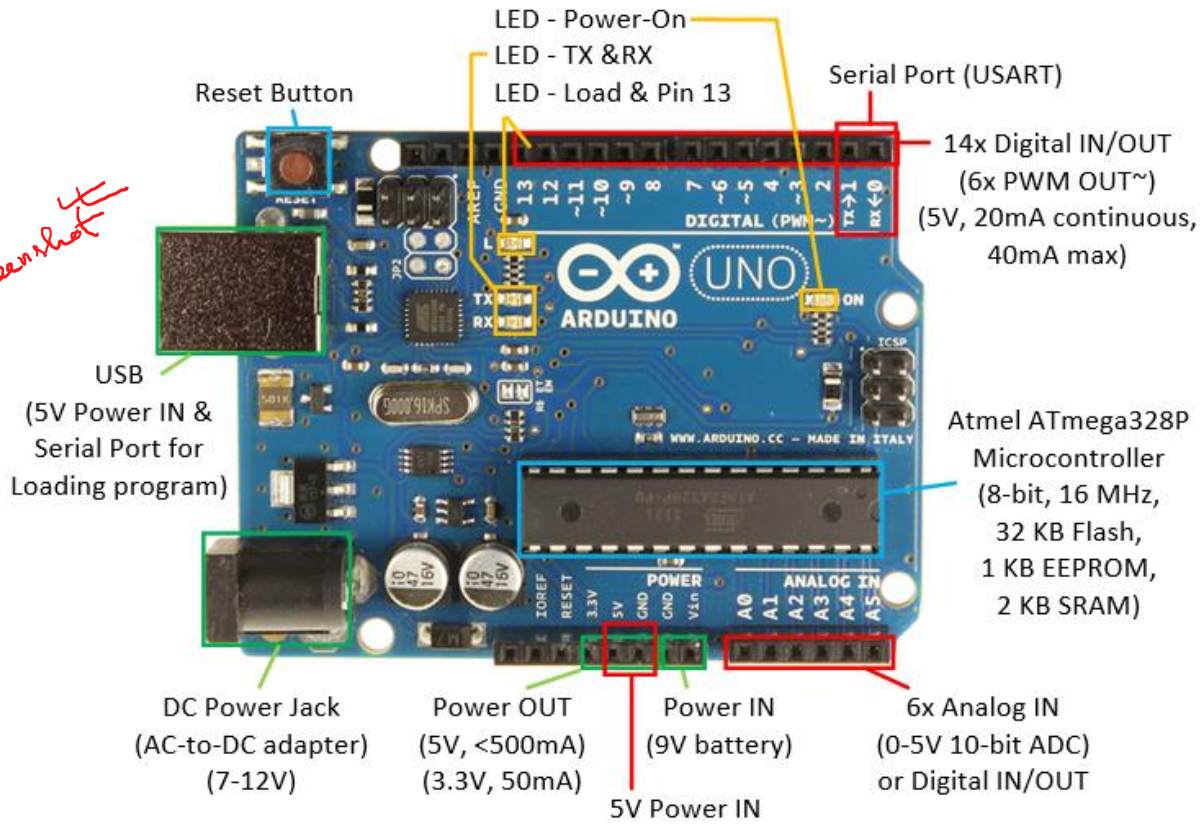
The Arduino Mega is like the UNO's big brother, it is mostly used in projects require digital i/o pins. It has lots (*54*) of digital input/output pins (14 can be used as PWM outputs), 16 analog inputs, a USB connection, a power jack, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 employs an ATMega 2560 as the main MCU, which has 54 general I/Os to enable you to interface with many more devices. The Mega also has more ADC channels, and has four hardware serial interfaces (unlike the one serial interface found on the Uno).

Arduino Leonardo

The Leonardo is Arduino's first development board to use one microcontroller with built-in USB. This means that it can be cheaper and simpler. Also, because the board is handling USB directly, code libraries are available which allow the board to emulate a computer keyboard, mouse, and more. The Leonardo uses the 32U4 as the main microcontroller, which has a USB interface built in. Therefore, it doesn't need a secondary MCU to perform the serial-to-USB conversion. This cuts down on the cost and enables you to do unique things like

	emulate a joystick or a keyboard instead of a simple serial device. You will learn how to use these features in Chapter 6, "USB and Serial Communication."
	Unlike all the other Arduino variants, which use 8-bit AVR MCUs, the Due uses a 32-bit ARM Cortex M3 SAM3X MCU. The Due offers higher-precision ADCs, selectable resolution pulse-width modulation (PWM), Digital-to-Analog Converters (DACs), a USB host connector, and an 84 MHz clock speed.
	The Nano is designed to be mounted right into a breadboard socket. Its small form factor makes it perfect for use in more finished projects.

Overview of the board (Arduino Uno R3):



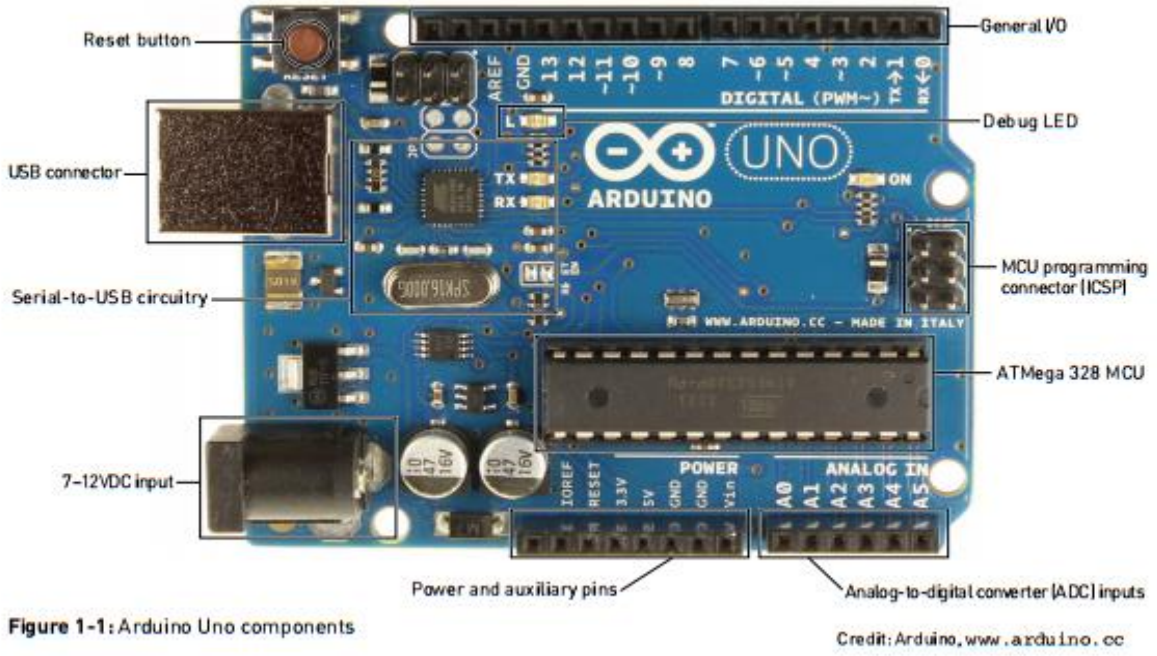


Figure 1-1: Arduino Uno components

1. DC Power Jack and USB connector

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack.

2. Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF):

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjuction with a breadboard and some wire. They usually have black plastic 'headers' that allow you

to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND** : Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V & 3.3V** : As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog**: The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog Input pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital** : Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM** : You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF** : Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

3. Reset Button:

Just like the original Nintendo, the Arduino has a reset button . Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

4. Power LED indicator:

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON'. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit.

5. TX RX LEDs:

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

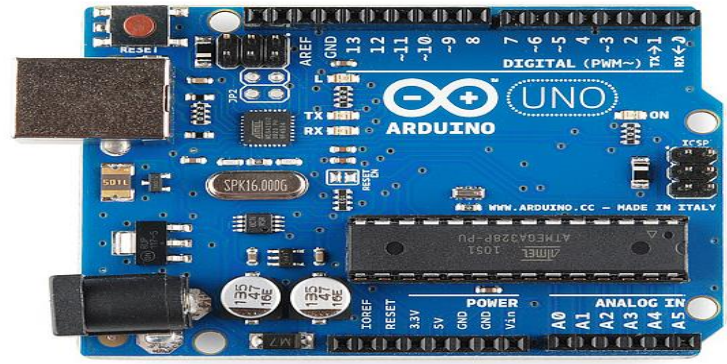
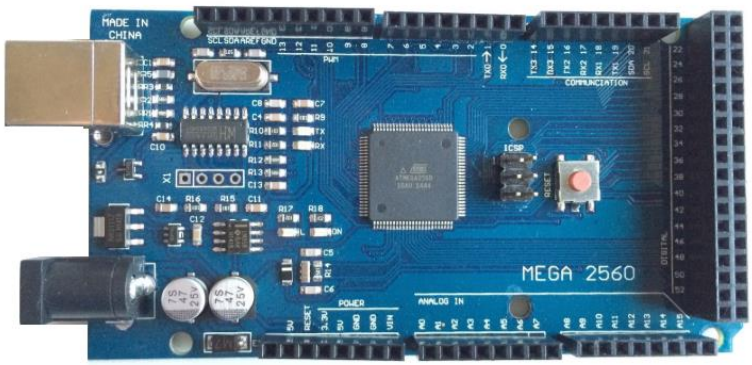
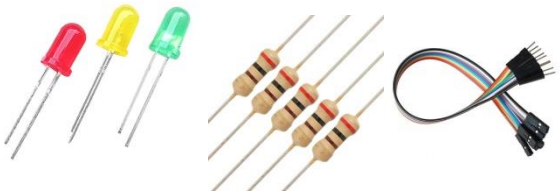
6. Main IC:

The black thing with all the metal legs is an IC, or Integrated Circuit . Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software.

7. Voltage Regulator:

The voltage regulator is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

Apparatus: (can list elements or use table below)

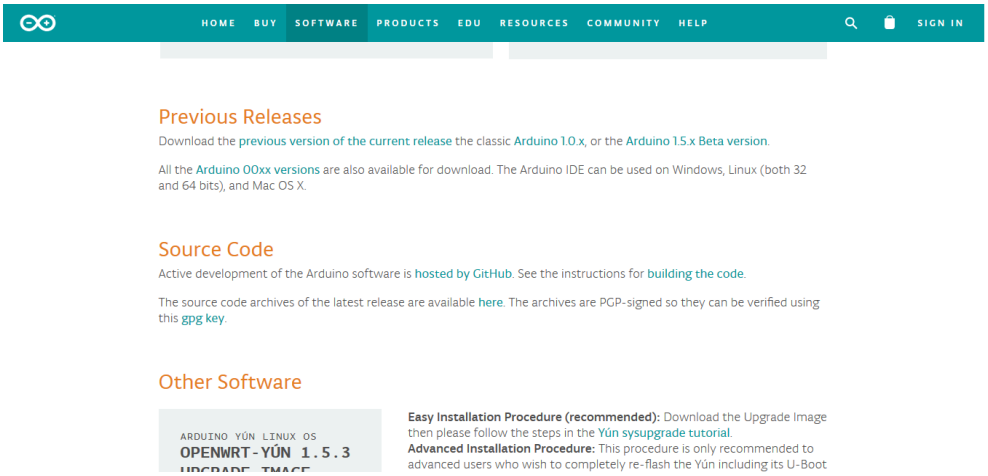
1) Arduino IDE (any version)	Software
2) Arduino Uno (R3) board or Arduino mega 2560 board	 
3) LED lights (RED,GREEN and YELLOW) and three 200 ohms resisters and jumper wires	

Setting up the Arduino IDE:

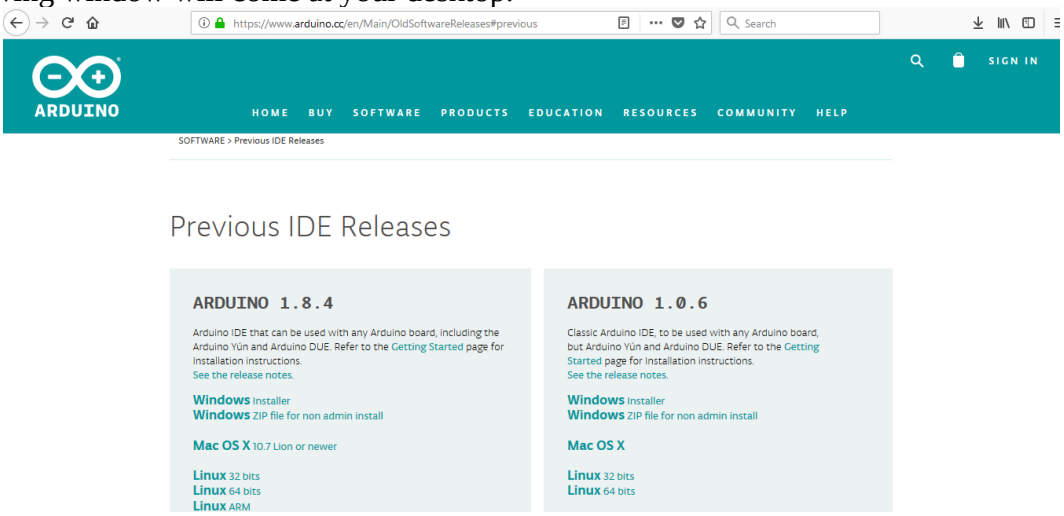
1. At first go to the following official Arduino IDE download page to download the latest version of Aduino IDE (before start downloading Aduino IDE, always keep in mind which operating system you are using in your PC).

Link: https://www.arduino.cc/en/Main/Software

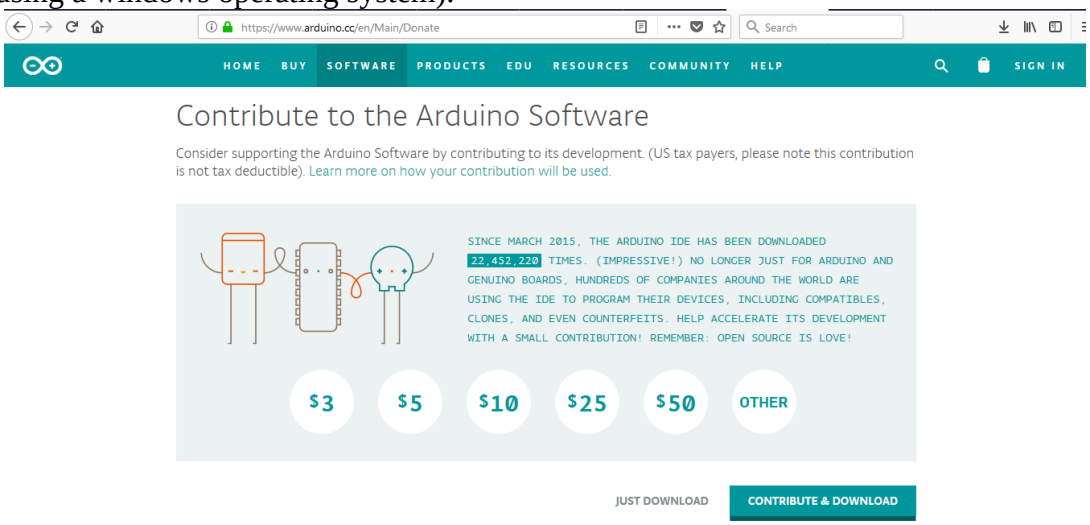
Now you will see a window like below showing the latest and previous version of Arduino IDE: -



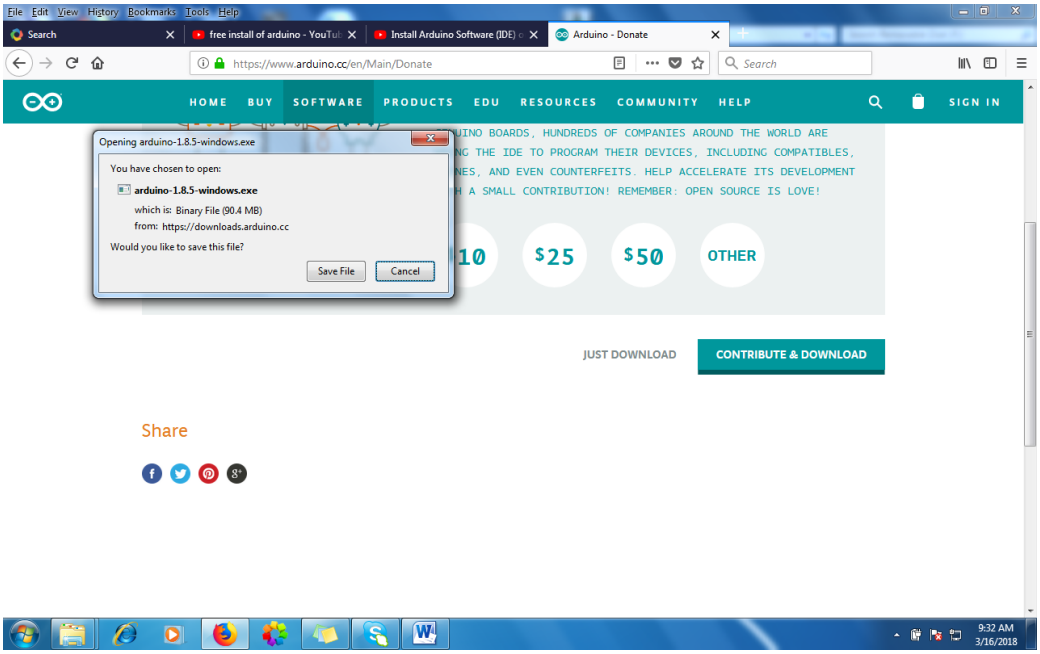
2. Now select the **previous version of current release** option from that page and the following window will come at your desktop.



2. Now select the option **Windows Installer** option from the previous window (considering you are using a windows operating system).

