Name: Gagan Sanjay Jain
Branch: IT
Year: TY
Roll No: 11
Sub: IOT LAB ITL-504
Year: 2023-2024

Experiment No: 2

Aim of the Experiment:

Getting started with Arduino-Board, Installation of Arduino IDE, Basic programs using Arduino-Uno.

Objective:

1. To study the installation of Arduino Programming.
2. To understand the Arduino IDE Environment.
3. To carry out basic programs using Arduino-Uno.
    a. Blinking the LED
    b. Fade in and fade-out the LED
    c.

Software/Hardware Required:

Theory:

1. The Function of the Arduino Board:

The Arduino board is a popular open-source electronics platform designed for building interactive projects and prototypes. It serves as the central component of the Arduino system and plays a crucial role in enabling various functionalities. The primary functions of the Arduino board are as follows:

Microcontroller: At the heart of the Arduino board is a microcontroller, which is a small computer on a single integrated circuit. The microcontroller is responsible for executing the program or code uploaded to the Arduino, controlling input and output operations, and managing other tasks.

Input/Output (I/O) Pins: Arduino boards come with a set of digital and analog input/output pins. These pins allow the board to interact with various sensors, actuators, and other electronic components. Digital pins can be used for reading digital signals (high or low), while analog pins can read analog voltages within a specific range.

Programming Port: The Arduino board usually features a USB port or another interface that allows you to connect it to a computer for programming. Using the Arduino Integrated Development Environment (IDE), you can write, compile, and upload code to the board via this port.

Power Supply: Arduino boards can be powered through a USB connection, a DC power adapter, or an external power source. They often have built-in voltage regulators to provide a stable and regulated voltage to the components.

Clock Crystal: The Arduino board includes a clock crystal that provides precise timing for the microcontroller, allowing it to carry out operations accurately.

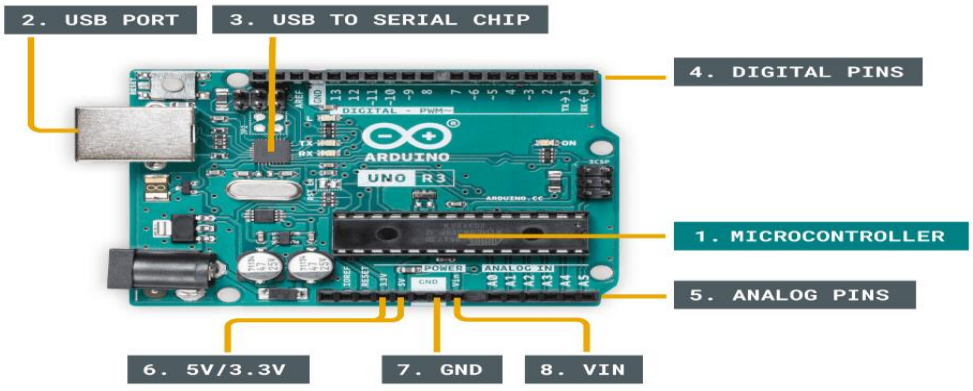
2. Different Types of Arduino Boards:

Here are some of the common Arduino boards:
Arduino Uno: The Arduino Uno is one of the most popular and widely used Arduino boards. It features an ATmega328P microcontroller, a USB interface for programming and communication, 14 digital I/O pins (6 of which can be used as PWM outputs), 6 analog input pins, and various other features. It is an excellent choice for beginners and most projects.
Arduino Nano: The Arduino Nano is a smaller version of the Arduino Uno, offering similar capabilities but in a more compact form factor. It is popular for projects with space constraints.
Arduino Mega: The Arduino Mega is designed for projects that require a large number of I/O pins. It uses the ATmega2560 microcontroller and provides 54 digital I/O pins and 16 analog input pins.

Arduino Leonardo: The Arduino Leonardo is based on the ATmega32U4 microcontroller and differs from other Arduino boards as it can act as a USB human interface device (HID), allowing it to emulate a mouse or keyboard. This feature makes it useful for projects involving computer interactions.
Arduino Due: The Arduino Due is based on the more powerful Atmel SAM3X8E ARM Cortex-M3 CPU. It operates at 3.3V (unlike most Arduino boards, which run at 5V) and offers a higher number of pins, enhanced performance, and additional features.

