
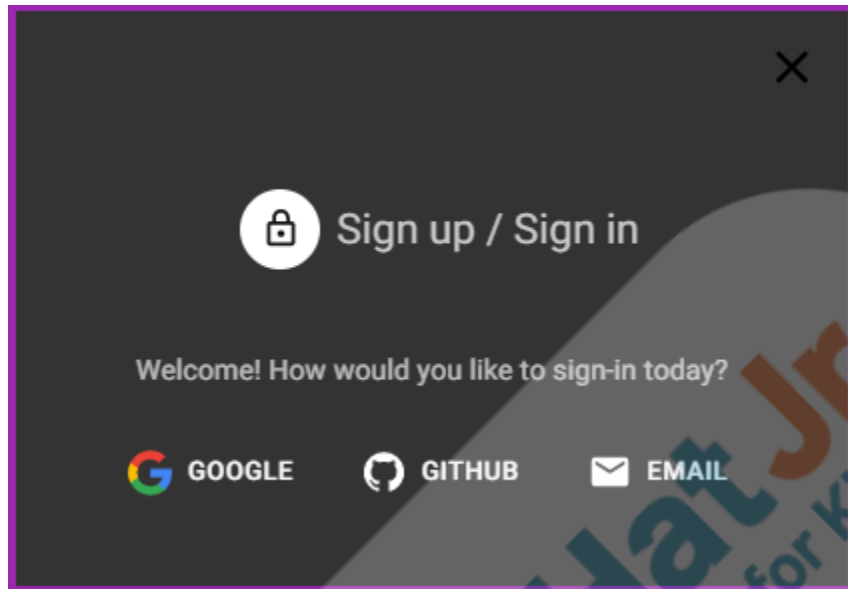


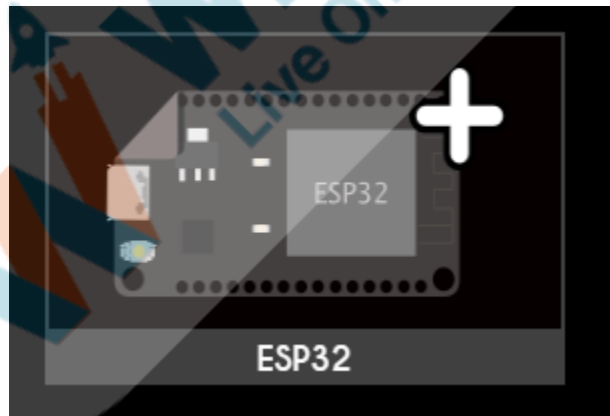
Topic	Tone Generator	
Class Description	Students will be introduced to the online simulator and they will explore basic activities on the simulator	
Class	PRO C254	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> <li>• Introduction to Wokwi Simulator</li> <li>• Melody Tone Generator</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>• Teacher Resources:               <ul style="list-style-type: none"> <li>○ Laptop with internet connectivity</li> <li>○ Earphones with mic</li> <li>○ Notebook and pen</li> <li>○ Smartphone</li> </ul> </li> <li>• Student Resources:               <ul style="list-style-type: none"> <li>○ Laptop with internet connectivity</li> <li>○ Earphones with mic</li> <li>○ Notebook and pen</li> </ul> </li> </ul>	
Class structure	<b>Warm-Up</b> <b>Student-Led Activity -1</b> <b>Student-Led Activity -2</b> <b>Wrap-Up</b>	<b>10 mins</b> <b>15 mins</b> <b>15 mins</b> <b>10 mins</b>
Credit & Permissions:	<b>Code samples used for Firebase-Google Authentication are licensed under the <a href="#">Apache 2.0 License</a>.</b> <b>Expo documentation used from - <a href="https://expo.io">https://expo.io</a></b> <b>Note: Keep this row section only if applicable</b>	
<b>WARM-UP SESSION - 10 mins</b>		
<b>Teacher Action</b>		<b>Student Action</b>

