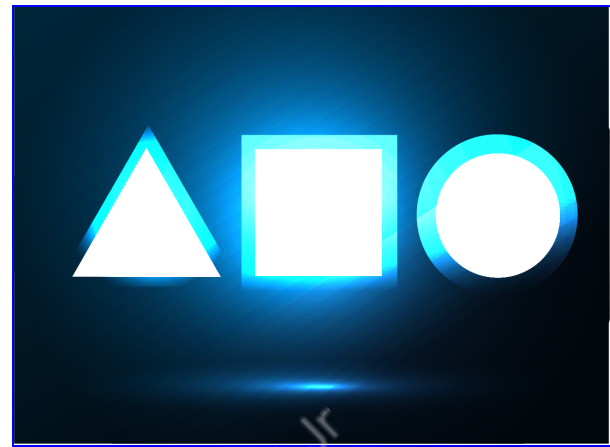


INTRODUCTION TO OLED



What is our GOAL for this CLASS?

In this class, we learned about **OLED (Organic Light-Emitting device)** and its working. We also learned how **OLED** can be used to display text, shapes.

What did we ACHIEVE in the class TODAY?

- We learned about the OLED
- Draw shapes on OLED
- Draw text on OLED

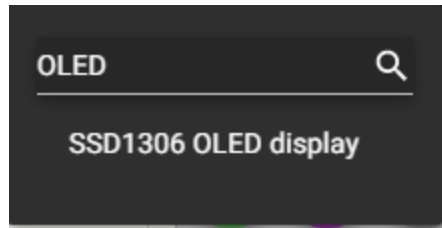
Which CONCEPTS/ CODING BLOCKS did we cover today?

- OLED:
 - An OLED stands for an organic light-emitting diode. An OLED display is made up of pixels that glow when electricity is applied to them. It's like the heating elements in a toaster, but with less heat and a better resolution. This effect is called **electroluminescence**
 - It is called organic because it is made up of organic substances, such as carbon.

How did we DO the activities?

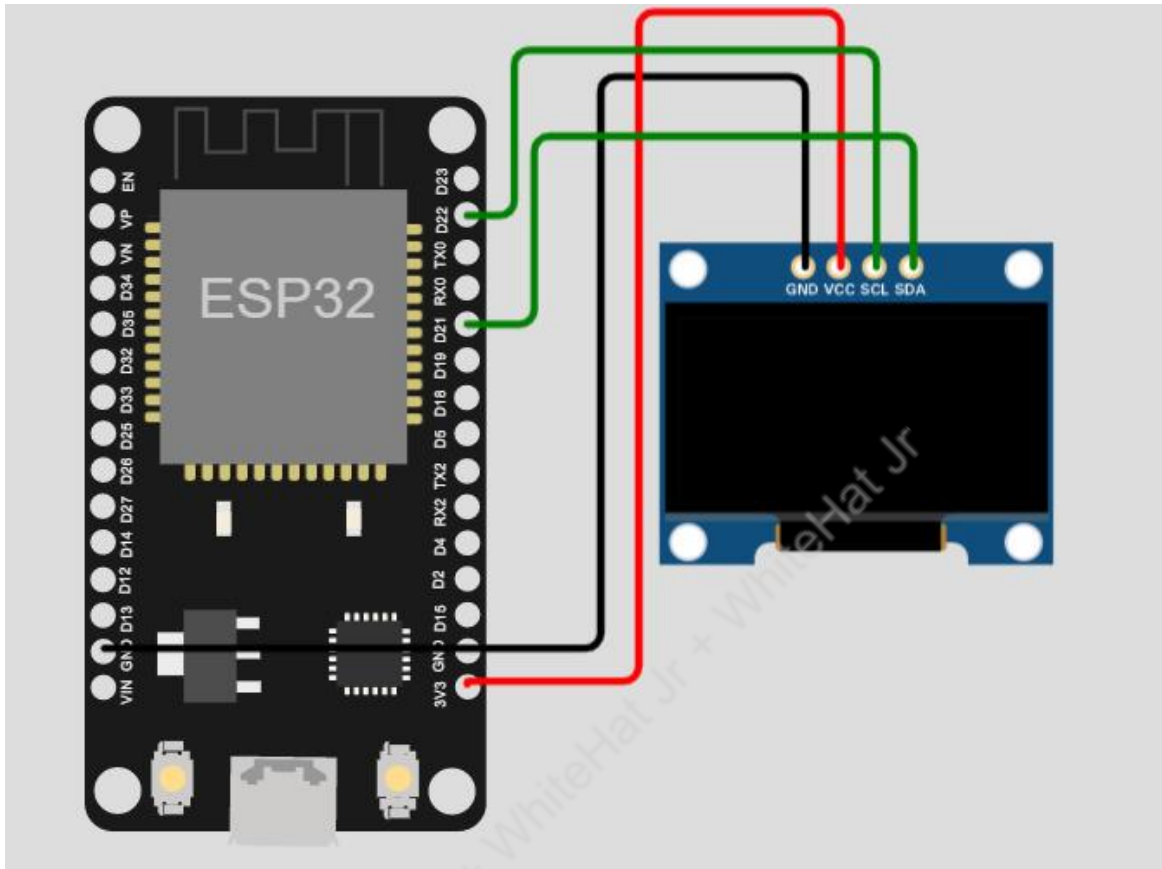
1. Display **WHITEHATJR** on the OLED.
 - **Collect the material**
 - 1 x ESP32

