BLINKING BANNERS

What is our GOAL for this CLASS?

In this class, we learned about the **ESP32 microcontroller** and to make patterns of light using an embedded programming language.

What did we ACHIEVE in the class TODAY?

- We learned about **electronics**
- We learned about **ESP32 microcontroller**
- We learned about **Arduino**
- We learned to make patterns of light using an embedded programming language.

Which CONCEPTS/ CODING BLOCKS did we cover today?

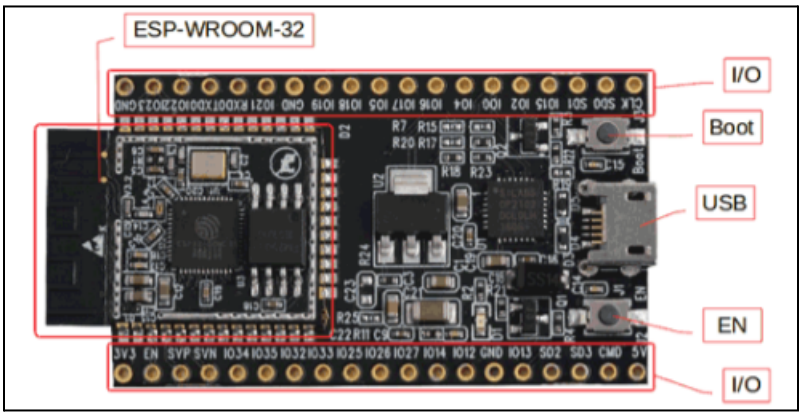
- We used **Arduino software**
- We used a **ESP32 microcontroller,** a **USB Cable,** a **few LEDs,** a **few 330-ohm Resistors,** a **few Jumper Wires,** a **Battery and** a **Battery Socket**

How did we DO the activities?

1. Understand **Microcontrollers**:

- What are **microcontrollers**?
  "A microcontroller is a compact integrated circuit (IC's) that includes processor, memory, and input/output (I/O) ports on a single chip" and it's called a microcontroller because we use very small electronic devices on that chip.

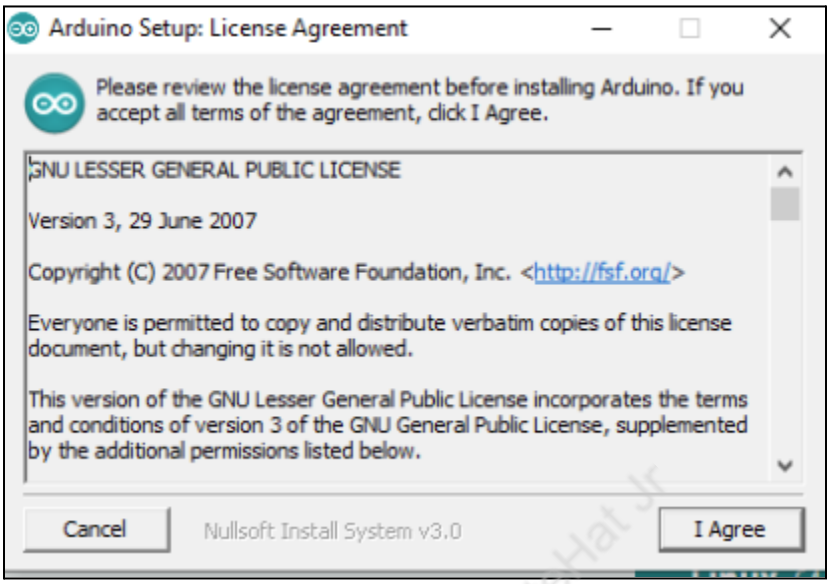
- **ESP32 microcontroller:**
  - **Micro-USB jack**: The micro USB jack is used to connect the ESP32 to our computer through a USB cable.
  - **EN Button**: The EN button is the reset button of the ESP module. Pressing this button will reset the code running on the ESP module
  - **Boot Button**: This button is used to upload the Program from Arduino to the ESP module. It has to be pressed after clicking on the upload icon on the Arduino IDE. When the Boot button is pressed along with the EN button, ESP enters into firmware uploading mode. Do not play with this mode unless you know what you are doing.
  - **Red LED**: The Red LED on the board is used to indicate the power supply. It glows red when the board is powered.
  - **Blue LED**: The Blue LED on the board is connected to the GPIO pin. It can be turned on or off through programming.
  - **I/O pins**: This is where major development has taken place. These pins are best to use as inputs, outputs. These pins have a lot of use. We will get more into that later.
  - **ESP-WROOM-32**: This is the heart of the ESP32 module. It is a 32-bit microprocessor.



