BOUNCING & DEBOUNCING SWITCH

What is our GOAL for this CLASS?

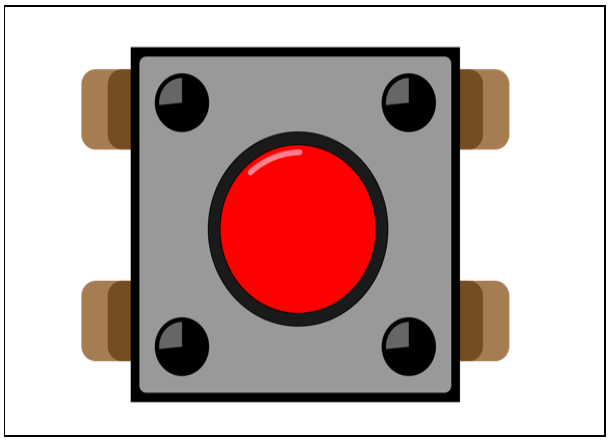
In this class, we learned about bouncing and debouncing, using a concept we operated relay switch using push-button.

What did we ACHIEVE in the class TODAY?

- We learned about Bouncing
- We learned about Debouncing

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Push Button:
    - **Push Button:** The push-button is used to control devices like turning on and off circuits or electronics devices.



- A push button usually has four pins that are connected internally in pairs.
- We only need to use two of the four pins, which are NOT in the same connected pair. Accordingly, there are four ways to do wiring with the button.

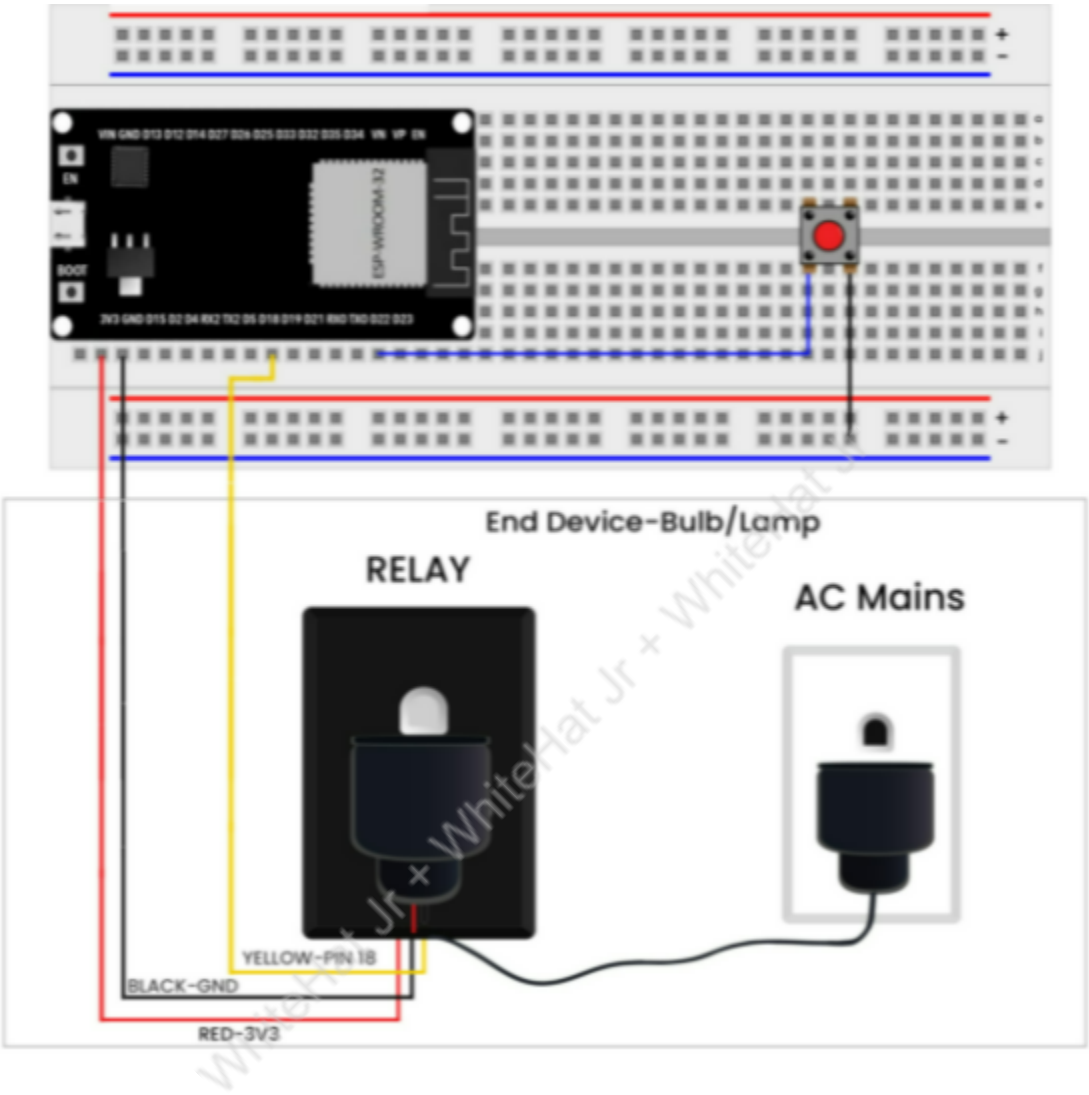
- We learned about bouncing & debouncing circuits