- **1 x OLED**



- **Connections:**

- Insert OLED into the breadboard
- Take four jumper wires.
- OLED VCC to ESP32 PIN VCC
- OLED GND to ESP32 PIN GND
- OLED Clk to ESP32 PIN GPIO22
- OLED Data to ESP32 PIN GPIO21



2. To control the **OLED** display, install libraries

- Click on the small triangle icon  next to Library Manager
- Select New File
- Name the file **libraries.txt**
- Write down **Adafruit SSD1306**

```
diagram.json  libraries.txt  OLED SA.ino  wokwi-project.txt  Library Manager ▼  
1  # Wokwi Library List  
2  # See https://docs.wokwi.com/guides/libraries  
3  
4  Adafruit SSD1306  
5  Adafruit GFX Library
```

3. Import Libraries:

- **SPI.h** **S**erial **P**eripheral **I**nterface (**SPI**) is a synchronous serial communication protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. When using SPI, there is always one master device (usually a microcontroller) that controls all peripheral devices.
- **Wire.h** This library allows you to communicate with I2C / devices. I2C is a **serial communication protocol**, so data is transferred bit by bit along a single wire.
- **Adafruit_GFX.h**: This library offers a common graphical syntax and set of functions for all LCD displays, OLED displays, and LED matrices.
- **Adafruit_SSD1306** : This library takes care of low-level communication with the hardware.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

4. Define SCREEN_WIDTH & SCREEN_HEIGHT for OLED

- **OLED** size is a 128x64

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

5. Declare **SSD1306** display that connects to **I2C** communication using **Wire** Library

- Initialize a **display** object with the **SCREEN_WIDTH & SCREEN_HEIGHT** defined earlier with I2C communication protocol.
- A value of **(-1)** indicates that our OLED display does not have a **RESET** pin. Sometimes OLED displays have a RESET pin on the OLED, in that case we should connect it to a GPIO and should include the GPIO number as a parameter.

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

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

- **Serial.begin(115200)**: Sets the **data rate** in bits per second (baud) for **serial** data transmission.
- Initialize the OLED display with the **begin()** method.
- If the OLED displays nothing, check the OLED address at **0x3C**. In our case, the address is **0x3C**.
- If we are not able to connect to the display, it prints a message on the Serial Monitor.

```
void setup() {
    Serial.begin(115200);

    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("SSD1306 allocation failed"));
        for(;;);
    }
}
```

7. Print data on OLED

- to initialize, add a **two second delay** before writing text
- Clear the display buffer with the **clearDisplay()** method after initializing the display
- To **write** text set the font size, color, and location where the text will be displayed in the OLED and data which need to be printed.
- Set the font size using the **setTextSize()** method
- Set the font color using the **setTextColor()** method. **WHITE** sets white font and black background.
- Using the **setCursor(x,y)** method, specify the starting point of the text. In this case, the text will be started at **(0,10)**.
- send the text to the display using the **println()** method
- Call the **display()** method to isplay the text on the screen.

```
delay(2000);
display.clearDisplay();

display.setTextSize(2);
display.setTextColor(WHITE);
display.setCursor(10, 20);
display.println("WHITEHATJR");
display.display();
}
```

