

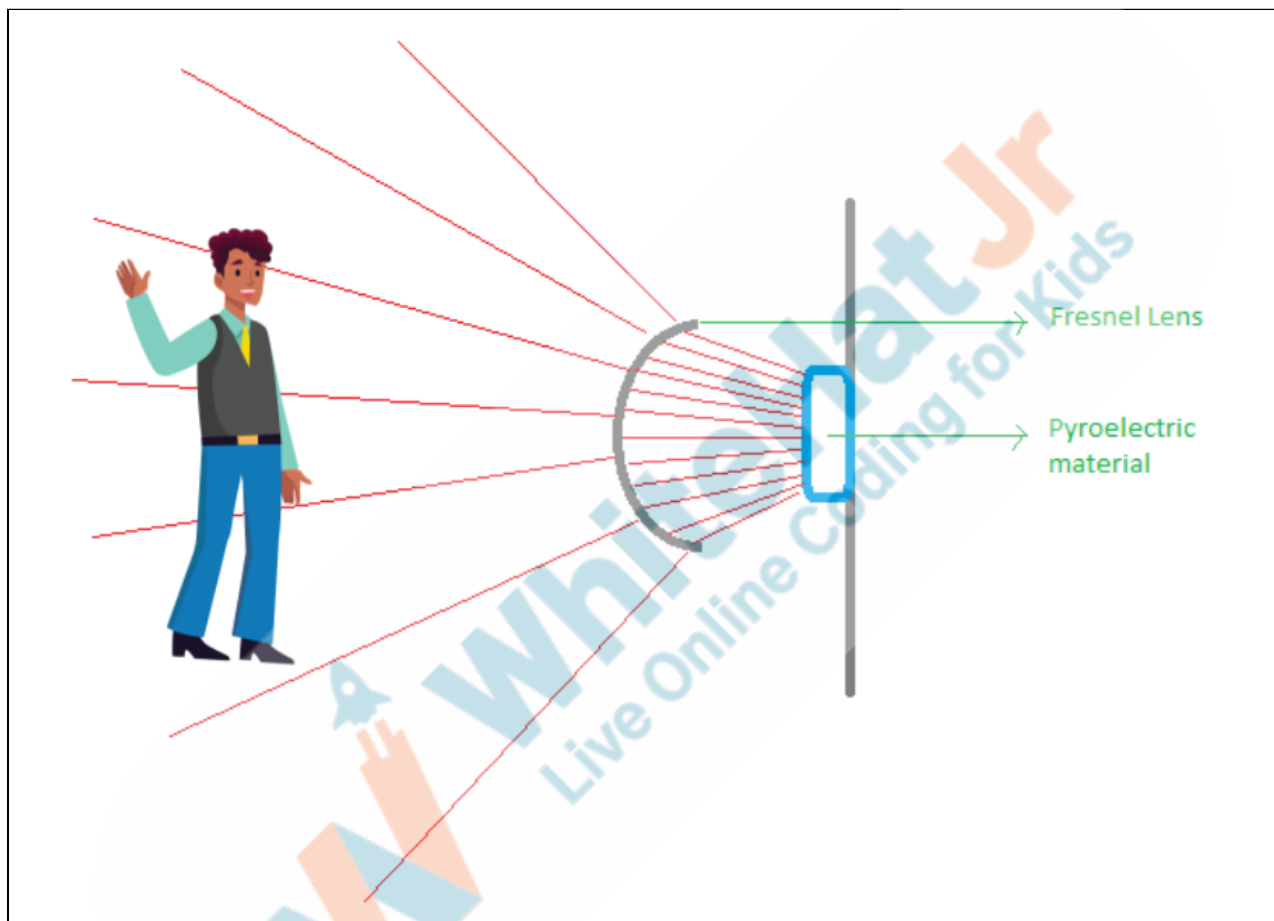
Topic	GESTURE CONTROLLED LIGHT	
Class Description	Students will learn about sensors. Students will also learn how to use a PIR motion sensor. Additionally, they will learn how to generate random patterns on a NeoPixel Ring.	
Class	PRO C262	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> Understanding sensors. Introduction to PIR sensor. Understanding NeoPixel Ring. Build a program to light up the NeoPixel Ring when a motion is detected. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Smartphone Student Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen 	
Class structure	Warm-Up Teacher-Led Activity Student-Led Activity Wrap-Up	10 mins 15 mins 15 mins 10 mins
WARM-UP SESSION - 10 mins		
Teacher Action		Student Action
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?		ESR: Hi, thanks! Yes, I am excited about it!

<p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Greet the student. • Revision of previous class activities. • Quizzes. 	<p>Click on the slide show tab and present the slides</p>
<p align="center">WARM-UP QUIZ Click on In-Class Quiz</p>	
<p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. 	
<p align="center">TEACHER-LED ACTIVITY 15mins</p>	
<p align="center">Teacher Initiates Screen Share</p>	
<ul style="list-style-type: none"> • PIR sensor connections. • Algorithm to light up an LED from NeoPixel Ring. 	
Teacher Action	Student Action
<p>Do you remember what we did in the previous class?</p> <p>Can you tell me about servo motors?</p> <p><i>If the student has any doubts, clarify the doubts.</i></p>	<p>ESR: We learned about Servo motors.</p> <p>ESR: These are special motors which can rotate 180 degrees. It is used in robotic arms, legs etc.</p>

Have you seen an automated door in a shopping mall?	ESR: Yes.
How does it know when someone walks in?	ESR: It can detect the person. Once it detects the person, it opens the door.
Great! So, it can detect its surroundings. How do we, human beings, detect our surroundings?	ESR: We can see our surroundings with our eyes. We can hear sound with our ears.
Exactly! We have 5 senses which help us detect our surroundings.	
Similarly, sensors help machines to detect their surroundings.	
Our senses help us to see, hear or detect touch etc. Sensors in machines can detect some physical parameters like - heat, light sound etc.	
In automated doors, there are special sensors called motion sensors . As the name suggests, these sensors help us detect motion.	
When a person walks towards an automated door, the motion sensor detects the person and sends this information to the microcontroller and this trigger opens the door.	