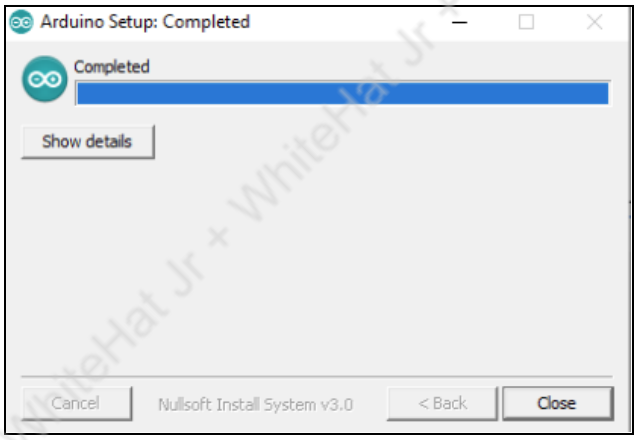
2. Install the software and learn a new language called **C language.**

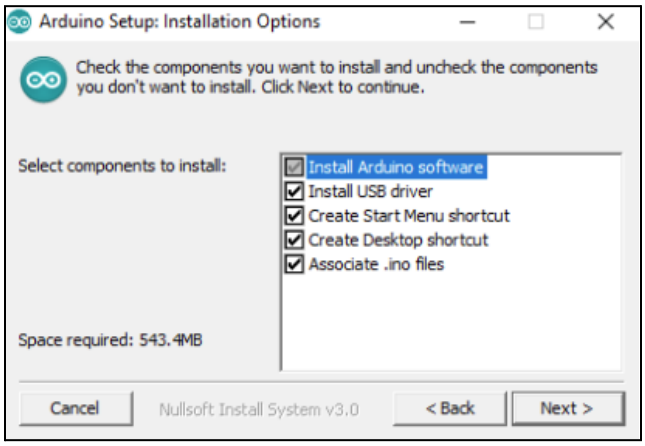
- Download **Arduino** software**.**

- Go to **Downloads**
    - Open Software
    - Click on **I Agree**

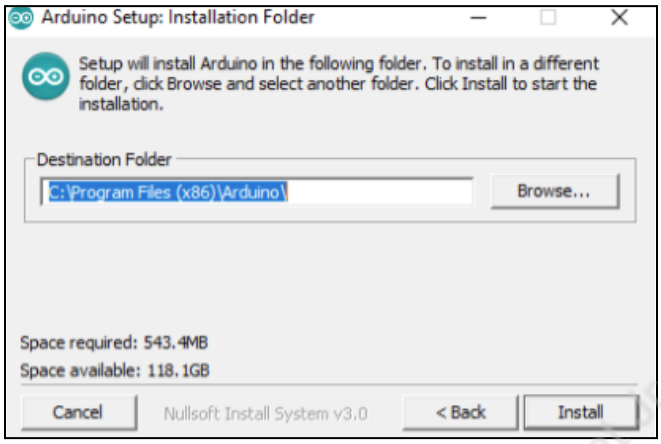


- Click on **Next**

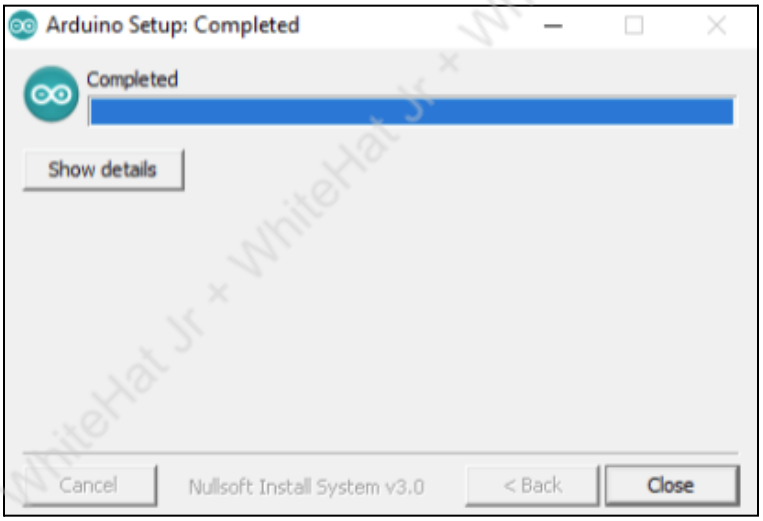