<p>Hey &lt;student's name&gt;. How are you? It's great to see you! Are you excited to learn something new today?</p> <p><b>Following are the WARM-UP session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Greet the student.</li> <li>• Revision of previous class activities.</li> <li>• Quizzes.</li> </ul>	<p><b>ESR:</b> Hi, thanks! Yes, I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<p align="center"><b>WARM-UP QUIZ</b> Click on In-Class Quiz</p>	
<p><b>Activity Details</b></p> <p><b>Following are the session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Appreciate the student.</li> <li>• Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</li> </ul>	
<p align="center"><b>TEACHER-LED ACTIVITY- 15mins</b></p>	
<p align="center"><b>Teacher Initiates Screen Share</b></p>	
<ul style="list-style-type: none"> <li>• <b>Introduction to Wokwi</b></li> </ul>	
Teacher Action	Student Action
<p>So in the last class, we learned about bouncing and debouncing concepts.</p> <p>Any doubts from the last class</p> <p><i>If the student has any doubts, clarify the doubts</i></p> <p>So today, we are moving towards the simulator, As we get the basics of how electronics circuit works.</p>	

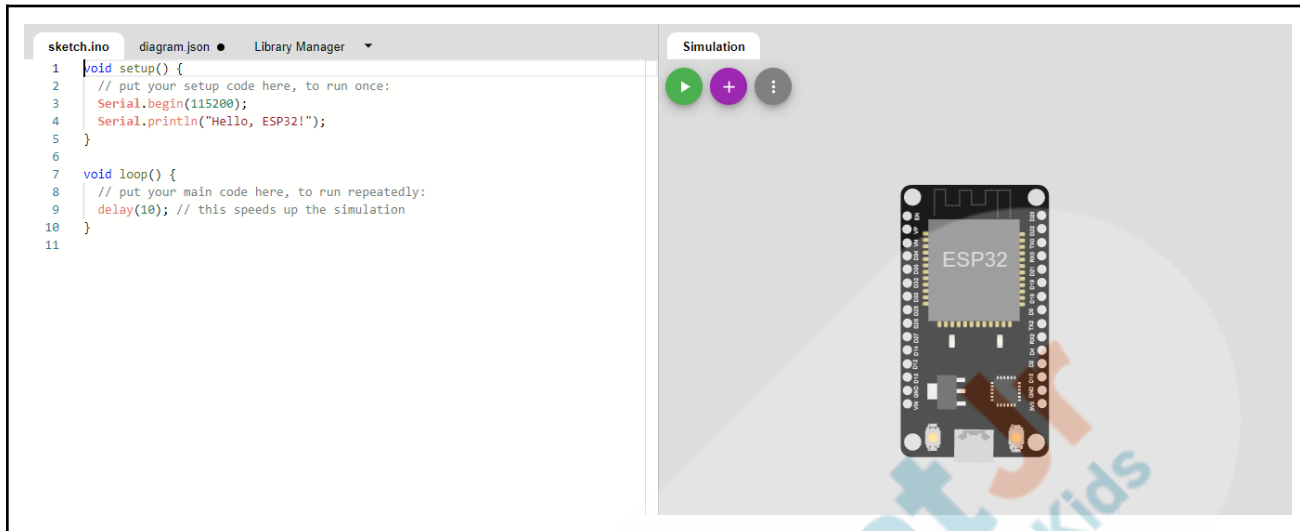
<p>We practiced a lot of circuits on the breadboard. Let's learn more about electronics and their components but this time on a simulator.</p> <p>The online simulator works exactly like the actual circuit, we need to do the wiring and all but this time it will be on the simulator.</p>	
<p>Teacher clicks on <a href="#">Teacher Activity 1</a></p>	<p>Student click on <a href="#">Student Activity 1</a></p>
<p>First, we need to <b>sign up</b></p>	
<div data-bbox="678 890 894 1056">  </div> <p>Choose any option for Sign UP:</p>	



After signing in you can select the board. As we can see a lot of boards are there, we will select ESP32 BOARD as earlier we were doing all activities on the ESP32 board.



