PRO-C262

GESTURE CONTROLLED LIGHTS

What is our GOAL for this CLASS?

In this class, we learned about sensors. We also learned how to use a PIR motion sensor. Additionally, we generated random patterns on a NeoPixel Ring.

What did we ACHIEVE in the class TODAY?

- Learned about sensors.
- Understood the working principle and usage of PIR motion sensors.
- Learned about the NeoPixel Ring.
- Built a program to light up the NeoPixel Ring on motion detection.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Concepts : Generating random numbers , conditional statements, loops, .

- Coding blocks : **random** method, **if-else** statements, **for** loops,.

How did we DO the activities?

1. Sensors:

   Sensors help machines to detect their surroundings. Sensors in machines can detect physical parameters like - heat, light sound etc.

2. PIR sensor:

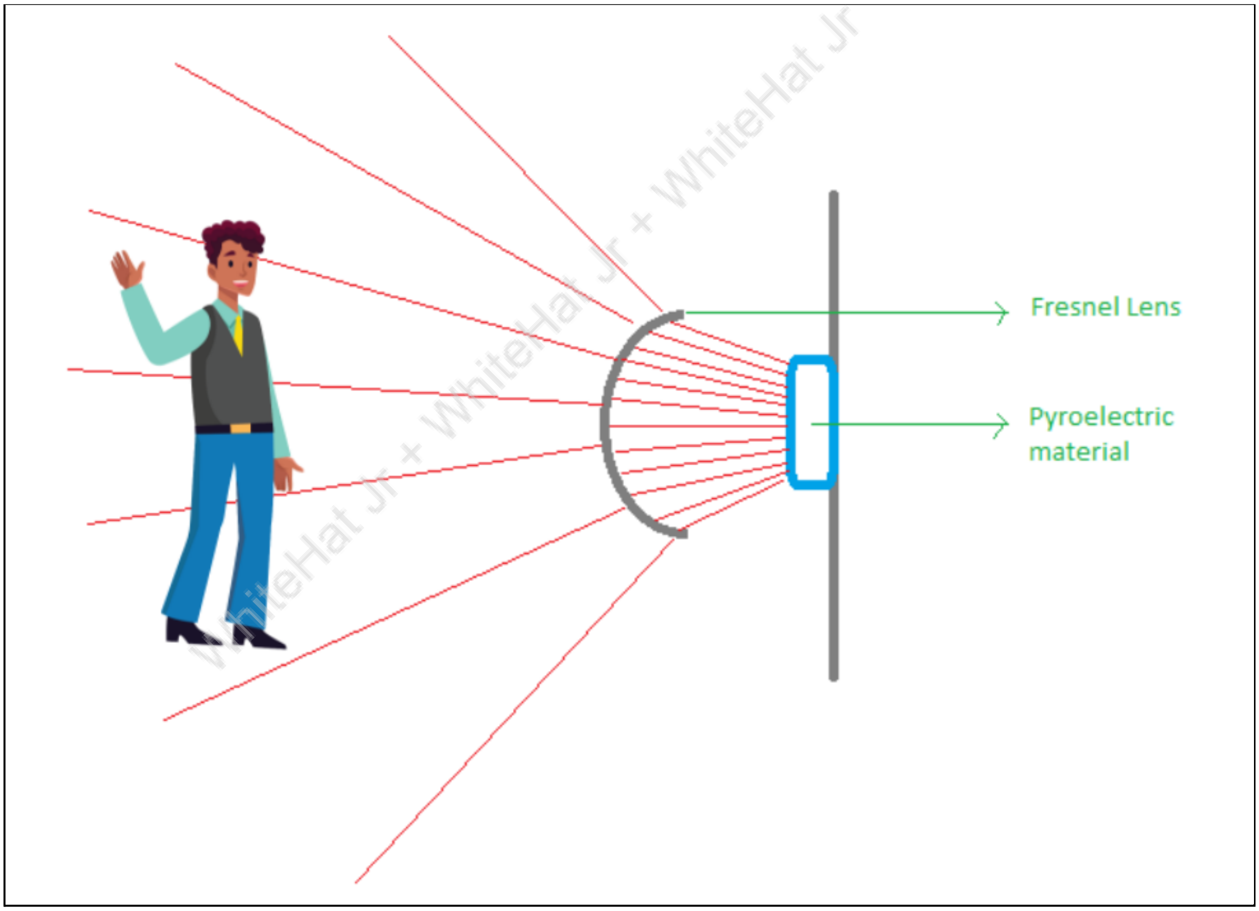
   PIR sensor stands for Passive infrared sensor. It is a motion sensor which can detect a living being. It is unable to detect non-living objects.

