```
#include <FastLED.h>

#define LED_PIN     5
#define COLOR_ORDER GRB
#define CHIPSET     WS2812B
#define NUM_LEDS    150

#define BRIGHTNESS  200
#define FRAMES_PER_SECOND 60

CRGB leds[NUM_LEDS];

void setup() {
  delay(3000); // sanity delay
  FastLED.addLeds<CHIPSET, LED_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection( TypicalLEDStrip );
  FastLED.setBrightness( BRIGHTNESS );
}

void loop()
{
  // Add entropy to random number generator; we use a lot of it.
  random16_add_entropy( random());

  Fire2012(); // run simulation frame

  FastLED.show(); // display this frame
  FastLED.delay(1000 / FRAMES_PER_SECOND);
}


// Fire2012 by Mark Kriegsman, July 2012
// as part of "Five Elements" shown here: http://youtu.be/knWiGsmgycY
////
// This basic one-dimensional 'fire' simulation works roughly as follows:
// There's a underlying array of 'heat' cells, that model the temperature
// at each point along the line.  Every cycle through the simulation,
// four steps are performed:
//  1) All cells cool down a little bit, losing heat to the air
//  2) The heat from each cell drifts 'up' and diffuses a little
//  3) Sometimes randomly new 'sparks' of heat are added at the bottom
//  4) The heat from each cell is rendered as a color into the leds array
//     The heat-to-color mapping uses a black-body radiation approximation.
```

```
//
// Temperature is in arbitrary units from 0 (cold black) to 255 (white hot).
//
// This simulation scales it self a bit depending on NUM_LEDS; it should look
// "OK" on anywhere from 20 to 100 LEDs without too much tweaking.
//
// I recommend running this simulation at anywhere from 30-100 frames per second,
// meaning an interframe delay of about 10-35 milliseconds.
//
// Looks best on a high-density LED setup (60+ pixels/meter).
//
//
// There are two main parameters you can play with to control the look and
// feel of your fire: COOLING (used in step 1 above), and SPARKING (used
// in step 3 above).
//
// COOLING: How much does the air cool as it rises?
// Less cooling = taller flames.  More cooling = shorter flames.
// Default 50, suggested range 20-100
#define COOLING  55

// SPARKING: What chance (out of 255) is there that a new spark will be lit?
// Higher chance = more roaring fire.  Lower chance = more flickery fire.
// Default 120, suggested range 50-200.
#define SPARKING 120


void Fire2012()
{
// Array of temperature readings at each simulation cell
  static byte heat[NUM_LEDS];

  // Step 1.  Cool down every cell a little
    for( int i = 0; i < NUM_LEDS; i++) {
      heat[i] = qsub8( heat[i],  random8(0, ((COOLING * 10) / NUM_LEDS) + 2));
    }

    // Step 2.  Heat from each cell drifts 'up' and diffuses a little
    for( int k= NUM_LEDS - 1; k >= 2; k--) {
      heat[k] = (heat[k - 1] + heat[k - 2] + heat[k - 2] ) / 3;
    }

    // Step 3.  Randomly ignite new 'sparks' of heat near the bottom
```

```
    if( random8() < SPARKING ) {
      int y = random8(7);
      heat[y] = qadd8( heat[y], random8(160,255) );
    }

    // Step 4.  Map from heat cells to LED colors
    for( int j = 0; j < NUM_LEDS; j++) {
        leds[j] = HeatColor( heat[j]);
    }
}
```