
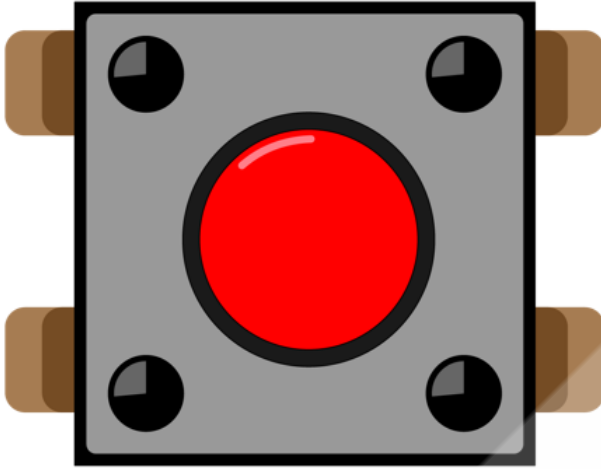


Topic	MELODY	
Class Description	Students will be introduced to how to control LEDs with push buttons and create a virtual piano using circuits	
Class	PRO C244	
Class time	55 Mins	
Goal	<ul style="list-style-type: none"> • Learn about the switch bounce & debounce • Banana switch 	
Resources Required	<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 20 mins 20 mins 05 mins
Credit & Permissions:	Code samples used for Firebase-Google Authentication are licensed under the Apache 2.0 License . Expo documentation used from - https://expo.io If applicable	

WARM-UP SESSION - 10 mins		
Teacher Action		Student Action
<p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <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>ESR: Hi, thanks! Yes, I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<p>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>Teacher Ends Slideshow </p>		
TEACHER-LED ACTIVITY - 10 mins		
Teacher Initiates Screen Share		
<p><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Push Button • Buzzer 		
Teacher Action		Student Action

Have you seen the lift?	ESR Varied!
Have you noticed the type of button used in lifts?	ESR Varied!
Do you know what that button is called?	
Basically, that is called push buttons, even though we used the same in our first circuit design right!	
The push-button is used to control devices like turning on and off a light-emitting diode (LED) when the push button is pressed or not.	
In the electronics industry, push-button plays an important role.	
Earlier we controlled our LED with push-button, but how a microcontroller plays a role with push buttons. Push buttons automate the process using a microcontroller.	
So today's task is to control the led using a push button with a microcontroller.	



A push-button usually has four pins that are connected internally in pairs.

We only need to use two of the pins, which are NOT in the same connected pair. Accordingly, there are four ways to do wiring with the button.

