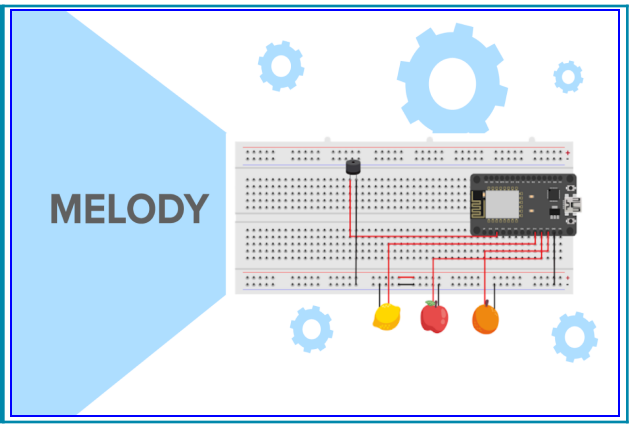
MELODY



What is our GOAL for this CLASS?

In this class, we learned how to control LEDs with push buttons and create a virtual piano using vegetables and fruits..

What did we ACHIEVE in the class TODAY?

- We learned about the basics of push buttons.
- We learned about **controlling LEDs using a push button with a microcontroller.**
- We learned about the basics **of Buzzer.**
- We learned about **the ESP32 chip and its inbuilt touch sensors.**
- **We designed our Touch Piano with the help of the ESP32 Touch pin.**

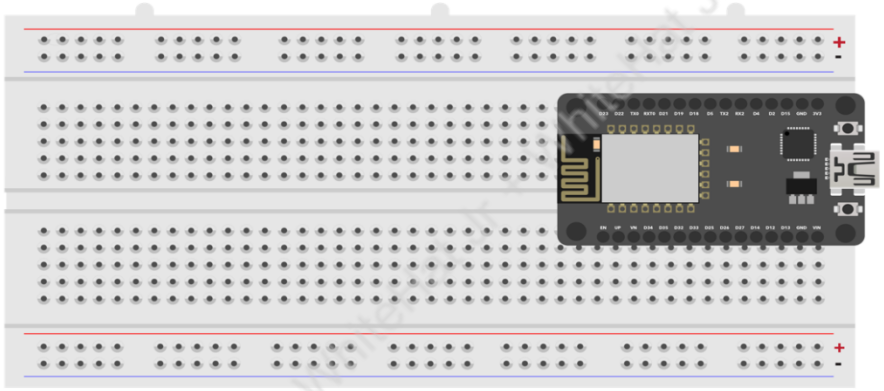
Which CONCEPTS/ CODING BLOCKS did we cover today?

- Normally open/closed Push Button
- Buzzer Programming
- Concept of frequency
- ESP32 Touch pin

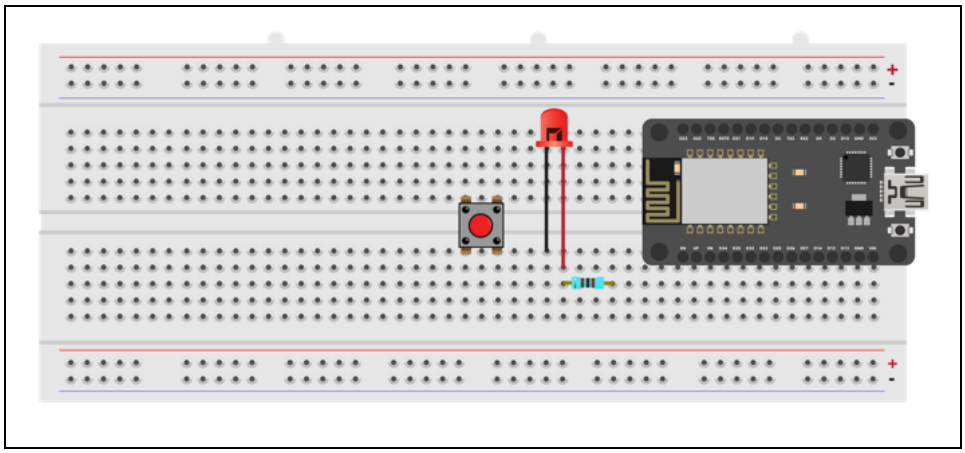
How did we DO the activities?