- Click on **Install**

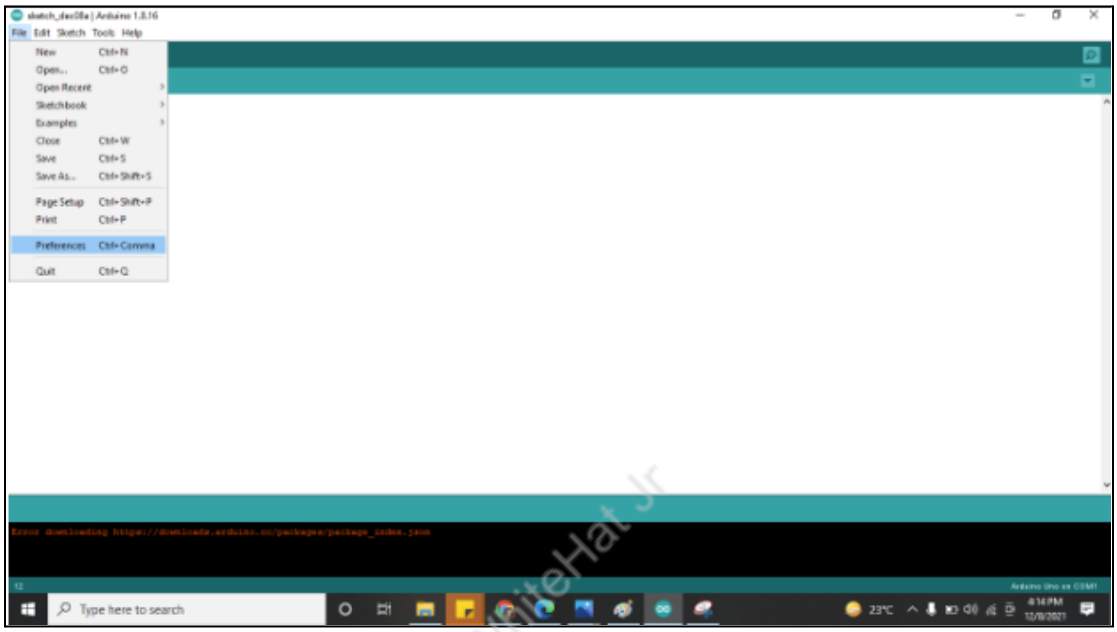


- Click on **close.**

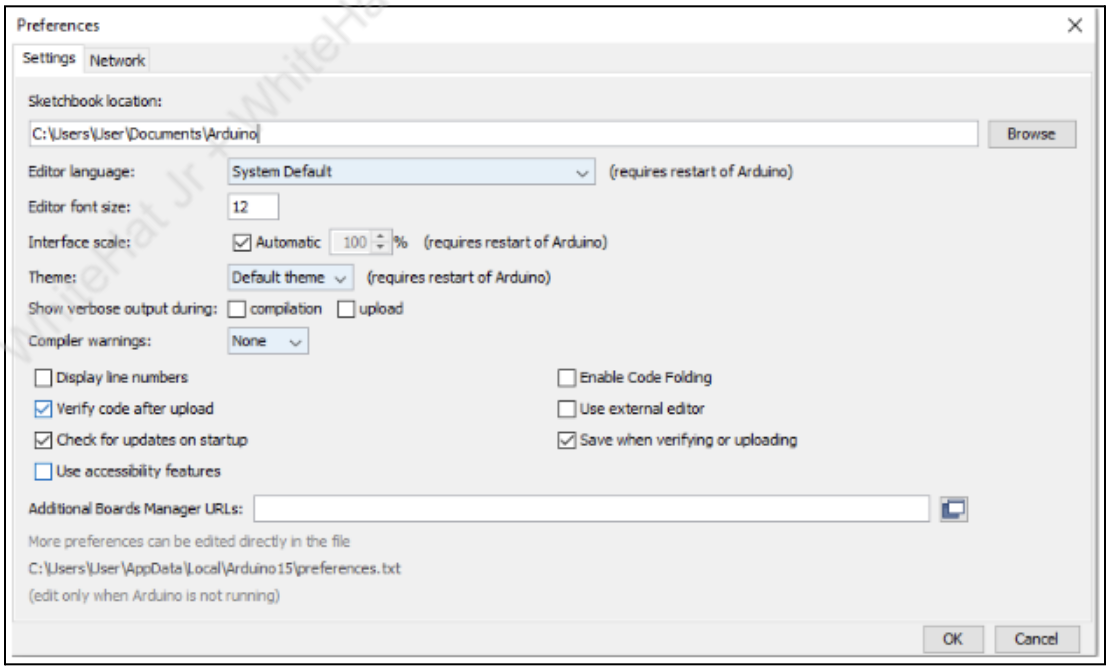


- Open **Software**

- To make it compatible with **ESP 32** change some **settings:**
  Click on **File > Preferences**

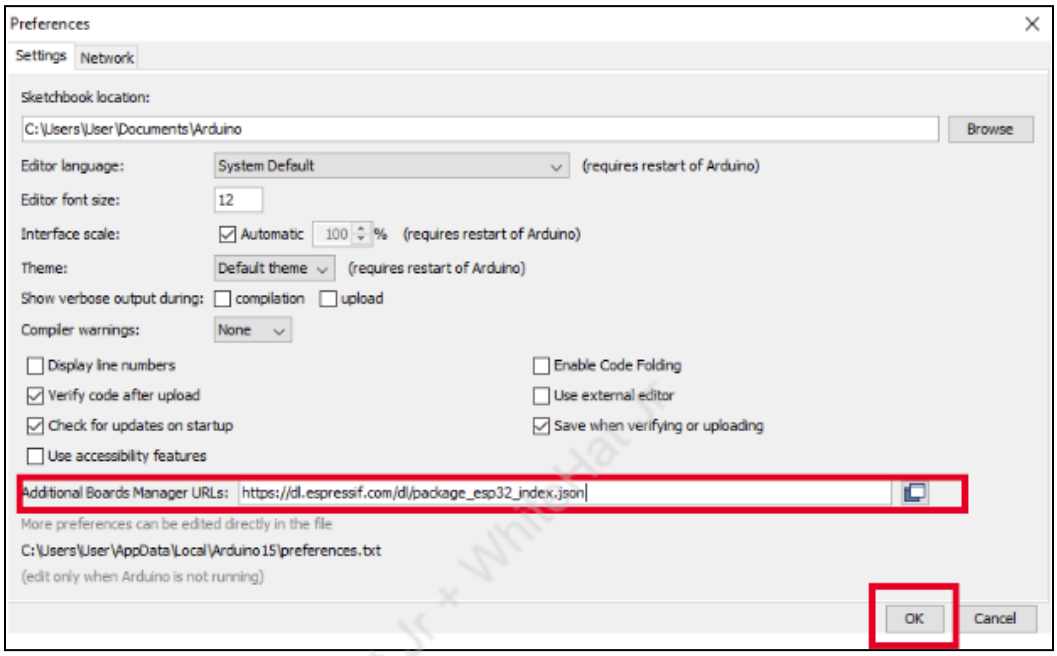


- The window will appear like below:

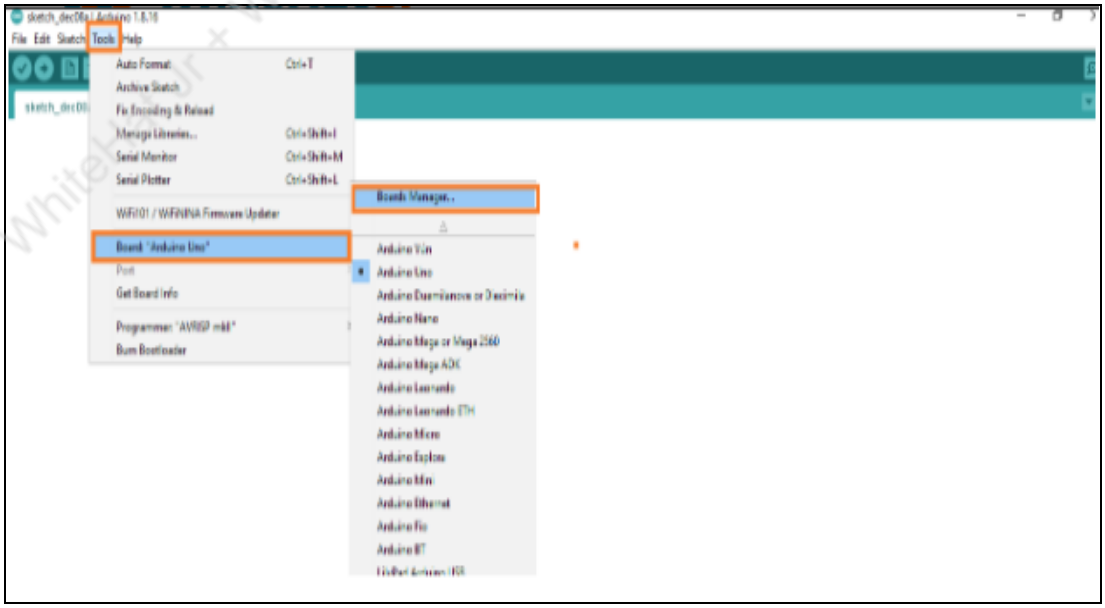


- Now copy the below link and paste the copied text into **Additional Boards Manager URLs- https://dl.espressif.com/dl/package_esp32_index.json**

Click on **OK**



- Go to the **Tools > Board Arduino Uno> BoardManager**



- Type **esp32** and then click on **Install.** Click on **close**

3. Programming using **Arduino**: **void setup()** and **void loop()** are two mandatory functions in Arduino.
    - **void setup()** function is called only once at the very beginning of the program.

      Use of **setup() function :**

        - To initialize variables and their values.
        - To set up communications (ex: Serial).
        - To set up modes for digital pins (input/output).
        - To Initialize any hardware component that plugged into the Arduino.

      The void setup, as its name suggests, is made for you to do any setup required at the beginning of the program.

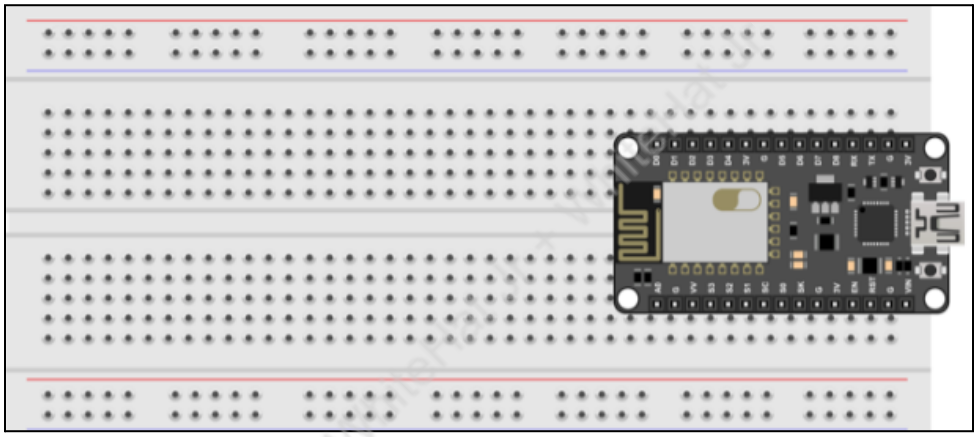
    - **void loop():** In the void loop, we will write our main program, knowing that the initialization is already done. In this function, always keep in mind that the last line is followed by the first line.

        - There's no need to write all the code directly in the function.
        - We can create as many other functions as we want and call those functions in the void loop.