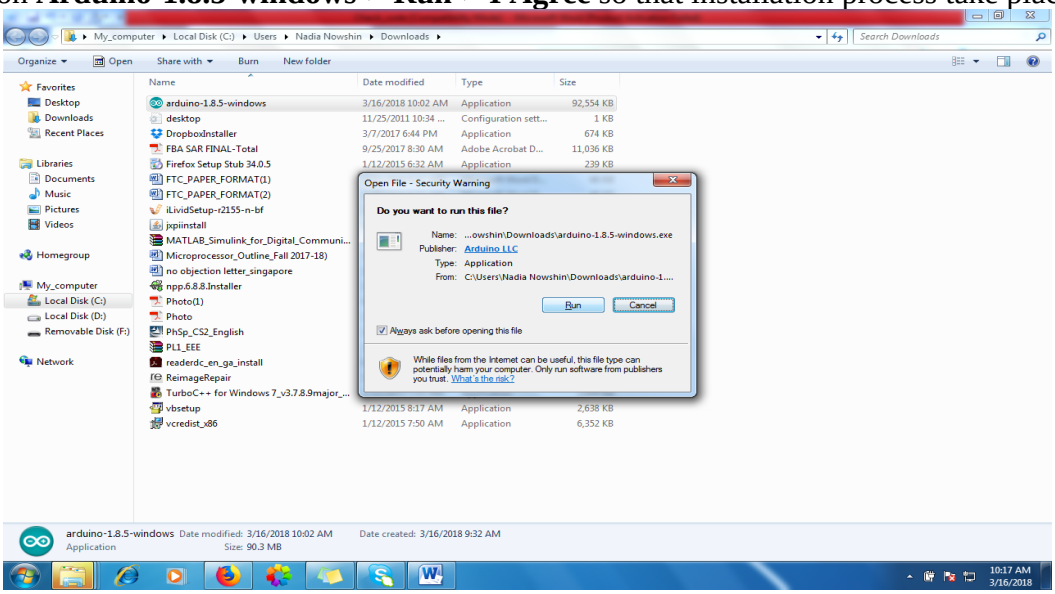


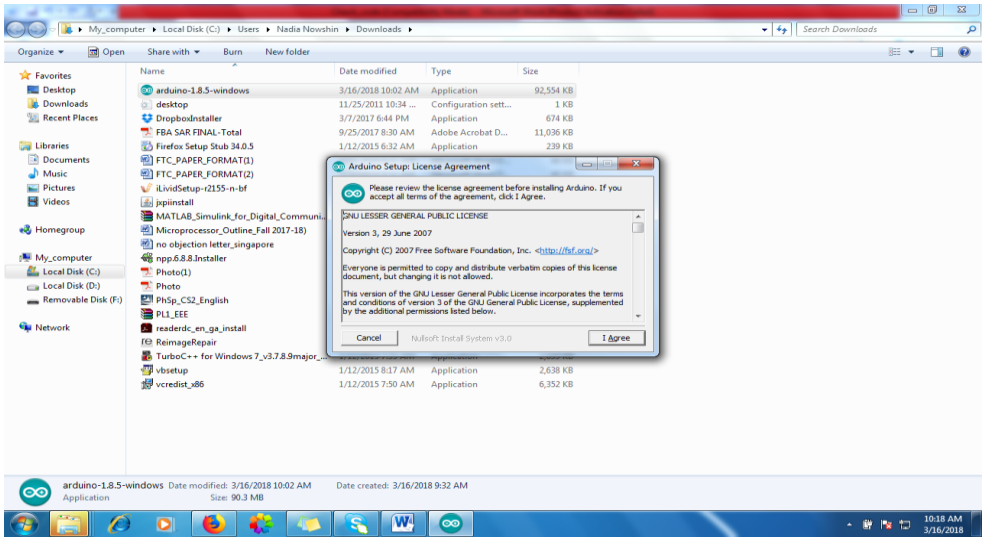
3. Now select the **JUST DOWNLOAD** option from the previous window then the next window will come up and then choose the **Save File** option, so that downloaded software will save in a folder of your PC.



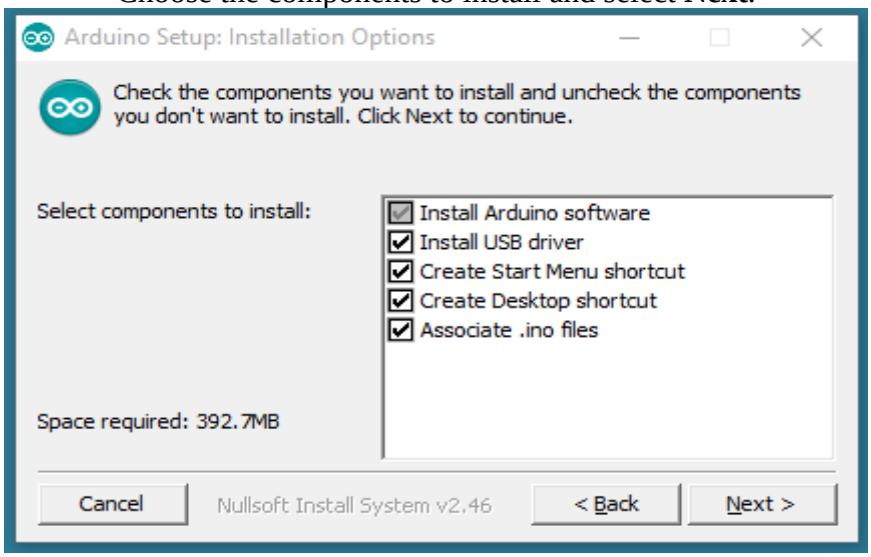
4. When the download finishes, proceed with the installation and through the following steps:-

click on **Arduino-1.8.5-windows-> Run-> I Agree** so that installation process take place.

