TONE GENERATOR

What is our GOAL for this CLASS?

In this class, we learned about Tone **Generator**

What did we ACHIEVE in the class TODAY?

- We learned about Buzzer Tone Generator
- We learned about Musical Notes & Frequency

Which CONCEPTS/ CODING BLOCKS did we cover today?

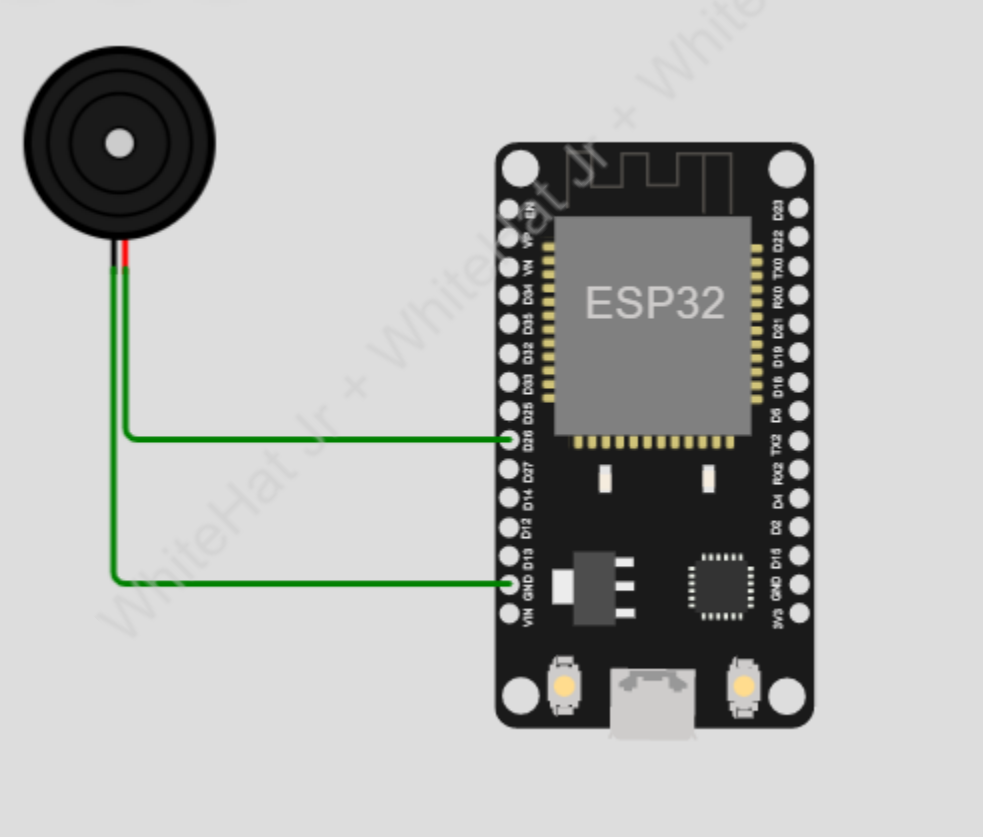
- We learned how a buzzer produces the sound and how using Music Notes and Frequency we can convert it into Melody.
- We learned how to check the sound of the buzzer and to control it using a push button.
- Melody is an array of notes and beats based on each note's relative length it will lengthen or shorten the tones.
- PWM stands for **PULSE WIDTH MODULATION** which is used to check 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 in microseconds. **PULSE WIDTH MODULATION** (PWM) is used to create that vibration.

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr

How did we DO the activities?

1. Select the material
   - 1 x ESP32
   - 1 x Buzzer

2. 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.



3. Write the program:
   - Define the setup function

   - Initialize using **void setup()** function

```
void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(BUZZER_PIN, OUTPUT);
}
```

- Write the logic part under **void loop()**

```
void loop() {
  int button_state = digitalRead(BUTTON_PIN);
  if (button_state == LOW) {
    Serial.println("Buzzer ON");
    digitalWrite(BUZZER_PIN, HIGH);
  }
  else if (button_state == HIGH) {
    Serial.println("Buzzer OFF");
    digitalWrite(BUZZER_PIN, LOW);
  }
}
```

4. **Output:**
   - Compile and upload the program to the ESP32 board using Arduino IDE
   - Verify the program by clicking the Tick option
   - Upload the program by clicking the arrow option
   - If the port is not selected, insert the USB cable in Computer's port and select the port.
   - Make sure hardware is connected properly.
   - Press the push button and the buzzer sound will be played, when its released the buzzer will be turned off.