- Call the main function using **void loop()**

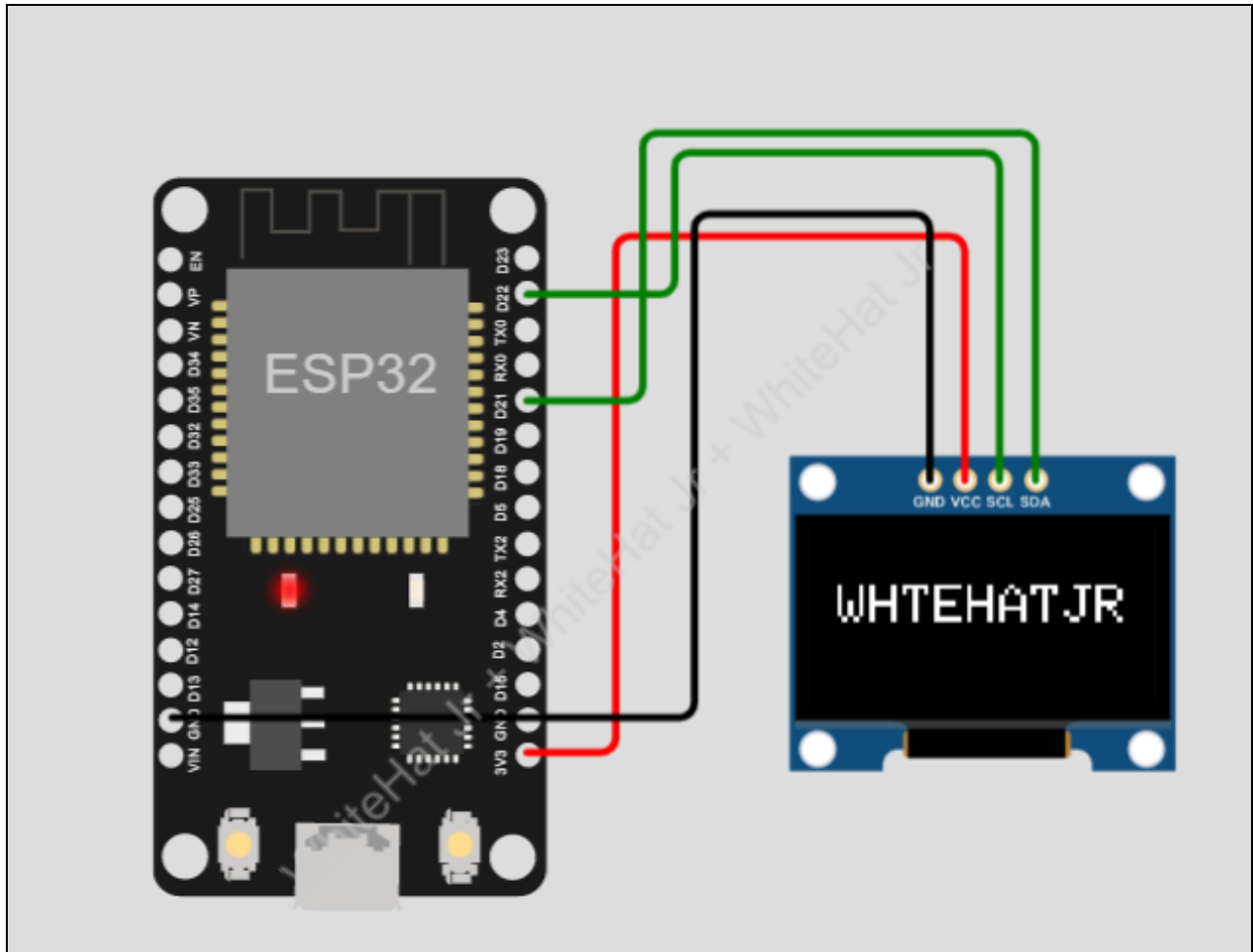
```
void loop() {

}
```

8. 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

- If your OLED display is not showing anything:
- Check that the OLED display is properly wired