After selecting ESP32, you will see the below window



We can see three buttons over there,



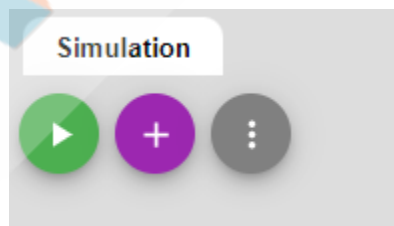
**Add component button**









**Start Simulation**



**Settings Property**

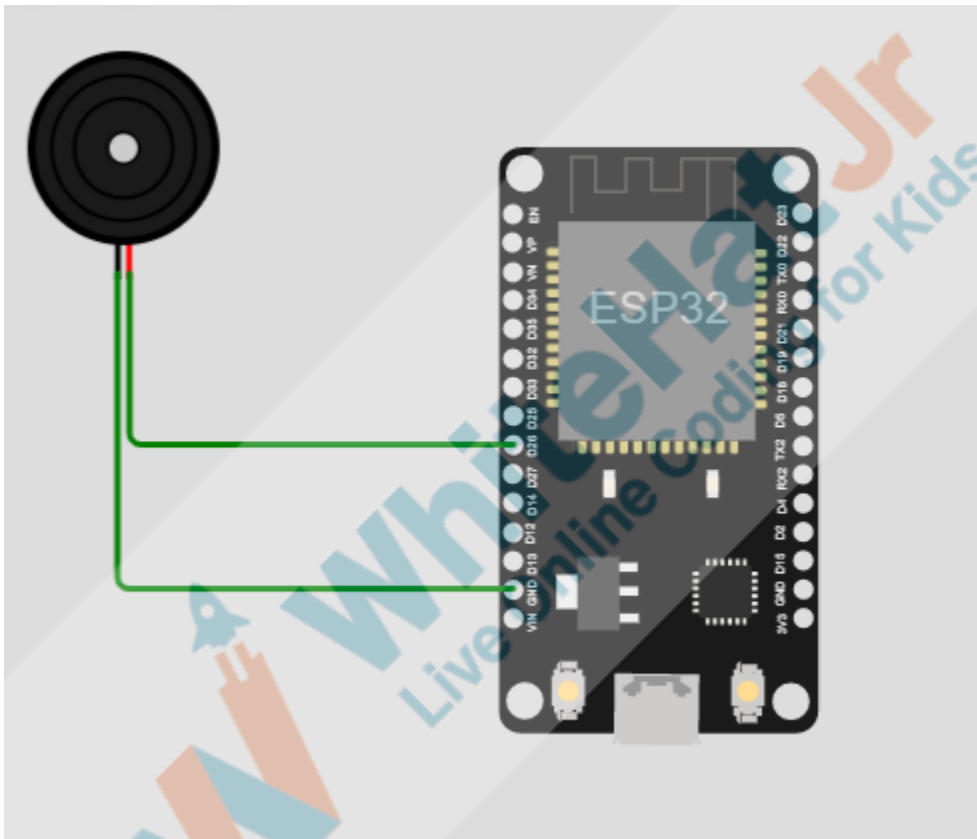


To save, rename, delete, create files, and upload files **Click on the small triangle icon**  **next to Library**

<p><b>Delete:</b> Select the file by clicking on the file and then click on the <b>small triangle icon</b>  <b>to delete a particular file.</b></p> <p><b>Create:</b> Click on the <b>small triangle icon</b>  to create a new file.</p> <p><b>Rename:</b> Select the file by clicking on the file and then click on the <b>small triangle icon</b>  <b>to rename a particular file.</b></p> <p><b>Save:</b> After renaming, Click on <b>Save</b> Button, it will automatically save your program in your Project list.</p> <p><b>Download:</b> To save it on your computer you can directly <b>download Project zip</b> option</p> <p><b>Upload:</b> Click on the <b>small triangle icon</b>  and select the <b>Upload file(s)</b> option.</p> <p><i>Note: We can upload multiple files once while uploading code but the circuit diagram needs to be designed every time.</i></p>	
<p>Let's try a very basic activity first on the simulator. We will turn on and off Buzzer.</p>	
<p><b>Step -1:Select the material from the Simulator</b></p> <ul style="list-style-type: none"> <li>• 1 x ESP32</li> <li>• 1 x Buzzer Click on + Sign and select Buzzer</li> </ul>	
<div data-bbox="581 1598 987 1755"> <input type="text" value="buzzer"/>   <div>Buzzer</div> </div>	

**Step -2: Let's do connections:**

- Click on OLED Black wire and then drag it to the ESP32 GND pin.
- Click on Buzzer Red wire and then drag it to the ESP32 Pin no 26.



Let's write the code.