3. Working principle of **PIR sensor**:

   PIR sensors use pyroelectric sensors. Pyroelectric materials are special materials which can detect infrared radiations (radiant heat). Pyroelectric materials have the ability to generate a temporary voltage when they are heated.

   Every living being emits some low levels of radiation. The PIR sensor can detect the radiation when a living being moves into the range of the PIR sensor.

   When a person walks into the range of the PIR sensor, the radiation is detected by the sensor. It also has a dome-shaped lens (also known as Fresnel Lens) which helps to concentrate the IR radiation on the Pyroelectric material.



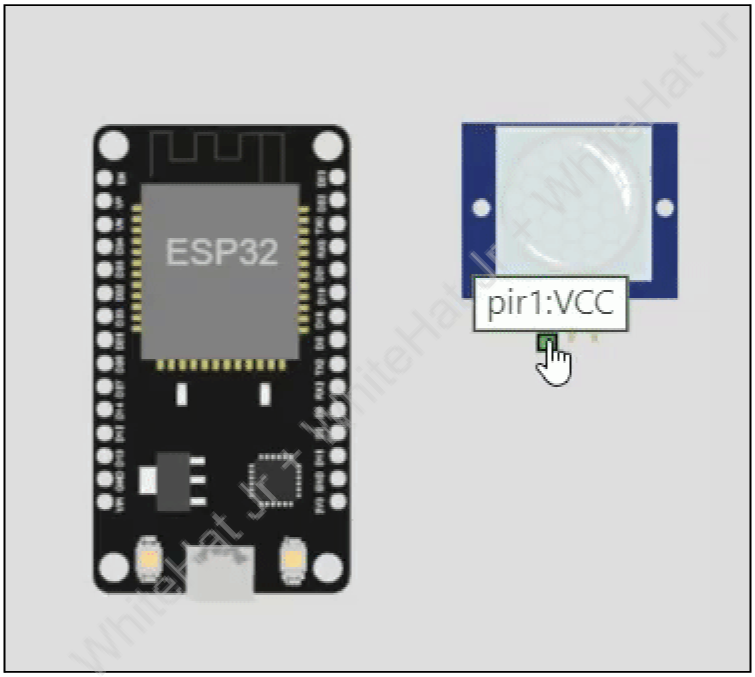
4. Create the circuit:

   a. Select the components

      - 1 x **ESP32**
      - 1 x **PIR Motion Sensor**

b. Let's connect:

PIR Sensor	ESP32 PIN
VCC	3V3
GND	GND.1
OUT	GPIO D2 (It can be connected to any GPIO port)



Click here to view the reference video.

5. Code for the PIR sensor:

a. First, define a constant which will hold the port number in which the PIR sensor is connected to.

```
const byte pir_pin= 2;
```

b. The PIR sensor sends an input to the controller. So, the next task would be to define the PIR sensor's pin mode as INPUT. Define pin mode in the **setup() method-**

```
pinMode(pir_pin, INPUT);
```

c. Now, let's check if our **PIR sensor** is working or not with the following methods-

- **digitalRead()** - method reads the value from a specified digital pin. Read the pin which is connected to the PIR sensor-

  digitalRead(pir_pin);

- **Serial.println()** - method will print the value that we pass through this method. To view the value returned by the **PIR sensor**-