Arduino Hardware:

Anatomy of an Arduino Board:



1. Microcontroller - this is the brain of an Arduino, and is the component that we load programs into. Think of it as a tiny computer, designed to execute only a specific number of things.

2. USB port - used to connect your Arduino board to a computer.
3. USB to Serial chip - the USB to Serial is an important component, as it helps translating data that comes from e.g. a computer to the on-board microcontroller. This is what makes it possible to program the Arduino board from your computer.
4. Digital pins - pins that use digital logic (0,1 or LOW/HIGH). Commonly used for switches and to turn on/off an LED.
5. Analog pins - pins that can read analog values in a 10 bit resolution (0-1023).
6. 5V / 3.3V pins- these pins are used to power external components.

Basic Operation :

The program that is loaded to the microcontroller will start execution as soon as it is powered. Every program has a function called "loop". Inside the loop function, you can for example:

- Read a sensor.
- Turn on a light.
- Check whether a condition is met.
- All of the above.

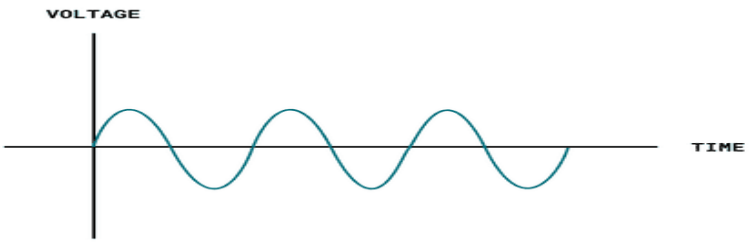
Circuit Basics :

Circuits consist of at least one active electronic component, and a conductive material, such as wires, so that current can pass through. When working with an Arduino, you will in most cases build a circuit for your project.

Electronic Signals :

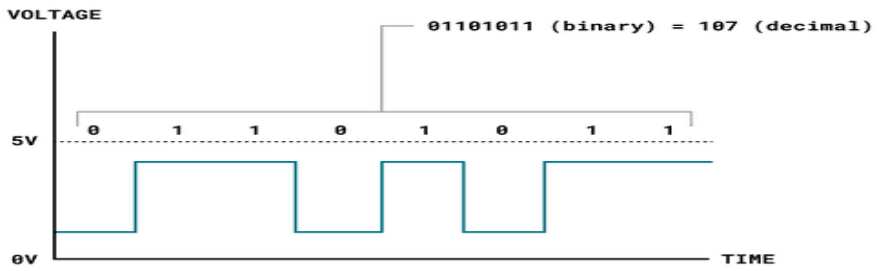
All communication between electronic components are facilitated by **electronic signals.** There are two main types of electronic signals: **analog & digital**.

Analog Signal :



An analog signal is generally bound to a range. In an Arduino, that range is typically 0-5V, or 0-3.3V.

Digital Signal :



A digital signal works a bit differently, representing only two binary states (0 or 1) that are read as high or low states in the program. This is the most common signal type in modern technology.
You can easily read and write digital signals on an Arduino, which is useful to for example read button states, or to turn something on or off.

Sensors & Actuators :

A sensor, in simple terms, is used to *sense* its environment, meaning it records a physical parameter, for example temperature, and converts it into an electronic signal.
An actuator, in simple terms, is used to *actuate* or *change a physical state*. Some examples are:

- A light (such as an LED).
- A motor.
- A switch.

Actuators converts electrical signals into e.g. radiant energy (light) or mechanical energy (movement).

Serial Communication Protocols :

There are several serial communication protocols that uses the aforementioned digital signals to send data. The most common are **UART, SPI & I²C**.

Memory :

The "standard" Arduino typically has two memories: SRAM and Flash memory. The SRAM (Static Random-Access Memory) is used to for example store the value of a variable (such as the state of a Boolean). When powered off, this memory resets. The Flash memory is primarily used to store the main program, or the instructions for the microcontroller.

Embedded Components :

Many new Arduino boards come equipped with **embedded sensors**. For example, the Nano 33 BLE Sense has 7 embedded sensors, but is only **45x18mm** (the size of a thumb). These are all connected via the I²C protocol as mentioned above, and has a unique address.