4. To blink an **LED** using **ESP32 controller** & **Arduino IDE :**

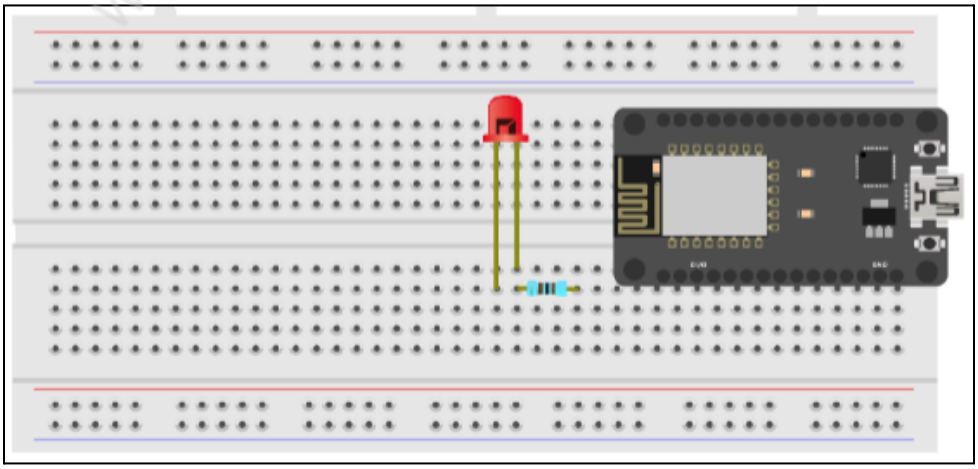
- Gather the following material from WHJR-IoT Kit
  - **ESP32**
  - **USB Cable**
  - **LED**
  - **330-ohm Resistors**
  - **Jumper Wires**
  - **Battery**
  - **Battery Socket**

- Mount the **ESP32** on the **breadboard** across both sides of the middle breakers.

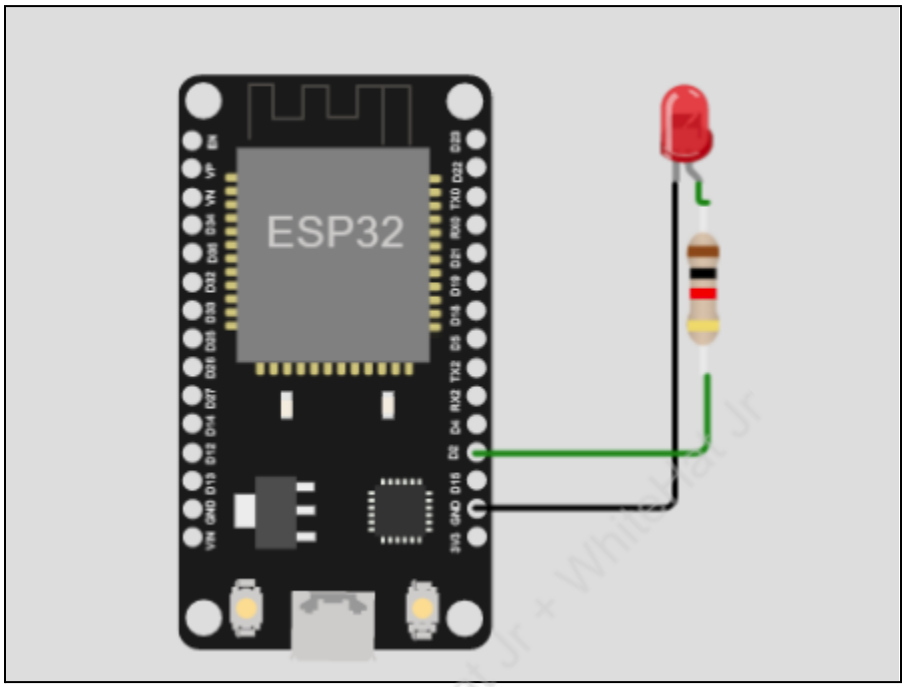


- Mount **LED** & **Resistor**

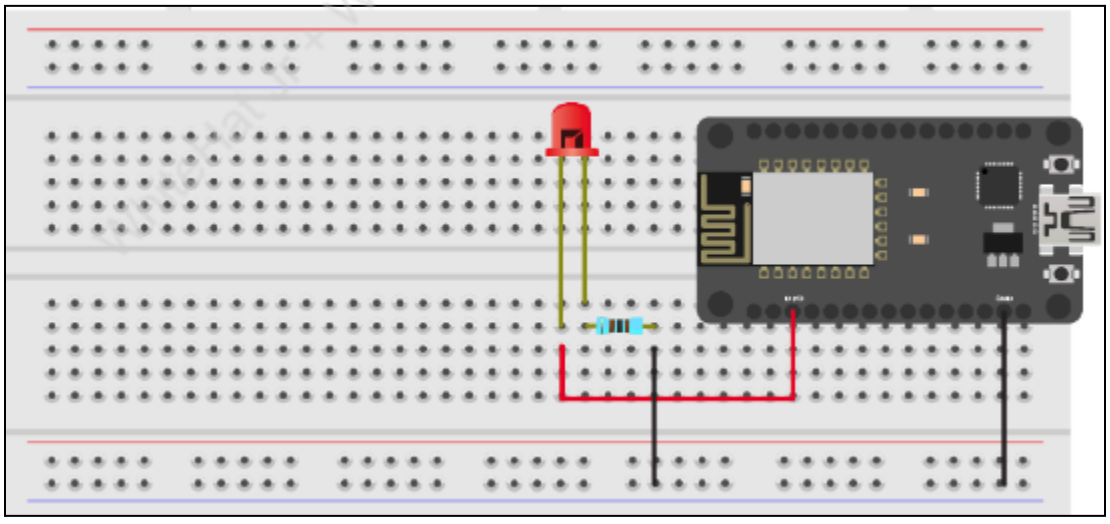
  ○ Insert the **LED** into the breadboard
  ○ Connect the shorter leg of the **LED** to one end of the **resistor** as shown



- The actual Circuit Diagram looks like this:



- Complete connections **:**



5. **Program in the Arduino**:

- Initialize **LED**
- Declare the **LED** before void **setup()**
- Configure this **LED pin** with **ESP32**
    - In **ESP32** the **LED** on board is connected to pin number **18**.
    - To configure the pin we use the **pinMode()** function
    - The **pinMode()** configures the specified pin to behave either as an input or an output. As we want to act this **pin** as output we are writing **OUTPUT** here
    - **Syntax**: **pinMode(pin, mode)**
      pin: **pin number** we want to set
      mode: Set the mode **INPUT/OUTPUT**

```
#define LED 18

void setup()
{
  pinMode(LED, OUTPUT);
}
```

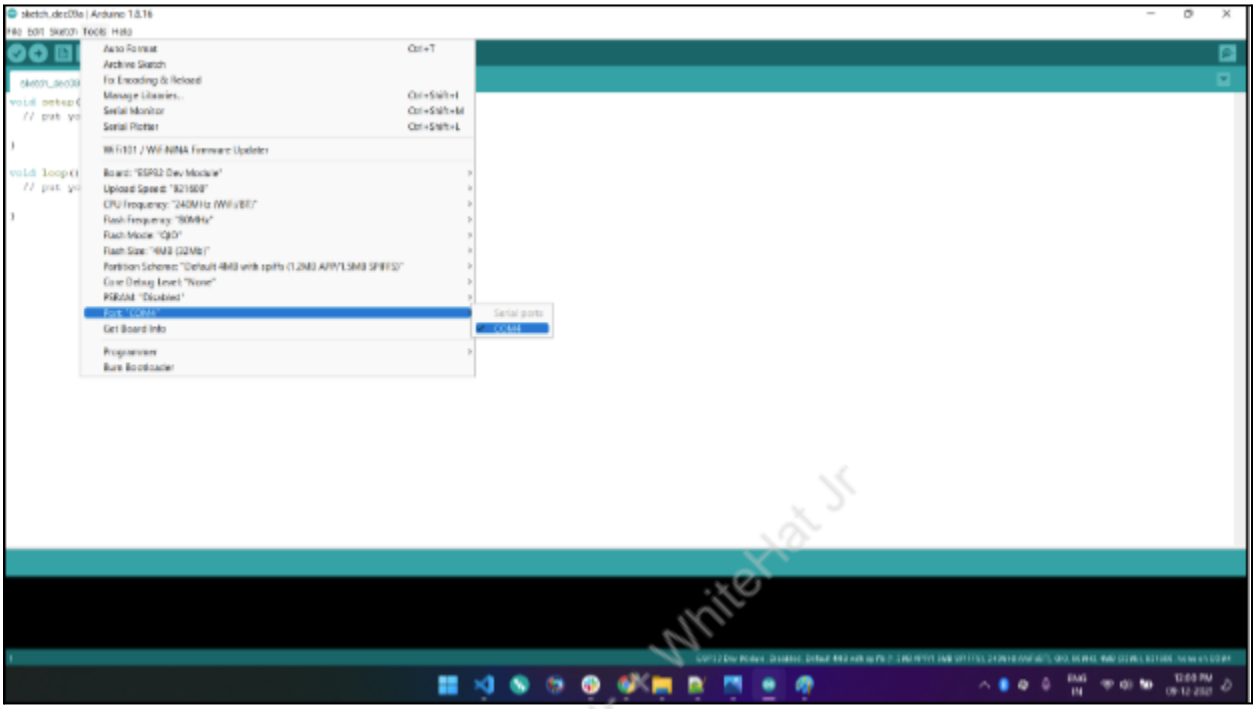
● To make it **on** or **off** we need to use a function called **digital write():**

```
void loop()
{
  digitalWrite(LED, HIGH);
  delay(500);
  digitalWrite(LED, LOW);
  delay(500);
}
```

.

● To check: Take the **USB** cable, from WHJR-IoT KiT. Insert one end of the **USB** cable in the **ESP32 USB port** and another end in the **Computer's USB port.**
Now select the port for communication:

1. Go to the Tools:
2. Select the Serial Port
3. Select the Port No



- Click on the **tick** to verify the syntax Or Click on **Sketch > Verify/Compile**

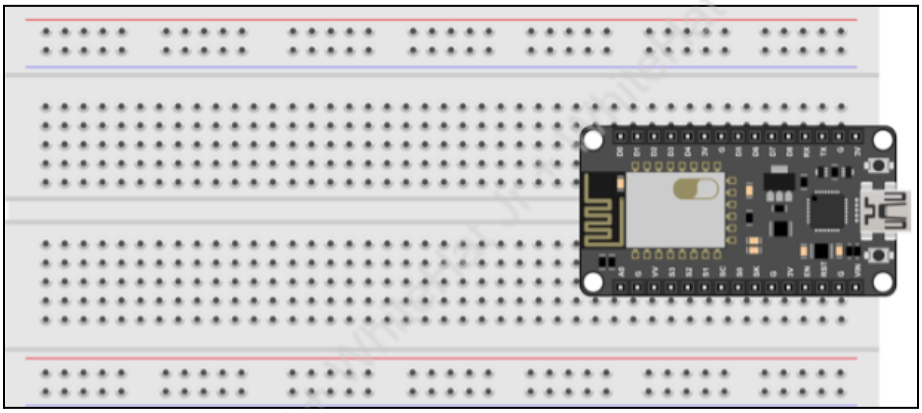
- Click on the **arrow** to upload the program Or Click on **Sketch >> Upload**

OUTPUT: If you look at your **ESP32,** you can see the 2 LEDs near Tx and Rx blinking. This is an indication of successful communication between your **PC** and **ESP32** board.
Success: "Done Uploading".
If the uploading process was not successful, you will see an error message accordingly.