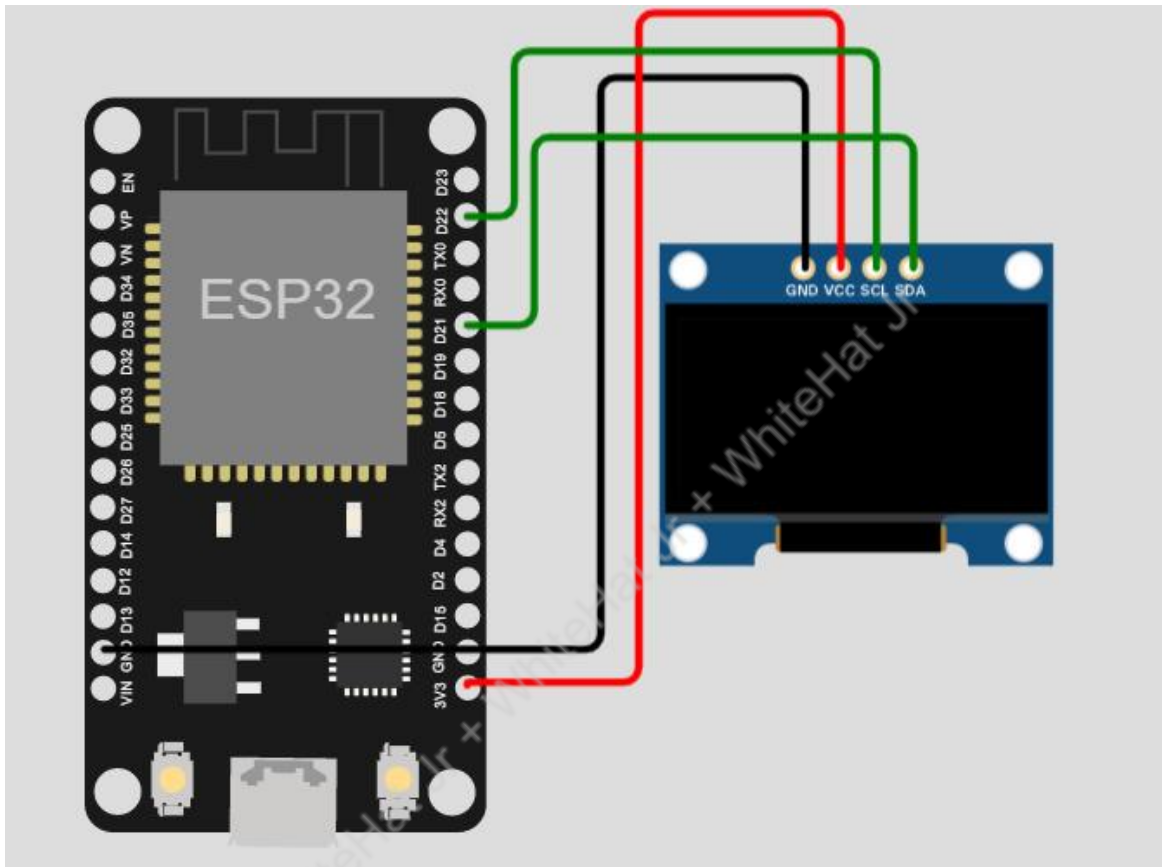
9. To print shapes on **OLED**.

- **Collect the material**
 - **1 x ESP32**
 - **1 x OLED**

10. Let's do connections:

- Insert OLED into the breadboard
- Take four jumper wires.
- Connect OLED **PIN VCC** to **ESP32 PIN 3.3V**
- Connect OLED **PIN GND** to **ESP32 PIN GND**
- Connect OLED **PIN CLK** to **ESP32 PIN GPIO22**

- Connect OLED PIN DATA to ESP32 PIN GPIO21



11. Import Libraries:

- **SPI.h Serial Peripheral Interface (SPI)** is a synchronous serial communication protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. When using SPI, there is always one master device (usually a microcontroller) that controls all peripheral devices.
- **Wire.h** This library allows you to communicate with I2C / devices. I2C is a **serial communication protocol**, so data is transferred bit by bit along a single wire.
- **Adafruit_GFX.h:** This library offers a common graphical syntax and set of functions for all LCD displays, OLED displays, and LED matrices.
- **Adafruit_SSD1306:** This library takes care of low-level communication with the hardware.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