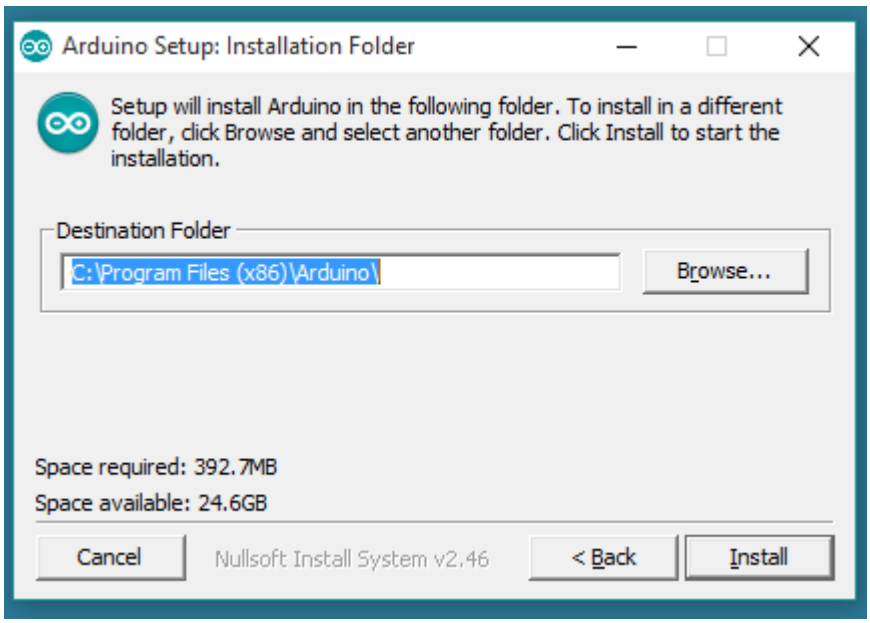


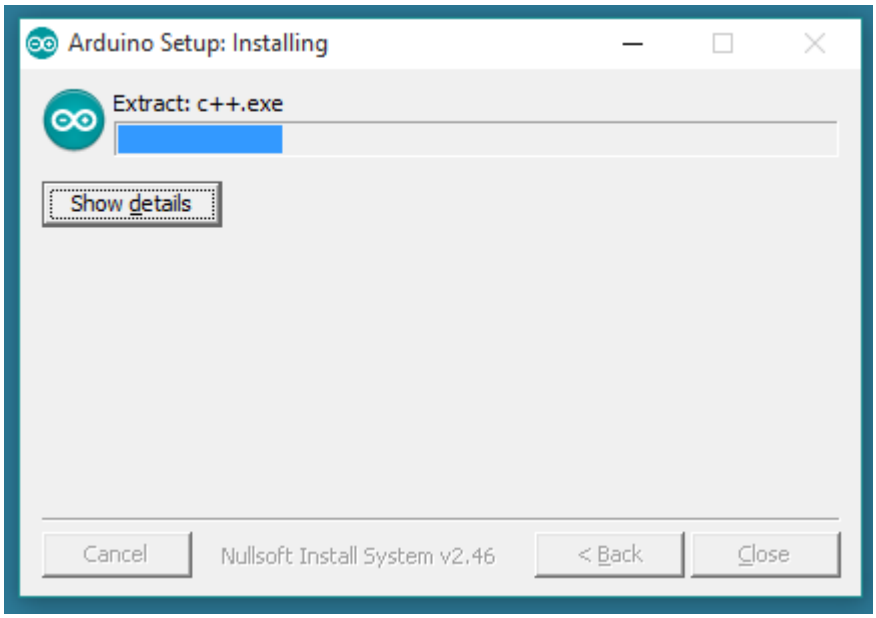


Choose the components to install and select **Next**.

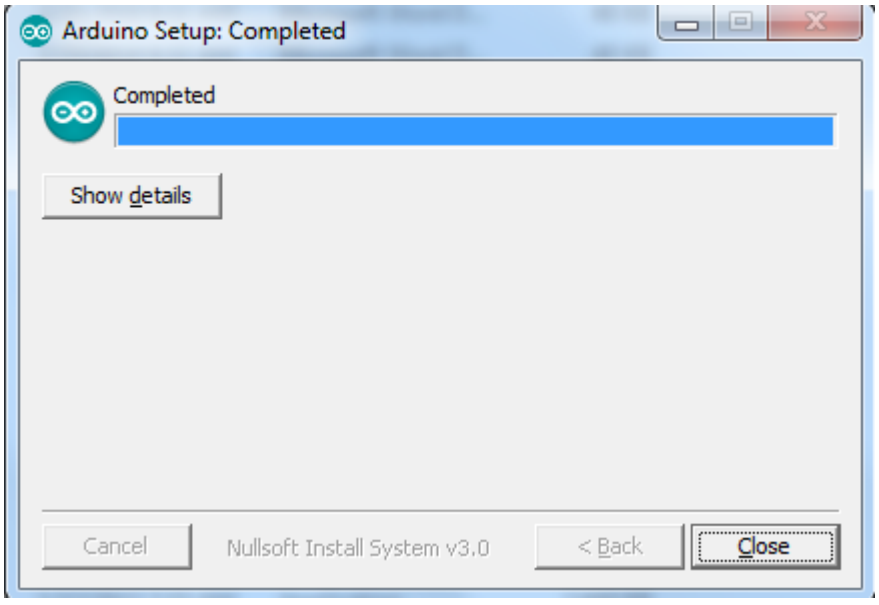


Choose the installation directory (I suggest to keep the default one) and choose **Install**





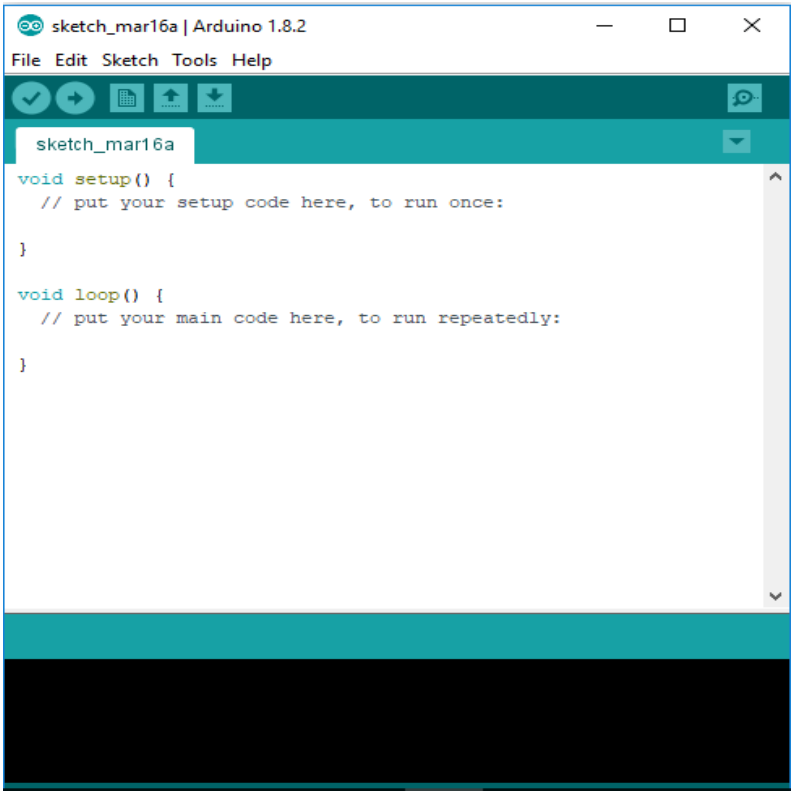
The process will extract and install all the required files to execute properly the Arduino Software (IDE) and don't forget to allow the driver installation process when you get a warning from the operating system.



