

**MICROCONTROLLERS FOR MECHATRONICS – MECA442**

Experiment 2: multitasking in arduino

**Jana Kabrit1, Oussama Al Ahmad Matar1**

**1** Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

*Abstract*

*The objective of this experiment is to study, program, and simulate a circuit that will have two LEDs blink at the same time by programming an Arduino. One of the LEDs should blink 2 times per second, and the other should blink 4 times per second. To get them to blink at the same time, we must code the Arduino using the millis function for multitasking the microcontroller. Second, we have to connect the Arduino and LEDs using a breadboard on the TinkerCad simulation.*

**Keywords:** Arduino, Led, Code, millis, TinkerCad.

1. INTRODUCTION

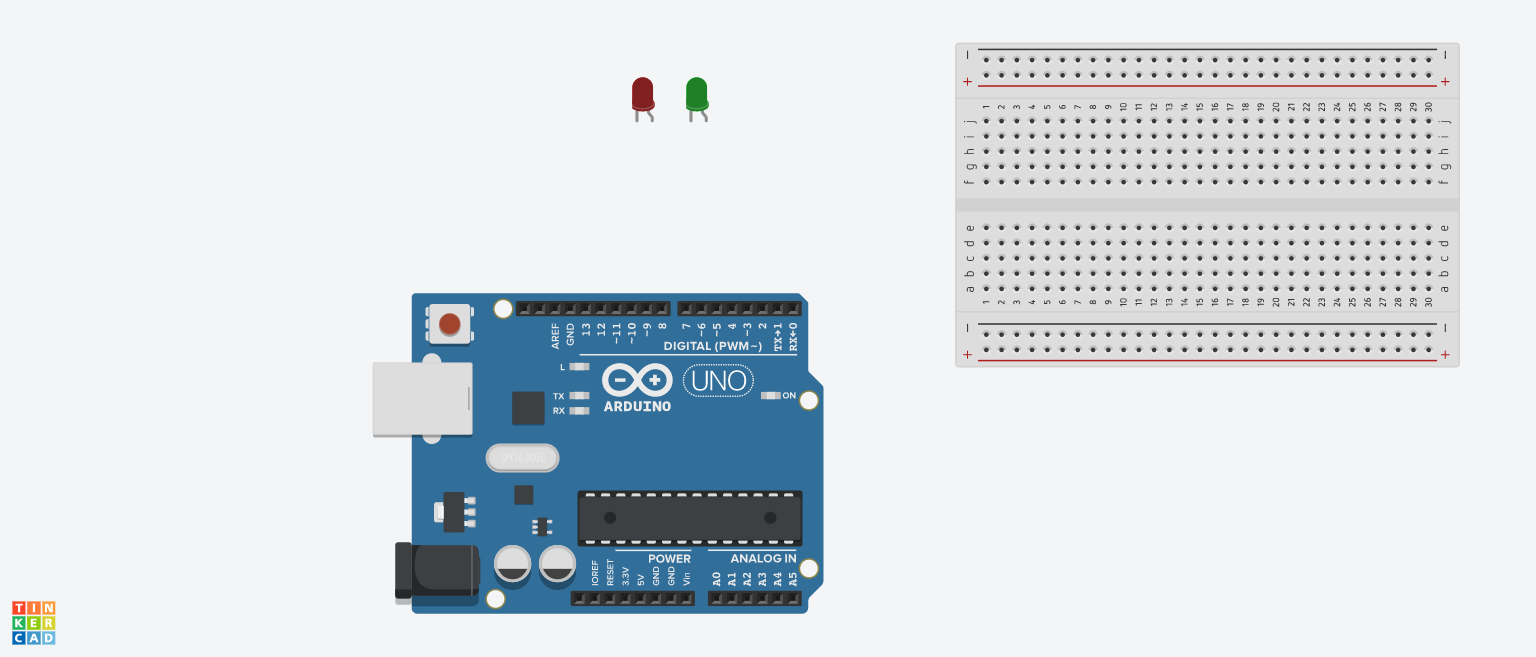
In this experiment, we need to code an Arduino to make two LEDs blink at the same time at different frequencies, and then simulate the circuit on TinkerCad. To do that, we have to start by writing the code that will allow the Arduino to multitask. We will be using the millis function to allow the microcontroller to run more than one task at a time. The first LED has to blink 2 times per second, and the third one has to blink 4 times per second. Second, we will connect the circuit on the simulation website using the proper material. The result we wish to obtain is a fully functional simulation where the LEDs are blinking at the same time at different frequencies.