5. Convert the buzzer sound into a melody.

6. make an array of Melody

```
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 };
```

7. click on Frequency Reference and use frequencies

```
#define  C     2109.89
#define  D     1879.69
#define  E     1674.62
#define  f     1580.63
#define  G     1408.18
#define  R     0
```

8. Initialize using **void setup()** function

9. Define the datatypes for **Buzzer, check**

- Int is used for integer values,
- **pinMode()** configures the specified pin to behave either as input or output. Since we want this pin for output, we are writing **OUTPUT** here.
    - **Syntax**: *pinMode(pin, mode)*

- **pin**: The pin do we need to set
- **mode**: Set the mode INPUT, OUTPUT,
- **Serial. begin(9600)** is used for data exchange speed. speed parameters. 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.

```
int Buzzer = 14;
int check = 1;
void setup() {
  pinMode(Buzzer, OUTPUT);
  if (check) {
    Serial.begin(9600);
  }
}
```

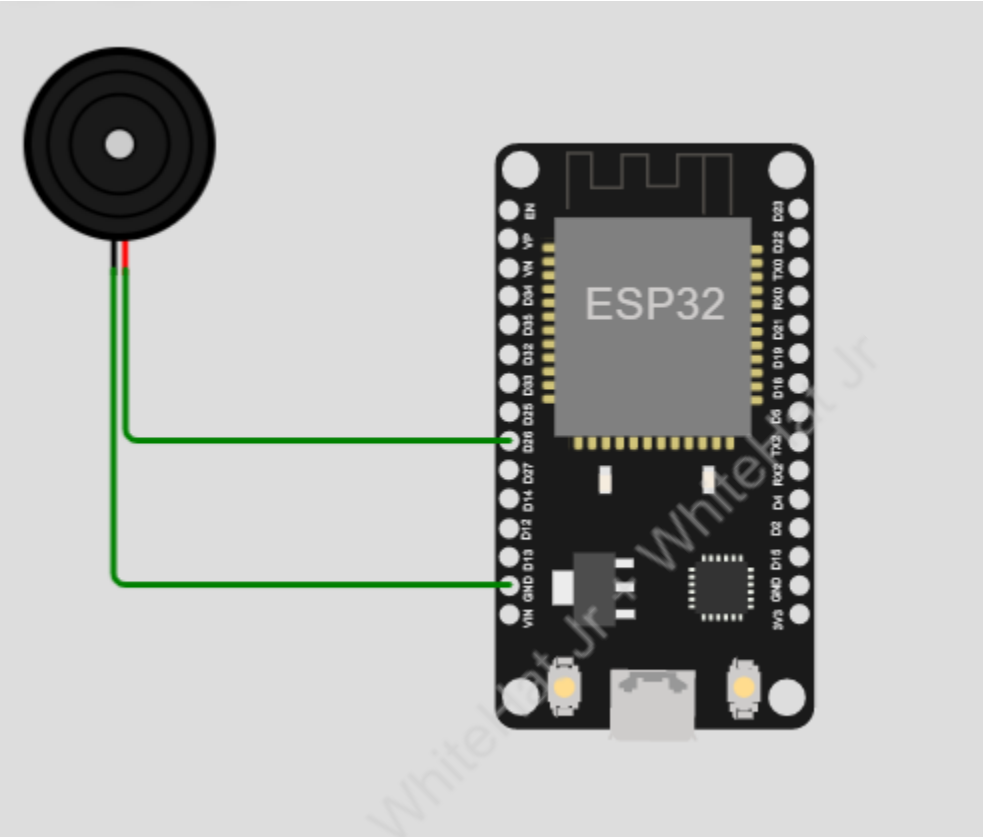
10. Select the material

- 1 x ESP32
- 1 x Buzzer

11. 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.

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr



12. 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
- Set the **rest_length** between melody
- Define the **tone** and variable **beat** and **set value =0**

```
int MAX_Length = sizeof(melody)
long tempo = 10000;
int paaus = 1000;
int rest_length= 2;
int tone_ = 0;
int beat = 0;
long duration  = 0;
```

13. send the **Pulse** to the speaker to play a tone for a certain amount of time
    - Define the variable **elased_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.
    - 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 makes the **BUZZER OUTPUT HIGH.**
    - If the tone has played longer than **'duration'** then makes the **BUZZER OUTPUT LOW.**
    - **digitalWrite()** function changes the state of the Buzzer **HIGH or LOW.**
    - **delay microseconds function** stops the program for the amount of time (in microseconds) specified by the parameter.
    - Syntax: **delayMicroseconds(us)** us: the number of microseconds to pause.
    - Keep track of the **tone** and how long we pulsed
    - To repeat the melody after **rest_count** use the **for** loop
    - Else make the **Buzzer LOW**

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

14. Call the **main loop**

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

   duration = beat * tempo;
   playTone();
   delayMicroseconds(paaus);
 }
}
```

15. **Output:**
    - Compile and upload the program to the ESP32 board using Arduino IDE
    - Verify the program by clicking the Tick option
    - Upload the program by clicking the arrow option
    - If the port is not selected, insert the USB cable in the Computer's port and select the port
    - Go to Tools and select Serial **Monitor**
    - See the **brightness of the LED and Value**

16. We learned about RGB LED and how to fade an LED.

What's NEXT?

In the **next class**, we will learn about **Electronic Voting Machines.**

Expand Your Knowledge

To know more about **PWM** click here.

WhiteHat Jr + WhiteHat Jr