**1. INTRODUCTION**

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a 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 physical board.

A pre-assembled Arduino board includes a microcontroller, which is programmed using Arduino programming language and the Arduino development environment. In essence, this platform provides a way to build and program electronic components. Arduino programming language is a simplified from of C/C++ programming language based on what Arduino calls "sketches," which use basic programming structures, variables and functions. These are then converted into a C++ program.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

### Why Arduino?

Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia’s BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
* **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of module in order to understand how it works and save money.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

**2. REQUIREMENT SPECIFICATIONS**

**2.1 SOFTWARE REQUIREMENTS**

**2.1.1 ARDUINO IDE**

**IDE:**

The Arduino integrated development environment is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

### Sketch

A program written with the Arduino IDE is called a *sketch*. Sketches are saved on the development computer as text files with the file extension **.ino**. Arduino Software (IDE) pre-1.0 saved sketches with the extension **.pde**.

A minimal Arduino C/C++ program consist of only two functions:

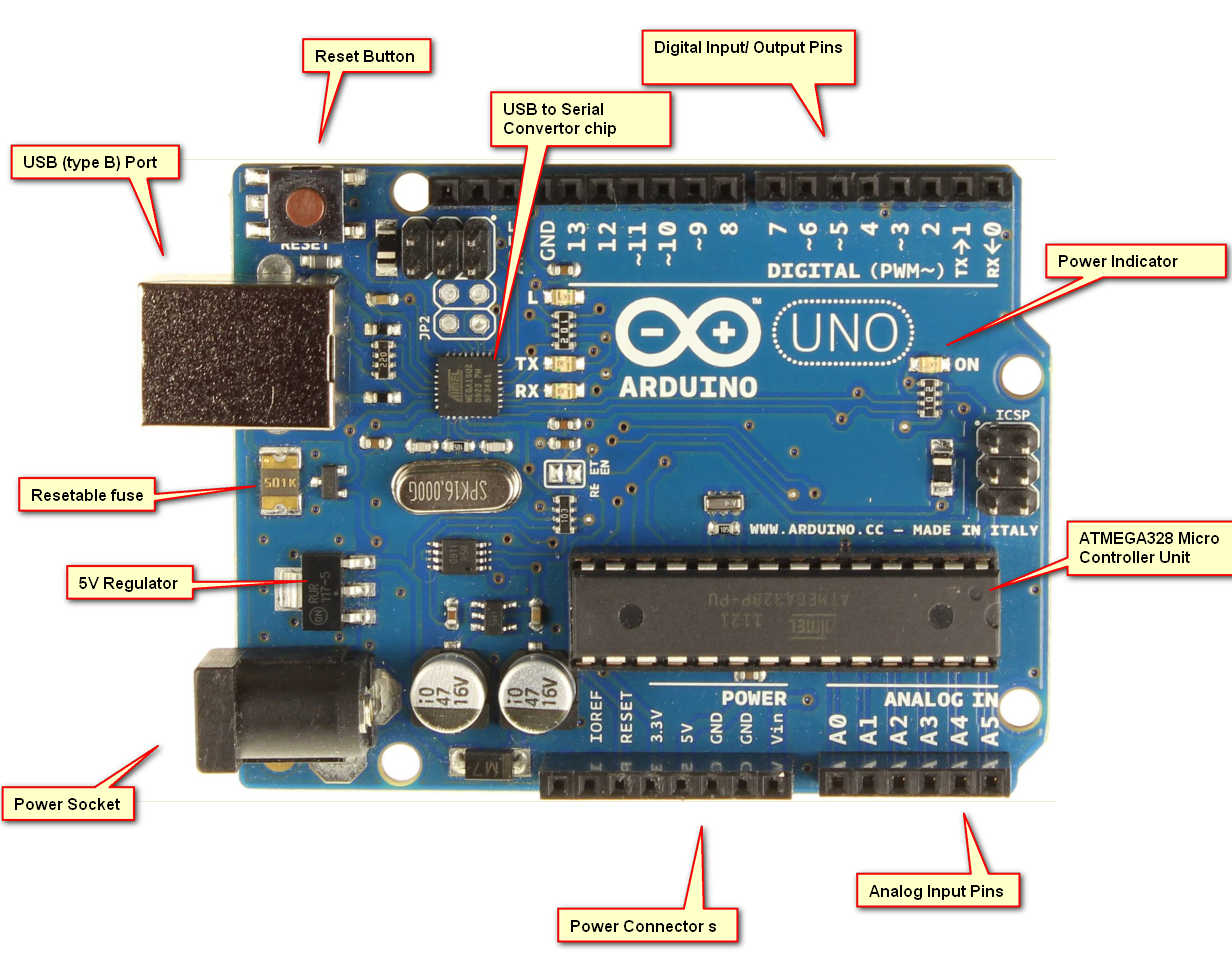
* setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
* loop(): After setup() function exits (ends), the loop() function is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

