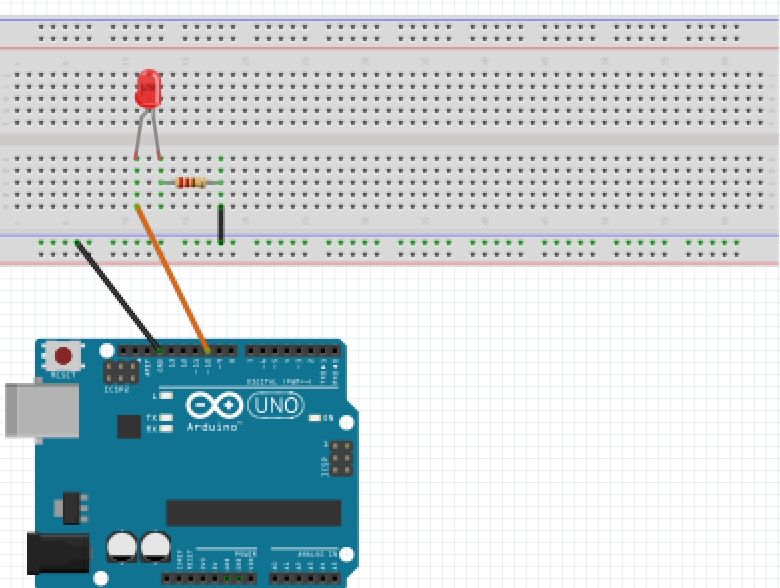
**ARDUINO LED PROJECT**

**Fading LED light.**

For this project we will need:

* Arduino board.
* LED light.
* Breadboard.
* 470 Ohm resistor.
* 3 Male to Male wires.

Circuit Design:

1. First make sure that the Arduino is powered off (no USB cable plugged to power).
2. Check the LED, you will see that one of the leg is shorter than the other one.
3. Plug the longer leg of the LED(anode) to a horizontal line on the breadboard. Connect using an orange wire from this line to pin 12 of the arduino.
4. Plug the shorter leg of the LED(cathode) to the breadboard. Separate horizontal line. Use a 470 ohm resistor from this line to connect to the common ground of the breadboard (marked by a blue (-) line).
5. Plug the common ground of the breadboard to the ground pin of the arduino using a black male jumper wire.

.

|  |
| --- |
| const int ledPin=10; // declare ARDUINO pin connected to LED anode  int brightness = 0;  const int fadeAmount = 5;  void setup() // assign the Led to output function.  pinMode(ledPin, OUTPUT);  Serial.begin(9600);  }  void loop(){  analogWrite(ledPin,brightness); // initial state of LED.  brightness = brightness + fadeAmount;  if(brightness <=0 || brightness>=255){ // condition  fadeAmount = - fadeAmount; // state  Serial.println(brightness);  }  delay(30);  } |

const int ledPin = 10;

First, we create a reference for the digital pin we intend to use and define a variable for that pin number. This will enable us to reference the pin by that variable name ledPin instead of the hard-coded number. In subsequent designs, if you need to use a different digital pin (for example pin 11), then you just need to change the number here and it will update it everywhere in your design program.

const shows that the value assigned doesn’t change during program execution

int shows the value is a number.Values declared without the keyword const can be modified later in the program execution.

pinMode(ledPin, OUTPUT);

After the execution of this line, the digital pin 10 will be set as output, and this will enable us to send information to it and control the LED.

Serial.begin(9600);

This initializes communication between the Arduino and your computer .It allows us to check on program execution and input /output operations. The value passed in parenthesis (9600) refers to the rate of flow of data (bitrate).

analogWrite(ledPin, brightness);

The pin we selected is unique.. Pin 10 along other pins marked with ~, allow us to perform analog operations.

In addition to passing discrete values(HIGH/LOW), this pin allows us to pass analog signals(in the range 0-255).

if(brightness <=0 || brightness>=255){}

This statement allows the program to make a decision if a set of conditions are achieved.

It checks

State = before making a decision it first checks the state of the program

Condition = if the state is within the ONLY conditions required it carries out specified instruction.

In this case it limits the brightness value to the allowed Analog range(0-255).

The state will be checked every time the loop gets executed

This state determines whether to increase or decrease the brightness value when the particular conditions are met

Since it is under the loop() it will keep increasing the brightness every time it is executed.

When the value exceeds the limits set it reverses.

Serial.println(brightness);

This tells the Serial monitor to display the value passed to it.

delay(30);

This determines how long the state will be maintained. The value passed is measured in milliseconds.

void setup (){}

This initializes the arduino and assigns functionality to its pins.

This also provides required resources for monitoring.

void loop(){}

After executing the void setup() function, we enter the void loop() and this function is executed continuously and repeatedly, until you Arduino is powered off.