Internet of Things (IoT) :

Most modern Arduino boards now come equipped with a radio module, designed to communicate wirelessly. There are several different ones: Wi-Fi, Bluetooth®, LoRa®, GSM, NB-IoT and more. Each are designed to communicate using the various technologies available on the market.

Arduino API:

The Arduino API, aka the "Arduino Programming Language", consists of several functions, variables and structures based on the C/C++ language.

1. Main Parts :

Functions: for controlling the Arduino board and performing computations. For example, to read or write a state to a digital pin, map a value or use serial communication.

Variables: the Arduino constants, data types and conversions.

Structure: the elements of the Arduino (C++) code, such as *sketch,control structure,arithmetic operators,comparison operators.*

2. Program Structure :

The absolute minimum requirement of an Arduino program is the use of two functions: void setup() and void loop().The "void" indicates that nothing is returned on execution.

3. The "Sketch" :

In the Arduino project, a program is referred to as a "sketch". A sketch is a file that you write your program inside. It has the .ino extension, and is always stored in a folder of the same name.
The folder can include other files, such as a header file, that can be included in your sketch.

4. Libraries :

Arduino libraries are an extension of the standard Arduino API, and consists of thousands of libraries, both official and contributed by the community.Libraries simplify the use of otherwise complex code, such as reading a specific sensor, controlling a motor or connecting to the Internet. Instead of having to write all of this code yourself, you can just install a library, include it at the top of your code, and use any of the available functionalities of it. All Arduino libraries are open source and free to use by anyone.

5. Core Specific API :

Every Arduino board requires a "core", or "package", that needs to be installed in order to program it. All packages contain the standard Arduino API, but also a specific API that can only be used with specific boards.

Arduino Software Tools :

The Arduino IDE, as it is commonly referred to, is an integrated development environment.

A Typical Workflow :

To upload code to an Arduino board using the IDE, one typically does the following:
1. Install your board - this means installing the right "package" for your board. Without the package, you can simply not use your board. Installing is done directly in the IDE, and is a quick and easy operation.

2. Create a new sketch - a sketch is your main program file. Here we write a set of instructions we want to execute on the microcontroller.
3. Compile your sketch - the code we write is not exactly how it looks like when uploaded to our Arduino: compiling code means that we check it for errors, and convert it into a binary file (1s and 0s). If something fails, you will get this in the error console.
4. Upload your sketch - once the compilation is successful, the code can be uploaded to your board. In this step, we connect the board to the computer physically, and select the right serial port.
5. Serial Monitor (optional) - for most Arduino projects, it is important to know what's going on on your board. The Serial Monitor tool available in all IDEs allow for data to be sent from your board to your computer.

Arduino IDE 1.8.x : For what is now considered the "legacy" editor, the Arduino IDE 1.8.X, or "Java IDE", is the editor that was first released back when Arduino started.

Arduino IDE 2 : For what is now considered the "legacy" editor, the Arduino IDE 1.8.X, or "Java IDE", is the editor that was first released back when Arduino started.

Arduino IoT Cloud : The Arduino IoT Cloud allows you to configure, program and control/monitor your devices - all in one web based application. With the use of things, or your "digital twin", you can control and monitor variables directly from dashboards. The service also supports webhooks and integrations with other services, such as Amazon Alexa.The cloud is made for anyone to use, and it does not require much previous experience to get started.

