REMOTE CONTROLLED LIGHT

What is our GOAL for this CLASS?

In this class, we learned about IR receivers. We also learned how remote control devices transmit messages to IR receivers. Using this knowledge, we have created a project to control an RGB LED remotely.

What did we ACHIEVE in the class TODAY?

- Understood IR receivers' working principle.
- Decoded messages received by IR receivers from a remote control device.
- Understood how RGB LEDs function.
- Built a program to light up the RGB LED when a signal is detected.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Concepts : IR receivers, remote control, conditional statements.
- Coding blocks : **if-else** statements, **functions**.

How did we DO the activities?

1. Working principle of **Remote control** devices:

   A Remote control flashes patterns of light which transmits our messages towards the target devices. But this light emitted by the remote control cannot be seen. It's called infrared light. Infrared light is beyond the visible light range.

   When different buttons are pressed on the remote control, it sends different signals.

2. Create the circuit:

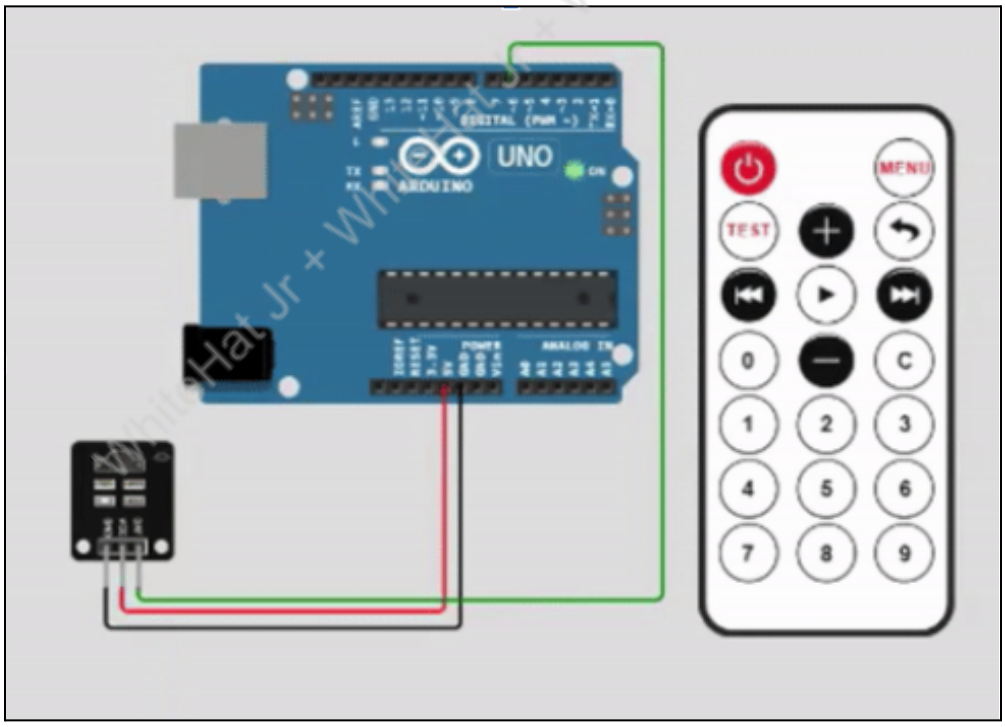
   a. Components:

- 1 x **Arduino Uno board**
- 1 x **IR Receiver**
- 1 x **IR Remote**

b. Connections:

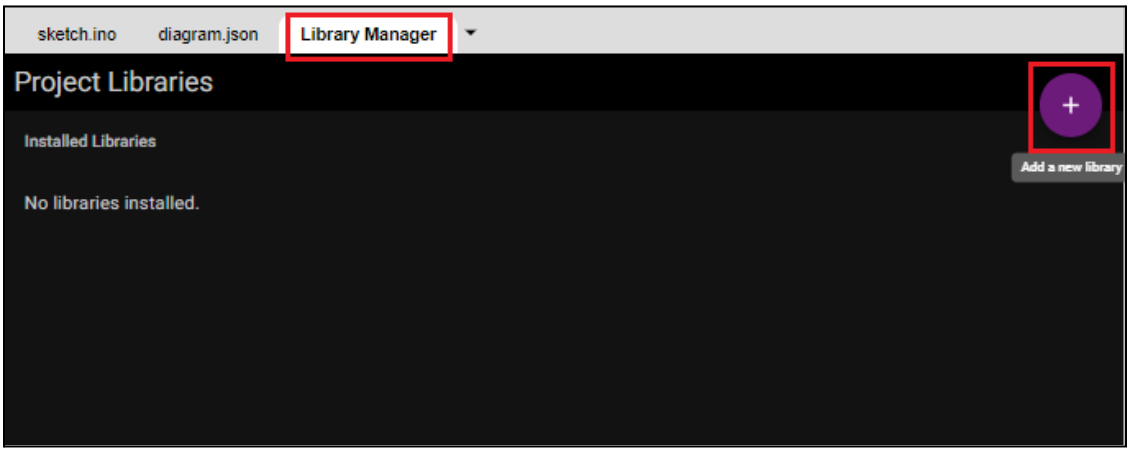
The circuit of this project consists of an Arduino Uno board, an **IR Receiver** and **IR Remote**. Connect the Arduino Uno board and IR Receiver. Don't need to connect the IR Remote.

IR Receiver	**Arduino Uno PIN**
VCC	5V
GND	GND.2
DAT	D6



3. Code:

a. First, go to the **Library Manager** and **Add a new library** called **IRremote**. A new file named **libraries.txt** will automatically be created.



b. Now, go to **sketch.ino,** add the header file.

```
#include <IRremote.h>
```

c. Now, add a variable which holds the pin number to which the **IR receiver** is connected.

```
const byte recv_pin=6;
```

d. Let's make an object of the receiver class.

```
IRrecv receiver(recv_pin);
```

e. Now, let's write the following code inside the **setup()** function.

- enable/start the IR receiver.

```
receiver.enableIRIn();
```

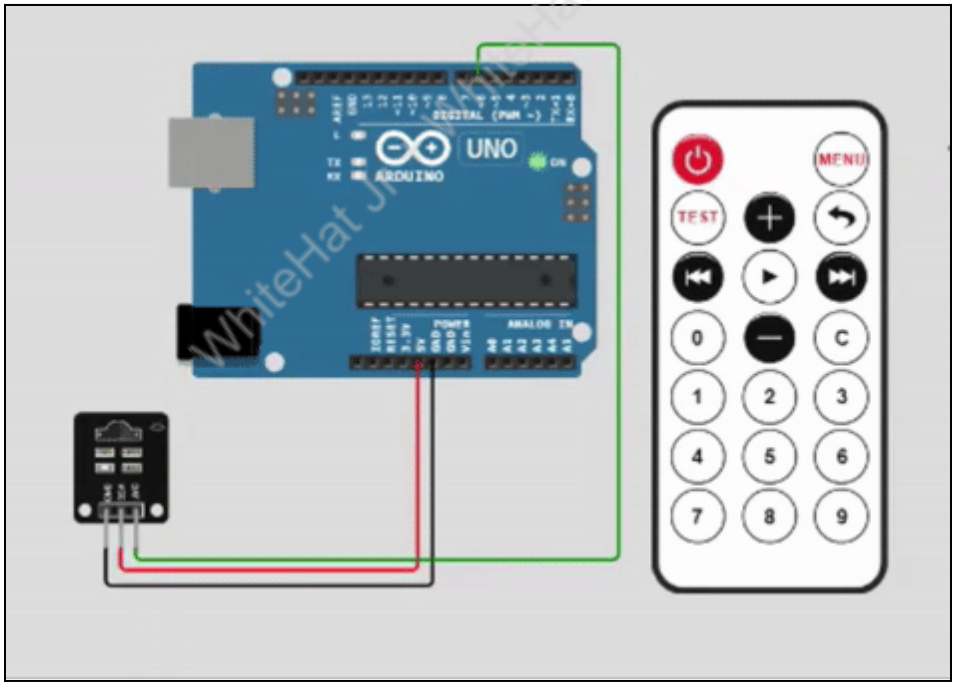
- To test if the remote is receiving from the IR sensor or not, use the blink13() function. There is an in-built LED connected to the 13th pin which will glow up when the IR receiver receives a message –

```
receiver.blink13(true);
```