**2.2 HARDWARE REQUIREMENTS**

#### ****2.2.1 ARDUINO UNO BOARD****

**Arduino Uno** is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header 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. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

**"Uno"** means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.



*ARDUINO UNO*

**3.NEED FOR TRAFFIC SIGNALS:**

The use of personal vehicles is very common now a days and a result, the number of vehicles on the roads are exponentially increasing. Roads without any supervision or guidance can lead in to traffic congestions and accidents.

Traffic Lights or Traffic Signals are signalling devices that are used to control the flow of traffic. Generally, they are positioned at junctions, intersections, ‘X’ roads, pedestrian crossings etc. and alternate the priority of who has to wait and who has to go.

The traffic lights will provide instructions to the users (drivers and pedestrians) by displaying lights of standard color. The three colors used in traffic lights are Red, Yellow and Green.

The system must be used to control the traffic lights for smooth and safe movement of traffic. These control systems consists of electro mechanical controllers with clockwork mechanisms or modern solid state computerised systems with easy setup and maintenance.

In this project, an Arduino based Traffic Light Controller system is designed. It is a simple implementation of traffic lights system but can be extended to a real time system with programmable timings, pedestrian lighting etc.

Having personal vehicles is very common today and a result, vehicles on the roads are exponentially increasing in numbers. Roads without any traffic lights or guidance can lead in to traffic congestions or could lead to accidents.The traffic light system provides instructions to the users (drivers and pedestrians) by displaying lights of standard colour on four cross streets.

The colours used in traffic lights are Red, Yellow and Green for stop, slow and go respectively. The system is programmed to control the traffic lights for smooth and safe movement of vehicle traffic. The system consists of electro mechanical controllers with modern solid state computerized systems which has easy setup and maintenance. This project is an Arduino board based Traffic Light system.

Nowadays traffic lights are quite complex within and have various kinds of sensors, timers, and also traffic monitoring systems that helps to control the traffic.

**4.THREE WAY TRAFFIC CONTROLLER**

**COMPONENTS:**

1. Arduino UNO With Ide Cable
2. 1KΩ Resistor X 3
3. Red LEDs X 3
4. Yellow LEDs X 3
5. Green LEDs X 3
6. Connecting wires
7. Bread board

**COMPONENTS DESCRIPTION:**

**ARDUINO UNO:**

The main part of the Traffic Light Controller is the controller itself. Arduino UNO will serve the purpose in this project to handle all the switching of the LEDs and controlling their timings.

**LED's:**

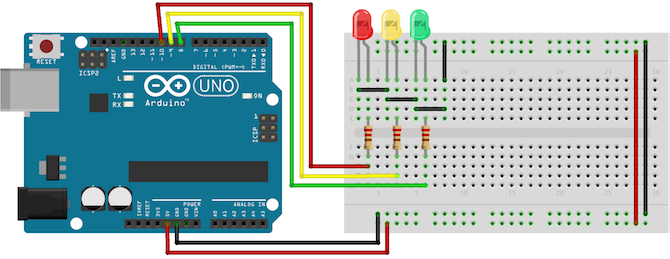
The LEDs used in the project are basic 5mm LEDs of Red, Yellow and Green colors. The maximum current that can be allowed through these LEDs (Red, Yellow and Green in particular) is 20mA. (Foe Blue LED, the maximum current can be up to 30mA).