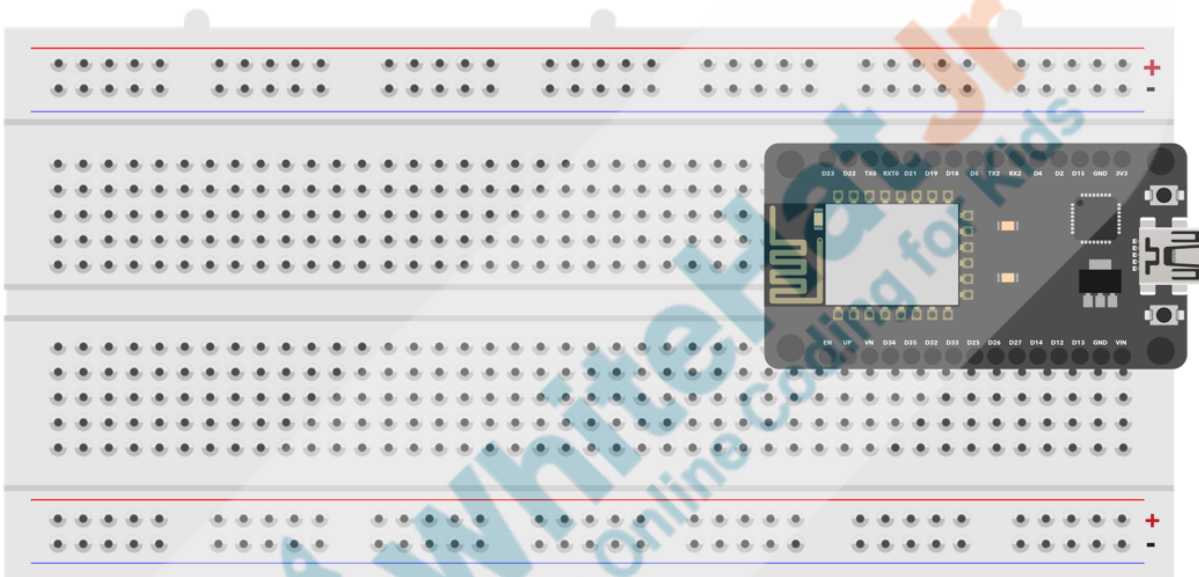


Step -1: Gather the below material from IoT Kit

- 1 X ESP32
- 1 x Resistor 330 ohm
- 1 x Push Button

- 1 x USB Cable
- 1 x LED
- 1 x Breadboard
- 6 x Jumper Wires

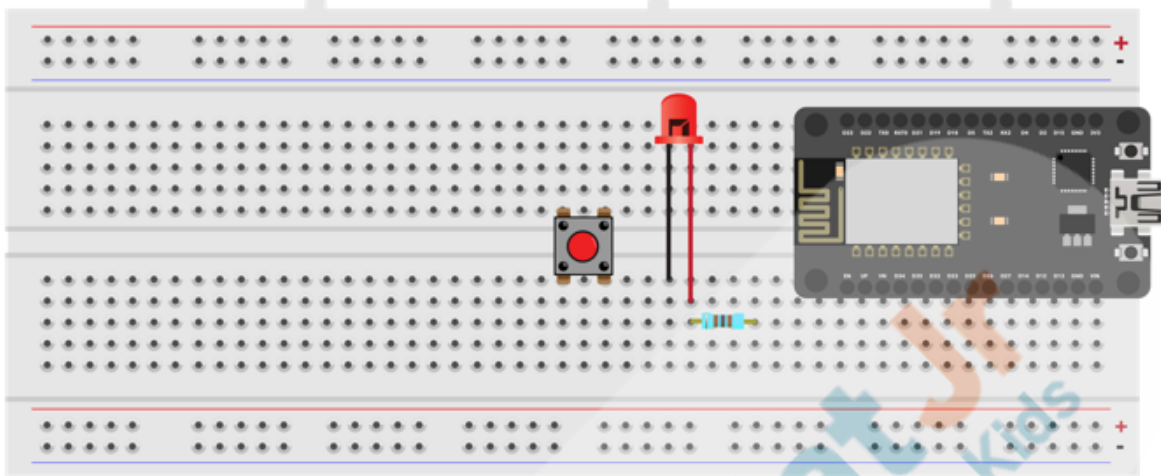
The first and foremost step is to mount ESP32 on the breadboard as shown below:



Step -2 Mount Components

Mount **Resistors, LED, Pushbuttons** on the breadboard as shown below:

- Connect the longer leg of the LED to one end of the resistor as shown below:
- As we can only use two ends of the push-button, mount it that way so it covers the middle breaker.



Push Button: When we use a push button with ESP32, we have to use the GPIO pins as digital inputs. We will read the state of the push button. It will either be high or low. The push-button can be interfaced with ESP32 either through a pull-up resistor or pull-down resistor.