The code should look like this now -

```
#include <IRremote.h>

const byte recv_pin=6;

IRrecv receiver(recv_pin);

void setup() {

  Serial.begin(9600);

  receiver.enableIRIn();

  receiver.blink13(true);

}

void loop() {

}
```

Let's observe the output-



Click here to view the output video.

Observe that the red light beside the 13th pin blinks when any button is pressed.

4. The Arduino Uno board should successfully receive messages from the remote

control now. There are two things to understand about the IR signals-

a. IR Interference:

Every living being emits some infrared light. There are many other sources of infrared rays like- the sun, fire, animals, human beings. The IR receiver might pick those signals up which we don't want.

- Solution:

  To prevent this problem, IR signals from a remote control are modulated. To modulate means to adjust a property of a signal in a certain way.

  To make the IR signals (from remote control) stand out among all the interference, the signal is modulated at a certain frequency (38KHz is the most common carrier frequency).

b. How do we know which button is pressed?

Each button has a unique code so that we can identify each button separately. Decode it to make the program understand the signal. For that, add a few lines of code in the **loop()** method-

- decode() - method looks for IR signals. If received, it decodes them.

```
void loop(){
  if (receiver.decode()){

  }
}
```

- Now, store the decoded IR response in a variable and print it on the console.

```
if (receiver.decode()){

  int response = receiver.decodedIRData.command;
  Serial.println(response);
```

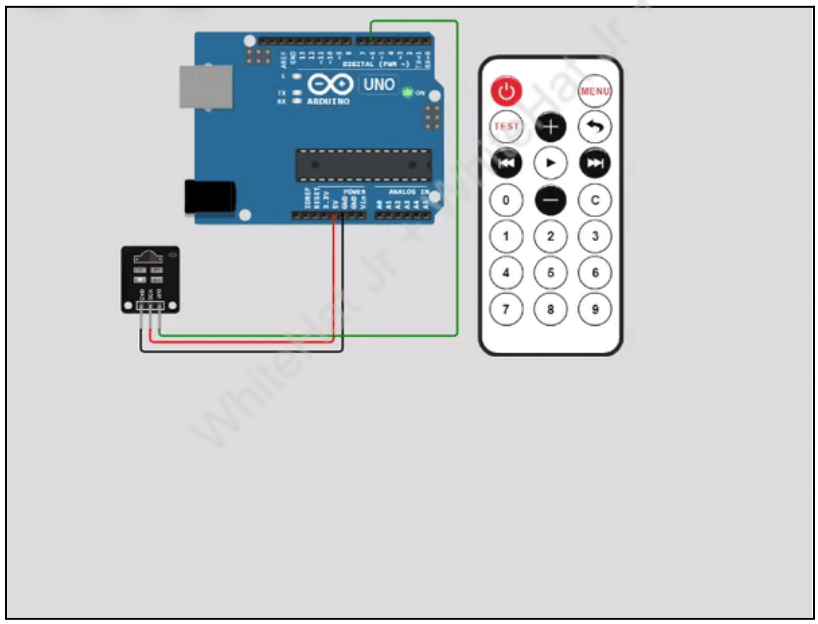
- Once a signal is received, resume with the next signal. Write -

  receiver.resume();

- Add a delay() method outside the if condition to make the simulator wait a little in between and improve it's working.

```
void loop(){

  if (receiver.decode()){

    int response = receiver.decodedIRData.command;
    Serial.println(response);

    receiver.resume();
  }

  delay(10);
}
```

Let's look at the output-



Click here to view the output video.