**Step-1: Define and declare variables:**

- Define **buzzer** and assign I/O pin D26

Initialize using **void setup()** function

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

<ul style="list-style-type: none"> <li>• <b>void setup()</b> is used to initialize</li> <li>• Describe <b>pinMode()</b> for <b>Buzzer</b> <b>Pin Mode()</b> : <b>PinMode()</b> will declare <b>Buzzer</b> as digital <b>OUTPUT</b></li> <li>• <b>Serial.begin()</b> <b>Serial.begin(9600)</b> 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.</li> <li>• Syntax for serial.begin : <i>Serial.begin(speed)</i></li> <li>• Set up <b>delay()</b></li> <li>• <b>digitalWrite()</b> will make the buzzer value LOW at the beginning.</li> </ul>	
<pre>int buzzerPin = 26; void setup() {   pinMode(buzzerPin, OUTPUT); }</pre>	
<p><b>Step-3: Execution of the main process:</b></p> <p><b>void loop()</b> function is used to execute the main process.</p> <p>The active buzzer will only generate a sound when it is electrified. If Pin is electrified then using <b>digitalWrite()</b> function changes the state of the Buzzer <b>High or Low</b>.</p> <p>Set a delay of 100 ms between the <b>HIGH and LOW</b> state of <b>Buzzer</b></p>	



```
void loop() {

    unsigned char j;
    while (1)
    {
        for (int j = 0; j < 100; j++)
        {
            digitalWrite(buzzerPin, HIGH);
            delay(2);
            digitalWrite(buzzerPin, LOW);
            delay(2);
        }
        delay(100);
    }
}
```

#### Output:

Click on the Save button and then click on the simulation button

- You will hear the buzzer sound

But you will be wondering why Ma'am is using Buzzer.  
Here is the challenge.

Now using the same buzzer you need to generate some melody.

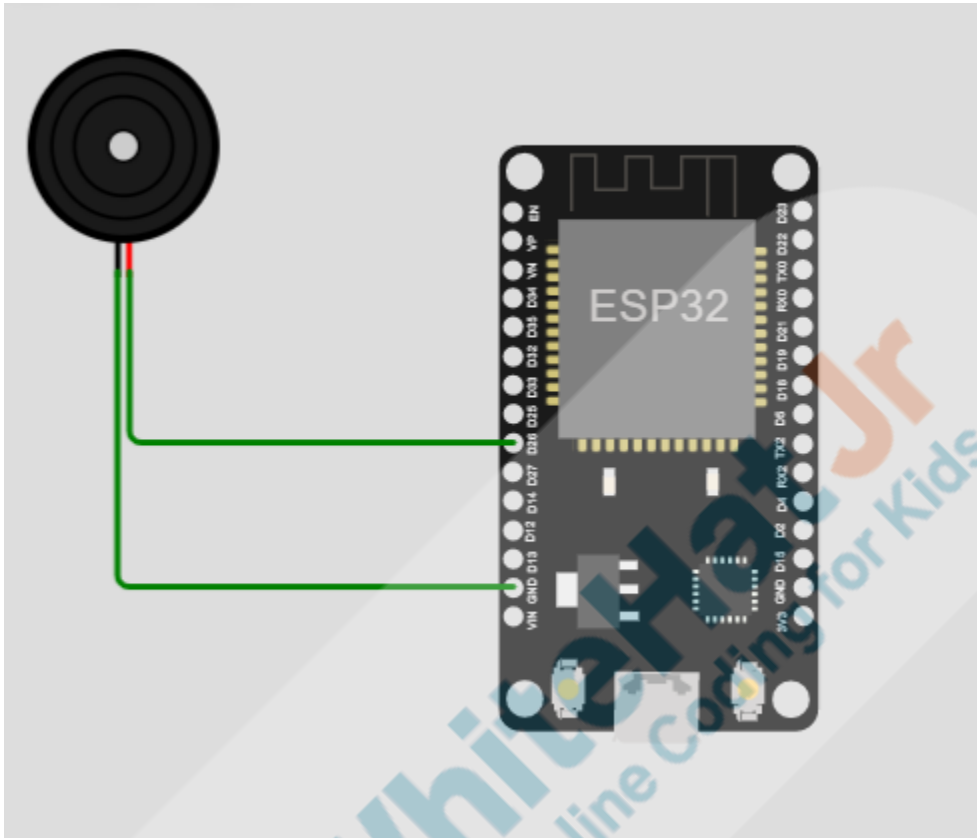
we can play any type of Melody with a buzzer. The only thing we need to know about is Music Notes & their Frequencies.

Every song/Melody has musical Notes and Frequencies.

So Let's see how using Music Notes and Frequency we can convert it into Melody.