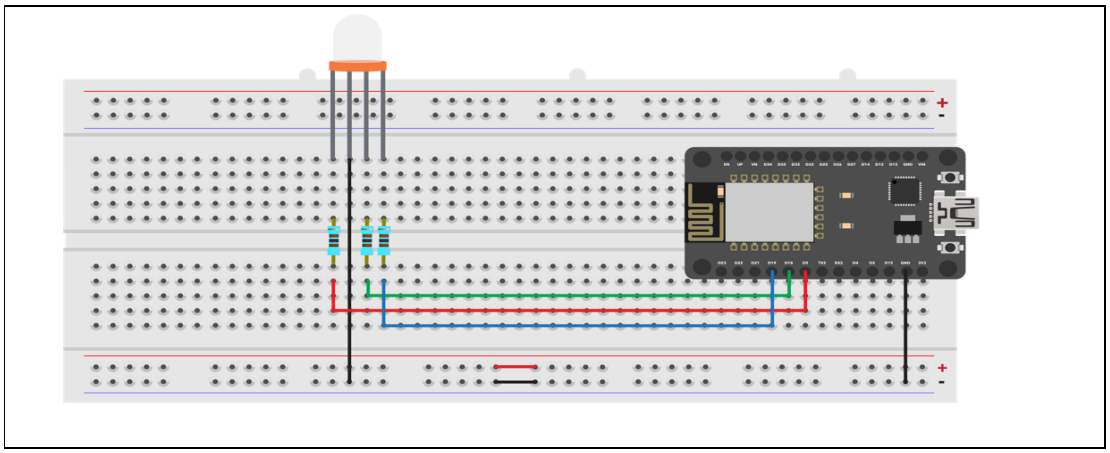
How did we DO the activities?

1. Control a relay switch with a push-button and understand bouncing and debouncing circuits in electronics.

2. Gather the material from the IoT kit Collect the material
   - 1 x ESP32
   - 1 x USB Cable
   - 1 x Breadboard
   - 4 x Jumper wires
   - 1 x Push Button
   - 1 x Relay
   - 1 x Mosquito Repellant Machine/Lamp/Bulb with holder

3. Do connections:
   - Insert pushbutton on the breadboard
   - Connect one end of the pushbutton with ESP32 GPIO pin no
   - Connect another end of the pushbutton with GND of the ESP32
   - Take the Relay(Black Box), Insert the relay Plug into **AC mains**
   - Connect relay with ESP32 BOARD
   - Connect **Black with GND, Red with 3.3V**, and **Yellow with ESP32 GPIO PIN 18**
   - Take one device like **Mosquito Repellant Machine/Lamp/Bulb with holder and insert them into relay switch.**





4. Write the program:
   - Define the setup function
     - Define GPIO pin for pushbutton i.e **PUSHBUTTON_PIN 22**
     - Define GPIO pin for relay i.e **RELAY_PIN** 18

```
#define BUTTON_PIN 22
#define RELAY_PIN  18
```

   - Define the dataypes for button_states
     - Int is used for integer values,
     - declare **int** for **relay_state, button_state, last_button_state**
     - **Last_button_state** will be store the last value of **push_buton**, **button_state** will store the current value of button, **relay_state** will store the value of relay.

```
int relay_state = LOW;
int button_state;
int last_button_state;
```

   - Initialize using **void setup()** function
     - **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.
     - **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, INPUT_PULLUP, INPUT_PULLDOWN,
     - **PULLUP** condition for push button will ensure the state on the pin is low.
     - **PULLDOWN** condition for push button will ensure the state on the pin is high