In the output, we can observe there are certain codes for each key on the remote. e.g. Key 1 has the code - 48, Key 2 has the code - 24 etc.

5. RGB LED:

Before you start with the next task. Let's understand **RGB LED**s first.

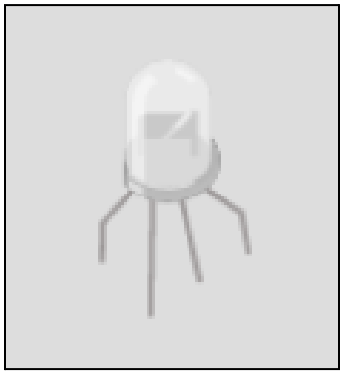


Image an RGB LED as three different red, green and blue LEDs. A normal Led has two legs- a cathode and an anode. An RGB LED has four legs, which are-

- Common pin (COM) - By default, the common pin is the anode (positive). It can be changed by setting the "common" attribute to "cathode". You imagine the common pin as all the cathode legs/ anode legs of the three LEDs tied together.
- Red LED (R) - if the COM pin is "anode", R would be the "cathode" of the Red LED and vice versa.
- Green LED (G) - if the COM pin is "anode", G would be the "cathode" of the Green LED and vice versa.
- Blue LED (B) - if the COM pin is "anode", B would be the "cathode" of the Blue LED and vice versa.

6. <u>Add RGB LED to the circuit diagram:</u>
    a. Add the following components-
        - 1x **RGB LED**
        - 3x **Resistors**

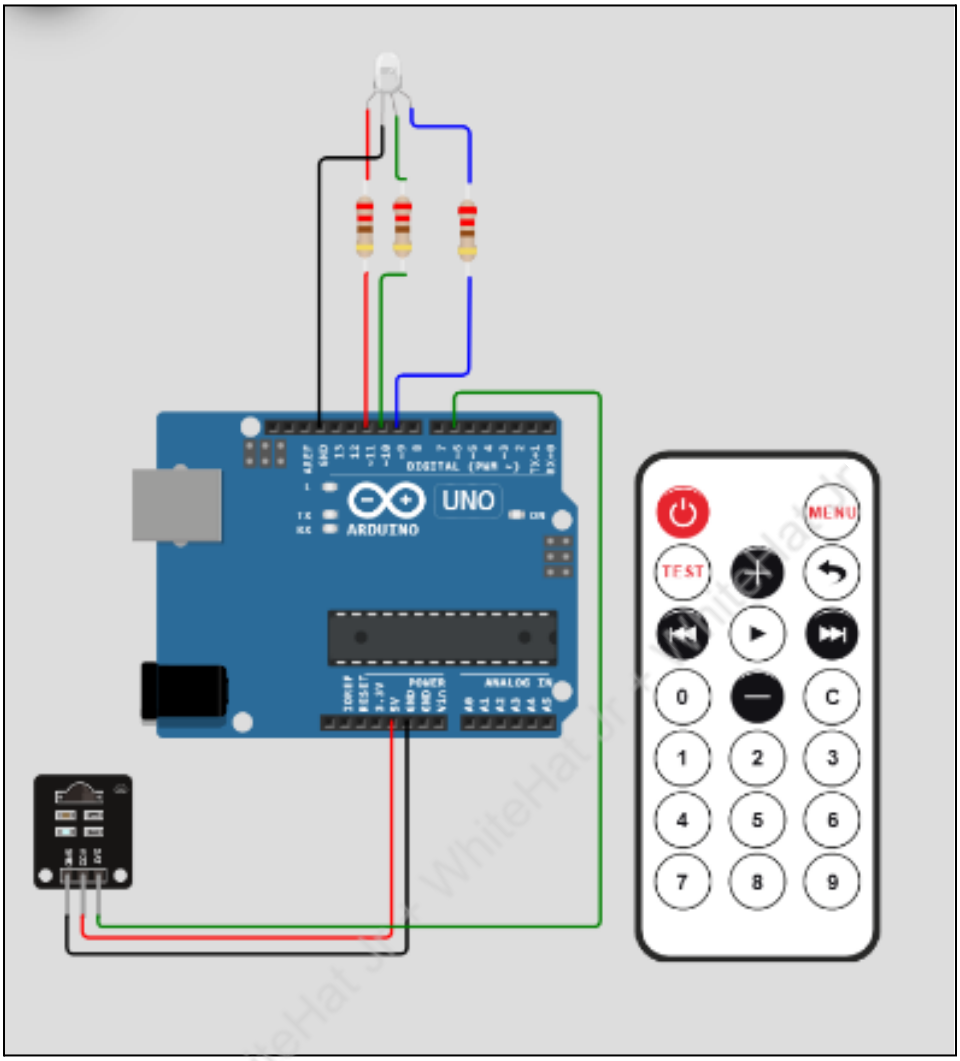
    b. Change the COM pin of the RGB LED to "cathode".
        - Go to **diagram.json** → find **wokwi-rgb-led** in parts → change the **common** attribute as **cathode.**

c. Let's change the resistors value. By default it is 1000 ohm, which is very high. Let's change it to 220 ohm. (ohm is the unit with which resistance is measured).
   - Go to **diagram.json** → find **wokwi-resistor** in parts → change the **value** attribute to **220.**
   - Do it for all three resistors.

d. Let's connect the RGB LED as per the instructions given below:

RGB LED	**Arduino Uno PIN**
COM	GND.1
R	Connect it to pin 11 through a resistor
G	Connect it to pin 10 through a resistor
B	Connect it to pin 9 through a resistor

7. **Reference circuit:**



8. Code:

   i. Initiate 3 const to store the pin values to which R,G,B pins are connected to.

```
const byte r_pin=11;
const byte g_pin=10;
const byte b_pin=9;
```

   ii. The RGB LED receives output from the controller. So, the next task would be to define the pin mode as OUTPUT.

   Go to **setup()** method and define pin mode for R, G, B pins.

```
pinMode(r_pin, OUTPUT);
pinMode(g_pin, OUTPUT);
```

```
pinMode(b_pin, OUTPUT);
```

iii. We use analogWrite(pin, value) function to light up RGB LEDs.

To light up the LED with red color-

```
analogWrite(r_pin,255);
```

To light up the LED with blue color-

```
analogWrite(b_pin,255);
```

To light up the LED with green color-

```
analogWrite(g_pin,255);
```

iv. Now, if to add a compound color (e.g. Yellow , purple, cyan etc.), write the **analogWrite()** function for all three pins.

Write the **lit_led()** function. Use this function to light up the LED. Pass three values i.e. r, g, b through this function -

```
53
54  void lit_led(int r,int g, int b){
55
56      analogWrite(r_pin,r);
57      analogWrite(g_pin,g);
58      analogWrite(b_pin,b);
59
60  }
61
```

v. Light up the **RGB LED** with different colors for different buttons on the remote control.

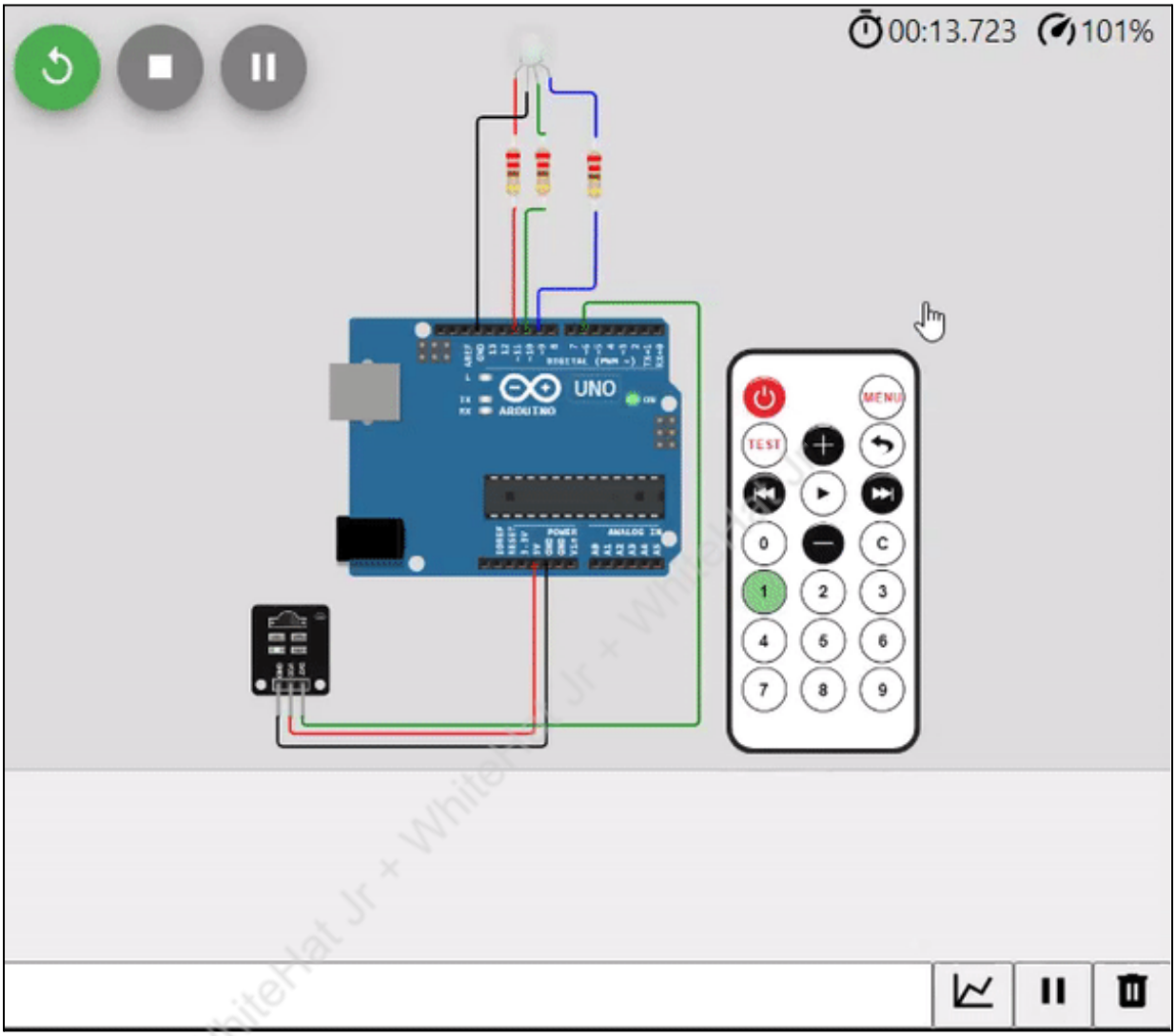
Write a function for that and name the function as **action().** Pass an argument through this function. This argument will hold the decoded value of the button pressed on the remote control -

```
35   void action(int num){
36
37     if(num== 48)
38       lit_led(255,0,0);
39     else if(num == 24)
40       lit_led(0,255,0);
41     else if(num == 122)
42       lit_led(0,0,255);
43     else if(num == 16)
44       lit_led(255,255,0);
45     else if(num == 56)
46       lit_led(0,255,255);
47     else if(num == 90)
48       lit_led(255,0,255);
49     else
50       lit_led(255,255,255);
51
52   }
```

vi. Let's call the **action()** method inside the **loop()** method. Also, pass the **response** variable through the function -

```
void loop() {

  if(receiver.decode()){
    int response=receiver.decodedIRData.command;
    action(response);
    Serial.println(response);
    receiver.resume();
  }

}
```

Reference Output:



Click here to view the reference video.

What's NEXT?

In the **next class**, we will learn about a new television component and add it to the circuit. We will also learn to display images and text on it.

Expand Your Knowledge

To know more about **IR receivers** on wokwi, click here.
To know more about **IR remote** on wokwi, click here.