<p>We will learn about one such motion sensor today. It is called the PIR sensor.</p> <p><u>PIR sensor:</u></p> <p>We are going to use a sensor named PIR sensor. Do you know what a PIR sensor is?</p> <p>PIR sensor stands for Passive infrared sensor. PIR sensor is actually a motion sensor. But it can only detect a living being. It is unable to detect non-living objects.</p> <p>We will understand why it cannot detect non-living objects once we understand how it works.</p>	<p>ESR: Varied.</p>
<p><u>Working principle of PIR sensor:</u></p> <p>PIR sensors use pyroelectric sensors.</p> <p>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. These materials can detect levels of infrared radiation.</p> <p>Every living being emits some low levels of radiation. The hotter something is, the more radiation it emits.</p> <p>The PIR sensor can detect the radiation when a living being moves into the range of the PIR sensor.</p>	

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.



Students can open this image from [Student activity 2](#).

Any idea which machines might use these sensors?

Yes, very good. These sensors can be used in security systems for homes and offices. It can be used for automated doors, escalators, automated light switches and many other machines. These sensors are small and

ESR: In security systems.

inexpensive, so they are widely used in different machines.

Let's try to add a PIR sensor and ESP32 controller in the wokwi simulator.

Go to the [wokwi](https://wokwi.com) simulator and create a new ESP32 project,

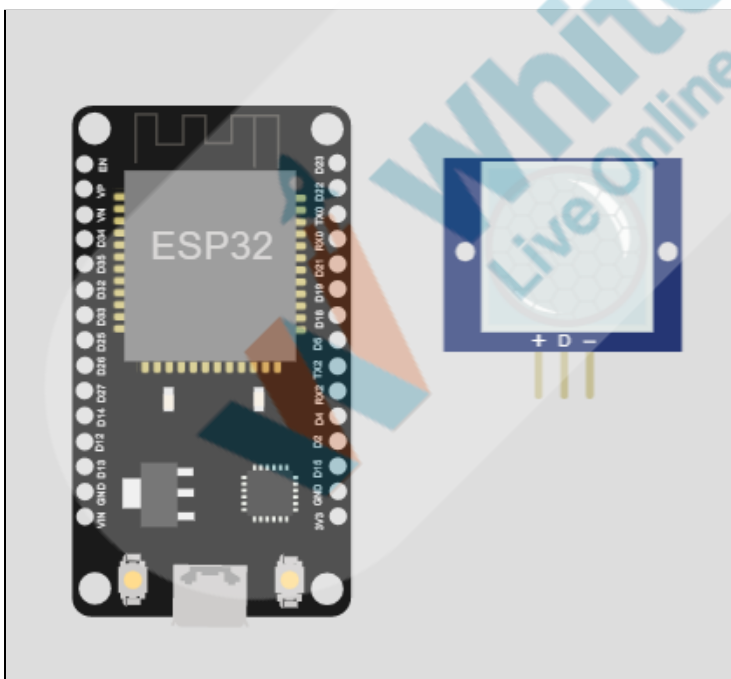
Let's try to create the circuit first.

Step -1: Select the components

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

Step -2: Let's do connections:

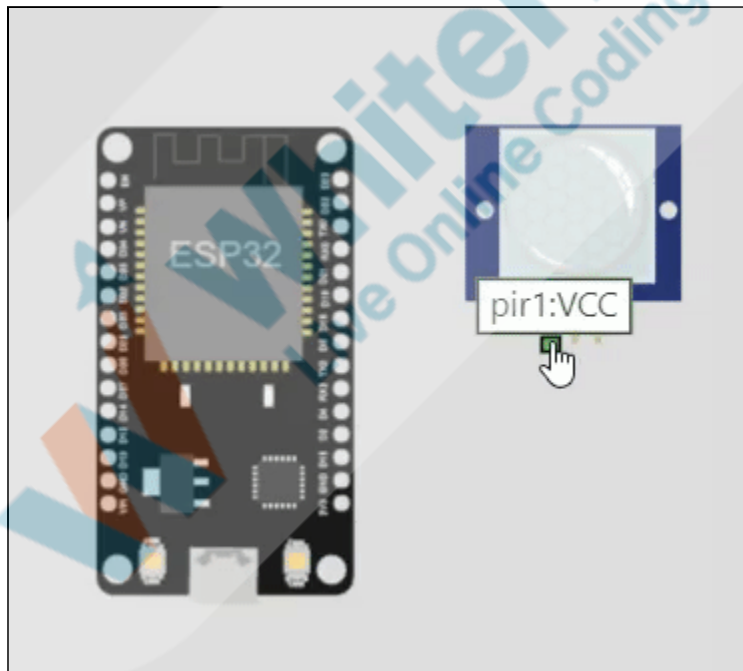
The circuit of this project consists of an **ESP32** Controller, and a **PIR Motion Sensor**.



- Select **PIR Motion Sensor** and connect as per the below instructions:

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

*Note: Wire color can be changed by **clicking over it** and selecting the color, or via **diagram.json** file. Go to the **diagram.json** wire and change the color of the wire. Any design changes or color changes can be done via the **diagram.json** file. Keep the track of the component and change the design settings.*



[Click here](#) to view the reference video.

Now it's time to write the code. For that, first go to the **sketch.ino** file and write the code.

1. We will first define a constant which will hold the port number in which the PIR sensor is connected to.

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

2. 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.

- Go to **setup()** method and define pin mode.

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

3. Now, let's check if our **PIR sensor** is working or not. To do that, we need two methods-

- **digitalRead()** - method reads the value from a specified digital pin. In this case, we want to read the pin which is connected to the PIR sensor. So, we can write-