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

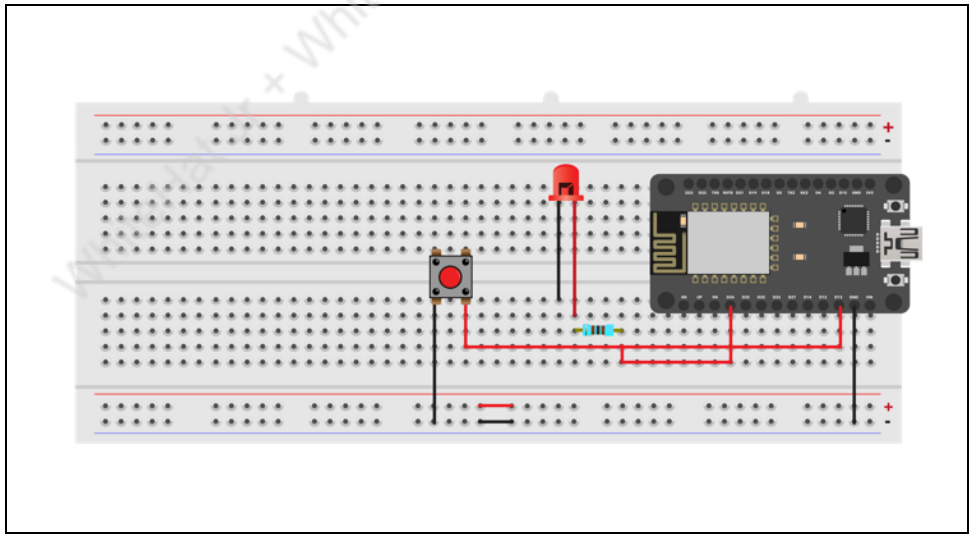
2. Mount **ESP32 on the breadboard** as shown below:



3. 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 and another end in the breadboard other component rails.
    - As we can only use two ends of the push-button, mount it that way to cover the middle breaker.



4. Provide **VCC (+ve) (5V**) and **(GND(-ve) to LED & resistor** respectively.
    - Connect the other end of the **resistor with ESP32 pin D26** (D31, D32, D25, D26, D12, D13, D17, D19) that are called Input/Output pins. To supply positive voltage, we can use any one of the mentioned pins.
    - Connect the shorter leg of the LED to the negative part to the **(GND(-ve))** terminal of the ESP32.
    - Connect push button one terminal is with **D12** (I/O) pin of ESP32 and other terminals of a push-button is connected to **GND**



5. Open the **Arduino IDE** and write the program.
    - 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,PushButton**

    - **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 **void loop()** function, **Digital read()** function read 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 augment 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.
- **Digital Write()** will help to change the state of LED

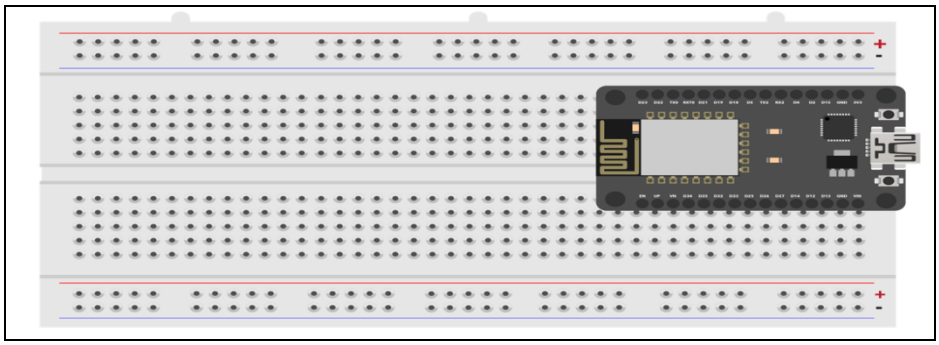
```
#define PUSHBUTTON_PIN  12
#define LED_PIN    26

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);
  }
}
```