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

  button_state = digitalRead(BUTTON_PIN);
}
```

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

- Variable last_button_state will store current value of push button state
- **digitalRead()** will check the state of button
- **Serial.println** is used to print the statement
- The pin needs to be programmed to be either ON or OFF, that is, we can command it to be ON (output 5 volts), or OFF (output 0 volts).
- To switch it on and off, use a function called **digitalWrite().**

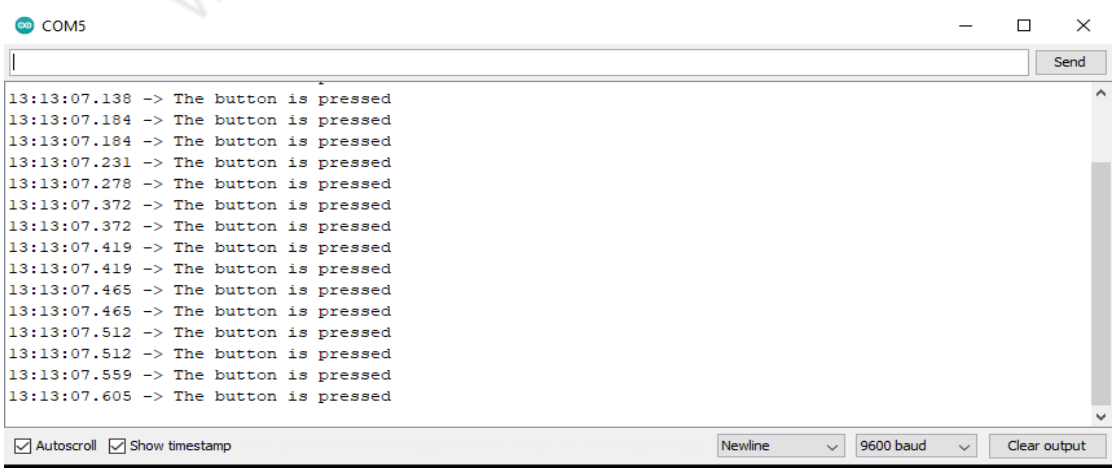
```
void loop() {
  last_button_state = button_state;
  button_state = digitalRead(BUTTON_PIN);

  if (last_button_state == HIGH && button_state == LOW) {
    Serial.println("The button is pressed");

    // toggle state of relay
    relay_state = !relay_state;

    // control relay arccoding to the toggled state
    digitalWrite(RELAY_PIN, relay_state);
  }
}
```

5. **Output:**
   - Compile and upload the program to 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 once and keep it several seconds and then release it and you'll see the light flickering of your connected device.
- You will see that you pressed the button once but your lamp/mosquito repellent LED will flicker on and off multiple times.
- It is called **BOUNCING** since it shows multiple stages of 0 and 1. Ideally, it should turn on and off once when a button is pressed. Consequently, it will give false signals.
- False signals cause a lot of problems with electronics circuits. To rectify this we must learn about **Debouncing** circuits

6. This problem occurs when buttons are used, especially when run at first time. To remove this type of issue use a pre defined library called **ezButton.** This library is used with pushbuttons and various Switches.

7. Define Pins
   - Define GPIO pin for pushbutton i.e **PUSHBUTTON_PIN 22**
   - Define GPIO pin for relay i.e **RELAY_PIN** 18

```
#include <ezButton.h>

#define BUTTON_PIN 22
#define RELAY_PIN  18
```

8. Define the dataypes for button_states,
   - Int is used for integer values,
   - declare **int** for **relay_state** and store the value **LOW**
   - Create **ezbutton** object **button**

```
ezButton button(BUTTON_PIN);
int relay_state = LOW;
```

9. Initialize using **void setup()** function
   - **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.
   - **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,
   - **setDebounceTime** of 50 ms

```
void setup() {
  Serial.begin(9600);
  pinMode(RELAY_PIN, OUTPUT);
  button.setDebounceTime(50);
}
```

10. Write the logic in **void loop()**

- Call the loop function first
- Variable last_button_state will store current value of push button state
- **Serial.println** is used to print the statement
- ! is use to toggle the state of relay in case of flickering
- The pin needs to be programmed to be either ON or OFF, that is, we can command it to be ON (output 5 volts), or OFF (output 0 volts).
- To switch it on and off, we need to use a function called **digitalWrite().**

```
void loop() {
  button.loop();

  if (button.isPressed()) {
    Serial.println("The button is pressed");


    relay_state = !relay_state;


    digitalWrite(RELAY_PIN, relay_state);
  }
```

11. **Output:**

- Compile and upload the program to 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.
- Go to Tools and select **Serial Monitor**
- Press the push button once and keep it several seconds and then release it and you see there is no light flickering.
- You will see that you pressed the button once in result your lamp/mosquito repellent LED will turn on and off once only instead of multiple times
- It is called **DEBOUNCING** since it shows one stage of 0 and 1. This is Ideal solution of the circuit.

What's NEXT?

In the next class, we will learn about OLED

Expand Your Knowledge

To know more about **Push Button** click here.