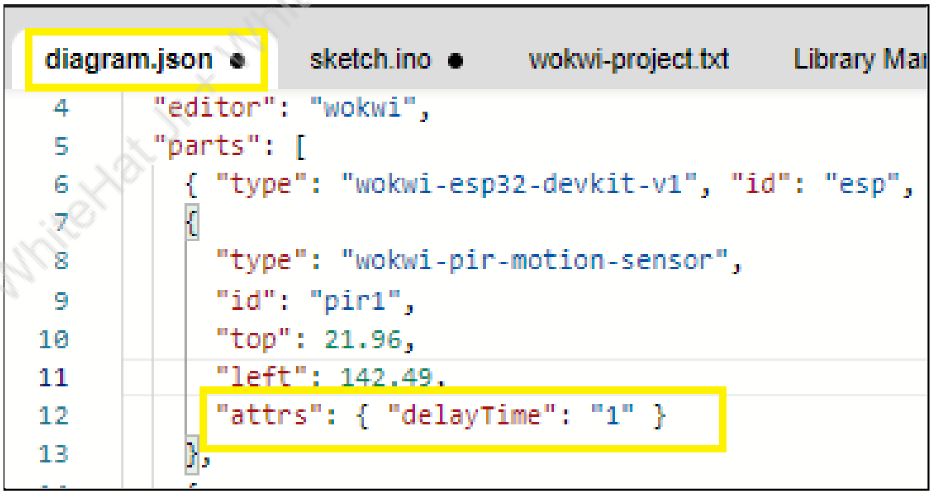
  Serial.println(digitalRead(pir_pin);

  write this line of code inside the **loop()** method.

d. The **PIR sensor** returns **1 for a certain time** when a motion is detected. Let's change the time lapse for which 1 should be generated after the detection of motion.

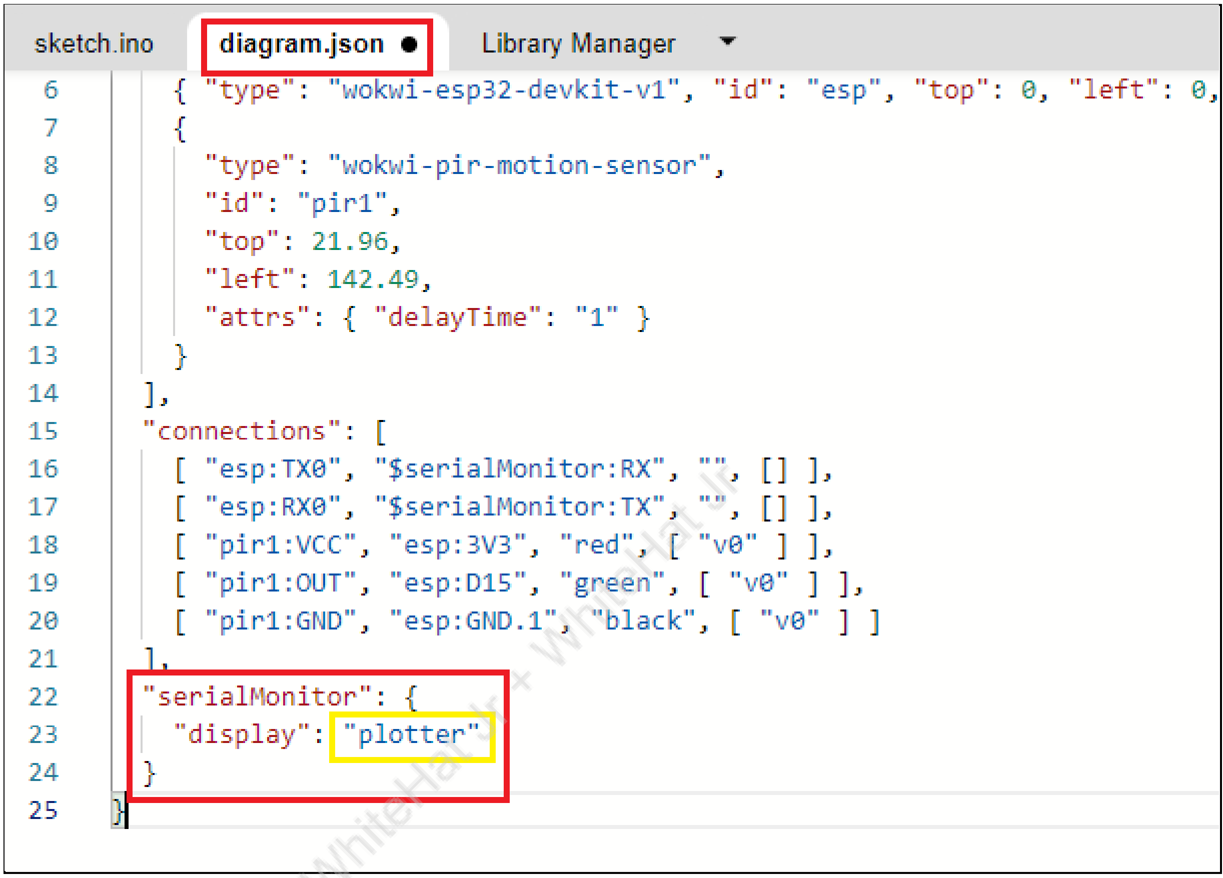
- Go to **diagram.json** → find "**wokwi-pir-motion-sensor**" under parts property → add "**delayTime**" as "1" under "attrs" property.