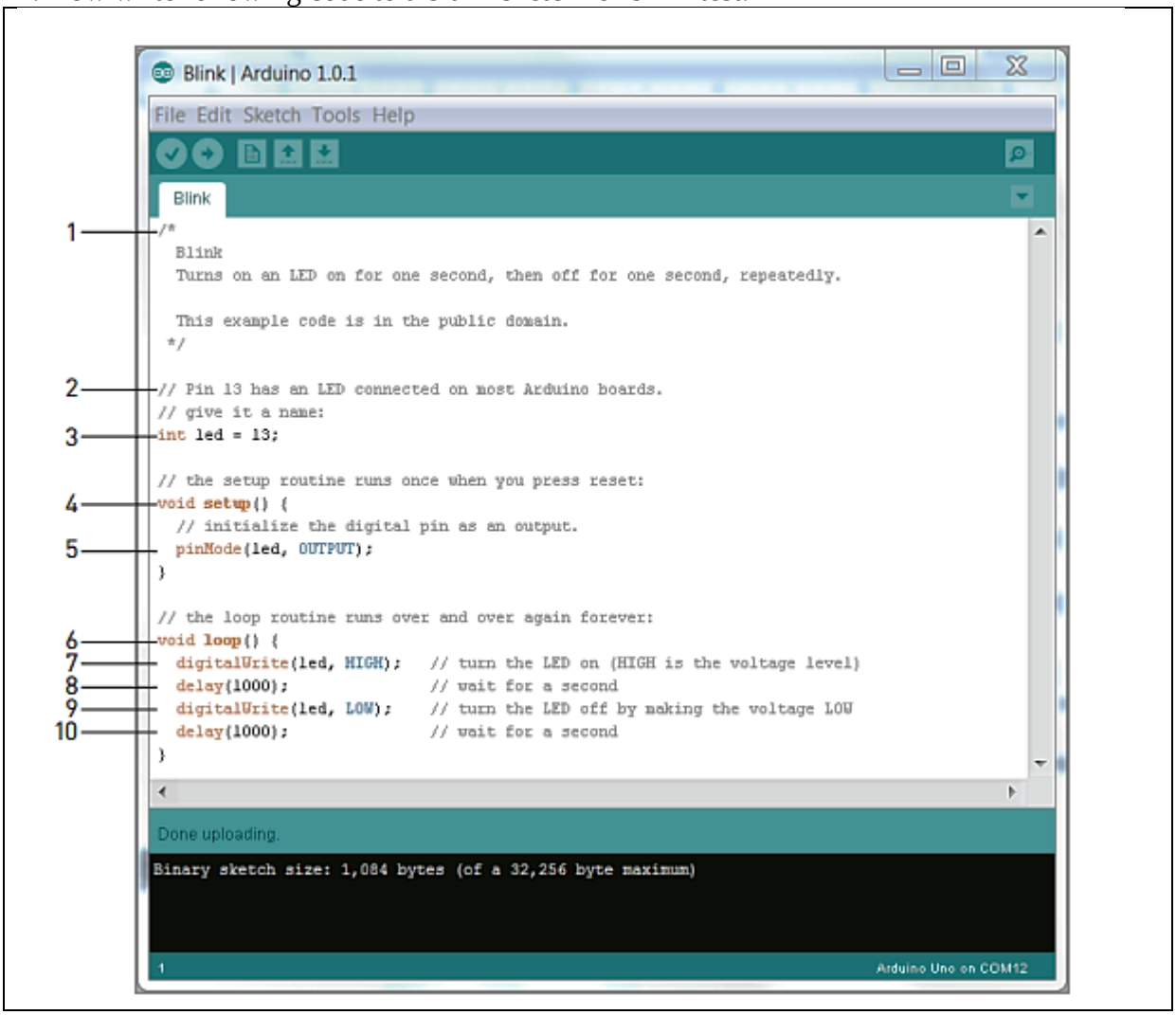
Now your Arduino IDE is ready to start for code writing.

Using Arduino IDE to write code

1. Open the Arduino Uno IDE 1.8.2 and a blank sketch will open. The following window will come up on your PC: -



2. Now write following code to a blank sketch for blink test.



Here's how the code works:

1. This is a multiline comment. Comments are important for documenting a code. Everything written between these symbols will not be compiled or even seen by Arduino. Multiline comments start with /* and end with */.

2. This is a single-line comment. When // is put on any line, the compiler ignores all text after that symbol on the same line. This is great for annotating specific lines of code or for "commenting out" a particular line of code.

3. This code is a variable declaration. A variable is a place in the Arduino's memory that holds information. Variables have different types. In this case, it's of type int, which means it will hold an integer. In this case, an integer variable called led is being set to the value of 13, the pin that the LED is connected to on the Arduino Uno. Throughout the rest of the program, we can simply use led whenever we want to control pin 13. Setting variables is useful because we can just change this one line if we hook up a LED to a different I/O pin later on; the rest of the code will still work as expected.

4. *void setup()* is one of two functions that must be included in every Arduino program. A *function* is a piece of code that does a specific task. Code within the curly braces of the *setup()* function is executed once at the start of the program. This is useful for one-time settings, such as setting the direction of pins, initializing communication interfaces, and so on.

5. The Arduino's digital pins can function as input or outputs. To configure their direction, use the command *pinMode().* This command takes two arguments. An *argument* gives commands information on how they should operate. Arguments are placed inside the parentheses following a command. The first argument to *pinMode* determines which pin is having its direction set. Because the led variable was defined earlier in the program, the command sets the direction of pin 13. The second argument sets the direction of the pin: INPUT or OUTPUT.

6. The second required function in all Arduino programs is *void loop().* The contents of the loop function repeat forever as long as the Arduino is on.

7. *digitalWrite()* is used to set the state of an output pin. It can set the pin to either 5V or 0V. When an LED and resistor is connected to a pin, setting it to 5V will enable to light up the LED. The first argument to *digitalWrite()* is the pin you want to control. The second argument is the value we want to set it to, either HIGH (5V) or LOW (0V). The pin remains in this state until it is changed in the code.

8. The *delay()* function accepts one argument: a delay time in milliseconds. When calling *delay(),* the Arduino stops doing anything for the amount of time specified. In this case, the processor is delaying the program for 1000ms, or 1 second. This results in the LED staying on for 1 second before the next command is executed.

9. Here, *digitalWrite()* is used to turn the LED off, by setting the pin state to LOW.

10. Again, we can set the delay for 1 second to keep the LED in the off state before the loop repeats and switches to the on state again.