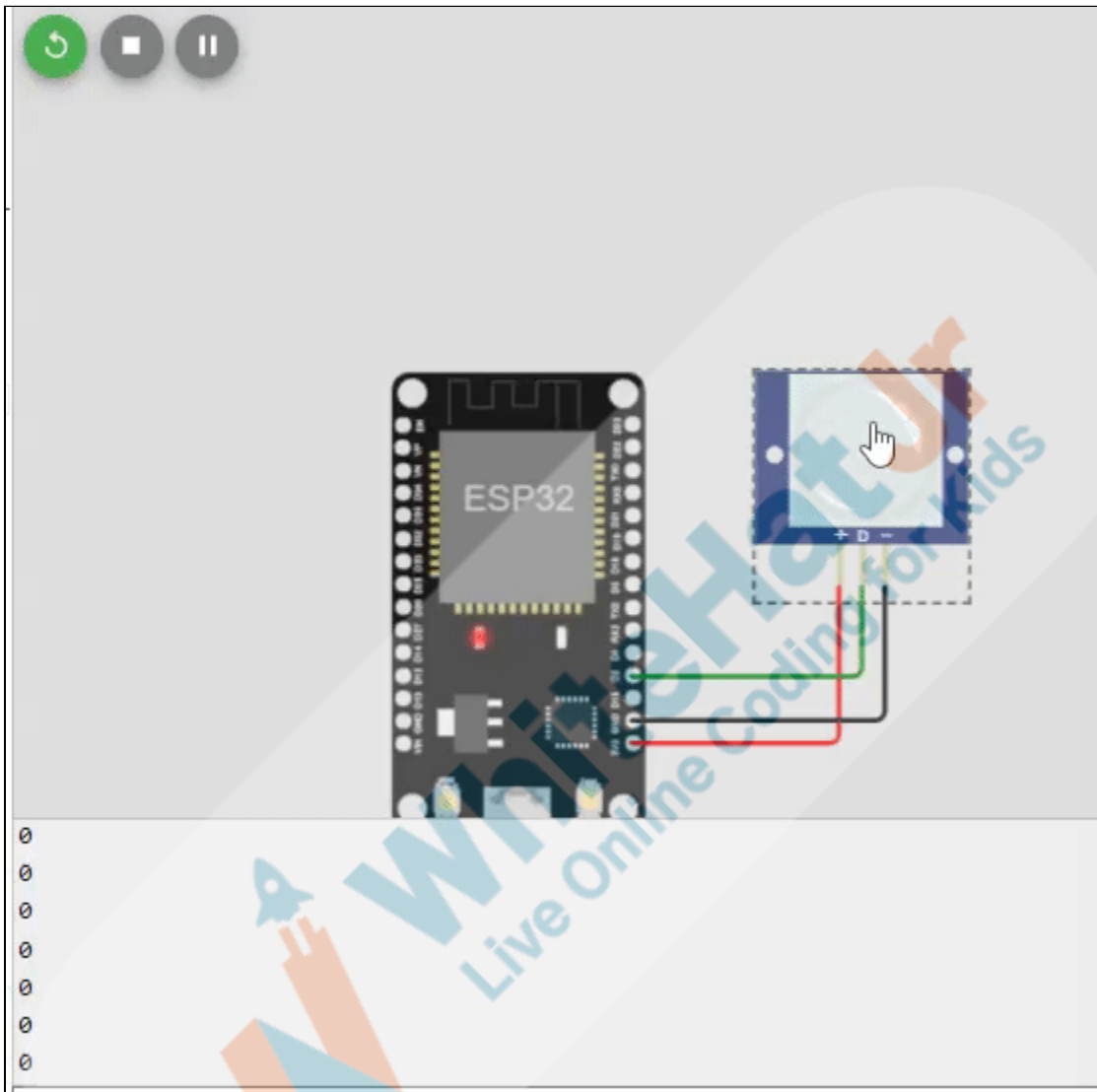
```
digitalRead(pir_pin);
```

- **Serial.println()** - method will print the value that we pass through this method. In this case, we want to see the value returned by the **PIR sensor**. Hence, we can write-

```
Serial.println(digitalRead(pir_pin);
```

We will write this line of code inside the **loop()** method. It will make sure that we read the **PIR sensor's** output continuously.

Let's observe the output now.



When we run this code, we can see that it prints a steady stream of 0's. 0 means our PIR sensor has not sensed any motion yet.

Now, click on the PIR sensor and you will see a pop-up window. Here, you have to click on the Simulate motion button.

In real life, when a living being moves near the PIR sensor

it detects the movement. Here, we are using a simulator. The Simulate motion button simulates the same effect (movement of a living being) on the simulator.

Once you click the Simulate **motion** button, you will see a stream of 1's being generated for some time. So, the **PIR sensor** returns **1 for a certain time** when a motion is detected. This time lapse for which 1 should be generated can be changed.

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 value will make our project wait for 5 seconds before it can detect a motion again. That's why we changed it 1.)



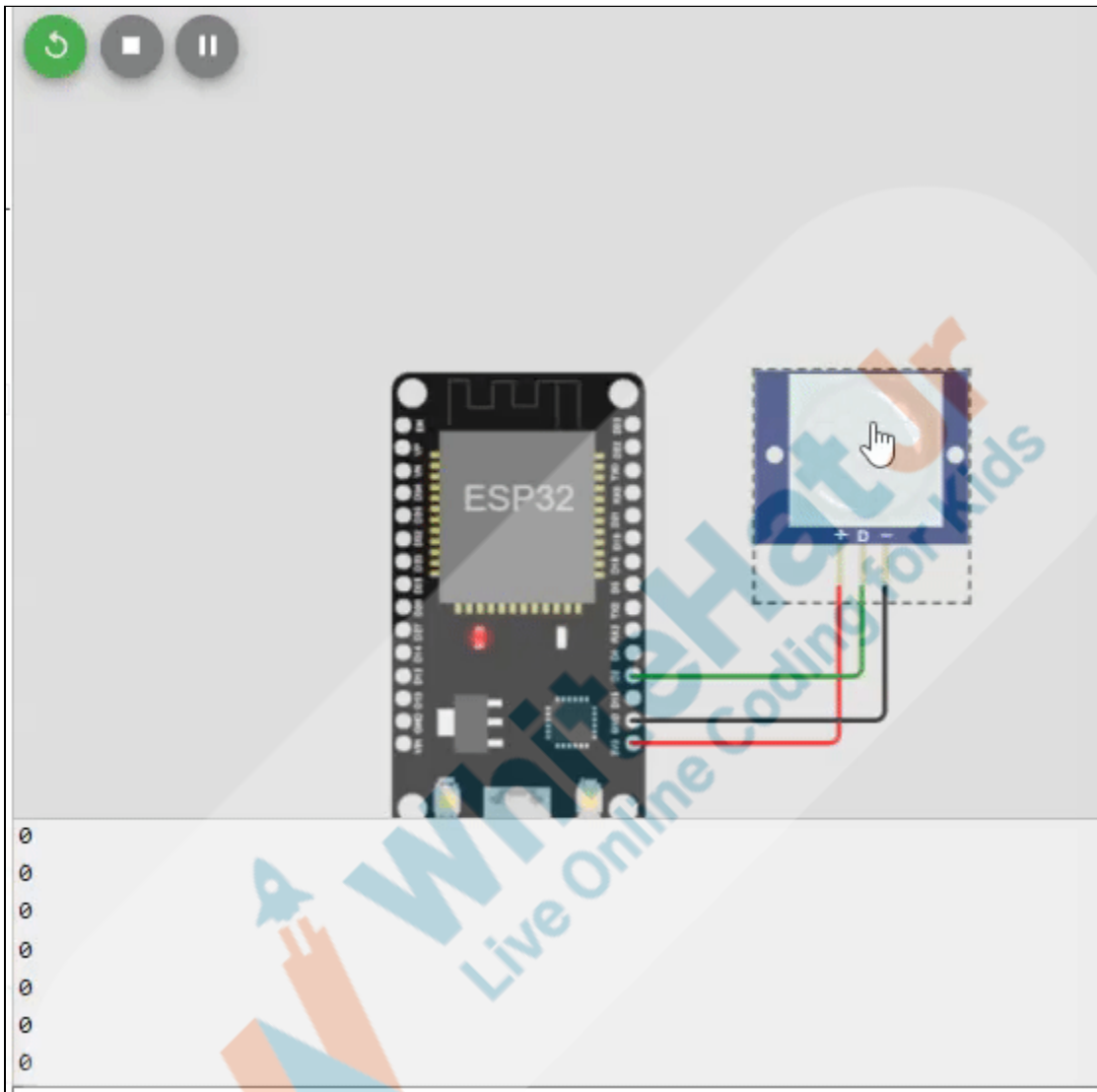
```
4  "editor": "wokwi",
5  "parts": [
6    { "type": "wokwi-esp32-devkit-v1", "id": "esp",
7      {
8        "type": "wokwi-pir-motion-sensor",
9        "id": "pir1",
10       "top": 21.96,
11       "left": 142.49,
12       "attrs": { "delayTime": "1" }
13     },
14   ],
```