1. **MATERIALS AND METHODS**

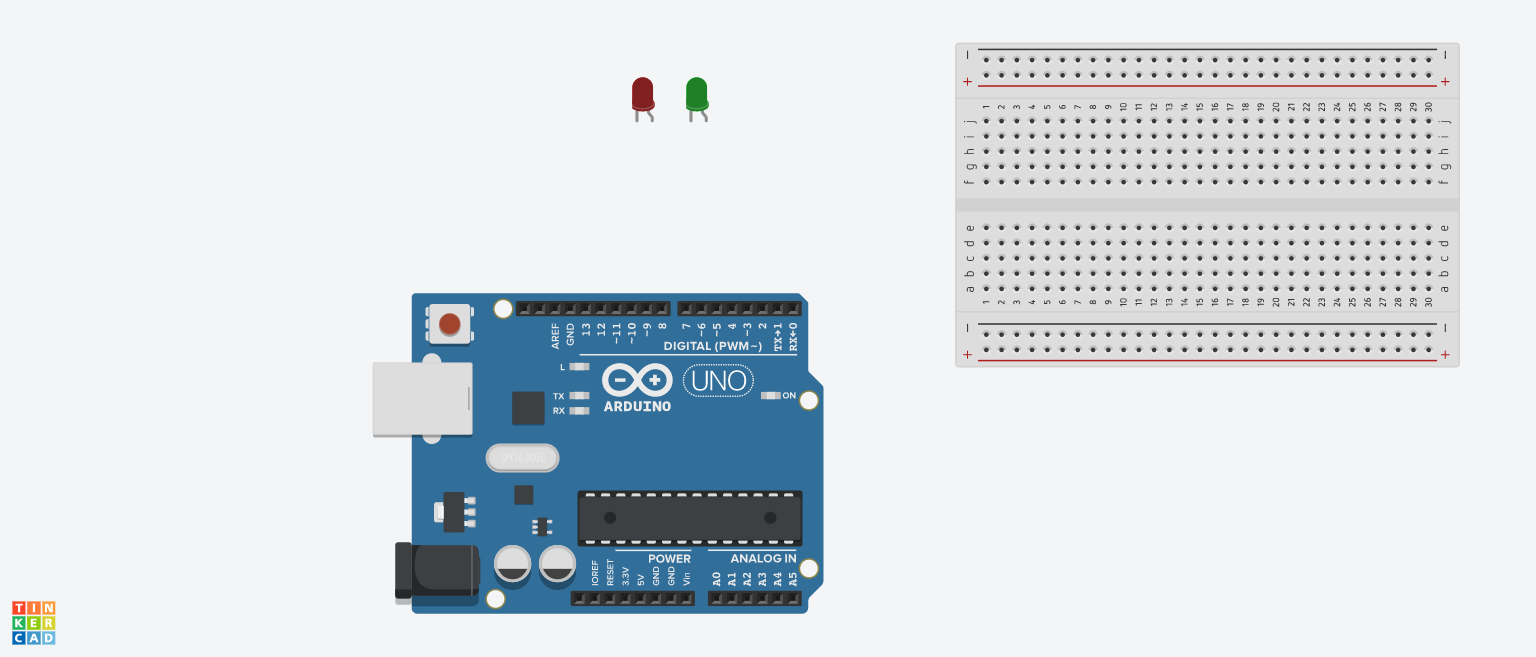
**2.1 Materials**

**2.1.1 Simulated Electronics Components**

1. Arduino Uno:

The Arduino Uno (Figure 1) is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.[1] In this experiment, we need the Arduino so that we can program it to make the LEDs blink the at the frequency and timings we want.

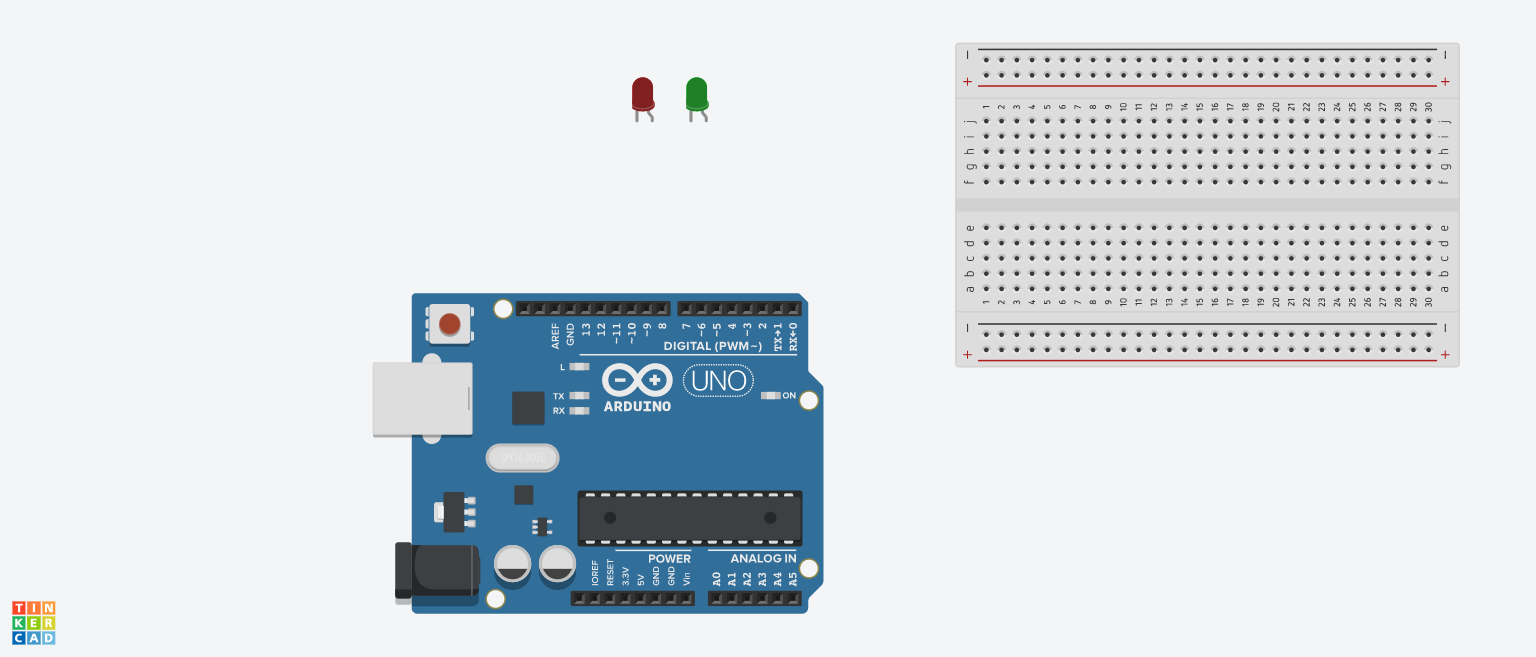
**FIGURE 1:** ARDUINO UNO BOARD

1. Breadboard:

A breadboard (Figure 2) is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to prototype (meaning to build and test an early version of) an electronic circuit, like this one with a battery, switch, resistor, and an LED (light-emitting diode).[2]

**FIGURE 2:** BREADBOARD

1. LEDs:

Light Emitting Diode (Figure 3) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the band gap of the semiconductor.[3] In this experiment, we want to make two LEDs, red and green, blink at 2Hz and 4Hz at the same time.

**FIGURE 3:** LEDs

* + 1. **Code Components**

1. const: A variable qualifier that modifies the behavior of the variable, making a variable "read-only" and constant.
2. long int: Long variables are extended size variables for number storage, and store 32 bits (4 bytes) instead of the usual 16 bits (2 bytes).
3. unsigned int: Same as int in that it stores a 2-byte value. Instead of storing negative numbers however they only store positive values, yielding a useful range of 0 to 65,535.
4. void setup: The function is called whenever the program starts. It is used to initialize variables, pin modes, etc. It will only run once after each powerup or reset of the Arduino board.
5. pinMode(): A function used to configure a specific pin to behave either as an input or an output, usually in the void setup.
6. void loop: The function that holds the code inside and runs over and over as long as the board is turned on.
7. millis(): A command that returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.
8. digitalWrite(): This function is used to write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value 3.3V-5V for HIGH, 0V (ground) for LOW.
9. HIGH/LOW: digitalWrite values can be HIGH or LOW. HIGH means you get a digital 1 on the output. LOW means you get a digital 0 on the output.[4]
   1. **Methods**

**2.2.1 Code:**

The code for the program is:

const int ledPin2 = 7; // the number of the LED pins

const int ledPin4 = 8;

int ledState2 = LOW; // used to set the state of the LEDs

int ledState4 = LOW;

unsigned long previousMillis2 = 0; // will store last time LED was updated

unsigned long previousMillis4 = 0;

const long interval2 = 500; // interval at which to blink (milliseconds)

const long interval4 = 250;

void setup() {

pinMode(ledPin2, OUTPUT); // sets the digital pin as output

pinMode(ledPin4, OUTPUT);

}

void loop() {

unsigned long currentMillis = millis(); // save the last time you blinked the LED

if (currentMillis - previousMillis2 >= interval2) { // if the LED is off turn it on and vice-versa

previousMillis2 = currentMillis;

if (ledState2 == LOW) {

ledState2 = HIGH;

}

else {

ledState2 = LOW;

}

digitalWrite(ledPin2, ledState2);

}

if (currentMillis - previousMillis4 >= interval4) {

previousMillis4 = currentMillis;

if (ledState4 == LOW) {

ledState4 = HIGH;

}

else {

ledState4 = LOW;

}

digitalWrite(ledPin4, ledState4);

}

}

**2.2.2 Description:**

The code shown above allows the Arduino Uno microcontroller to multitask, performing two or more processes in parallel. This can be done by constantly monitoring the time taken by each task without using the delay() function which would pause all the processing and tasks of the program. The function that allows the Arduino to monitor the time taken since it began running the program is millis() that measures in milliseconds. It has an unsigned limit of 50 days before overflowing and going back to zero.

In this activity, we’ll be controlling two blinking LEDs in parallel, each with a different frequency of blinking. The red LED blinks twice per second or at 2Hz, while the green LED blinks 4 times per seconds or at 4Hz, thus the time interval set for each in this code is 500ms and 250ms respectively. The red LED, connected to pin 7, is initially set to LOW such that once the difference of the current time and the last time the LED blinked (initially set to 0ms) reaches 500ms, the LED state is set to HIGH turning it on and sets a new value for the previous time to keep track of the time taken in each iteration. The code then proceeds to execute other tasks while not interfering with the red LED until the difference of the current and previous time is equal to the interval of 500ms again, in which turns off the LED if it was on, and vise versa. The same concept is applied to the green LED, connected to pin 8, but at a time interval of 250ms which allows it blink twice as fast.

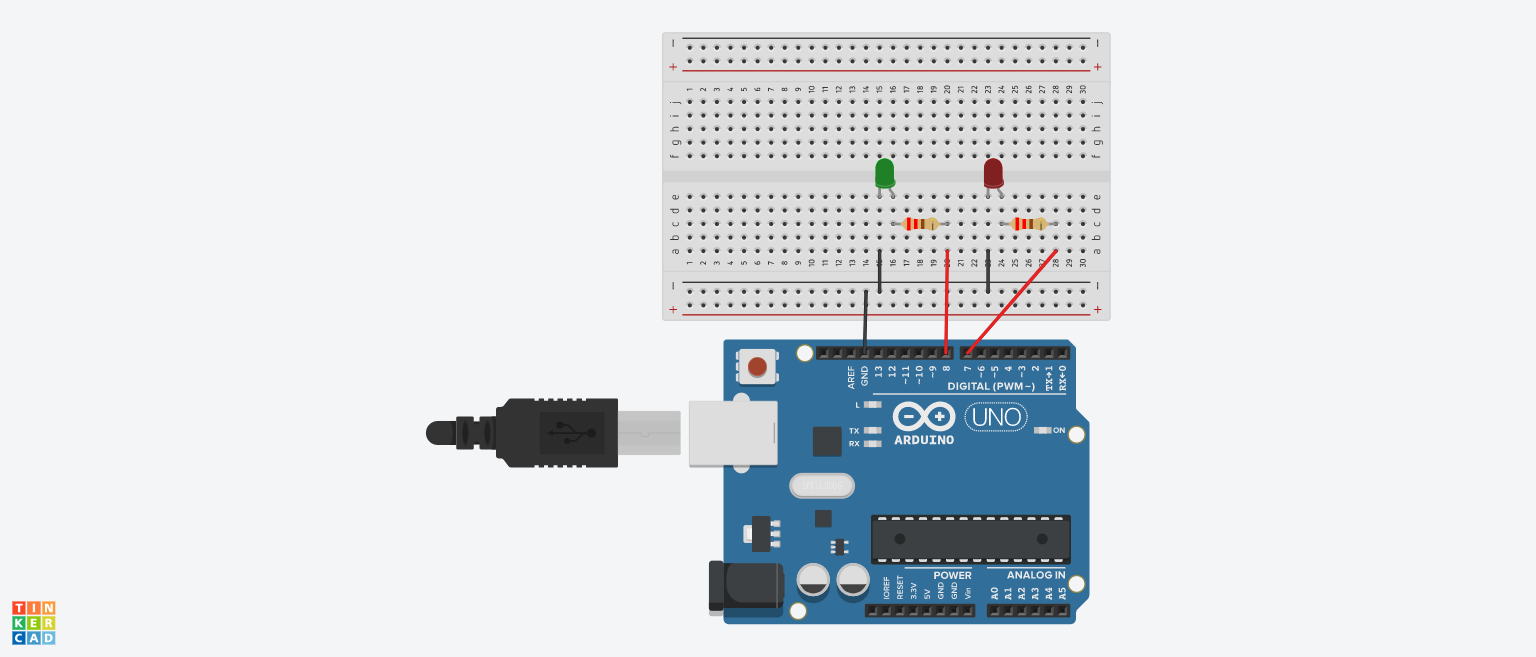
**2.2.3 Connections on TinkerCad:**

1. First, we added all the components we need for the simulation which are the Arduino Uno, Breadboard, green and red LEDs, and two 220 Ohm resistors.
2. Then we connected the ground of the Arduino to the ground of the breadboard (black line).
3. The assigned Pins of each LED is then connected to the LED through the 220 Ohm resistor (to prevent damaging the LED). The pins are connected directly to the LED and not through the 5V pin of the Arduino because LEDs need very little voltage to light that can be provided from the pins.
4. Lastly, we run the simulation.
5. **RESULTS AND DISCUSSION**

**3.1 Simulation**

|  |  |
| --- | --- |
| 1 | Arduino Uno Board |
| 2 | Breadboard |
| 3 | LEDs |
| 4 | 220 Ohm Resistors |

We implemented the circuit on TinkerCad, adding the code to the Arduino Simulation, and we obtained the circuit shown in figure 4.



**FIGURE 4:** CIRCUIT SIMULATION USING TINKERCAD

TABLE 1: Legend

**3.2 Analysis**

After running the simulation, the circuit showed the two LEDs blinking at the same time, each one at a different frequency. The red LED blinks at 2Hz, and the green one at 4Hz. The video of the simulation can be found at [this link](https://drive.google.com/file/d/1CzHzrPiron7m0njU9FzfUOCn4ECGEeYv/view?usp=sharing).

1. **CONCLUSION**

The objective of this experiment is to program an Arduino to multitask blinking two LEDs at the same time at different frequencies. This was done by using specific loops and functions, like millis, to multitask the microcontroller. The whole circuit was simulated on TinkerCad to test it. The experiment was successful, and the simulation showed two LEDs blinking at the same time at 2Hz and 4Hz.

**REFERENCES**

[1] Pololu Robotics & Electronics. *Arduino Uno*, 2020

https://www.pololu.com/product/2191#:~:text=Overview,header%2C%20and%20a%20reset%20button.

[2] Science Buddies. *How to Use a BreadBoard*, 2020

https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard

[3] Wikipedea. *Light-emitting diode,* 2020

https://en.wikipedia.org/wiki/Light-emitting\_diode

[4] Arduino Official Website  
 https://www.arduino.cc