**HOW TO RUN A CODE**

To run the code written in Embedded C for an ESP32 microcontroller using the Arduino IDE, we need to follow a series of steps to ensure everything runs smoothly. Here's a detailed guide:

**1. Install Arduino IDE:** Download and install the Arduino IDE from the official website. Make sure to select the correct version compatible with our operating system. Here we have installed version **v1.8.19**

**2. Install ESP32 Board Package:** The Arduino IDE doesn't natively support the ESP32. You'll need to install the ESP32 board package to enable programming for this microcontroller. Navigate to "**File**" > "**Preferences**" in the Arduino IDE, and paste the following URL into the "Additional Board Manager URLs" field: **https://dl.espressif.com/dl/package\_esp32\_index.json.** Then, go to "**Tools**" > "**Board**" > "**Boards Manager**", search for "**ESP32**" and install the package.

**3. Select the ESP32 Board:** Once the ESP32 board package is installed, go to "Tools" > "Board" and select your ESP32 board from the list. Ensure you choose the correct board variant matching your hardware. Here we have installed **ESP32Servo-1.1.2**

**4. Connect the ESP32:** Connect the ESP32 microcontroller to the computer using a USB cable. Make sure it's properly connected and recognized by the operating system.

**5. Open the Code:** Open the code in the Arduino IDE.

**6. Verify and Compile:** Click the verify button (checkmark icon) to compile the code. This step checks for any syntax errors in the code. If there are errors, the IDE will highlight them in the console at the bottom of the window.

**7. Upload the Code:** Once your code compiles successfully, click on the upload button (right arrow icon) to upload the code to your ESP32. The IDE will compile your code again and then upload it to the microcontroller. Make sure the correct port is selected under "**Tools**" > "**Port**".

**8. Debugging:** If the code isn't working as expected, double-check your connections, code logic, and any external components you're using (e.g., sensors, actuators). You can also use serial debugging statements to print out variable values and messages to the serial monitor for troubleshooting.