  (delayTime is 5 seconds by default. This will make the project wait for 5 seconds before it can detect a motion again. change it to 1.)



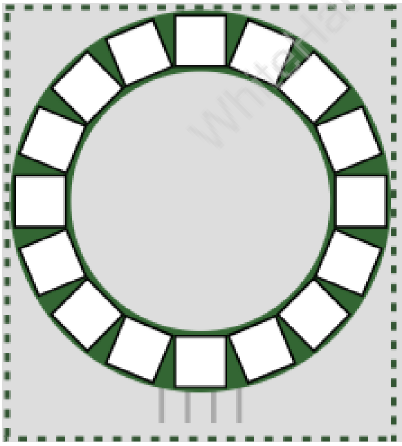
```
"editor": "wokwi",
"parts": [
  { "type": "wokwi-esp32-devkit-v1", "id": "esp",
  {
    "type": "wokwi-pir-motion-sensor",
    "id": "pir1",
    "top": 21.96,
    "left": 142.49,
    "attrs": { "delayTime": "1" }
  },
```

e. Add another property called "serialMonitor" at the end of the diagram.json which will help us plot our output on a graph



```
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0,
    {
      "type": "wokwi-pir-motion-sensor",
      "id": "pir1",
      "top": 21.96,
      "left": 142.49,
      "attrs": { "delayTime": "1" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "pir1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "pir1:OUT", "esp:D15", "green", [ "v0" ] ],
    [ "pir1:GND", "esp:GND.1", "black", [ "v0" ] ]
  ],
  "serialMonitor": {
    "display": "plotter"
  }
}
```