<p>So first we will check the sound of the buzzer and we will control it using a push button.</p> <p>But before that, we must learn about Musical Notes &amp; Frequencies.</p> <p>Suppose we want to play <b>Jingle bells</b> on the buzzer.</p> <ol style="list-style-type: none"> <li>1. We must know the piano notes for the Jingle bells.</li> <li>2. We must have frequencies for notes.</li> </ol>	
<p>Teacher clicks on <a href="#">Teacher Activity 2</a></p>	<p>Student click on <a href="#">Student Activity 2</a></p>
<p>Let's make an array of Melody</p>	
<pre>int melody[] = {E, E, E,R, E, E, E,R, E, G, C, D, E, R, f, f, f,f, f, E, E,E, E, D ,D,E, D, R, G ,R, E, E, E,R, E, E, E,R, E, G, C, D, E, R, f, f, f,f, f, E, E, E, G,G, f, D, C,R };</pre>	
<p>After making Melody, the next task is to use frequencies</p>	
<p>Teacher clicks on <a href="#">Teacher Activity 3</a></p>	<p>Student clicks on <a href="#">Student Activity 3</a></p>
<pre>#define C      2109.89 #define D      1879.69 #define E      1674.62 #define f      1580.63 #define G      1408.18 #define R      0</pre>	
<p>Now, you make the connections and complete the activity</p>	
<p>Student Stops Screen Share</p>	

So its time to do the second Activity, now you are comfortable with LEDs 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-2 - 15 mins	
<ul style="list-style-type: none"> <li>Ask the student to press the ESC key to come back to the panel.</li> <li>Guide the student to start Screen Share.</li> <li>The teacher gets into Full Screen.</li> </ul>	
Student Initiates Screen Share	
<u>ACTIVITY</u>  <ul style="list-style-type: none"> <li>melody</li> </ul>	
Teacher Action	Student Action
Teacher guides student to download the boilerplate code	Student downloads from <a href="#">Student Activity 4</a>
<b>Step -1: Select the material from the Simulator</b> <ul style="list-style-type: none"> <li>1 x ESP32</li> <li>1 x Buzzer Click on + Sign and select Buzzer</li> </ul>	!
<b>Step -2: Let's do connections:</b> <ul style="list-style-type: none"> <li>Click on Buzzer Black wire and then drag it to the ESP32 GND pin.</li> <li>Click on Buzzer Red wire and then drag it to the ESP32 Pin no 26.</li> </ul>	



The next step is to set the Melody timings, speed, stop time, and rest time between two melodies

**Set the MELODY Length, tempo, pause time, and rest time**

- Define the datatypes, Int is used for integer values
- **MAX\_LENGTH** is used to set the size or total time of the MELODY
- Tempo can be defined as **the pace or speed at which a section of music is played**. Set the **tempo**
- Set the **Pause** time between melody

*Note: Don't use the pause word as its Arduino method.*

- Set the **rest\_length** between melody
- Define the **tone** and variable **beat** and **set value =0**

```
int MAX_COUNT = sizeof(melody) / 2;

long tempo = 10000;

int paus = 1000;

int rest_count = 2;

int tone_ = 0;
int beat = 0;
long duration = 0;
```

Now you must be wondering about the same circuit but two different outputs just with music notes and frequencies?

How do these notes & frequencies help to create different sounds?

Any idea?

To produce a tone, a speaker is pulsed rapidly on and off and this can be done using PWM (**PULSE WIDTH MODULATION**) to create frequencies.

Do you remember the PULSE WIDTH MODULATION?

PWM stands for **PULSE WIDTH MODULATION** which is used to check the portion of the time the signal spends ON versus the time that the signal spends OFF. PWM ON-OFF pattern can simulate voltages in between the full **Vcc** of the board (e.g., 5 V/3.3 V ON and OFF (0 Volts) The duration of "on time" is called the pulse width.

Each note has its own frequency, which is created by varying the period of the note vibration, which is measured

**ESR: Varied!**

in microseconds. **PULSE WIDTH MODULATION (PWM)** is used to create that vibration.