CODE for Traffic Control system :

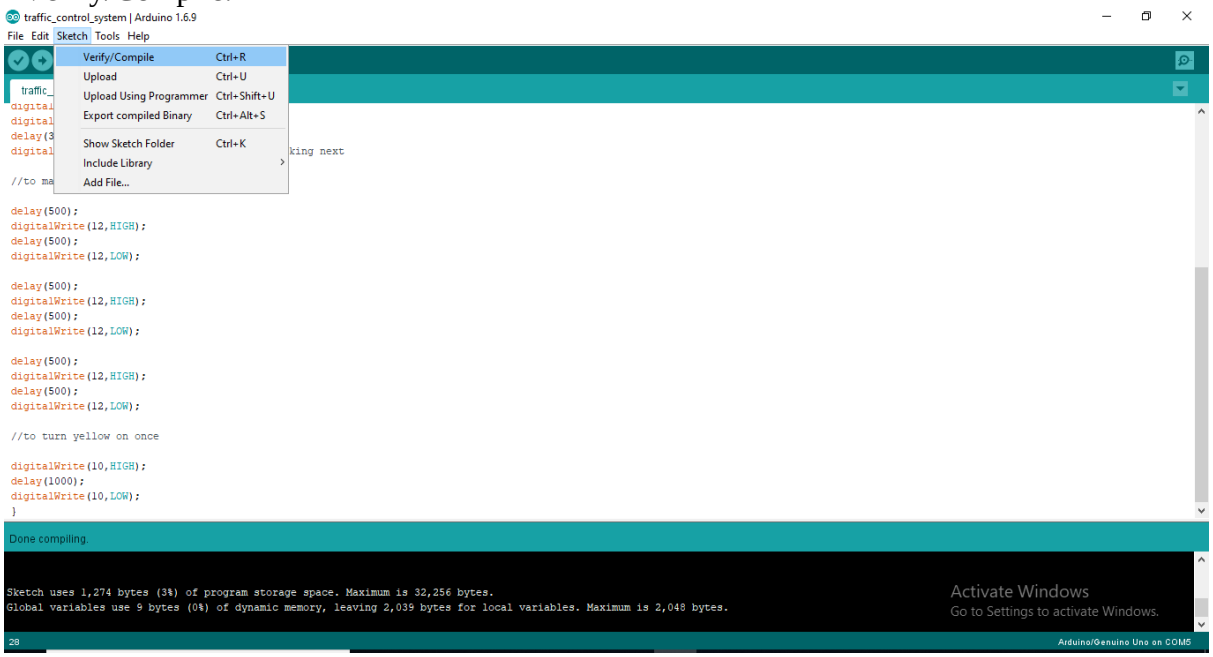
```
void setup() {
  // pin connections for the LED lights

pinMode(8,OUTPUT);
```

```
pinMode(10,OUTPUT);
pinMode(12,OUTPUT);
}

void loop() {
  // turning on voltage at output 8(for red LED)

digitalWrite(8,HIGH);
delay(3000);     // red LED is on
// turning on voltage at output 8(for red LED)

digitalWrite(10,HIGH);
delay(1000);   // yellow LED is on

//for turning off red and yellow and turning on green

digitalWrite(8,LOW);
digitalWrite(10,LOW);
digitalWrite(12,HIGH);
delay(3000);
digitalWrite(12,LOW);    //green is off for blinking next

//to make green on and off for 3 times

delay(500);
digitalWrite(12,HIGH);
delay(500);
digitalWrite(12,LOW);

delay(500);
digitalWrite(12,HIGH);
delay(500);
digitalWrite(12,LOW);

delay(500);
digitalWrite(12,HIGH);
delay(500);
digitalWrite(12,LOW);

//to turn yellow on once

digitalWrite(10,HIGH);
delay(1000);
digitalWrite(10,LOW);
}
```

3. After the writing the code you have to save the sketch, go to File->Save As->give a File name(Traffic_control_light)-> Select Save.

N.B. After saving your code a sketch file with the desired file name will be stored in a folder with same file name.

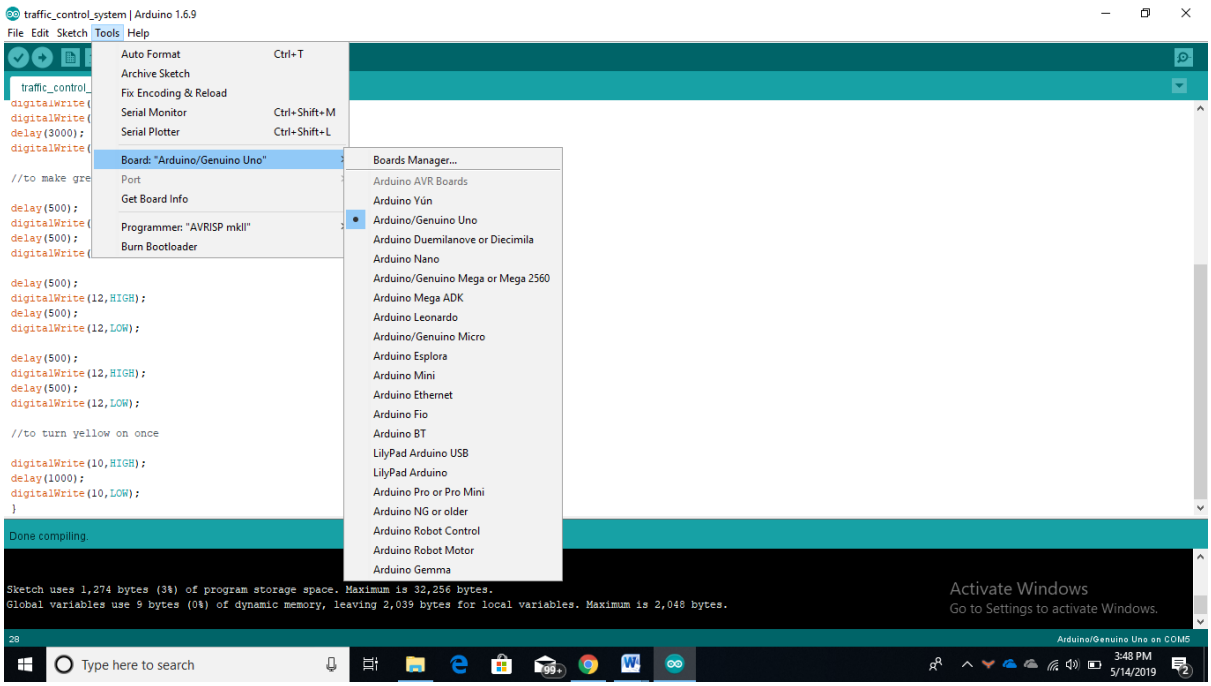
4.Now you need to verify/compile your code to find out and correct the errors, go to Sketch->Verify/Compile.