We will also add another property which will help us plot our output on a graph.

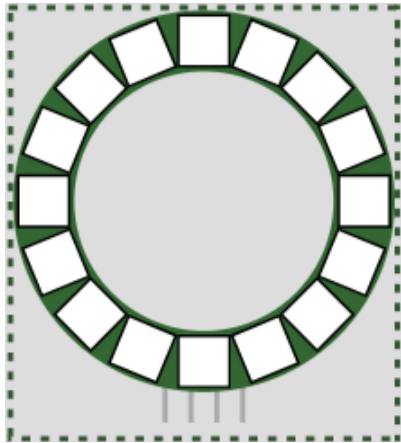
Add another property called "**serialMonitor**" at the end of the diagram.json

```
sketch.ino  diagram.json ●  Library Manager ▼
6      { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0,
7      {
8        "type": "wokwi-pir-motion-sensor",
9        "id": "pir1",
10       "top": 21.96,
11       "left": 142.49,
12       "attrs": { "delayTime": "1" }
13     }
14   ],
15   "connections": [
16     [ "esp:TX0", "$serialMonitor:RX", "", [ ] ],
17     [ "esp:RX0", "$serialMonitor:TX", "", [ ] ],
18     [ "pir1:VCC", "esp:3V3", "red", [ "v0" ] ],
19     [ "pir1:OUT", "esp:D15", "green", [ "v0" ] ],
20     [ "pir1:GND", "esp:GND.1", "black", [ "v0" ] ]
21   ],
22   "serialMonitor": {
23     "display": "plotter"
24   }
25 }
```

It will give us the following output.



Now, depending on the PIR sensor's output we want to light up a neopixel ring.

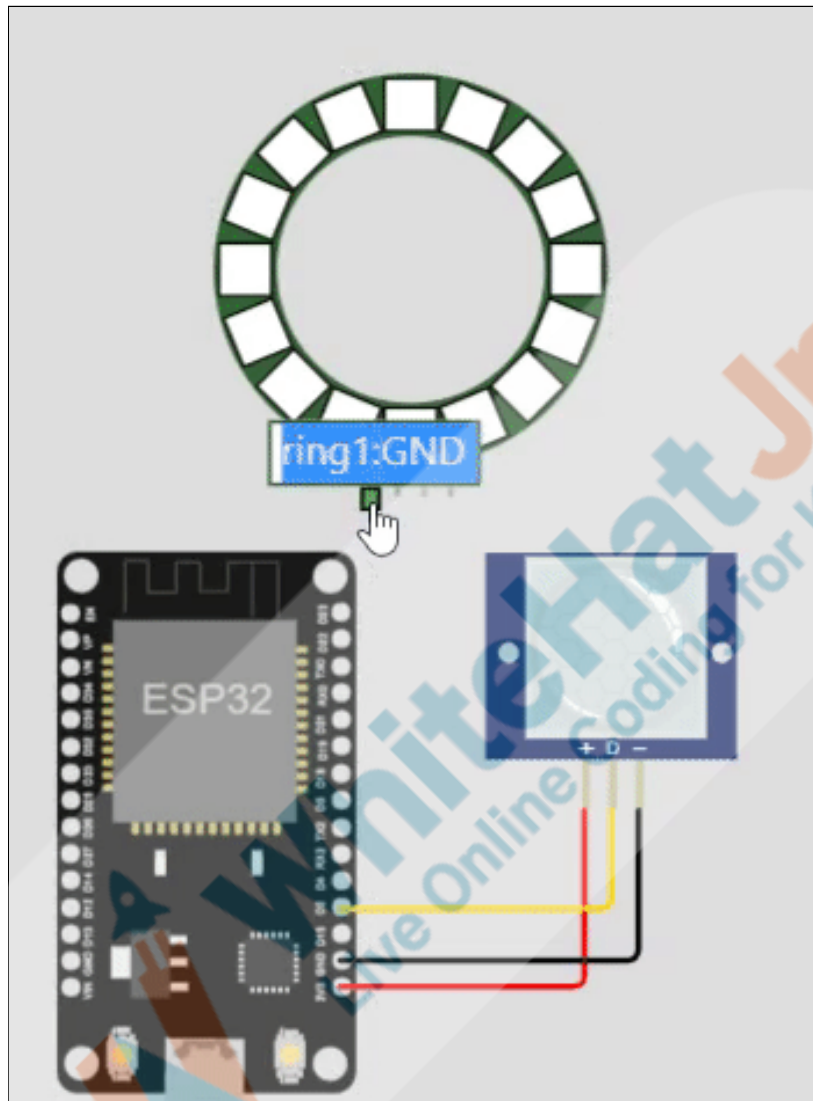


NeoPixel ring is a chain of 16 LED lights connected together in the shape of a ring. These lights can be controlled by one input pin.

Let's make the connections,

- Connect as per the below instructions:

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.

Now it's time to write the code to light up the NeoPixel LEDs. For that, go to the **sketch.ino** file and write the code.