**1KΩ RESISTOR:**

The resistor in the circuit servers the useful function of limiting the amount of current sent to the LED.there is a need for limiting because LED’s eat up current like a kid eats candy.They try to gobble up as much as you give them.But LED’s run into a problem 🡪they burn themselves if they draw too much current.

**WORKING:**

Let’s start small. A basic, single traffic light is a good place to start. Here’s the circuit:



*Circuit Diagram 1*

Connect the anode (long leg) of each LED to digital pins eight, nine, and ten (via a 220? resistor). Connect the cathodes (short leg) to Arduino ground.

**The Code**

1.Start by defining variables so that we can address the lights by name rather than a number.

int red = 10;

int yellow = 9;

int green = 8;

2.Next, let’s add the setup function, where’ll we configure the red, yellow and green LEDs to be outputs.

void setup(){

pinMode(red, OUTPUT);

pinMode(yellow, OUTPUT);

pinMode(green, OUTPUT);

}

3 .Now for the difficult part -– the actual logic of a traffic light. Create a separate function for changing the lights

void loop(){

changeLights();

delay(15000);

}

void changeLights(){

// green off, yellow on for 3 seconds

digitalWrite(green, LOW);

digitalWrite(yellow, HIGH);

delay(3000);

// turn off yellow, then turn red on for 5 seconds

digitalWrite(yellow, LOW);

digitalWrite(red, HIGH);

delay(5000);

// red and yellow on for 2 seconds (red is already on though)

digitalWrite(yellow, HIGH);

delay(2000);

// turn off red and yellow, then turn on green

digitalWrite(yellow, LOW);

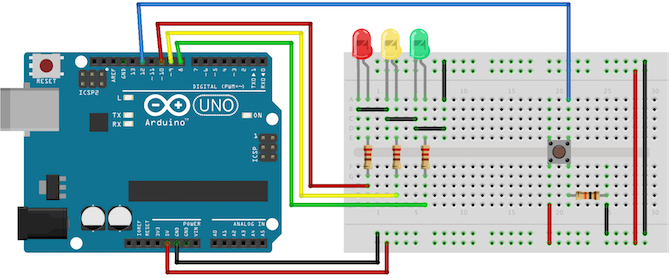
digitalWrite(red, LOW);

digitalWrite(green, HIGH);

delay(3000);

}

Now that you know the basics, let’s improve it. Add in a pushbutton for pedestrians to change the light whenever they like:



*Circuit Diagram 2*

Notice how the traffic light is exactly the same as the previous example. Connect the button to digital pin 12. You’ll notice that the switch has a high-impedance 10k? resistor attached to it, and you may be wondering why. This is called a pull-down resistor. It’s a difficult concept to grasp at first, but bear with me.

A switch either lets the current flow, or doesn’t. This seems simple enough, but in a logic circuit, the current should be always flowing in either a high or low state (remember – 1 or 0, high or low). You might assume that a pushbutton switch that isn’t actually being pushed would be defined as being in a low state, but in fact it’s said to be ‘floating’, because no current is being drawn at all.

In this floating state, it’s possible that a false reading will occur as it fluctuates with electrical interference. In other words, a floating switch is giving neither a reliable high, nor low state reading. A pull down resistor keeps a small amount of current flowing when the switch is closed, thereby ensuring an accurate low state reading. In other logic circuits, you may find a pull-up resistor instead – this works on the same principle, but in reverse, making sure that particular logic gate defaults to high.

Now, in the loop part of the code, instead of changing the lights every 15 seconds, we’re going to read the state of the pushbutton switch instead, and only change the lights when it’s activated.