5. After compiling is done you need to upload your code into the Arduino Uno board. To upload the code connect you Arduino Uno R3 board to your PC with USB cable. Before uploading the code select the board type and port at your Arduino IDE, go to

- Tools-> Board:"Arduino/Genuino Uno" -> Arduino/Genuino Uno.
- Tools->Ports-> COMx

After you have selected the board and port select the upload option at the Arduino IDE (see the following figure) to upload the code.



Familiarization with the Arduino Commands

In this section,

1. We will learn about some common Arduino commands that will help write code.
2. This section also focuses on the standard Library functions associated with the IDE.

a) ****pinMode(X, INPUT) or pinMode(X,OUTPUT) ****
this command will configure any pin at the Arduino board as either input/output.

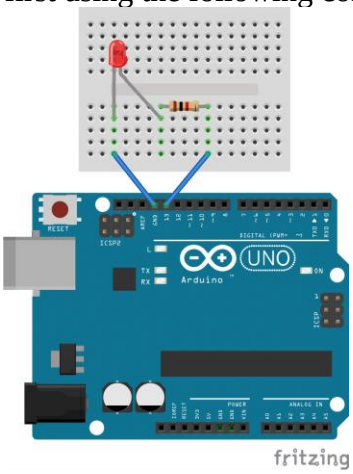
b) ****digitalWrite(X, LOW) or digitalWrite(X, HIGH) ****
this command will provide a HIGH/LOW value to any digital output pin at the Arduino board .

Setting up the Circuit The main task of our lab is to understand and implement a traffic control system after understanding to blink a LED light.

Experimental Procedure:

You have all you need to start your Lab work.

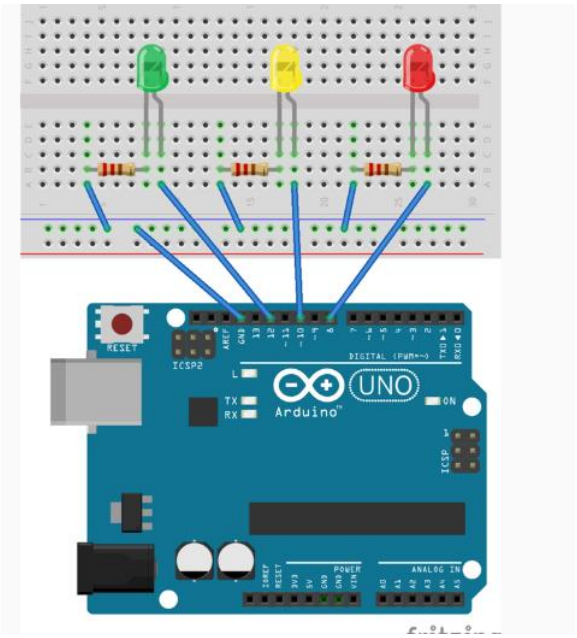
1) Make the circuit first using the following connection system between all the elements



2) Understand the normal operation of a traffic control system .

3) Write the code accordingly and see the output operation

2) Traffic control system
Screenshot of ckt & copy code from TinkerCAD



How It Works:

Turn on the red signal (pin 8)
wait for 3 sec
Turn on the yellow signal (pin 10)
wait 1 sec
turn off red signal (pin 8)
turn off yellow signal (pin 10)
turn on green signal (pin 12)
wait 3 sec
turn off green signal (pin 12)
turn on green signal (pin 12)
wait 0.5 sec
turn off green signal (pin 12)
turn on green signal (pin 12)
wait 0.5 sec
turn off green signal (pin 12)
turn on green signal (pin 12)
wait 0.5 sec
turn off green signal (pin 12)
turn on yellow signal (pin 12)
wait 1 sec
turn off yellow signal (pin 12)

In own words

In traffic system, red light turns on after green light and yellow light is turned off, so firstly when we need to stop the vehicles, red light is used to turn on for 3 sec and then yellow light is on for 1 sec . Both red and yellow lights are off to turn green light on afterwards for 3 sec.

To turn on red light again, the green light needs to blink for 0.5 secs for 3 times and then yellow light is on for 1 sec and after turning off yellow light, we can again see that red light is on.

4) Let us rewrite the code again for same operation and same circuit by defining the delays.

```
#define RED_PIN 8
#define YELLOW_PIN 10
#define GREEN_PIN 12

int red_on = 3000;
int red_yellow_on = 1000;
int green_on = 3000;
int green_blink = 500;
int yellow_on = 1000;

void setup() {
  //ports for connecting LEDs
  pinMode(RED_PIN, OUTPUT);
  pinMode(YELLOW_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);

}

void loop() {
  //turning on voltage at output red LED
  digitalWrite(RED_PIN, HIGH);
  //to make red LED on
  delay(red_on);
  //to turn yellow LED on
  digitalWrite(YELLOW_PIN, HIGH);
  delay(red_yellow_on);

  //turning off RED_PIN and YELLOW_PIN, and turrning on greenLEd
  digitalWrite(RED_PIN, LOW);
  digitalWrite(YELLOW_PIN, LOW);
  digitalWrite(GREEN_PIN, HIGH);
  delay(green_on);
  digitalWrite(GREEN_PIN, LOW);

  //for turning green Led on and off for 3 times
 for(int i = 0; i < 3; i = i+1)
  {
    delay(green_blink);
    digitalWrite(GREEN_PIN, HIGH);
    delay(green_blink);
    digitalWrite(GREEN_PIN, LOW);
  }
    //for turning on yellow LED
    digitalWrite(YELLOW_PIN, HIGH);
    delay(yellow_on);
    digitalWrite(YELLOW_PIN, LOW);

}
```

Questions for report writing:

1) Include all codes and scripts into lab report following the writing template mentioned in appendix A of Laboratory Sheet Experiment 2.

Reference(s):

1) https://www.arduino.cc/.
2) https://www.coursera.org/learn/arduino/lecture/ei4ni/1-10-first-glance-at-a-program
3) Jeremy Blue; Exploring Arduino: Tools and Techniques for Engineering Wizardry