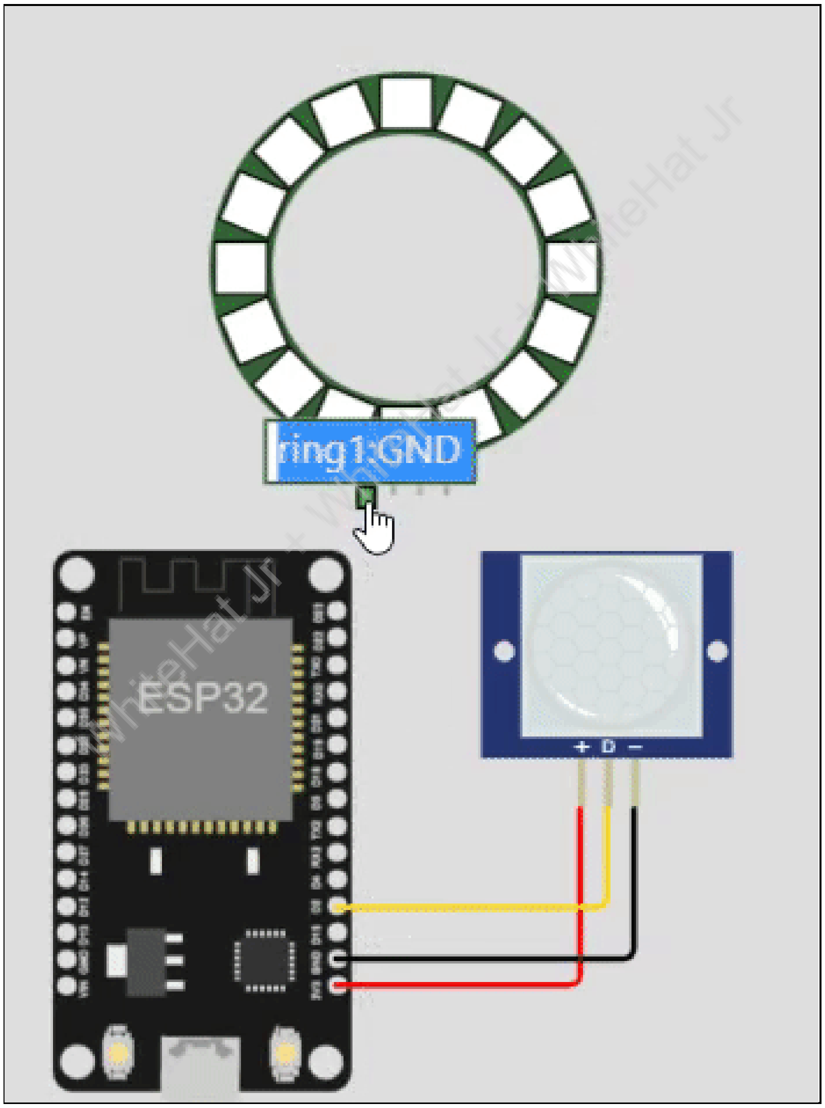
6. NeoPixel Ring:



NeoPixel ring is a chain of 16 LED lights connected together in the shape of a ring. These lights can be controlled by one input pin.

a. Let's make the connections,

KEYPAD	ESP32 PIN
GND	GND.1
VCC	3V3
DIN	GPIO D15 (It can be connected to any GPIO port)



Click here to view the reference video.

7. Code to light up the NeoPixel LEDs:

go to the **sketch.ino** file and write the code.

a. Include header file for the NeoPixel Ring.

```
#include <Adafruit_NeoPixel.h>
```

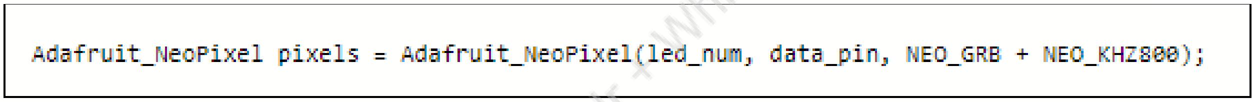
b. Define a constant which will hold the port number in which the NeoPixel ring is connected to.

```
const byte pir_pin= 2;
```

Define another constant which will hold the number of LEDs in the NeoPixel ring.

```
const byte led_num= 16;
```

c. Create an instance of Adafruit_NeoPixel and name it as pixels.



```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(led_num, data_pin, NEO_GRB + NEO_KHZ800);
```

d. Call two methods in the setup() method-

- **pixels.begin()** - prepares the data pin for NeoPixel output.
- **pixels.show()** - will initialize all pixels to "off".

e. Define a new method named **generate_random_pattern()**. Within the function call the following functions-

- **setPixelColor(index, r, g, b)** - This function helps to light up a certain LED on the NeoPixel Ring. Parameters-
  a. **index**- LEDs can be accessed by their indices which start from 0.
  b. **r**- this indicates red. Value can be between 0-255.
  c. **g**- this indicates green. Value can be between 0-255.
  d. **b**- this indicates blue. Value can be between 0-255.