6. To blink 3 **LEDs** using **ESP32 controller** & **Arduino IDE :**

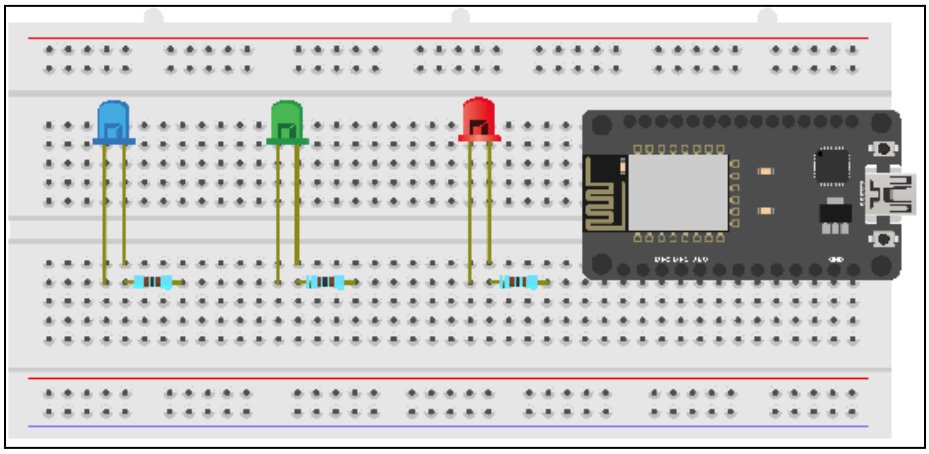
- Gather the following material from WHJR-IoT Kit
  - **ESP32**
  - **USB Cable**
  - **3 x LED**
  - **330-ohm Resistors**
  - **Jumper Wires**
  - **Battery**
  - **Battery Socket**

- Mount the **ESP32** on the **breadboard** across both sides of the middle breakers.

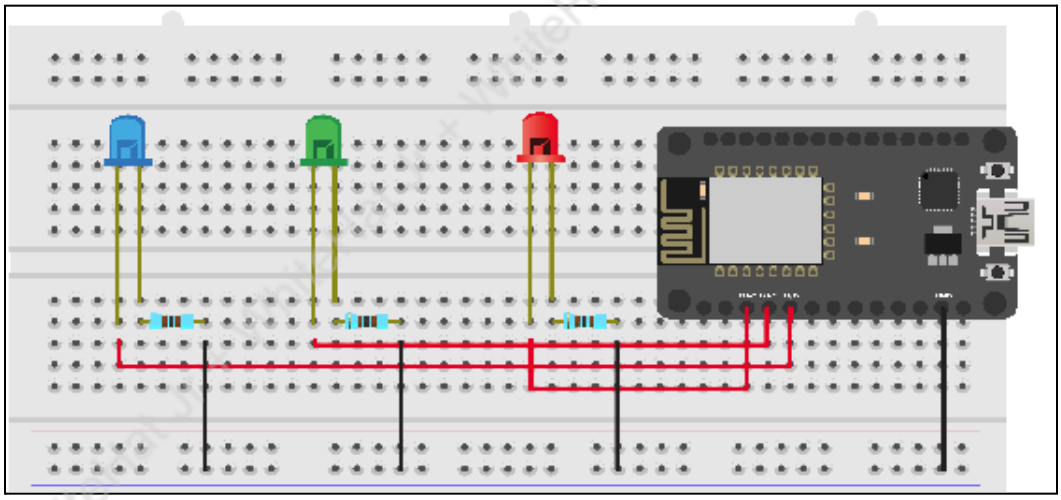


- Mount **LEDs** & **Resistors**

  ○ Insert **five LEDs** into the breadboard & **five resistors**
  ○ Connect the shorter leg of the **LED** to one end of the **resistor** as shown
  ○ Do the same for other **LEDs** & other four **resistors** too.



- Wire everything up
    - Provide **+ve** (5V) and **GND**(-ve) to **LED** & **resistor** respectively.
    - Connect the positive part (Longer Leg) of the LEDs one by one with **ESP32** pins **D25,D26, D27, D31, D32** respectively.Just below the **LED** pin, insert the male jumper wire (as all the pins in the breadboard just below the **LED** will behave the same) and drag it to the **D25**. Do the same for other **LEDs** too.
    - Connect the negative part (Shorter Leg) of the **LED** to one of the **resistor** terminals.
    - Now connect the resistor of one of the open terminals with the **GND(-ve)** pin of the **ESP32**. Take a male **jumper** wire and connect just below the **resistor** and drag it to the **GND**(-ve) terminal of the **ESP32**.Do the same for the other four **resistors** too.



