LED blinking with Arduino uno

Title:

Basic LED blinking with Arduino

Objective:

- **Understanding Microcontroller Basics**: Learning how to program and use a microcontroller, like the Arduino Uno, is the primary goal. The blinking LED project is often the first step toward understanding how microcontrollers work.
- **Learning Digital Output Control**: By turning the LED on and off, users understand how to control digital output pins of the Arduino. This teaches how to send signals to external components.
- **Practicing Arduino Programming**: Writing simple programs in the Arduino IDE introduces fundamental programming concepts such as loops (e.g., loop() function) and delays (e.g., using delay()).

Outcome:

Students will know about popular boards available in the market which are used for IoT.

Theory:

What is Arduino?

Arduino is an open-source**,** board that has a Microchip ATmega328P microcontroller on it. This microcontroller has a set of Digital & Analog input and output pins. The operating voltage of the board is 5V. It has 14 digital I/O pins & 6 Analog input pins. The clock frequency of the microcontroller is 16 MHz.

What is LED?

The flow of charge carriers (electrons and holes) across the P-N junction drives the activity of an LED. When a forward voltage (anode positive in comparison to the cathode) is applied, electrons and holes recombine at the junction, releasing energy in the form of photons (light). The semiconductor chip is linked to external terminals known as the anode (+) and the cathode (-). The anode is linked to the P-region, and the cathode to the N-region.

Blinking an LED

Blinking an LED is an introductory Arduino project in which we control an LED using Arduino. LED blinking refers to the process of continuously turning an LED (Light Emitting Diode) and off in a repetitive pattern. It is a simple and common demonstration in electronics and microcontroller-based projects

Arduino Uno:

The Arduino Uno R3 is a microcontroller board based on the ATmega328. It has 14

digital input/output pins (of which six can be used as PWM outputs), six analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. Simply connect it to a computer via a USB cable or power it with a AC-to-DC adapter or battery to get started.

Working Procedure

setup() and loop() are two fundamental Arduino functions for controlling the behavior of your board. The Arduino framework automatically calls these functions, which form the foundation of any Arduino program.

The setup() function is only called once when the Arduino board boots up or is reset. Its goal is to set pin modes, initialize variables, and execute any other necessary setup tasks before the main loop begins. This function can be used to configure settings that should only be changed once over the board's lifespan.

The loop() function is the heart of an Arduino program. After the setup() function is executed, the loop() function starts running repeatedly until the Arduino is powered off or reset. It contains the main code that performs the desired tasks, controls the board, user input. Whatever is included in the loop() function will be executed in a continuous loop, allowing the Arduino to perform its intended functions continuously.

In the code, we have declared two integers, **LEDpin** and **delayT. LEDpin** represents the pin number of the Arduino where LEDs need to be connected, and **delayT** is an integer variable for the delay() function. The delay() function accepts values in milliseconds.