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

6. **Touch piano using ESP32 Touch pin:**
    - Materials required:
        - 3 x Fruits, Vegetables
        - 8 x Jumper wires
        - 1 x ESP 32
        - 1 x Buzzer

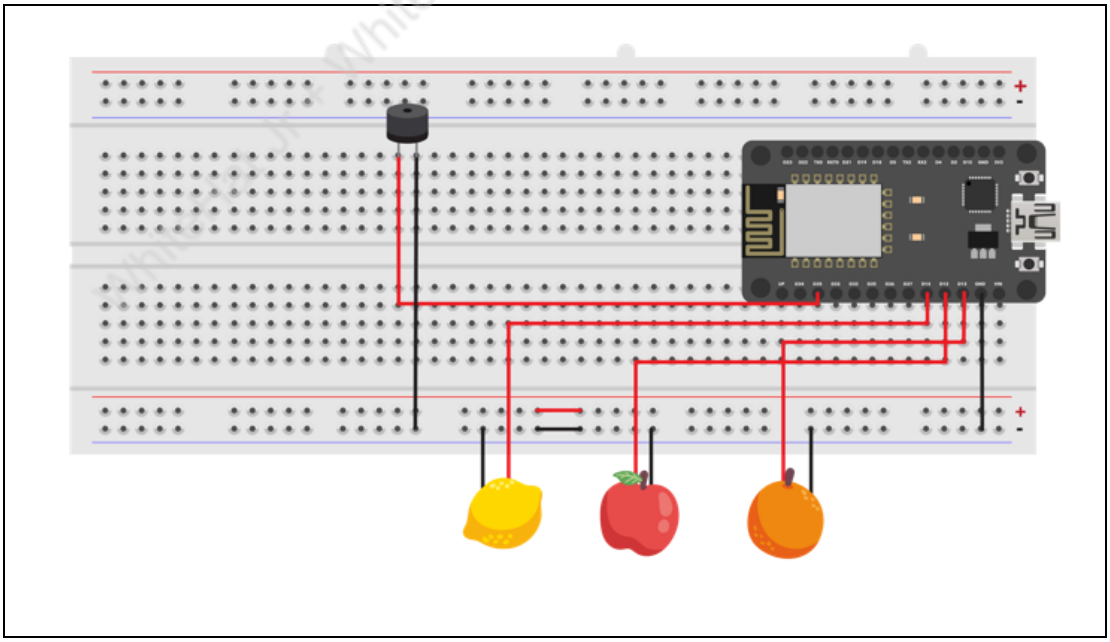
7. **Mount the ESP32 on the breadboard**. Try to mount it from one end and to leave a

few terminals open on one side as shown below



8. Mount the components on the breadboard
    - Connect positive part **(VCC (+ve))** of the buzzer with ESP32 pin D12. Take the jumper wire and insert it into just below the buzzer **VCC (+ve)** part.
    - Connect negative part **(GND(-ve))** of the buzzer with ESP32 **GND(GND(-ve))** pin.

9. Fruits & Connections
    - Take three fruits/vegetables and insert male jumper wires one by one in each fruit/vegetable and other ends into ESP32 D25, D26, D32



10. Open the **Arduino IDE** and write the program.
    - Define **buzzer** and assign I/O pin **26**
    - 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

```
#define Buzzer 26

const int VALUE_THRESHOLD = 30;

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

11. 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 a starting.
    -

```
void setup(){

  pinMode(Buzzer, OUTPUT);
  Serial.begin(115200);
  delay(2000);

  digitalWrite(Buzzer, LOW);
}
```

12. Execution of the main process using **void loop()**

```
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);
```

- **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. These touch sensors can detect electrical changes on GPIO pins.
    - **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.
    - 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

13. Conditions
    - The **active buzzer** will only generate a sound when it will be electrified. It generates sound at only
    - If **TOUCH_SENSOR_VALUE_1** digitalWrite() function writes or changes the state of the Buzzer

```
  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);
  }

}
```

- Check the circuit. Touch any fruit or vegetable and it should play the sound.

What's NEXT?

In the next class will be introduced to the concept of **Analog inputs** and **sensors**.

Expand Your Knowledge

To know more about **Basics of Electronics** click here