12. Define **SCREEN_WIDTH** & **SCREEN_HEIGHT** for OLED

- Our **OLED** size is a **128x64**

```
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

13. Declaration of an **SSD1306** display that connects to **I2C** communication using **Wire** Library

- Initialize a **display** object with the **SCREEN_WIDTH** & **SCREEN_HEIGHT** defined earlier with I2C communication protocol.
- A value of **(-1)** indicates that our OLED display does not have a **RESET** pin. Sometimes OLED displays have a RESET pin on the OLED, in that case connect it to a GPIO and include the GPIO number as a parameter.

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

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

- **Serial.begin(115200)**: Sets the **data rate** in bits per second (baud) for **serial** data transmission.
- Initialize the OLED display with the **begin()** method
- If the OLED displays nothing, check the OLED address at **0x3C**. In our case, the address is **0x3C**.
- If we are not able to connect to the display, it prints a message on the Serial Monitor.
- If something fails, don't proceed further, try to repeat the process using **for()** loop
- Using the **setCursor(x,y)** method, specify the starting point of the text. In this case, the text will be started at **(0,0)**.


```
void setup() {  
    Serial.begin(9600);  
  
    // initialize OLED display with I2C address 0x3C  
    if (!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  
        Serial.println(F("failed to start SSD1306 OLED"));  
        while (1);  
    }  
  
    delay(2000); // wait two seconds for initializing  
    oled.setCursor(0, 0);  
}
```

15. Write code for shapes that need to display under the main function i.e **void loop()** function.

- Circle:
 - Clear the display buffer with the **clearDisplay()** method after initializing the display
 - **drawCircle** method is used to draw circle shape on the OLED. drawCircle will use **X** and **Y** coordinates along with Radius (CenterX, CenterY, Radius in Pixels, WHITE);
 - **display.display()** is used to apply the changes.
 - Set a **delay** of 1s
 - **fillCircle** method is used to fill color in the circle shape on the OLED. **fillCircle** will use **X** and **Y** coordinates along with Radius (CenterX, CenterY, Radius in Pixels, WHITE);

```
void loop() {  
    // draw a circle  
    display.clearDisplay();  
    display.drawCircle(50, 30, 30, WHITE);  
    display.display();  
    delay(1000);  
  
    // fill a circle  
    display.clearDisplay();  
    display.fillCircle(50, 30, 30, WHITE);  
    display.display();  
    delay(1000);  
}
```

- Triangle:
 - Clear the display buffer with the **clearDisplay()** method after initializing the display.
 - **drawTriangle()** method is used to draw a triangle shape on the OLED. **drawTriangle()** will use **X** and **Y** coordinates for three sides of a triangle

- along with color.
- **display.drawTriangle (FirstX , FirstY, SecondX, SecondY, ThirdX, ThirdY, WHITE).**
 - **display.display()** is used to apply the changes.
 - Set a **delay** of 1s.
 - **fillTriangle()** method is used to fill color in a triangle shape on the OLED. **drawtriangle()** method will use X and Y coordinates for three sides of a triangle along with color.

```
// draw a triangle
display.clearDisplay();
display.drawTriangle(50, 10, 0, 60, 60, 60, WHITE);
display.display();
delay(1000);