1. At first include the header file for the NeoPixel Ring.

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

2. We will define a constant which will hold the port number in which the NeoPixel ring is connected to.

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

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

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

3. Now, we will create an instance of Adafruit_NeoPixel and we will name it as pixels.

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

4. Now, we will need to call two methods in the setup() method-

- **pixels.begin()** - prepares the data pin for NeoPixel output.

```
pixels.begin();
```

- **pixels.show()** - will initialize all pixels to "off".

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

5. 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**- Our 16 LEDs can be accessed by their indices which lie between 0-15.
 - 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();
```

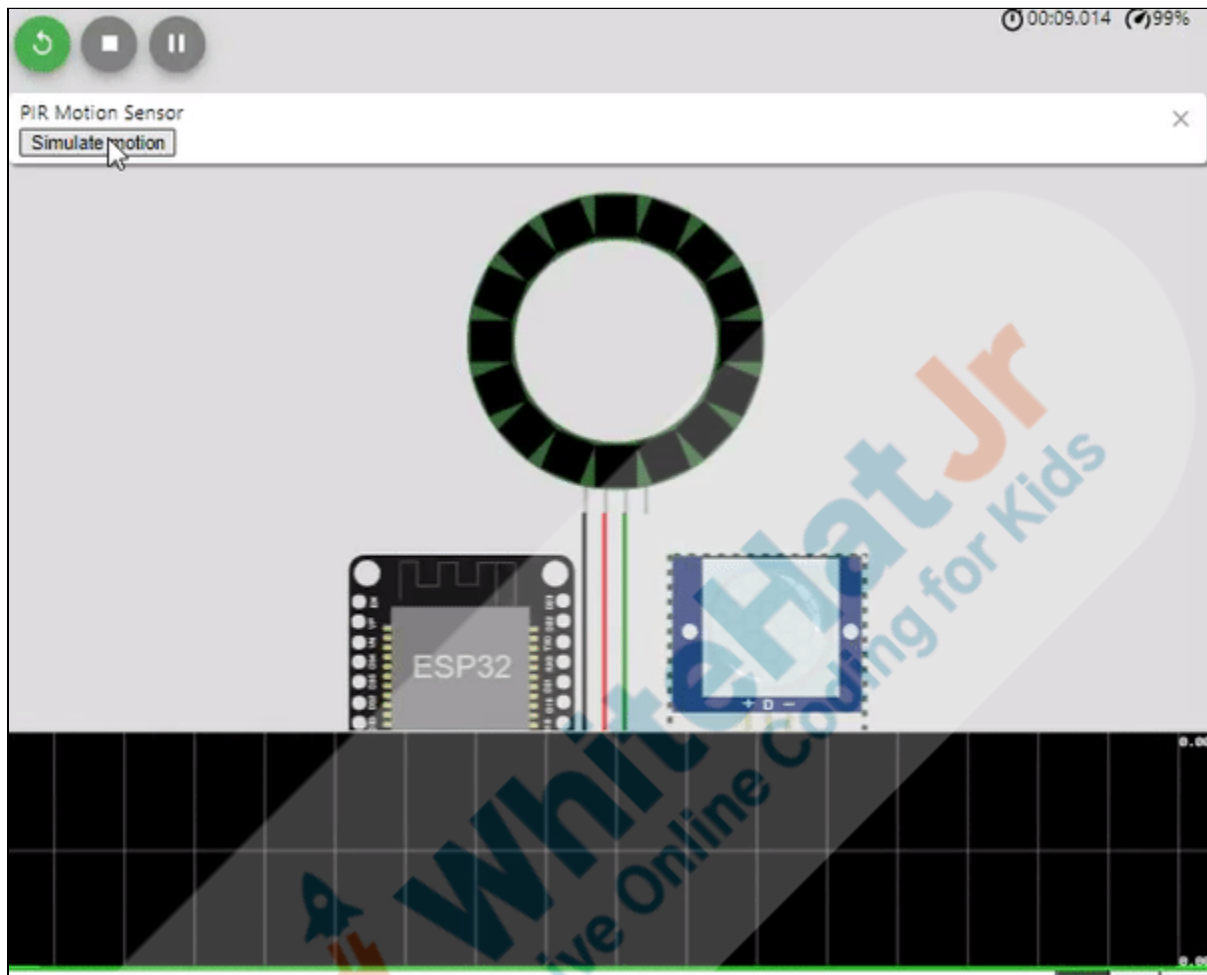
6. Call the **generate_random_pattern()** method inside the **loop()** method, when the PIR sensor detects a motion.

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

Your code should look like this-