7. **Program in the Arduino**:
    - Declare the global variables
        - Define variable along with datatype:
            - The '**int'**, '**const'** are called data types.
            - Datatype '**int'** is used to store an integer value
            - Define an array of **LEDS 'arr_pin'**
        - Initialize **'arr_pin'** with the pin numbers to which LEDs are attached to in the breadboard.
    - **Variables i,j** will be used for the '**for'** loop

```
int delay_ms = 200;

int arr_pin[5]={32,33,25,26,27};

int i, j;
```

- Initialize under **setup()** function
    - Range-based **for** loop for an array of **LEDs**.
    - To configure these **pin** we use the **pinMode()** function
      ● The **pinMode()** configures the specified **pin(arr_pin)** to behave either as an input or an output. As we want to act as output we are writing **OUTPUT** here
      ● **Syntax: pinMode(pin, mode)**
      ● **pin**: Which pin do we need to set
      ● **mode**: Set the mode INPUT, OUTPUT,
    - Set a delay of 200ms

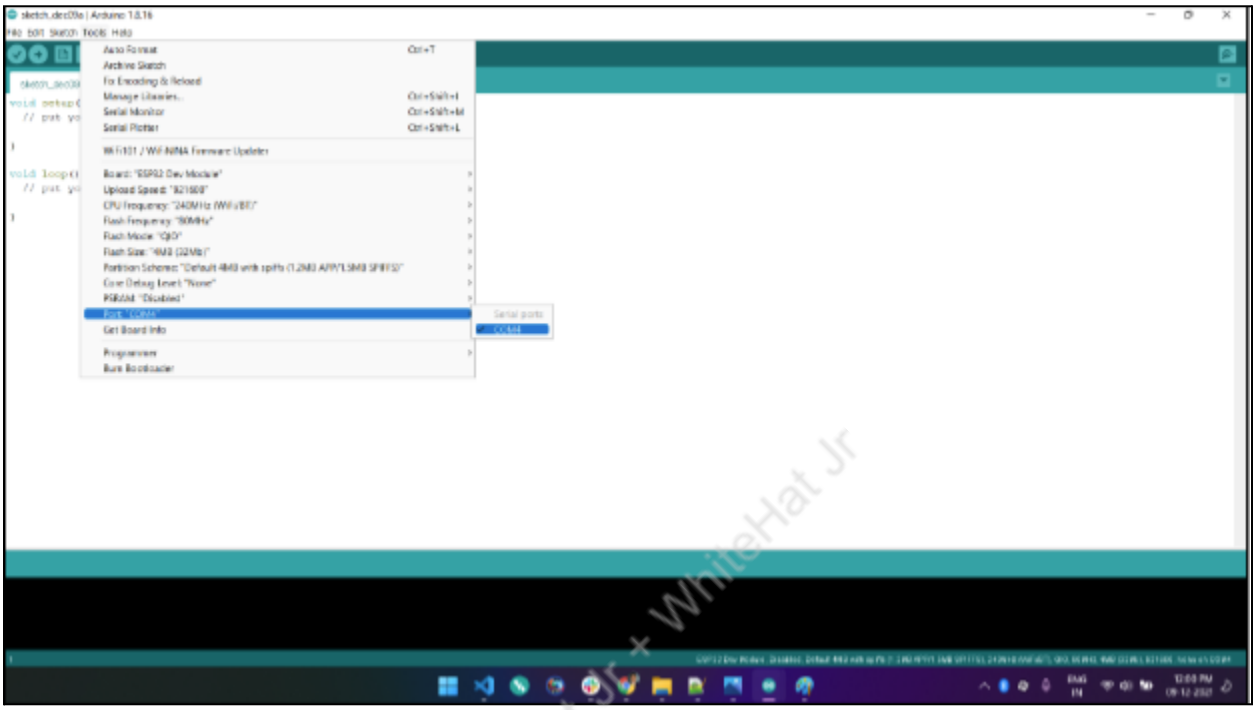
```
void setup() {
  for (int i=0; i<5; i++) {
    pinMode(arr_pin[i], OUTPUT);
  }
  delay(200);
}
```

- Blink an **LED** from higher to lower
    - All main processes need to be set under **void loop()**
    - **The for** loop is used to on and off **LED'S pin.**
    - The pin needs to be programmed to be either **ON** or **OFF,** we can command it to be **ON (output 5 volts),** or **OFF (output 0 volts).**
    - To make it on and off we need to use a function called **digitalWrite()**
    - Set up a delay

```
// loop from the highest pin to the lowest:
 for (j-4; j>-0; j--) {
  digitalWrite(arr_pin[j], HIGH);
  delay(delay_ms);
  digitalWrite(arr_pin[j], LOW);
  //delay(delay_ms);
 }
]
```

● To check: Take the **USB** cable, from WHJR-IoT KiT. Insert one end of the **USB** cable in the **ESP32 USB port** and another end in the **Computer's USB port.**
Now select the port for communication:

1. Go to the Tools:
2. Select the Serial Port
3. Select the Port No



● Click on the **tick** to verify the syntax Or Click on **Sketch > Verify/Compile**

● Click on the **arrow** to upload the program Or Click on **Sketch >> Upload**

OUTPUT: If you look at your **ESP32,** you can see the **2 LEDs** near **Tx** and **Rx** blinking. This is an indication of successful communication between your **PC** and **ESP32** board.
Success: "Done Uploading".
If the uploading process was not successful, you will see an error message.

You will see the LED blinking pattern from lowest to highest and highest to lowest in a pattern. We have successfully created **LEDs** blinking patterns using a microcontroller.

What's NEXT?

In the next class,_____.

Expand Your Knowledge

To know more about **Arduino** click here