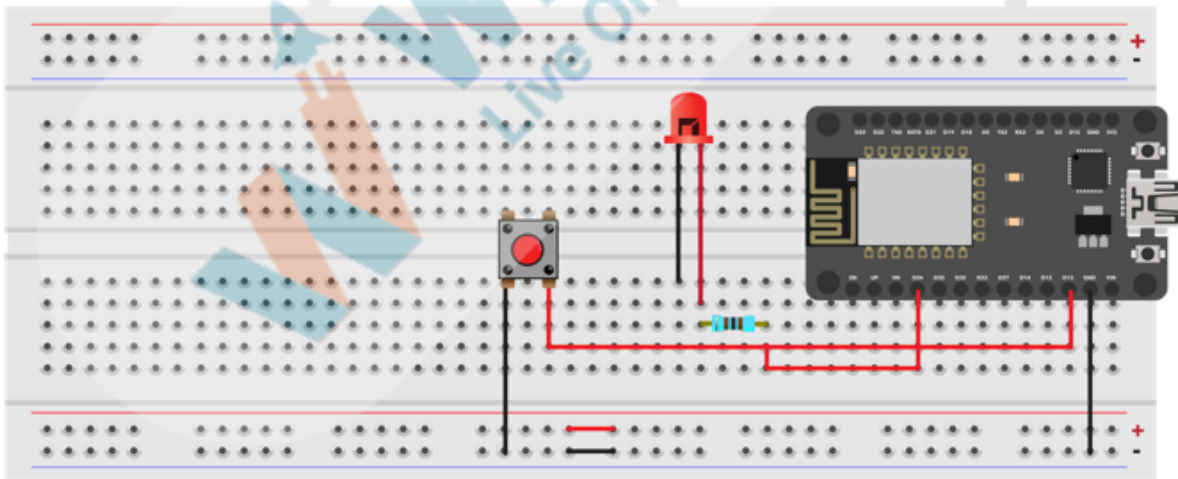
Push Button has 4 pins:

- In Pull up resistor mode, when the push button is not pressed, input to GPIO pin will be logical high or vice versa.
- In pull-down resistor mode, when the push button is pressed, input to the GPIO pin will be the logic low state and otherwise logic high state.

Step-3:Wiring Connections:

Provide power supply i.e **VCC (+ve)** and **(GND(-ve))** to LED & resistor.

- Connect the positive part of the LED (Longer Leg) with one end of the resistor and other end of the resistor to ESP32 GPIO pin **D26** (D31, D32, D25, D26, D12, D13) that are called Input/Output pins. To supply positive voltage, we can use any one of the mentioned pins. Just below the resistor pin, insert the male jumper wire and connect to ESP32 GPIO pins
- Now connect the **negative part (shorter leg) of the LED** and to one **(GND(-ve))** pin of the ESP32. Take a male jumper wire and connect just below the **shorter leg of the LED** and drag it to the **(GND(-ve))** terminal of the ESP32.
- Connect **push button** one terminal is with **ESP32 GPIO D12 (I/O)** pin of **ESP32** and other terminals of a **push-button** is connected to **GND**
- Will read these two states of the push button and turn on and turn off LED accordingly.



Now it's time for programming

Open the Arduino IDE

- Define a pin for the **PUSHBUTTON_PIN =12** (D12)
- Define a pin for the **LED_PIN =12** (D26)
- Declare variable **button**
- **void setup()** function is used to initialize everything
- Describe **pinMode()** for both LED, Push Button
Pin Mode() :PinMode() will declare LED as digital **OUTPUT**, & Push Button as **INPUT_PULL UP**
- **void loop()** function is used to execute the main process.
- In the void **loop()** function, **the digitalRead()** function reads the state of the push button and stores its value in the variable button.
- **digitalRead()** : The **digitalRead()** function is used to determine whether the input pin is **HIGH** or **LOW**. If the input pin state is **HIGH**, it is returned as HIGH and otherwise as **LOW**. You only need to pass the pin number as an argument to this function.
- If the condition is used to check the state of variable **Push_button_state**
- When **Push_button_state** is **HIGH**, **LED_PIN** will be turned on, and otherwise, it will remain off.

- **digitalWrite()** will help to change the state of LED

```

#define PUSHBUTTON_PIN 12
#define LED_PIN 26

int button = 0; // variable for reading the button status

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(PUSHBUTTON_PIN, INPUT_PULLUP);
}

void loop()
{
  button = digitalRead(PUSHBUTTON_PIN);
  if (button == LOW) {
    digitalWrite(LED_PIN, HIGH);
    delay(100);
  }
  else{
    digitalWrite(LED_PIN, LOW);
  }
}

```

So, Now press the push button, it will turn your LED on, when you release the button it will be off.


Now, I know you must do it on your own!

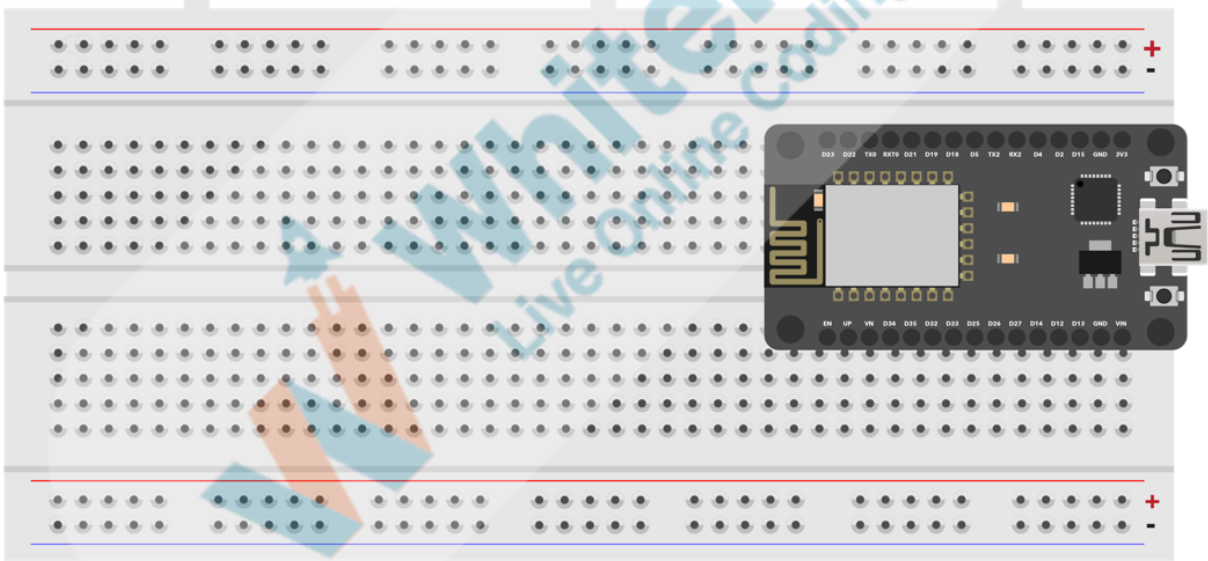
Teacher Stops Screen Share

So now it's your turn.
Please share your screen with me.

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

Let's try. I will guide you through it.	
STUDENT-LED ACTIVITY - 20 mins	
<ul style="list-style-type: none"> • 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 Fullscreen. 	
Student Initiates Screen Share	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Touch Piano 	
Teacher Action	Student Action
Before starting the activity, let me know if you have a touch piano or guitar at home.	ESR Varied!
No worries, that's ok, if you have or not!	
Isn't it fun to make our own Touch piano?	ESR Yes!
But for that, you need to bring some vegetables or fruits from your kitchen!	
Take out three fruits or vegetables like banana, apple, Potato, orange, tomato, chili anything	
Not you must be wondering how these fruits and vegetables will make piano	
The basic idea is based on touch sensing, where we touch fruits, and with the help of the ESP32 Touch pin, it will generate different frequency sounds corresponding to the	

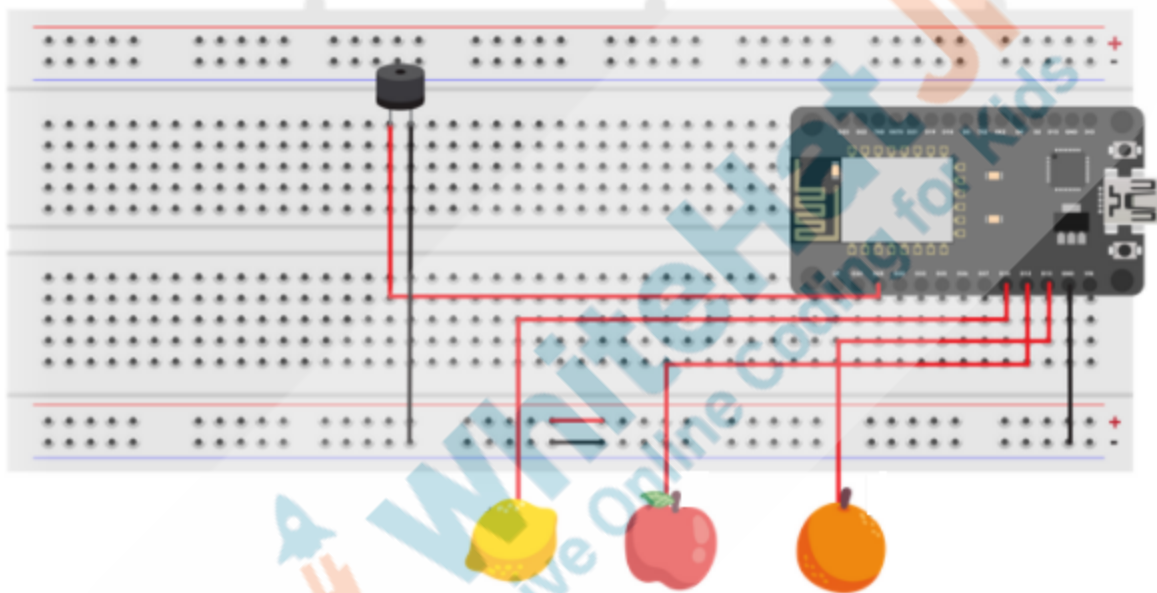
<p>various vegetables and fruits.</p> <p>But how will it produce sound?</p> <p>We need a buzzer to produce sound.</p> <p>Let's learn about buzzer first</p>	<p>ESR Varied!</p>
<p>Buzzer: A Buzzer works like a magnetic speaker, it needs voltage with a different frequency to be able to make a sound. It gets louder as the frequency increases and vice versa</p> <ul style="list-style-type: none"> • It makes a sound when the fruit is touched and stops it when it's not. • Basically, it will make a melody <p>The buzzer has two pins:</p> <p>GND pin: connect this pin to GND (0V) VCC pin: connect this pin to VCC (3.3V)</p>	
	

<p>Step-1 Gather the material from the IoT kit</p> <p>3 x Fruits, Vegetables</p> <p>8 x Jumper wires</p> <p>1 x ESP 32</p> <p>1 x Buzzer</p>	
<p>Step-1 Mount Components</p> <p>Mount the ESP32 on the breadboard. Try to mount it from one end and to leave a few terminals open on one side as shown below.</p>	
	
<p>Step-2: Mount the components on the breadboard</p> <ul style="list-style-type: none"> Connect the positive part(VCC (+ve)) of the buzzer with ESP32 pin D26. Take the jumper wire and insert it into just below the buzzer VCC (+ve) part and drag it to pin D26 of the ESP32 	

- Connect the negative part (**GND(-ve)**) of the buzzer with **ESP32(GND(-ve))** pin.

Step -3 Fruits & Connections

- Take three fruits/vegetables and insert male jumper wires one by one in each fruit/vegetable and other ends into ESP32 D13, D12, D14



Now, it's time to write a program

Open Arduino IDE

Step-1: Define and declare variables:

- Define **buzzer** and assign I/O pin **D26**
- Define variable along with datatype:
 - Int, const int is called data types, data type int is used to store an integer value
 - **VALUE_THRESHOLD**
 - **TOUCH_SENSOR_VALUE_1**, is used for Fruit/vegetable -1

- **TOUCH_SENSOR_VALUE_2**, is used for Fruit/vegetable -1
- **TOUCH_SENSOR_VALUE_3** is used for Fruit/vegetable 3

We are using touch sensors as the variable name because on touch fruit/vegetable will produce different frequencies

Note: Some ESP32 pins have multifunctional properties, the pins we are using for ESP32 I/O also act as touch sensor pins.

```
#define Buzzer 26

const int VALUE_THRESHOLD = 20;

int TOUCH_SENSOR_VALUE_1;
int TOUCH_SENSOR_VALUE_2;
int TOUCH_SENSOR_VALUE_3;
```

Step-2: Initialization under **setup()** function

- **void setup()** is used to initialize
- Describe **pinMode()** for **Buzzer**
Pin Mode() : **PinMode()** will declare **Buzzer** as digital **OUTPUT**
- **Serial.begin()** **Serial.begin(9600)** is used for data exchange data speed. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second. That's 9600 binary ones or zeros per second and is commonly called a baud rate.
- Syntax for serial.begin : *Serial.begin(speed)*
- Set up **delay()**
- **digitalWrite()** will make the buzzer value LOW at

the beginning.	
<pre>void setup() { pinMode(Buzzer, OUTPUT); Serial.begin(115200); delay(2000); digitalWrite(Buzzer, LOW); }</pre>	
<p>Step-3: Execution of the main process:</p> <p>void loop() function is used to execute the main process</p> <p>The ESP32 chip comes with inbuilt touch sensors. These touch sensors are the capacitive type. These touch sensors are shared with I/O pins of ESP32. As mentioned earlier, capacitive sensors can detect electrical changes on respective GPIO pins. For example, if you touch any of these pins, the GPIO pin will produce an output based on the electrical charge on your finger. Because the body also carries some electrical charge. Therefore these touch sensors can detect electrical changes on GPIO pins.</p> <ul style="list-style-type: none"> ● touchRead(touch_sensor_pin_number): This function is used to read the touch sensor value associated with the touch pin. We simply need to write the pin number we will be using. For example, if we are using touch pin zero, we would use this function like touchRead(T0) ● Store the touchRead() value of all fruit/Vegetable in respective variables. ● Serial.print() is used to print the data. ● Print the values of all touch sensors 	

```
void loop() {

    TOUCH_SENSOR_VALUE_1 = touchRead(T5);
    TOUCH_SENSOR_VALUE_2 = touchRead(T6);
    TOUCH_SENSOR_VALUE_3 = touchRead(T7);

    Serial.print("TOUCH_SENSOR_VALUES 1:");
    Serial.print(TOUCH_SENSOR_VALUE_1);
    Serial.print(" ");
    Serial.print("TOUCH_SENSOR_VALUES 2:");
    Serial.print(TOUCH_SENSOR_VALUE_2);
    Serial.print(" ");
    Serial.print("TOUCH_SENSOR_VALUES 3:");
    Serial.print(TOUCH_SENSOR_VALUE_3);
    Serial.println(" ");
    delay(500);
}
```

Step-4:Conditions:

The active buzzer will only generate a sound when it is electrified.If **TOUCH_SENSOR_VALUE_1** is less than **VALUE_THRESHOLD** then using **digitalWrite()** function changes the state of the Buzzer **High or Low**.

Set delay of 100 ms between **HIGH** and **LOW** state of **Buzzer**

```
if(TOUCH_SENSOR_VALUE_1 < VALUE_THRESHOLD) {
    for(int i=0; i<2; i++){
        digitalWrite(Buzzer, HIGH);
        delay(100);
        digitalWrite(Buzzer, LOW);
        delay(100);
    }
}
```

Our condition for one fruit/vegetable has been set, but we need to set conditions for other fruits and vegetables too. If no conditions match with the if conditions else make the **Buzzer Low**


```

if(TOUCH_SENSOR_VALUE_2 < VALUE_THRESHOLD){
    for(int i=0; i<5; i++){
        digitalWrite(Buzzer, HIGH);
        delay(50);
        digitalWrite(Buzzer, LOW);
        delay(50);
    }
}
if(TOUCH_SENSOR_VALUE_3 < VALUE_THRESHOLD){
    for(int i=0; i<8; i++){
        digitalWrite(Buzzer, HIGH);
        delay(25);
        digitalWrite(Buzzer, LOW);
        delay(25);
    }
}
else{
    digitalWrite(2, LOW);
}
}

```

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

You get “hats-off” for your excellent work!

In the next class, we will learn about RGB Led

Student Action

Make sure you have given at least 2 hats-off during the class for:

Creatively Solved Activities  +10

Great Question  +10

Strong Concentration  +10

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	LED_Reference Code	https://github.com/procodingclass/PRO-C244-Reference-Code
Teacher Reference 1	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/27fa64e2-c3f9-427e-8f70-a5fe43022223.docx
Student Activity 1	Reference Code	https://github.com/procodingclass/PRO-C244-Reference-Code-SA