```
1  #include <Adafruit_NeoPixel.h>
2
3  const byte pir_pin= 15;
4  const byte data_pin= //enter pin number which connects to the DIN of NeoPixel Ring here.
5  const byte led_num= 16;
6
7  Adafruit_NeoPixel pixels = Adafruit_NeoPixel(led_num, data_pin, NEO_GRB + NEO_KHZ800);
8
9  void setup() {
10     Serial.begin(115200);
11
12     pinMode(pir_pin, INPUT);
13
14     pixels.begin();
15     pixels.show();
16 }
17
18 void loop() {
19     Serial.println(digitalRead(pir_pin));
20
21     if(digitalRead(pir_pin)){
22         generate_random_pattern();
23     }
24
25     delay(10); // this speeds up the simulation
26 }
27
28 void generate_random_pattern(){
29     pixels.setPixelColor(0,255,0,0);
30     pixels.show();
31 }
```

Let's observe the output.



Here, when the green line is flat — it means 0 or PIR sensor has not detected anything.
When the green line goes up — it means 1 or PIR sensor has detected something.

Student Stops Screen Share

We have one more class challenge for you.
Can you solve it?

Let's try. I will guide you through it.

ESR: Yes, sure!

STUDENT-LED ACTIVITY- 15 mins

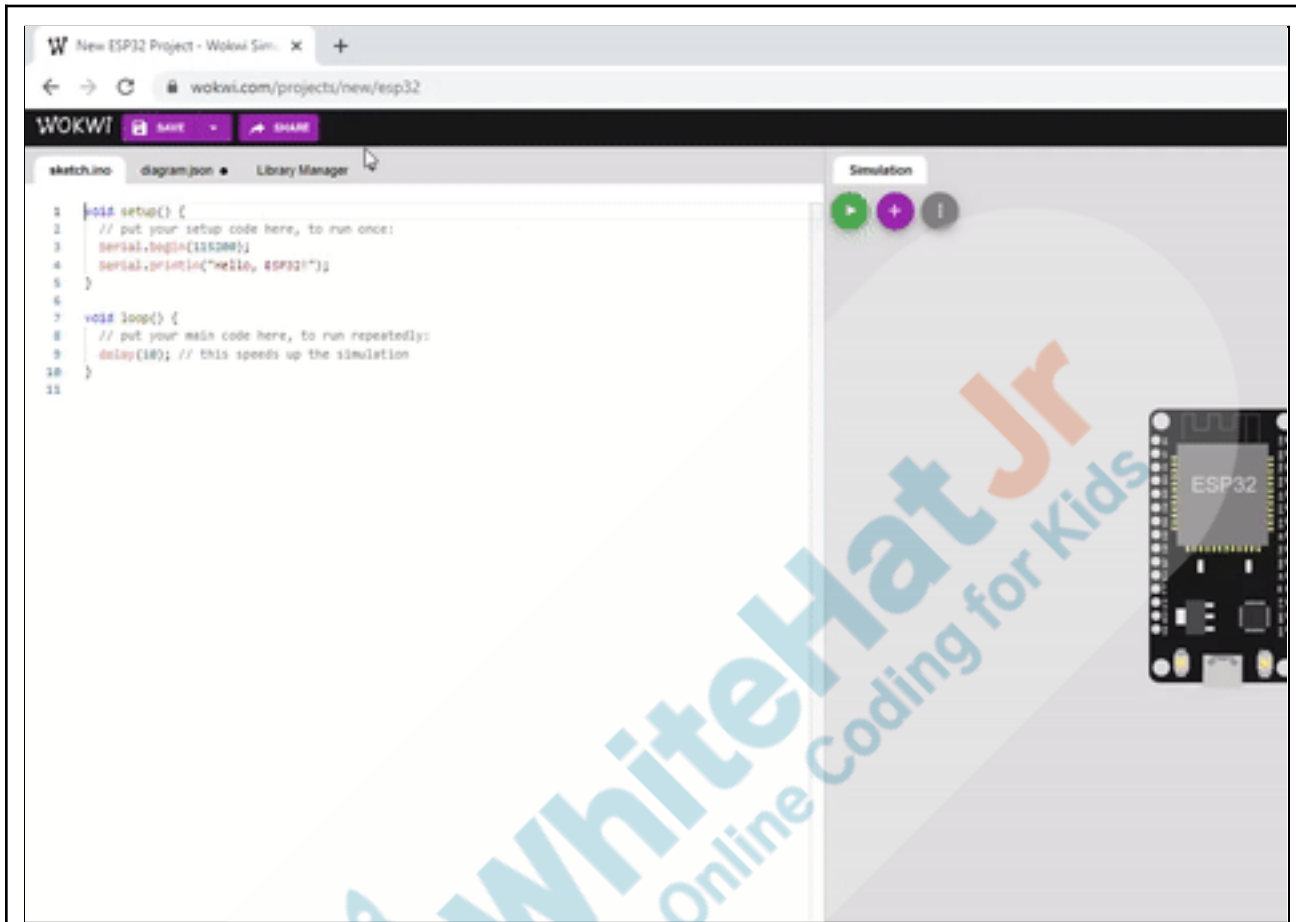
- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.
- The teacher gets into Full Screen.

Student Initiates Screen Share

ACTIVITY

- Connection between the NeoPixel Ring and the ESP32 controller.
- Algorithm to light up the NeoPixel Ring with different patterns.

Teacher Action	Student Action
<i>Teacher guides the student to download boilerPlate code from Student activity 3.</i>	Student opens wokwi simulator.
<i>Unzip the downloaded file and upload it in your wokwi project folder.</i>	



[Click here](#) to view the reference video.

Let's try to create the circuit diagram first.

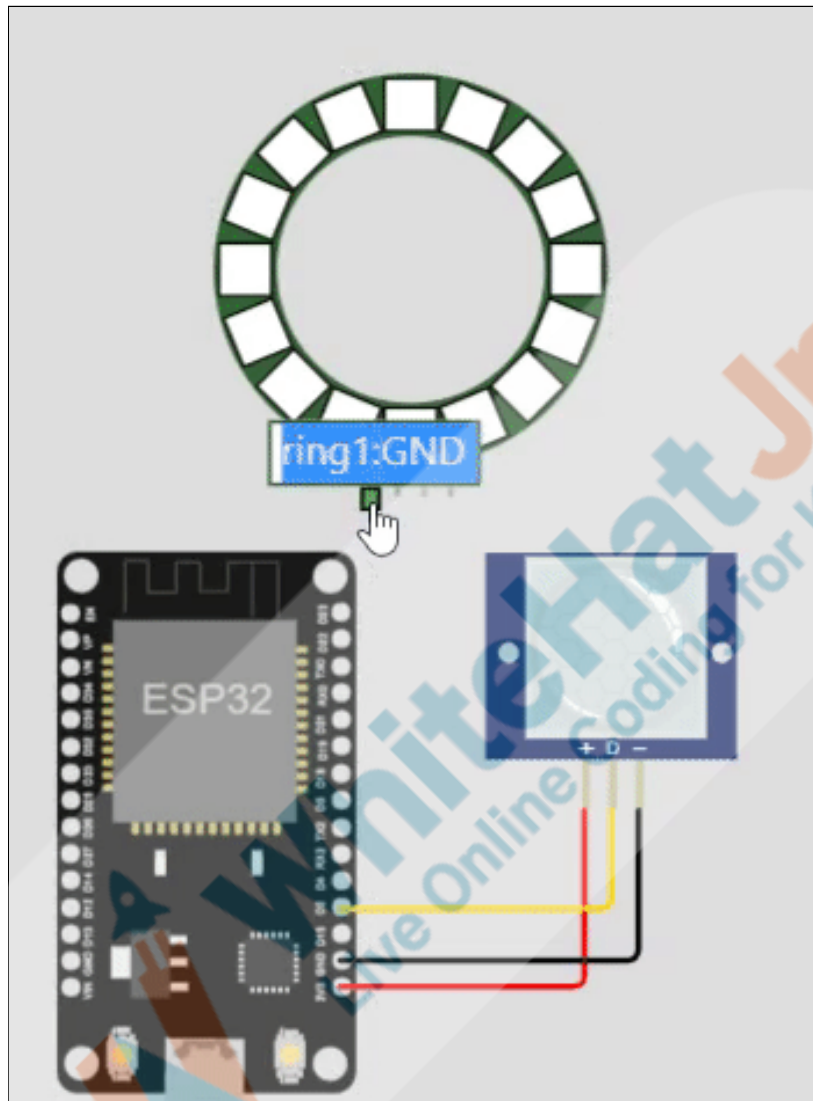
Step -1: Make the connection between the NeoPixel LED and ESP32 controller.

- Connect as per the below instructions:

KEYPAD	ESP32 PIN
GND	GND.1

VCC	3V3
DIN	GPIO D15 (It can be connected to any GPIO port)

*Note: Wire color can be changed by **clicking over it** and selecting the color, or via **diagram.json** file. Go to the diagram.json wire and change the color of the wire. Any design changes or color changes can be done via the diagram.json file. Keep the track of the component and change the design settings.*



[Click here](#) to view the reference video.

Let's work on the code now.

What should we do first?

Exactly, update the `data_pin` to 15. (If you connect the **DIN** pin of the **NeoPixel Ring** to D15. Else, update according to

ESR: We need to assign 15 to the `data_pin` constant.

<p>the GPIO port, you have connected it to.)</p> <p>What should we do next?</p> <p>How do you think we can do that?</p> <p>What if we want to light up all the LEDs at once.</p> <p>Can we do it with lesser lines of code?</p> <p>Great! Let's write the code then. We will write it in generate_random_pattern() method.</p> <p><i>Student write the code to light up all the LEDs. Teacher helps when required.</i></p>	<p>ESR: We can try lighting up the other LEDs in the NeoPixel Ring.</p> <p>ESR: We can change the index parameter in the setPixelColor() method.</p> <p>ESR: We need to write the setPixelColor() method 16 times.</p> <p>ESR: Yes. We can use a loop.</p>
---	---