  Let's say that we want to light up the LED with 0 index with red color. Then, we will write-

  ```
  pixels.setPixelColor(0,255,0,0);
  ```

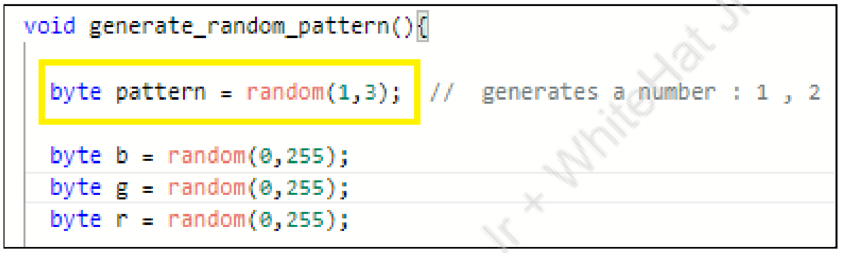
- To "push" the color data to the strip, call the show() method again.

```
pixels.show();
```

6. Call the **generate_random_pattern()** method inside the **loop()** method, when the PIR sensor detects a motion.

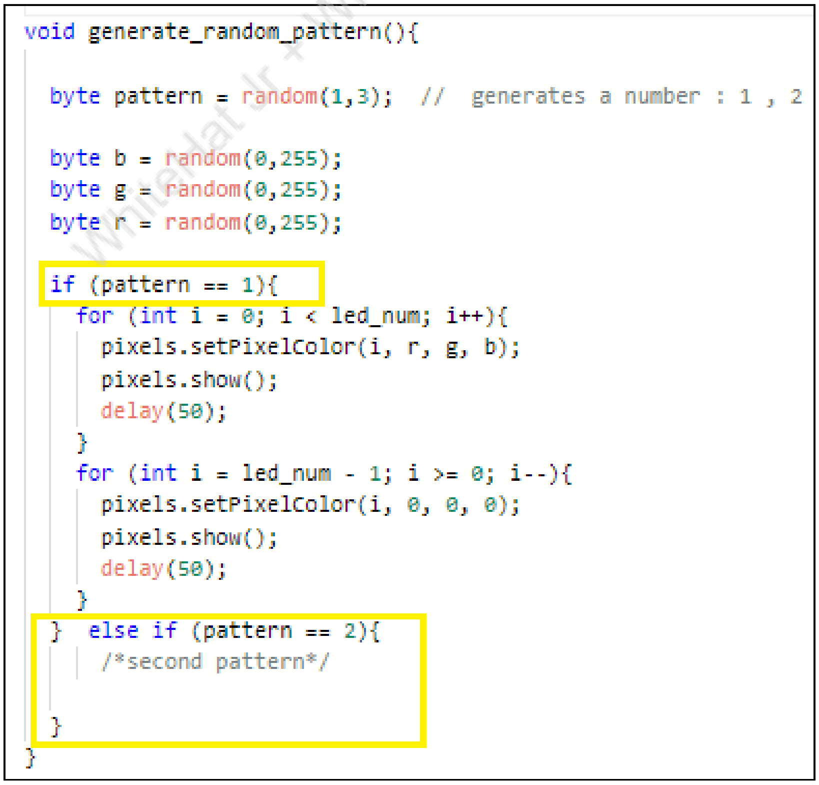
```
if(digitalRead(pir_pin)){
  generate_random_pattern();
}
```