// fill a triangle
display.clearDisplay();
display.fillTriangle(50, 10, 0, 60, 60, 60, WHITE);
display.display();
delay(1000);
```

- **Rectangle**
 - Clear the display buffer with the **clearDisplay()** method after initializing the display
 - **drawRectangle()** method is used to draw a rectangle shape on the OLED. **drawRectangle** will use X and Y coordinates, Width & Height in Pixels along with the color
 - **display.drawRect(StartX, StartY, Width in Pixels, Height in Pixels, WHITE);**
 - **isplay.display()** is used to apply the changes.
 - Set a **delay** of 1s
 - **fillRectangle()** method is used to fill color in a rectangle shape on the OLED. **drawRectangle** will use X and Y coordinates, Width & Height in Pixels along with the color

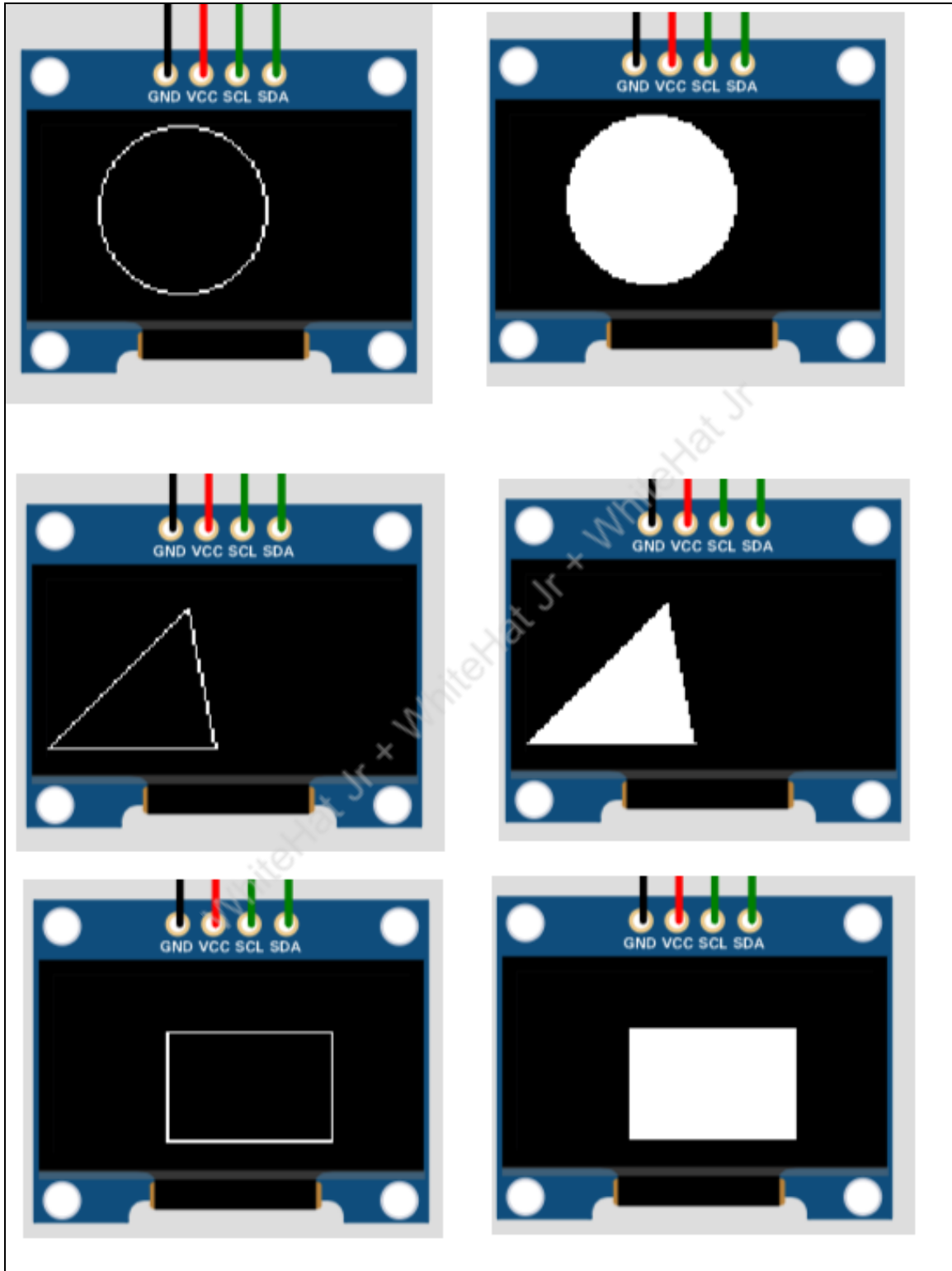
```
// draw a rectangle
display.clearDisplay();
display.drawRect(40, 20, 60, 40, WHITE);
display.display();
delay(1000);

// fill a rectangle
display.clearDisplay();
display.fillRect(40, 20, 60, 40, WHITE);
display.display();
delay(1000);

}
```

16. 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**
- **If OLED display is not showing anything: Check that the OLED display is properly wired**
- **We learned about OLED and how to display text and pattern on OLED.**



What's NEXT?

In the next class, we will learn about BIT ARRAYS

Expand Your Knowledge

To know more about OLED [click here](#).

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr