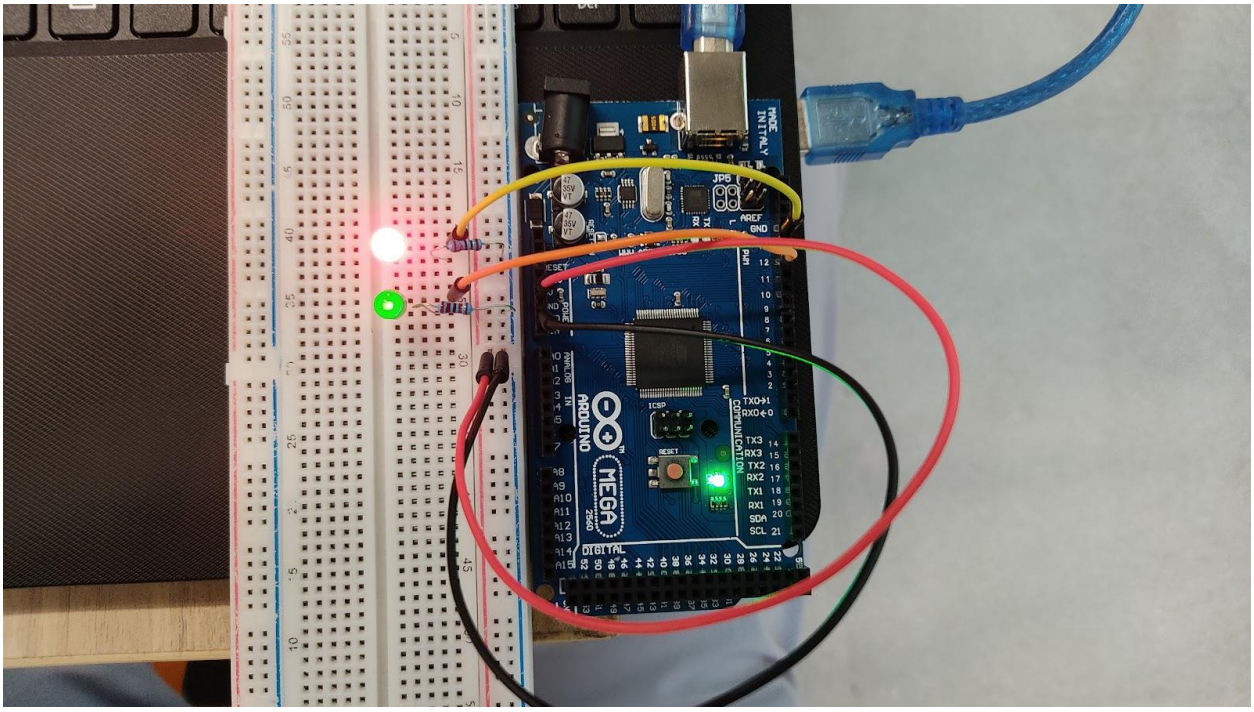
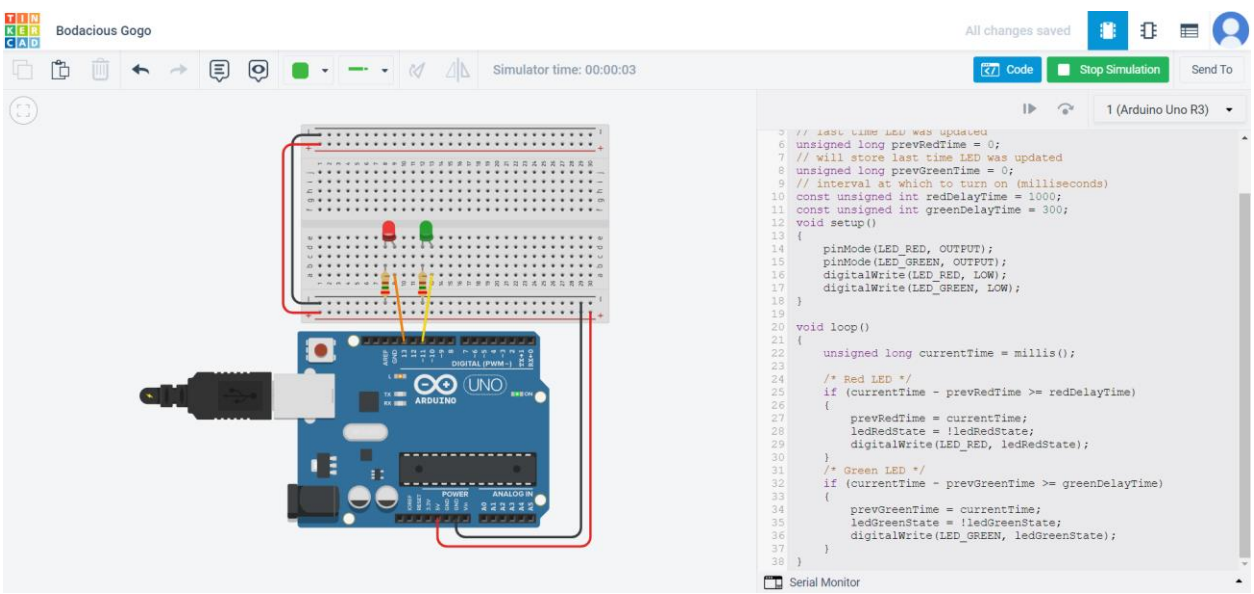
Group Work 1 & 2; Real Circuit Picture