Components Description

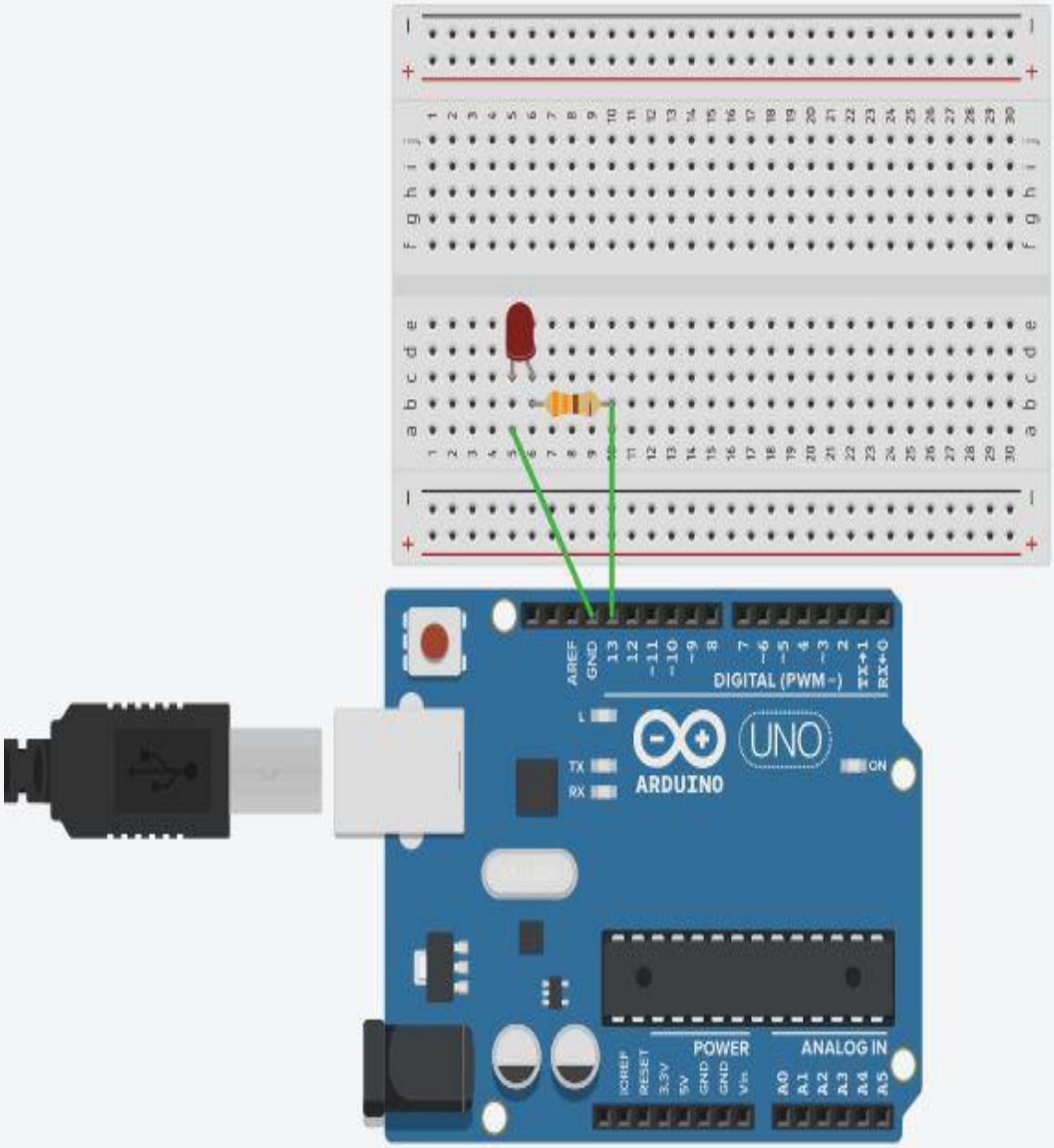
- **1 X LED:** We are controlling only one LED in this program.
- **1 X Resistor, 330 Ohm:** For every LED, we need one current limiting resistor.
- **Breadboard:** A breadboard is a fundamental tool used in electronics and prototyping to build and test circuits without soldering.
- **Arduino UNO R4** or earlier versions.
- **Jumper wires:** Jumper wires are simple electrical wires with connectors on both ends used to create connections between various electronic components or points on a circuit on a breadboard.

Arduino Code

C++

```
int LEDpin = 13;

int delayT = 1000;

void setup() {

// put your setup code here, to run once:

pinMode(LEDpin, OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(LEDpin, HIGH);

delay(delayT);

digitalWrite(LEDpin, LOW);

delay(delayT);

}
```

Outcome



Conclusion

In the circuit diagram, we used one 330-ohm resistor in series with the LED. This resistor is also called a current-limiting resistor. The Anode of the LED (the longer pin) is connected to one end of the resistor, and the cathode (the shorter pin) is connected to the ground. The other end of the resistor is connected to the Arduino pin. A step-by-step explanation is as follows:

1. **LED Connections:** Connect the LED to the breadboard. The LED has two legs, the longer of which is the anode (positive) and the shorter of which is the cathode (negative).

2. **Resistor Connection:** Insert one end of the resistor into the same row of the breadboard as the LED's Anode. The resistor's other end should be connected to the Arduino's digital output pin.

3. **Ground (GND) Connection:** Connect a jumper wire from the same row as the LED's cathode to any Arduino board GND (Ground) pin. This connects the circuit to the ground of the Arduino.