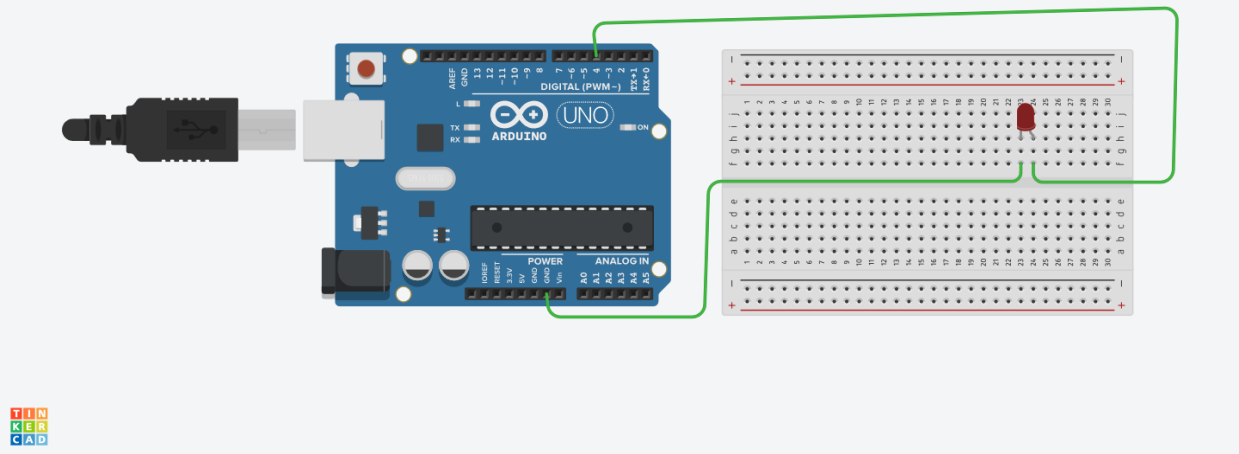
Web Editor : The Arduino Web Editor is an online IDE, part of the Arduino Cloud suite. Similar in function, this editor is completely web based, with online storage among other features. To use the Web Editor, you will need to register an Arduino account.

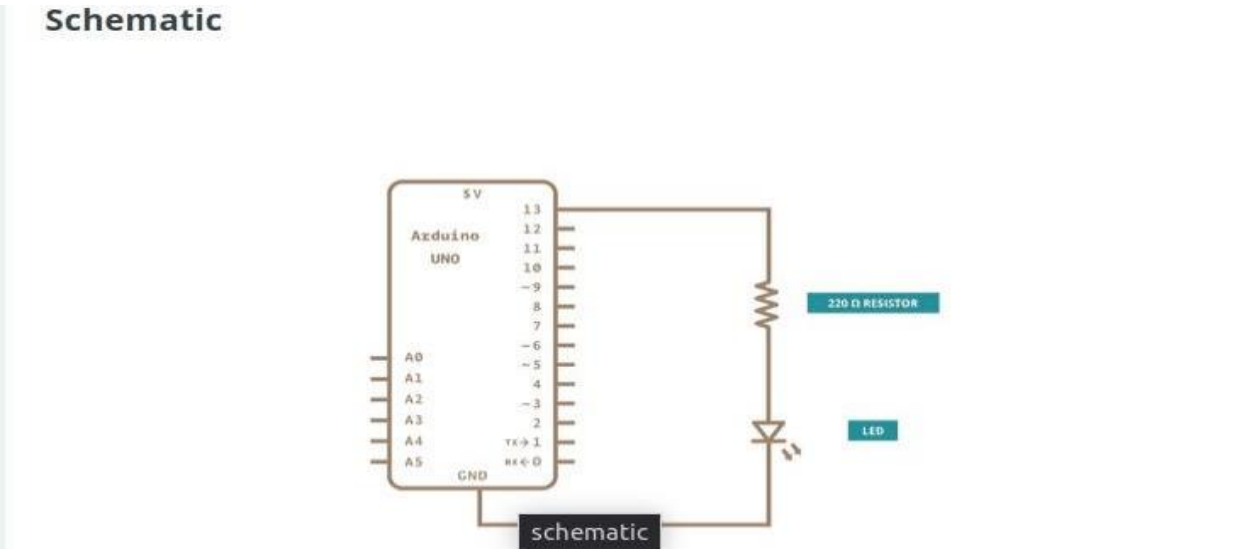
Library Manager : Every version of the IDE has a library manager for installing Arduino software libraries. Thousands of libraries, both official and contributed libraries, are available for direct download. Code examples for each library is made available on download.

Arduino CLI : The Arduino CLI is a command line tool that can be used to compile and upload code to your board. It has no visual UI, but is very useful for automation. It is designed for more advanced users.A proper use of the CLI can speed up your development time by far, as any operation is executed much faster than in the regular IDE.