Group Work 1

Tinkercad Picture



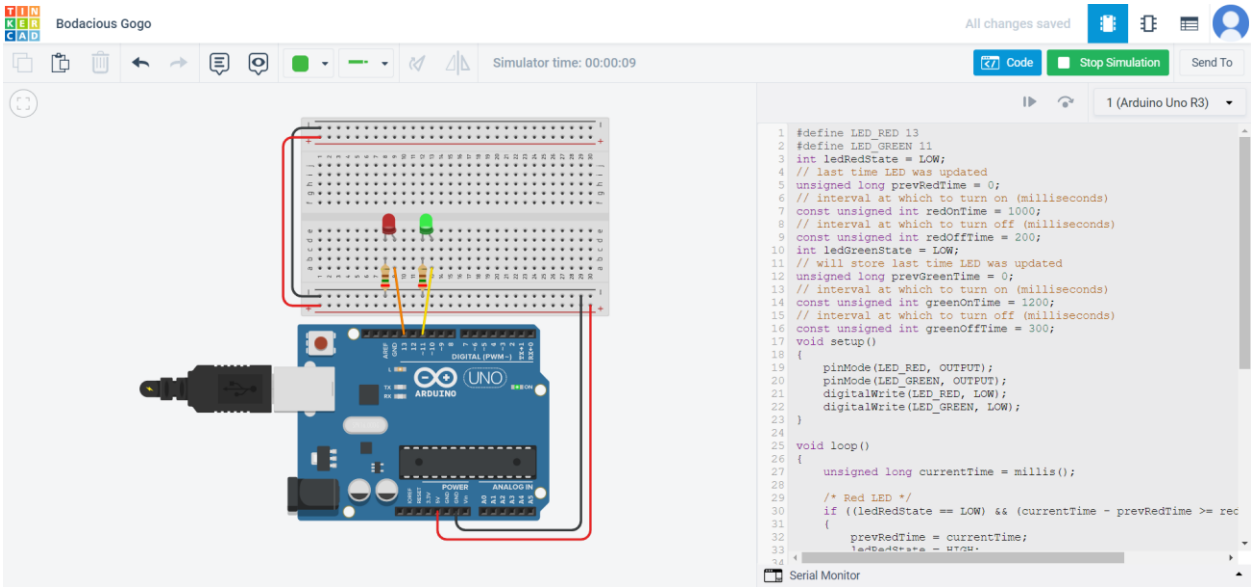
Code:

```
#define LED_RED 13
#define LED_GREEN 11
int ledRedState = LOW;
int ledGreenState = LOW;
// last time LED was updated
unsigned long prevRedTime = 0;
// will store last time LED was updated
unsigned long prevGreenTime = 0;
// interval at which to turn on (milliseconds)
const unsigned int redDelayTime = 1000;
const unsigned int greenDelayTime = 300;
void setup()
{
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
}

void loop()
{
    unsigned long currentTime = millis();

    /* Red LED */
    if (currentTime - prevRedTime >= redDelayTime)
    {
        prevRedTime = currentTime;
        ledRedState = !ledRedState;
        digitalWrite(LED_RED, ledRedState);
    }
    /* Green LED */
    if (currentTime - prevGreenTime >= greenDelayTime)
    {
        prevGreenTime = currentTime;
        ledGreenState = !ledGreenState;
        digitalWrite(LED_GREEN, ledGreenState);
    }
```

```
#define LED_RED 13
#define LED_GREEN 11
int ledRedState = LOW;
int ledGreenState = LOW;
// last time LED was updated
unsigned long prevRedTime = 0;
// will store last time LED was updated
unsigned long prevGreenTime = 0;
// interval at which to turn on (milliseconds)
const unsigned int redDelayTime = 1000;
const unsigned int greenDelayTime = 300;
void setup()
{
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
}

void loop()
{
    unsigned long currentTime = millis();

    /* Red LED */
    if (currentTime - prevRedTime ≥ redDelayTime)
    {
        prevRedTime = currentTime;
        ledRedState = !ledRedState;
        digitalWrite(LED_RED, ledRedState);
    }
    /* Green LED */
    if (currentTime - prevGreenTime ≥ greenDelayTime)
    {
        prevGreenTime = currentTime;
        ledGreenState = !ledGreenState;
        digitalWrite(LED_GREEN, ledGreenState);
    }
}
```

