```
#include <FastLED.h>
#include <vector>
#define LED_PIN 3
#define NUM_LEDS 30
#define BRIGHTNESS 255
#define LED_TYPE WS2811
#define COLOR_ORDER RGB
CRGB leds[NUM_LEDS];
int current_led = 0;
const int trigPin = 9;
const int echoPin = 10;
const int ledPin = 13;
// Defines variables
long duration;
int distance;
int safetyDistance;
std::vector<int> arrayDistance;
int average_distance = 0;
void setup() {
 FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection(TypicalLEDStrip);
 FastLED.setBrightness(BRIGHTNESS);
 pinMode(trigPin, OUTPUT);  // Sets the trigPin as an Output
 pinMode(echoPin, INPUT);   // Sets the echoPin as an Input
 pinMode(ledPin, OUTPUT);
 Serial.begin(9600);  // Starts the serial communication
 leds[3] = CRGB::Black;
 leds[5] = CRGB::Black;
 leds[1] = CRGB::Black;
 FastLED.show();}
void loop() {
 // Clears the trigPin
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 // Sets the trigPin on HIGH state for 10 micro seconds
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 // Reads the echoPin, returns the sound wave travel time in microseconds
 duration = pulseIn(echoPin, HIGH);
 // Calculating the distance (adjusting for upside-down sensor if needed)
 distance = duration / 100;  // Convert time to distance in cm
 if (distance <= 0 || distance > 120) {
  // If the sensor returns an invalid value, fallback to a default safe distance
  //distance = 5;
```

```
  //  Serial.println("Invalid distance detected. Using fallback value.");
  }
 // Serial.print("Raw distance: ");
  Serial.println(distance);
  arrayDistance.push_back(distance);
  if (arrayDistance.size() == 10) {
    average_distance = 0;
    arrayDistance.erase(arrayDistance.begin());
    for (int element : arrayDistance) {
      average_distance += element / 10.0;
    }
  }
  safetyDistance = average_distance;
 // Serial.print("Average distance: ");
  Serial.println(average_distance);
  if (safetyDistance <= 60) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  if (safetyDistance > 600) {
    leds[3] = CRGB::Black;
    leds[5] = CRGB::Black;
    leds[1] = CRGB::Black;
    FastLED.show();
  } else if (safetyDistance <= 600 && safetyDistance > 550) {
    leds[3] = CRGB(255, 0, 0);
    leds[5] = CRGB::Black;
    leds[1] = CRGB::Black;
    FastLED.show();
  } else if (safetyDistance <= 550 && safetyDistance > 300) {
    leds[5] = CRGB(255, 0, 0);
    leds[3] = CRGB::Black;
    leds[1] = CRGB::Black;
    FastLED.show();
  } else if (safetyDistance <= 300 && safetyDistance >= 0) {
    leds[1] = CRGB(255, 0, 0);
    leds[5] = CRGB::Black;
    leds[3] = CRGB::Black;
    FastLED.show();
  }
  delay(200);  // Small delay for stability
}
```