```
void generate_random_pattern(){  
    for (int i = 0; i < led_num; i++){  
        pixels.setPixelColor(i, 255, 0, 0);  
        delay(50);  
    }  
    pixels.show();  
}
```

If we put the **pixels.show()** function outside the for loop, it will light up all the LEDs at once.

And if we put the **pixels.show()** function outside the for loop, it will light up LEDs one by one.

```
void generate_random_pattern(){  
    for (int i = 0; i < led_num; i++){  
        pixels.setPixelColor(i, 255, 0, 0);  
        pixels.show();  
        delay(50);  
    }  
}
```

We will use this one in our project.

We will also want to generate a random color, every time the LED lights up. For that, let's declare 3 variables named **r**, **g**, **b** and assign a random value between 0 to 255.

Use these **r**, **g**, **b** variables as parameters in the **setPixelColor()** function.


```
void generate_random_pattern(){  
  
    byte b = random(0,255);  
    byte g = random(0,255);  
    byte r = random(0,255);  
  
    for (int i = 0; i < led_num; i++){  
        pixels.setPixelColor(i, r, g, b);  
        pixels.show();  
        delay(50);  
    }  
}
```

We also want to make it go off in the reverse order. So, we will write a for loop.

```
void generate_random_pattern(){  
  
    byte b = random(0,255);  
    byte g = random(0,255);  
    byte r = random(0,255);  
  
    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);  
    }  
}
```

So, we have generated one pattern till now. We want to generate a random pattern depending on a random number.

1. Generate a random number:

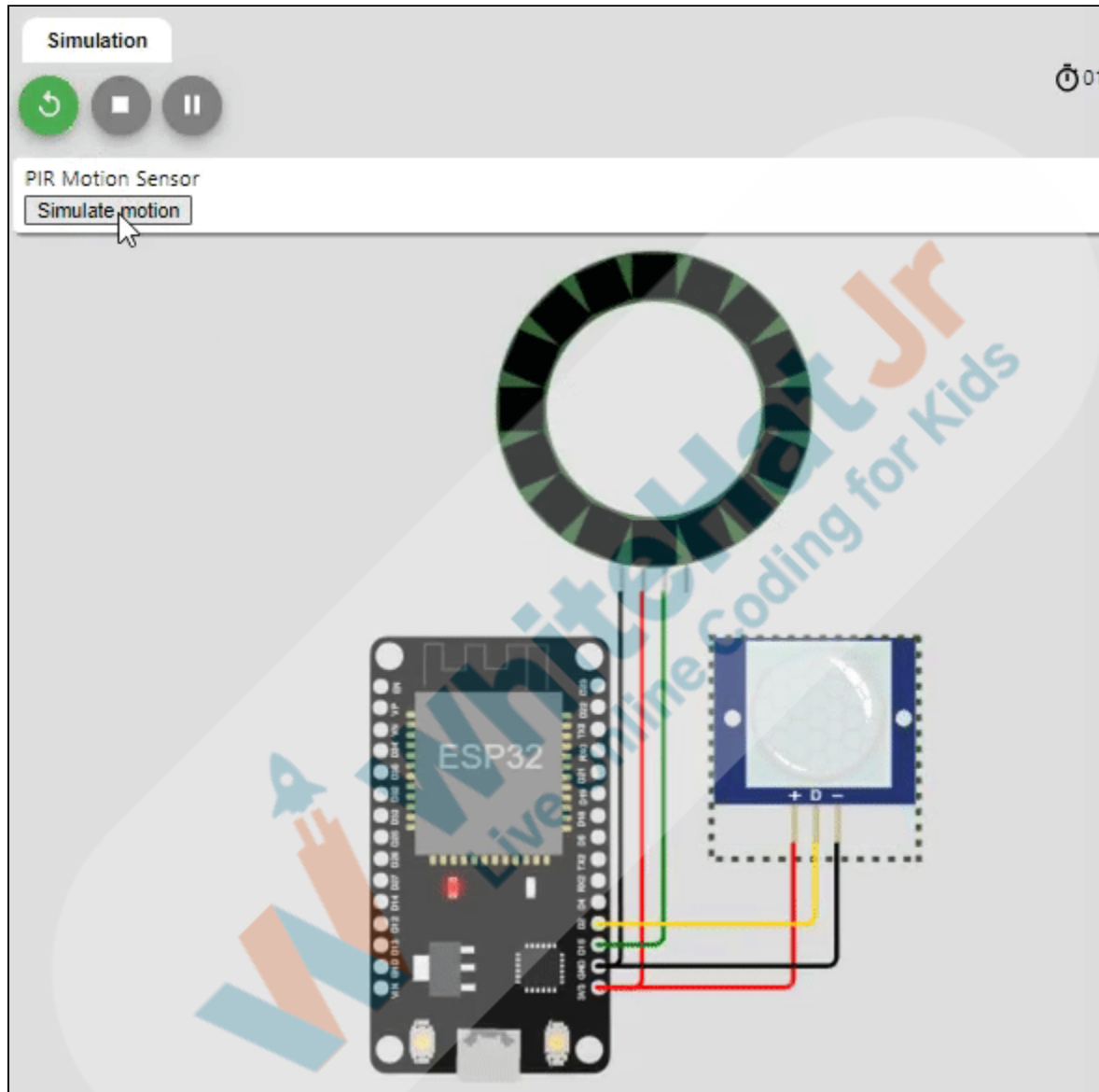
```
void generate_random_pattern(){  
    byte pattern = random(1,3); // generates a number : 1 , 2  
  
    byte b = random(0,255);  
    byte g = random(0,255);  
    byte r = random(0,255);  
}
```

2. 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*/  
    }  
}
```

3. Our second pattern will be a fade up and fade down pattern.

This is how it should look like-



Student opens [Student Activity 4](#) to view this effect.

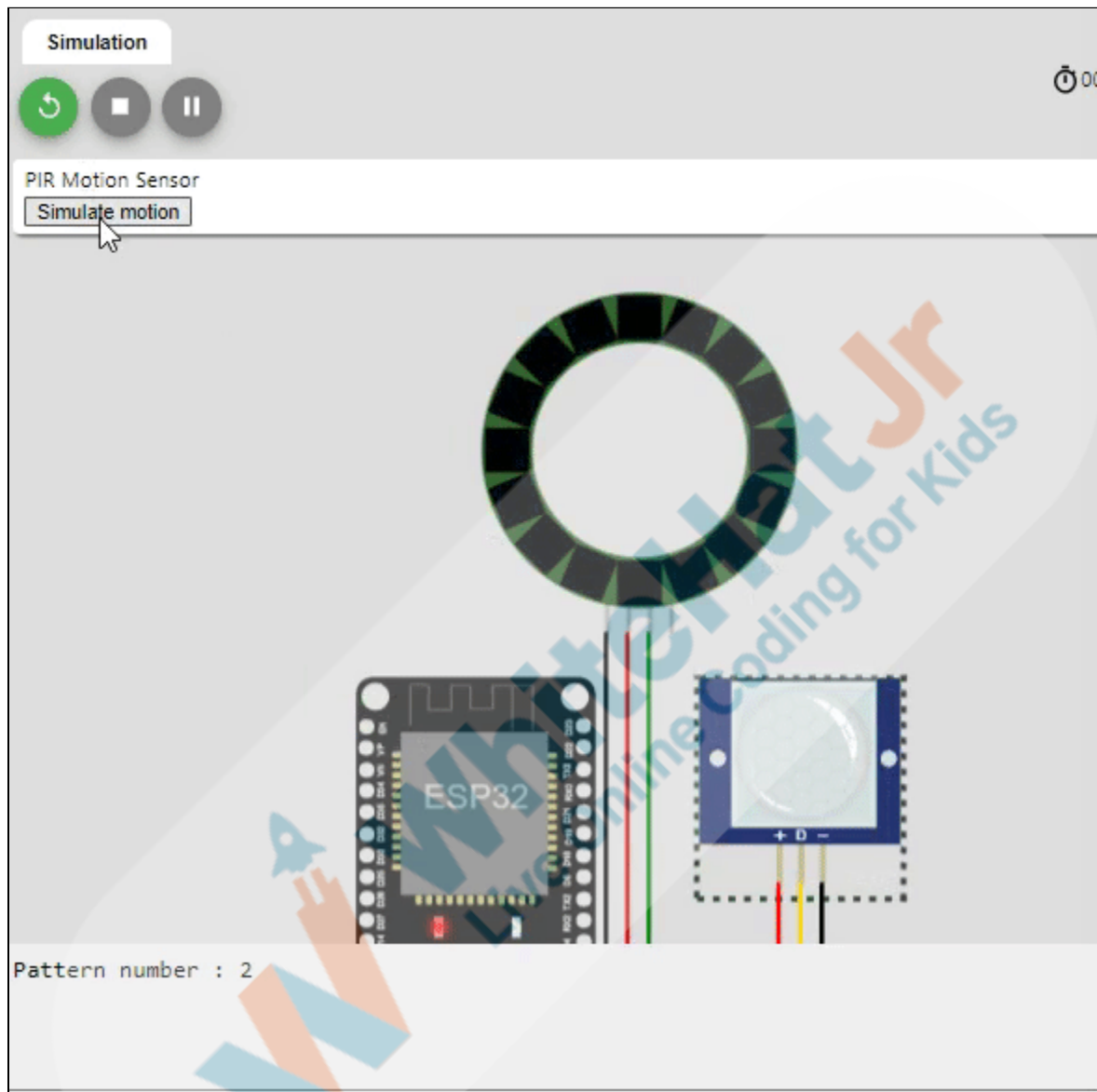
Try writing the code for it.

Note : Let the student try and write the code. Help when required.

ESR: Sure!

```
    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);  
    }  
}  
}
```

Reference Output:



[Click here](#) to view the reference video.

Great, we have now completed our project with PIR sensor and NeoPixel Ring.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ

Click on In-Class Quiz




Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
<p>You get “hats-off” for your excellent work!</p> <p>In the next class, we will learn about ultrasonic sensors and we will create an exciting project with it.</p>	<p><i>Make sure you have given at least 2 hats-off during the class for:</i></p> <div> <div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div> </div>

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks

✕ End Class

ADDITIONAL ACTIVITIES

(Optional)

Additional Activities

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Simulator	https://wokwi.com/
Teacher Activity 2	PIR motion sensor documentation	https://docs.wokwi.com/parts/wokwi-pir-motion-sensor
Teacher Reference 1	Reference Code	https://github.com/procodingclass/PRO-C262-Reference-Code
Teacher Reference 2	Project	https://s3-whjr-curriculum-uploads.whjr.online/ee9f0029-5245-44a1-b2ec-aa8539243f63.pdf
Teacher Reference 3	Project Solution	https://github.com/procodingclass/PRO-C262-Project-Solution
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/ed88afed-0c5c-4d84-bf29-83c917f219ba.pdf
Student Activity 1	Simulator	https://wokwi.com/
Student Activity 2	PIR sensor	https://s3-whjr-curriculum-uploads.whjr.online/962d790c-3e1e-4c62-8b57-80bbca6d9b68.png

Student Activity 3	Student Activity Boilerplate code	https://github.com/procodingclass/PRO-C262-Student-Boilerplate
Student Activity 4	Fading pattern reference	https://s3-whjr-curriculum-uploads.whjr.online/0f4f1110-a2da-450e-94d4-cee3f1c823cb.gif