Now let's see our actual project of 3 way traffic control

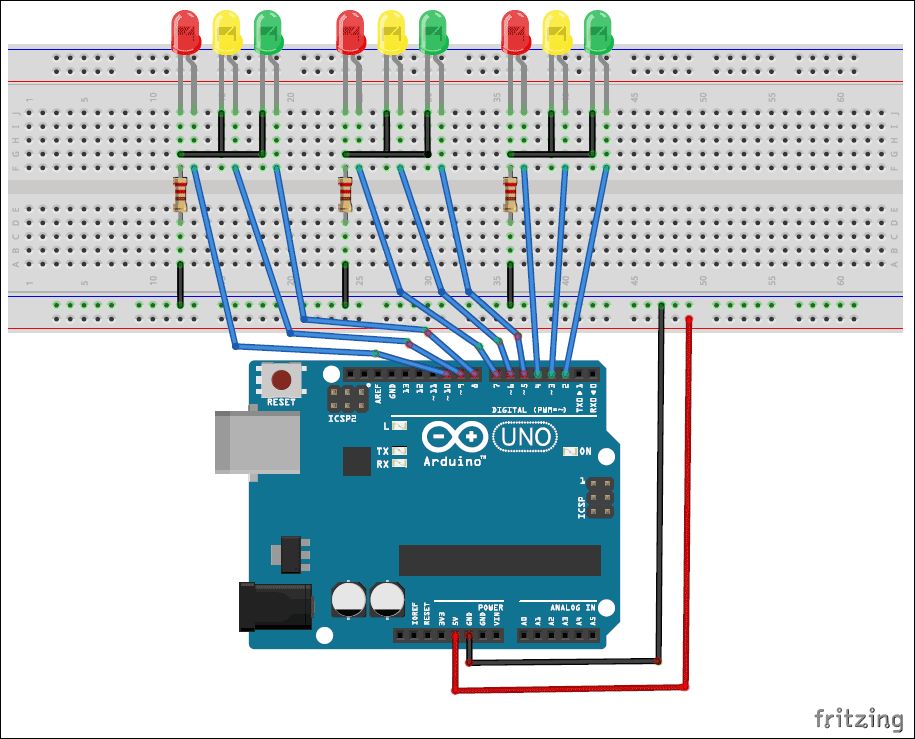
**CIRCUIT DESIGN:**

Since the project is a traffic light controller, the circuit consists of many LEDs (09 as a matter of fact) as we are implementing traffic lights at a 3 way intersection. The project is a simple representation of traffic light controller and hence no other extra components are used.

We need three LEDs of Red, Yellow and Green colors at each intersection. The intersection is divided in to four lanes: Lane1, Lane 2 and Lane 3

All the LEDs are connected to the Arduino UNO’s digital I/O pins through respective current limiting resistors of 1KΩ.

All the connections are made as per the circuit diagram. The complete wiring diagram of the circuit is shown below.



*Traffic Light Controller*

The real time traffic light controller is a complex piece of equipment which consists of power cabinet, main controller or processor, relays, control panel with switches or keys, communication ports etc.

In this project, a simple traffic light system for a 3 way intersection is implemented using Arduino UNO. Although it is not the ideal implementation for real life scenarios, it gives an idea of the process behind the traffic light control system

The aim of the project is to implement a simple traffic light controller using Arduino UNO, where the traffic is controlled in a pre-defined timing system. The working of the project is very simple and is explained below

In that, first the Lane 1 gets its Green light turned. Hence, in all the other Lanes, their corresponding Red lights are turned on. After a time delay of predefined time say 5 seconds, the Green light in the Lane 3 must be turned on and the Green light in the Lane 1 must be turned off.

As a warning indicator, the Yellow light in Lane 1 is tuned on indicating that the red light is about to light up. Similarly, the yellow light in the Lane 3 is also turned as an indication that the green light about to be turned on.

The yellow lights in Lanes 1 and 3 are turned for a small duration say 2 seconds after with the red light in the Lane 1 is turned on and green light in Lane 3 is also turned on.

The green light in Lane 3 is also turned on for a predefined time and the process moves forward to Lane 2.

The system then loops back to Lane 1 where the process mentioned above will be repeated all over again.

**LIMITATIONS:**

The project is not suitable for actual implementation but just a demonstration of the process behind the system.

Real time traffic light controller systems are generally run time programmable i.e. the operator (usually a policeman) can change the timings of each lane as per the intensity of the traffic in each lane.

There will also be a provision for either manual operation or pre-programmed operation.

**APPLICATIONS:**

A simple traffic light controller is implemented in this project with a real chance of expansion.

An external memory can be interface with the main controller so that the timings are not fixed during its programming but rather can be programmed during operation.

An efficient traffic light controller system will include a pedestrian signaling system.

**5. REFERENCES**

1.https://www.electronicshub.org

2.http://geeksforgeeks.com

3.NMIT library.