Our main purpose is to send the **Pulse** to the speaker to play a tone for a certain amount of time

1. Define the variable **elapsed\_time** which is used to keep the track of how long we pulsed. **long** is the data type like int, the only difference is used to store large range values.
2. The **BUZZER** will only generate a sound when it is electrified. Now make the **BUZZER Tone HIGH OR LOW** as per duration. If the tone has played less long than '**duration**' then make the **BUZZER OUTPUT HIGH**.
3. If the tone has played longer than '**duration**' then makes the **BUZZER OUTPUT LOW**.
4. **digitalWrite()** function changes the state of the Buzzer **HIGH or LOW**.
5. **delay microseconds function** stops the program for the amount of time (in microseconds) specified by the parameter.
6. Syntax: **delayMicroseconds(us)** us: the number of microseconds to pause.
7. Keep track of the **tone** and how long we pulsed
8. To repeat the melody after **rest\_count** use the **for** loop

else make the **Buzzer LOW**

```
void playTone() {  
    long elapsed_time = 0;  
    if (tone_ > 0) {  
  
        while (elapsed_time < duration) {  
            digitalWrite(speakerOut,HIGH);  
            delayMicroseconds(tone_ / 2);  
  
            digitalWrite(speakerOut, LOW);  
            delayMicroseconds(tone_ / 2);  
  
            elapsed_time += (tone_);  
        }  
    }  
    else {  
        for (int j = 0; j < rest_count; j++) {  
            delayMicroseconds(duration);  
        }  
    }  
}
```

#### Call the **main loop**

- Play the loop till MAX\_Length
- Set the bear 50
- Duration will be calculated as per tempo and beat
- Call the function **playTone()**
- Set the pause using **delayMicroseconds**

```

void loop() {
  for (int i=0; i<MAX_COUNT; i++) {
    tone_ = melody[i];
    beat = 50;

    duration = beat * tempo; // Set up timing

    playTone();
    // A paaus between notes...
    delayMicroseconds(paaus);
  }
}

```

**Output:**

Click on the Save button and then click on the simulation button

- You will hear the melody.

So, today we learned about how a single Buzzer can produce different sounds.

That's fun!

**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**

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

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

In the next class, we will learn about web servers

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	Simulator	<a href="https://wokwi.com/">https://wokwi.com/</a>
Teacher Activity 2	Music Notes	<a href="https://s3-whjr-curriculum-uploads.whjr.online/c3a552cb-66d0-4dcf-8186-fe695dcc47c4.png">https://s3-whjr-curriculum-uploads.whjr.online/c3a552cb-66d0-4dcf-8186-fe695dcc47c4.png</a>
Teacher Activity 3	Frequency Notes	<a href="https://s3-whjr-curriculum-uploads.whjr.online/1895612f-c676-46be-a7e4-94a7c9925c4f.png">https://s3-whjr-curriculum-uploads.whjr.online/1895612f-c676-46be-a7e4-94a7c9925c4f.png</a>
Teacher Activity 4	Teacher Activity -1	<a href="https://github.com/procodingclass/PRO-C254-Teacher-Activity-1">https://github.com/procodingclass/PRO-C254-Teacher-Activity-1</a>
Teacher Activity 5	Reference Code	<a href="https://github.com/procodingclass/PRO-C254-ReferenceCode">https://github.com/procodingclass/PRO-C254-ReferenceCode</a>
Student Activity 1	Simulator	<a href="https://wokwi.com/">https://wokwi.com/</a>
Student Activity 2	Music Notes	<a href="https://s3-whjr-curriculum-uploads.whjr.online/c3a552cb-66d0-4dcf-8186-fe695dcc47c4.png">https://s3-whjr-curriculum-uploads.whjr.online/c3a552cb-66d0-4dcf-8186-fe695dcc47c4.png</a>
Student Activity 3	Frequency Notes	<a href="https://s3-whjr-curriculum-uploads.whjr.online/1895612f-c676-46be-a7e4-94a7c9925c4f.png">https://s3-whjr-curriculum-uploads.whjr.online/1895612f-c676-46be-a7e4-94a7c9925c4f.png</a>

		<a href="#">e4-94a7c9925c4f.png</a>
Student Activity 4	Boiler Plate Code	<a href="https://github.com/procodingclass/-PRO-C254-StudentBoilerPlate">https://github.com/procodingclass/-PRO-C254-StudentBoilerPlate</a>
Teacher Reference	In_Class Quiz	<a href="https://s3-whjr-curriculum-uploads.whjr.online/b853d361-0c67-4d88-86e2-df792f594c33.docx">https://s3-whjr-curriculum-uploads.whjr.online/b853d361-0c67-4d88-86e2-df792f594c33.docx</a>

