Practical No: 1

Aim:

1. To study the basics of Arduino circuits and bread-boarding
2. Blinking of LEDs

Simulation Environment: TinkerCAD (Free online simulator)

Part A: Basics of Arduino Circuits

Theory:

Arduino is an open-source electronics platform that has gained immense popularity for its ease of use and versatility. It was created in 2005 by a group of Italian engineers and is now maintained and developed by the Arduino community.

The heart of the Arduino platform is a microcontroller, which is a small, programmable computer on a single integrated circuit (IC) chip.

Arduino boards, which house these microcontrollers, provide a user-friendly environment for creating interactive electronic projects, prototypes, and various applications.

Key Components of Arduino:

1. Microcontroller: The core of an Arduino board is the microcontroller. The most commonly used microcontroller in Arduino is the ATmega series from Atmel (now a part of Microchip Technology). These microcontrollers come in different variations and are the brains behind your Arduino projects.

2. Input/output Pins: Arduino boards have a set of digital and analog pins that can be used to read data (inputs) or send data (outputs). Digital pins work with binary signals (0 or 1), while analog pins can read a range of values. The number and types of pins vary among different Arduino board models.

3. Power Supply: Arduino boards can be powered via USB, an external power supply, or a battery. Some boards have built-in voltage regulators, which make them compatible with a range of power sources.

4. USB Port: Arduino boards often feature a USB port for programming and power supply. This allows you to connect the board to your computer and upload code.

5. Reset Button: A reset button is provided to restart the Arduino, allowing you to upload new code or reset the program.

6. LED Indicator: Many Arduino boards include a built-in LED (Light Emitting Diode) on pin 13, which can be used for testing and basic visual feedback.

Arduino Software:

The Arduino platform comes with its integrated development environment (IDE). The Arduino IDE is a software tool that allows you to write, compile, and upload code to the Arduino board. Key features of the IDE include:

- Programming Language: Arduino uses a simplified version of the C/C++ programming language. It provides a set of libraries and functions tailored for easy interaction with the hardware.

- Code Library: Arduino has a vast library of pre-written code and functions that simplify common tasks, making it accessible to beginners.

- Serial Monitor: The IDE includes a serial monitor that allows you to communicate with the Arduino board and view debugging information.

- Community Support: The Arduino community is large and active, offering forums, tutorials, and extensive documentation to help users troubleshoot issues and learn.

Part B: Blinking of LEDs

Components Used:

1. Arduino UNO
2. Breadboard
3. LED
4. Resistor (330 Ω)

Program 1:

Code: The following C++ code is used in the given case void

```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}
```

```
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Program 2:

Code: The following C++ code is used in the given case void

```
// C++ code
//
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(12, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(12, LOW);
```

```
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(11, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(11, LOW);
}
```

Practical No: 2

Aim:

To study the working of Light sensor using Arduino

Simulation Environment: TinkerCAD (Free online simulator) Components: Arduino UNO, LED, Photodiode and Resistors Theory:

The goal of this practical is to create a system that can automatically control the brightness of an LED based on the light detected by a photodiode. This project leverages the principles of light sensing and feedback control.

Components:

a) Photodiode: A photodiode is a light-sensitive semiconductor device that generates a current or voltage proportional to the incident light's intensity. It acts as the input sensor in this system.

b) LED: An LED (Light Emitting Diode) is used as the output device. It emits light and can be controlled to vary its brightness.

c) Arduino: The Arduino microcontroller is the brain of the project. It reads data from the photodiode, processes it, and controls the LED's brightness accordingly.

Working:

a) Photodiode Operation:

The photodiode is connected to one of the Arduino's analog input pins.

When exposed to light, the photodiode generates a current or voltage that is directly proportional to the light intensity.

Arduino reads the analog voltage from the photodiode using one of its analog pins.

b) Control Algorithm:

The Arduino is programmed with an algorithm that translates the analog reading from the photodiode into a control signal for the LED.

The algorithm typically involves mapping the photodiode's output to the LED's brightness. For example, when the photodiode detects more light, the LED becomes brighter, and vice versa.

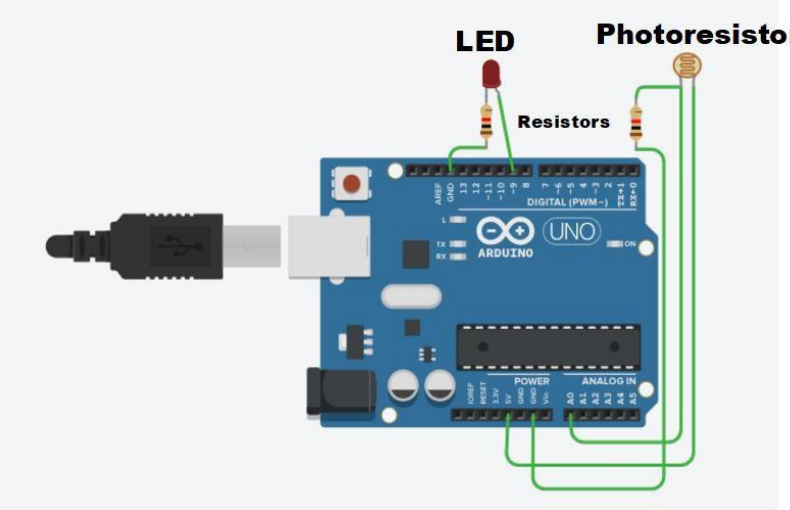
c) Feedback Loop:

The system operates in a feedback loop. As light conditions change, the photodiode detects the variations and sends this information to the Arduino.

The Arduino processes the data and adjusts the LED's brightness in real-time based on the input from the photodiode.

This closed-loop system ensures that the LED's brightness is always synchronized with the surrounding light levels.

Circuit Diagram:



Pin Connections:

Arduino	Photoresistor	LED
5V	Right pin	
GND (Power)	Left pin through a Series Resistor	
A_0	Left pin	
Pin 9		Anode
GND (Digital)		Cathode through a Series Resistor

Code: The following C++ code is used in the given case

```
int ldrPin = A0;    // LDR connected to analog pin A0
int ledPin = 9;     // LED connected to digital pin 9
int threshold = 500; // Threshold value for light intensity (adjust as needed)

void setup() {
  pinMode(ledPin, OUTPUT); // Set LED pin as output
  Serial.begin(9600);      // Begin serial communication
}
void loop() {
  int ldrValue = analogRead(ldrPin); // Read LDR value
  Serial.println(ldrValue);         // Print LDR value for debugging
  if (ldrValue < threshold) {
    digitalWrite(ledPin, HIGH); // Turn LED on if light intensity is low
  }
else {
    digitalWrite(ledPin, LOW);  // Turn LED off if light intensity is high
  }
  delay(1); // Small delay to stabilize readings
}
```