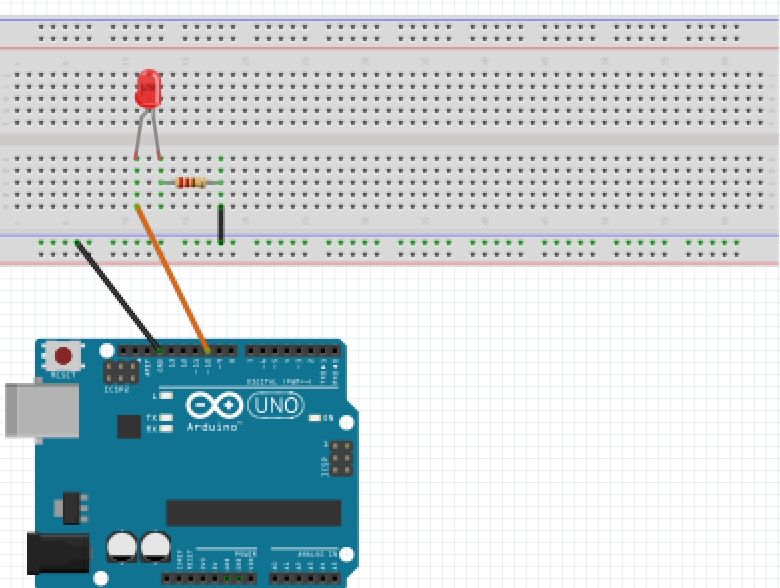
**ARDUINO LED PROJECT**

**Blinking LED light**



For this project we will need:

* Arduino board.
* LED light.
* Breadboard.
* 470 Ohm resistor.
* 2 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 a 470 ohm resistor to the horizontal line connected to the shorter leg(anode), then to the common ground of the breadboard.
6. 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 ledState = LOW;  unsigned long previousMillis = 0;  const long interval =1000;  void setup() // assign the Led to output function.  pinMode(ledPin, OUTPUT);  Serial.begin(9600);  }  void loop(){  unsigned long currentMillis = millis(); //checks current time.  Serial.println(currentMillis); // display to Serial monitor  if (currentMilis – previousMillis >= interval){  previousMillis = currentMillis;  if ( ledState == LOW ){  ledState = HIGH;  } else {  ledState = LOW;  }  digitalWrite(ledPin,ledState);  Serial.println(ledState); // Display to serial monitor  }  delay(100);  } |

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.

unsigned long previousMillis = 0;

unsigned long show that the value to be stored is expected to be a large number and have too many digits to store in int.

const long interval = 1000;

This refers to a value which will not change during program execution.

unsigned long currentMillis = millis();

millis() is used to check the time from the internal clock, which is then assigned a name for reference and expected values. Time is referenced to beginning of program execution.

Serial.begin(9600);

This initiates Serial communication with the Arduino and allows monitoring.The value passed indicates the bitrate(number of bits sent per second).

Serial.println(currentMillis);

This asks for the value passed to be printed to the serial monitor. This will be displayed on a new line every time every time the loop() is executed.

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.

if(currentMillis – previousMillis >= interval)

checks whether the time between current time and previous time the led was in another state is greater than our intended interval .This sets the condition.

previousMillis = currentMillis;

This changes the state.This overwrites the previous time and replaces it with the current time.

if (ledState == LOW);

This code block checks on the current state of the LED.If it meets the condition of being ON its switched OFF by passing corresponding LOW/HIGH signals

== is used to check on equality of the two values passed.

digitalWrite(ledPin, ledState);

This passes the signal to the LED in one of the two discrete states (HIGH/LOW);

Serial.println(ledState);

This prints the state the LED is in to the serial monitor. This allows us to verify the output from the LED.

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.