Group Work 2

Tinkercad Picture



Code:

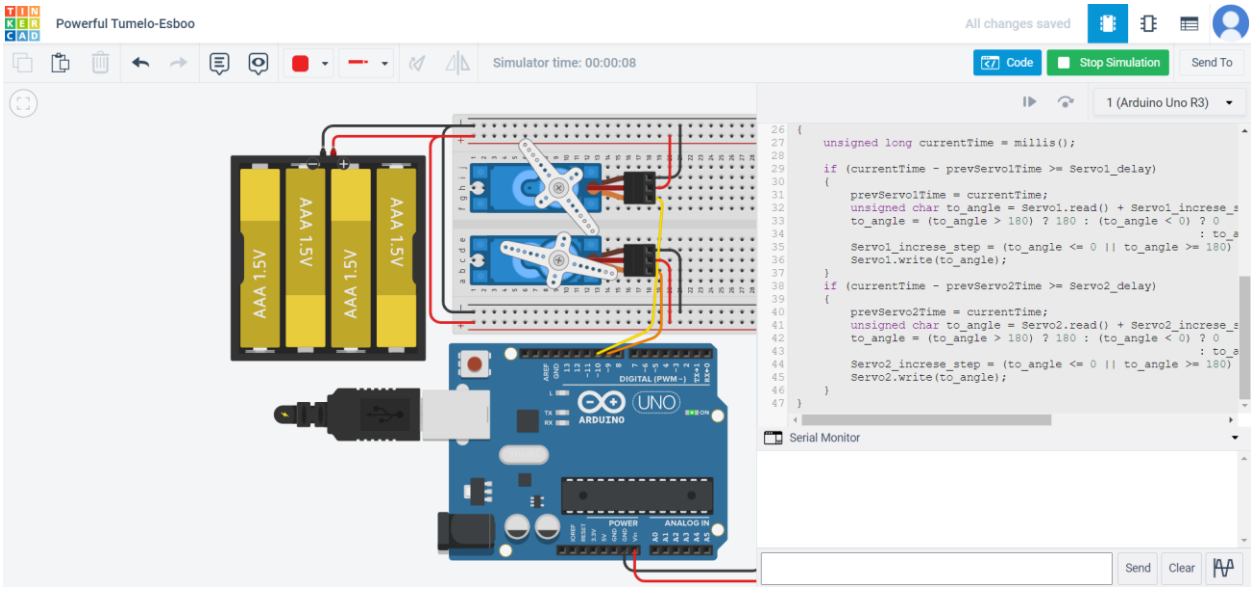
```
#define LED_RED 13
#define LED_GREEN 11
int ledRedState = LOW;
// last time LED was updated
unsigned long prevRedTime = 0;
// interval at which to turn on (milliseconds)
const unsigned int redOnTime = 1000;
// interval at which to turn off (milliseconds)
const unsigned int redOffTime = 200;
int ledGreenState = LOW;
// will store last time LED was updated
unsigned long prevGreenTime = 0;
// interval at which to turn on (milliseconds)
const unsigned int greenOnTime = 1200;
// interval at which to turn off (milliseconds)
const unsigned int greenOffTime = 300;
void setup()
{
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
}
```

```
void loop()
{
    unsigned long currentTime = millis();

    /* Red LED */
    if ((ledRedState == LOW) && (currentTime - prevRedTime >=
redOffTime))
    {
        prevRedTime = currentTime;
        ledRedState = HIGH;
        digitalWrite(LED_RED, ledRedState);
    }
    else if ((ledRedState == HIGH) && (currentTime - prevRedTime
>= redOnTime))
    {
        prevRedTime = currentTime;
        ledRedState = LOW;
        digitalWrite(LED_RED, ledRedState);
    }

    /* Green LED */
    if ((ledGreenState == LOW) && (currentTime - prevGreenTime >=
greenOffTime))
    {
        prevGreenTime = currentTime;
        ledGreenState = HIGH;
        digitalWrite(LED_GREEN, ledGreenState);
    }
    else if ((ledGreenState == HIGH) && (currentTime -
prevGreenTime >= greenOnTime))
    {
        prevGreenTime = currentTime;
        ledGreenState = LOW;
        digitalWrite(LED_GREEN, ledGreenState);
    }
}
```

```
#define LED_RED 13
#define LED_GREEN 11
int ledRedState = LOW;
// last time LED was updated
unsigned long prevRedTime = 0;
// interval at which to turn on (milliseconds)
const unsigned int redOnTime = 1000;
// interval at which to turn off (milliseconds)
const unsigned int redOffTime = 200;
int ledGreenState = LOW;
// will store last time LED was updated
unsigned long prevGreenTime = 0;
// interval at which to turn on (milliseconds)
const unsigned int greenOnTime = 1200;
// interval at which to turn off (milliseconds)
const unsigned int greenOffTime = 300;
void setup()
{
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    digitalWrite(LED_RED, LOW);
    digitalWrite(LED_GREEN, LOW);
}

void loop()
{
    unsigned long currentTime = millis();

    /* Red LED */
    if ((ledRedState == LOW) && (currentTime - prevRedTime ≥ redOffTime))
    {
        prevRedTime = currentTime;
        ledRedState = HIGH;
        digitalWrite(LED_RED, ledRedState);
    }
    else if ((ledRedState == HIGH) && (currentTime - prevRedTime ≥ redOnTime))
    {
        prevRedTime = currentTime;
        ledRedState = LOW;
        digitalWrite(LED_RED, ledRedState);
    }

    /* Green LED */
    if ((ledGreenState == LOW) && (currentTime - prevGreenTime ≥ greenOffTime))
    {
        prevGreenTime = currentTime;
        ledGreenState = HIGH;
        digitalWrite(LED_GREEN, ledGreenState);
    }
    else if ((ledGreenState == HIGH) && (currentTime - prevGreenTime ≥ greenOnTime))
    {
        prevGreenTime = currentTime;
        ledGreenState = LOW;
        digitalWrite(LED_GREEN, ledGreenState);
    }
}
```

Group Work 3

Tinkercad Picture



Code:

```
#include <Servo.h>

#define Servo1_pin 9
#define Servo2_pin 10

// delay in milliseconds
const unsigned int Servo1_delay = 100;
const unsigned int Servo2_delay = 200;

int Servo1_increse_step = 1;
int Servo2_increse_step = 1;

unsigned long prevServo1Time = 0;
unsigned long prevServo2Time = 0;

Servo Servo1;
Servo Servo2;

void setup()
{
    Servo1.attach(Servo1_pin);
    Servo2.attach(Servo2_pin);
}
```

```
void loop()
{
    unsigned long currentTime = millis();

    if (currentTime - prevServo1Time >= Servo1_delay)
    {
        prevServo1Time = currentTime;
        unsigned char to_angle = Servo1.read() + Servo1_increse_step;
        to_angle = (to_angle > 180) ? 180 : (to_angle < 0) ? 0
                                                            : to_angle;
        Servo1_increse_step = (to_angle <= 0 || to_angle >= 180) ? -
Servo1_increse_step : Servo1_increse_step;
        Servo1.write(to_angle);
    }
    if (currentTime - prevServo2Time >= Servo2_delay)
    {
        prevServo2Time = currentTime;
        unsigned char to_angle = Servo2.read() + Servo2_increse_step;
        to_angle = (to_angle > 180) ? 180 : (to_angle < 0) ? 0
                                                            : to_angle;
        Servo2_increse_step = (to_angle <= 0 || to_angle >= 180) ? -
Servo2_increse_step : Servo2_increse_step;
        Servo2.write(to_angle);
    }
}
```

```
#include <Servo.h>

#define Servo1_pin 9
#define Servo2_pin 10

// delay in milliseconds
const unsigned int Servo1_delay = 100;
const unsigned int Servo2_delay = 200;

int Servo1_increse_step = 1;
int Servo2_increse_step = 1;

unsigned long prevServo1Time = 0;
unsigned long prevServo2Time = 0;

Servo Servo1;
Servo Servo2;

void setup()
{
    Servo1.attach(Servo1_pin);
    Servo2.attach(Servo2_pin);
}

void loop()
{
    unsigned long currentTime = millis();

    if (currentTime - prevServo1Time ≥ Servo1_delay)
    {
        prevServo1Time = currentTime;
        unsigned char to_angle = Servo1.read() + Servo1_increse_step;
        to_angle = (to_angle > 180) ? 180 : (to_angle < 0) ? 0
                                                            : to_angle;
        Servo1_increse_step = (to_angle ≤ 0 || to_angle ≥ 180) ? -Servo1_increse_step : Servo1_increse_step;
        Servo1.write(to_angle);
    }
    if (currentTime - prevServo2Time ≥ Servo2_delay)
    {
        prevServo2Time = currentTime;
        unsigned char to_angle = Servo2.read() + Servo2_increse_step;
        to_angle = (to_angle > 180) ? 180 : (to_angle < 0) ? 0
                                                            : to_angle;
        Servo2_increse_step = (to_angle ≤ 0 || to_angle ≥ 180) ? -Servo2_increse_step : Servo2_increse_step;
        Servo2.write(to_angle);
    }
}
```