7. Generate a random pattern depending on a random number.
   a. Generate a random number:



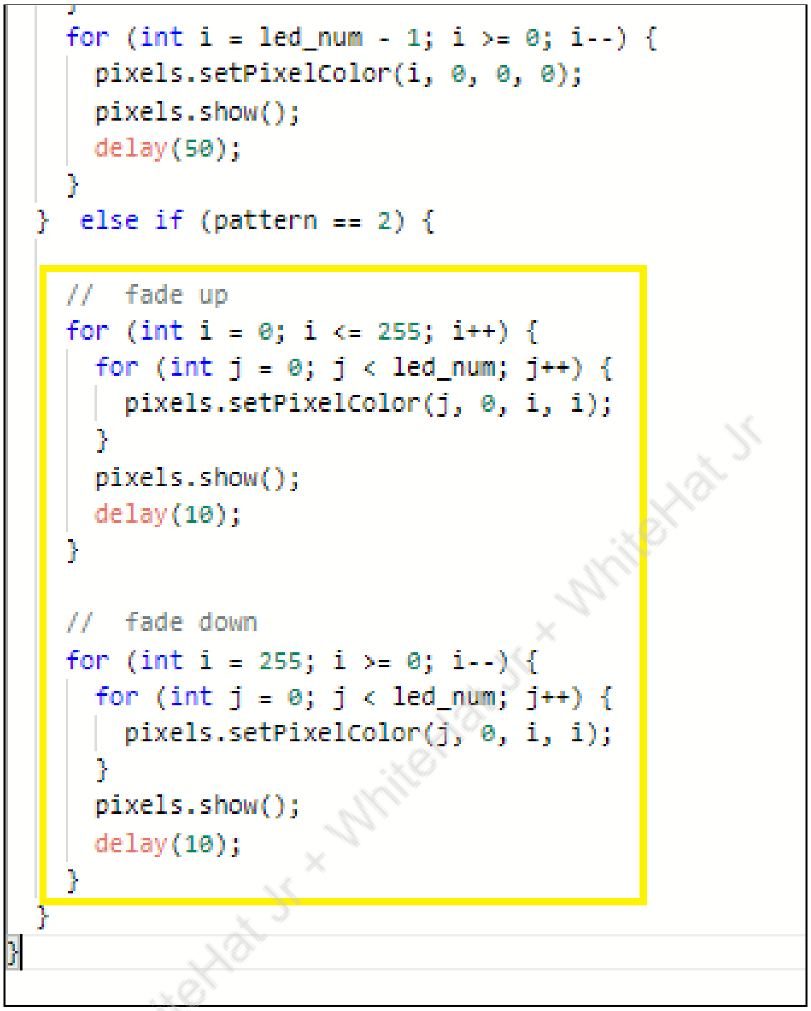
```
void generate_random_pattern(){

  byte pattern = random(1,3);  //  generates a number : 1 , 2

  byte b = random(0,255);
  byte g = random(0,255);
  byte r = random(0,255);
```

   b. Now, we want to generate the first pattern when the random number is 1. We will generate a new pattern when the random number is 2.



```
void generate_random_pattern(){

  byte pattern = random(1,3);  //  generates a number : 1 , 2

  byte b = random(0,255);
  byte g = random(0,255);
  byte r = random(0,255);

  if (pattern == 1){
    for (int i = 0; i < led_num; i++){
      pixels.setPixelColor(i, r, g, b);
      pixels.show();
      delay(50);
    }
    for (int i = led_num - 1; i >= 0; i--){
      pixels.setPixelColor(i, 0, 0, 0);
      pixels.show();
      delay(50);
    }
  } else if (pattern == 2){
    /*second pattern*/

  }
}
```

c. For the second pattern, write code for fade up and fade down effect:



```
    }
    for (int i = led_num - 1; i >= 0; i--) {
      pixels.setPixelColor(i, 0, 0, 0);
      pixels.show();
      delay(50);
    }
  }  else if (pattern == 2) {

    //  fade up
    for (int i = 0; i <= 255; i++) {
      for (int j = 0; j < led_num; j++) {
        pixels.setPixelColor(j, 0, i, i);
      }
      pixels.show();
      delay(10);
    }

    //  fade down
    for (int i = 255; i >= 0; i--) {
      for (int j = 0; j < led_num; j++) {
        pixels.setPixelColor(j, 0, i, i);
      }
      pixels.show();
      delay(10);
    }
  }
}
```

What's NEXT?

In the **next class**, we will learn about **ultrasonic sensors** and we will create an exciting project with it.

Expand Your Knowledge

To know more about **keypads** on wokwi**,** click here.