Blinking the LED and Fade-in and fade-out the LED :

- **Circuit Diagram:**

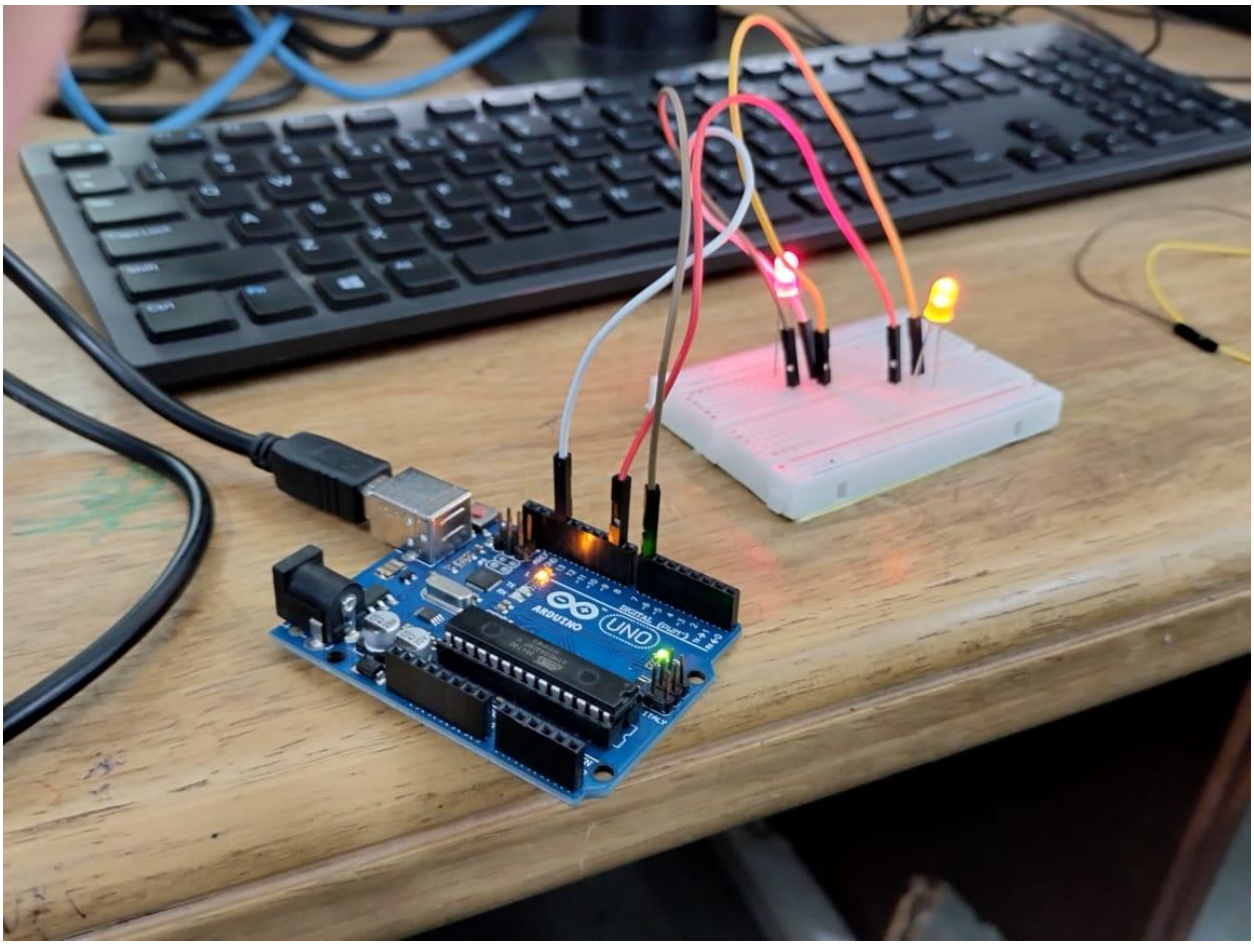




- **CODE Screen Shot:**

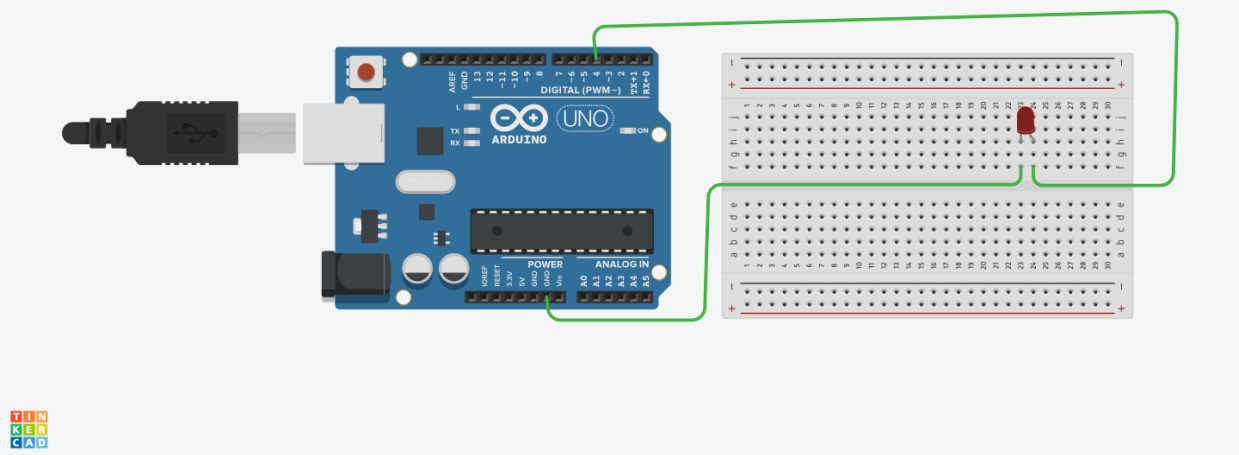
```
int lpin = 4;
int dt = 1000;
void setup() {
  pinMode(lpin, OUTPUT);
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(lpin,HIGH);
  delay(dt);
  digitalWrite(lpin,LOW);
  delay(dt);
}
```

Output:

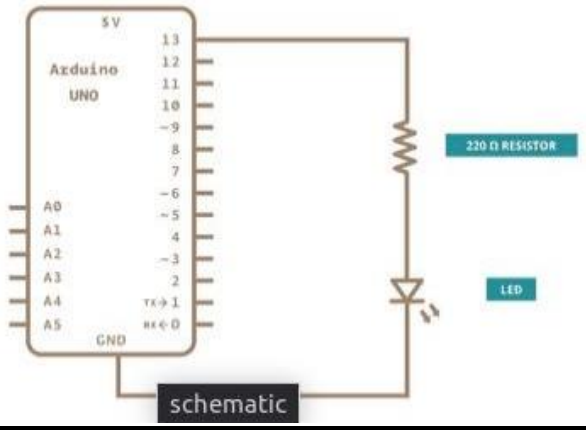


Fade-in and fade-out the LED

Circuit Diagram:



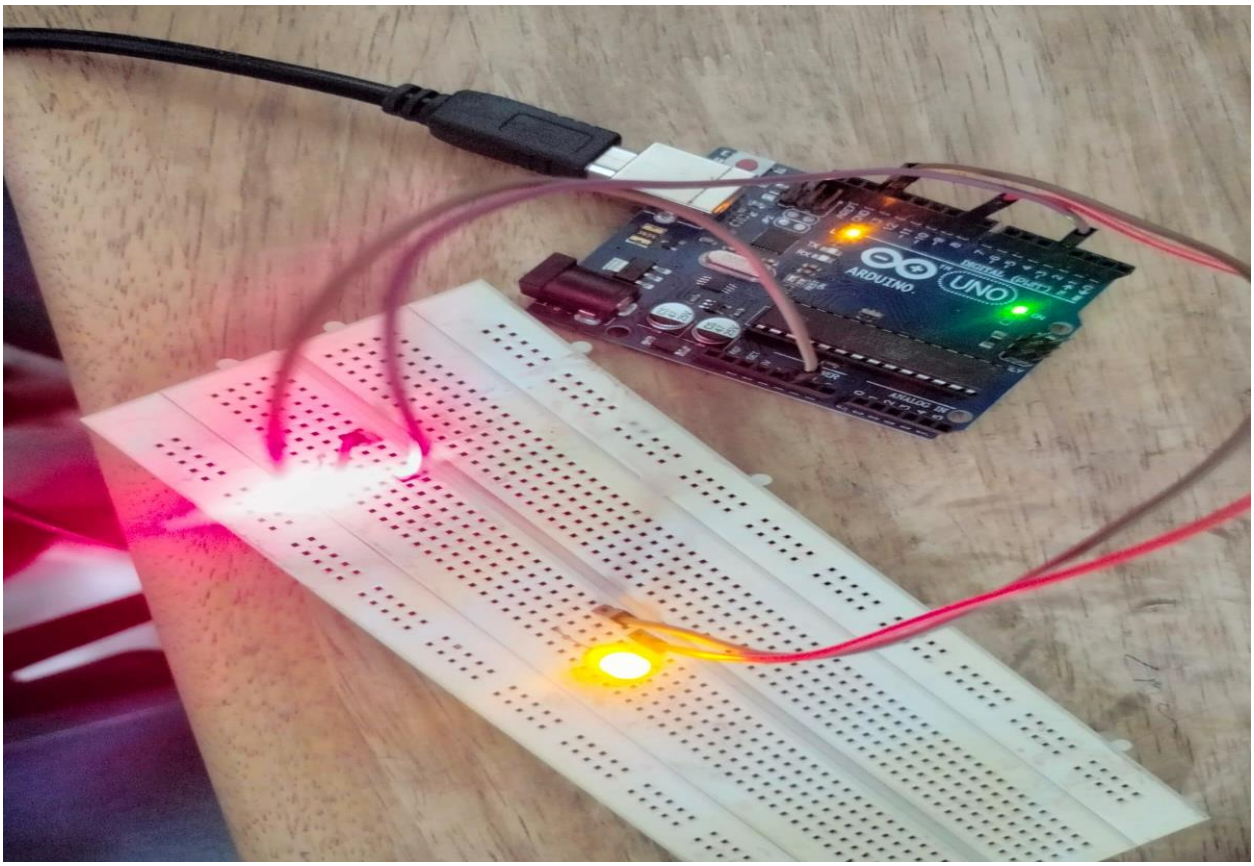
Schematic



CODE Screen Shot:

```
int led1 = 4;
int led2 = 9;         // the PWM pin the LED is attached to
int brightness = 0;  // how bright the LED is
int fadeAmount = 5;
int dt2 = 0;
int dt1 = 150; // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led2, OUTPUT);
  pinMode(led1, OUTPUT);
}

void loop() {
  analogWrite(led2, brightness);
  brightness = brightness + fadeAmount;
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(dt2);
  digitalWrite(led1, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(dt1);                       // wait for a second
  digitalWrite(led1, LOW);   // turn the LED off by making the voltage LOW
  delay(dt1);
}
```

Output:



Conclusion:

In conclusion, the Arduino Uno, a popular and accessible microcontroller board, serves as an excellent platform for beginners to carry out basic programs and learn about electronics and programming. By utilizing the digital and analog input/output pins on the board, two fundamental programs were explored: blinking the LED and creating a fade-in and fade-out effect.

Blinking the LED is a simple introductory project that involves turning an LED connected to one of the digital pins on and off in regular intervals. This exercise helps newcomers get familiar with uploading code to the Arduino, configuring digital pins as outputs, and controlling their states to control external components.

On the other hand, the fade-in and fade-out effect expands on the blinking concept by smoothly increasing and decreasing the LED's brightness instead of abrupt on-off transitions. This project introduces the concept of Pulse Width Modulation (PWM), which allows users to control the brightness of an LED or the speed of a motor, among other applications. Understanding PWM lays the foundation for more complex projects involving motor speed control, RGB LED lighting, and more.

Through these basic programs, beginners can gain hands-on experience with Arduino's capabilities and the principles of programming, electronics, and digital control. These foundational projects

serve as building blocks for exploring a vast array of creative and practical projects, ranging from home automation and robotics to interactive art installations and sensor-based applications. The Arduino platform's versatility and the wealth of online resources and community support ensure an exciting journey of exploration and learning for aspiring makers and engineers.