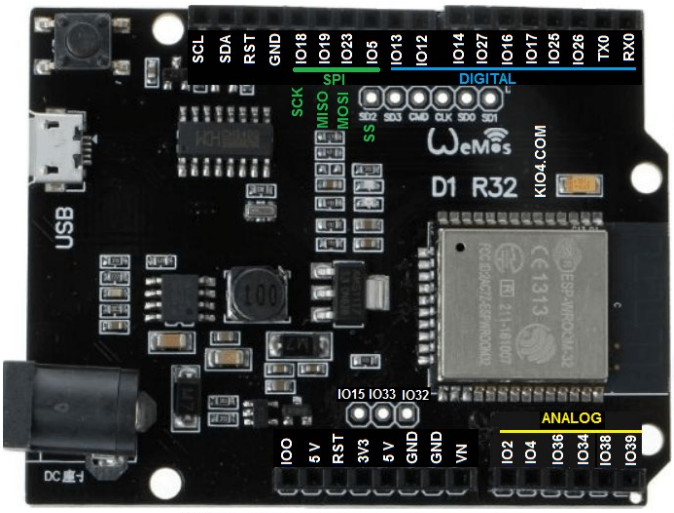
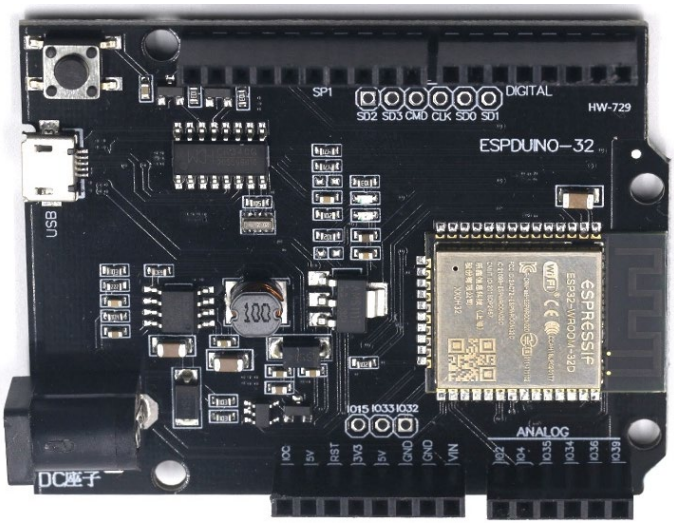
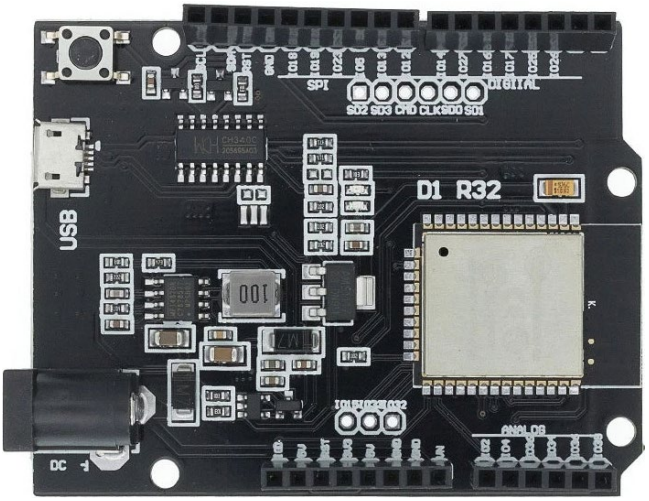
WEMOS® D1 R32 W/ ESP32 UNO R3







Be aware that the pins Arduino use are the GPIO numbers. i.e. GPIO1 would just be 1. Also, some pins are input only such as GPIO34, 35, 36, and GPIO39.[1]

Features

- **Standard UNO size and headers**
- **CPU and Memory: Xtensa® 32-bit LX6 Dua-core processor, up to 600 DMIPS**
- **4 MByte SPI Flash**
- **448 KByte ROM**
- **520 KByte SRAM**
- **Compact size of 68mm x 53mm x 3.1mm (±0.2mm)**
- **Supply Voltage: DC 5V to 12V**

Technical Specifications

WiFi

[1] https://designtech.blogs.auckland.ac.nz/d1-r32-esp32/

- **802.11 b/g/n/e/i**
- **802.11 n (2.4 GHz), up to 150 Mbps**
- **802.11 e: QoS for wireless multimedia technology.**
- **WMM-PS, UAPSD**
- **MPDU and A-MSDU aggregation**
- **Block ACK**
- **Fragmentation and Defragmentation**
- **Automatic Beacon monitoring/scanning**
- **802.11 i security features: pre-authentication and TSN**
- **Wi-Fi Protected Access (WPA)/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup (WPS)**
- **Infrastructure BSS Station mode/SoftAP mode**
- **Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode and P2P Power Management**
- **UMA compliant and certified**
- **Antenna diversity and selection**

Bluetooth

- **Compliant with Bluetooth v4.2 BR/EDR and BLE specification**
- **Class-1, class-2 and class-3 transmitter without external power amplifier**
- **Enhanced power control**
- **+10 dBm transmitting power**
- **NZIF receiver with -98 dBm sensitivity**
- **Adaptive Frequency Hopping (AFH)**
- **Standard HCI based on SDIO/SPI/UART ? High speed UART HCI, up to 4 Mbps**
- **BT 4.2 controller and host stack**
- **Service Discover Protocol (SDP)**
- **General Access Profile (GAP)**
- **Security Manage Protocol (SMP)**
- **Bluetooth Low Energy (BLE)**
- **ATT/GATT**
- **HID**
- **All GATT-based profile supported**
- **SPP-Like GATT-based profile**
- **BLE Beacon**

- **A2DP/AVRCP/SPP, HSP/HFP, RFCOMM**
- **CVSD and SBC for audio codec**
- **Bluetooth Piconet and Scatternet**

[!] Notes:[2]

Do NOT use this board with 5V shields or the shields which are not compatible with 3.3V. These shields can permanently damage the board! Use only with 3.3V compatible shields and modules.

Documents and Downloads

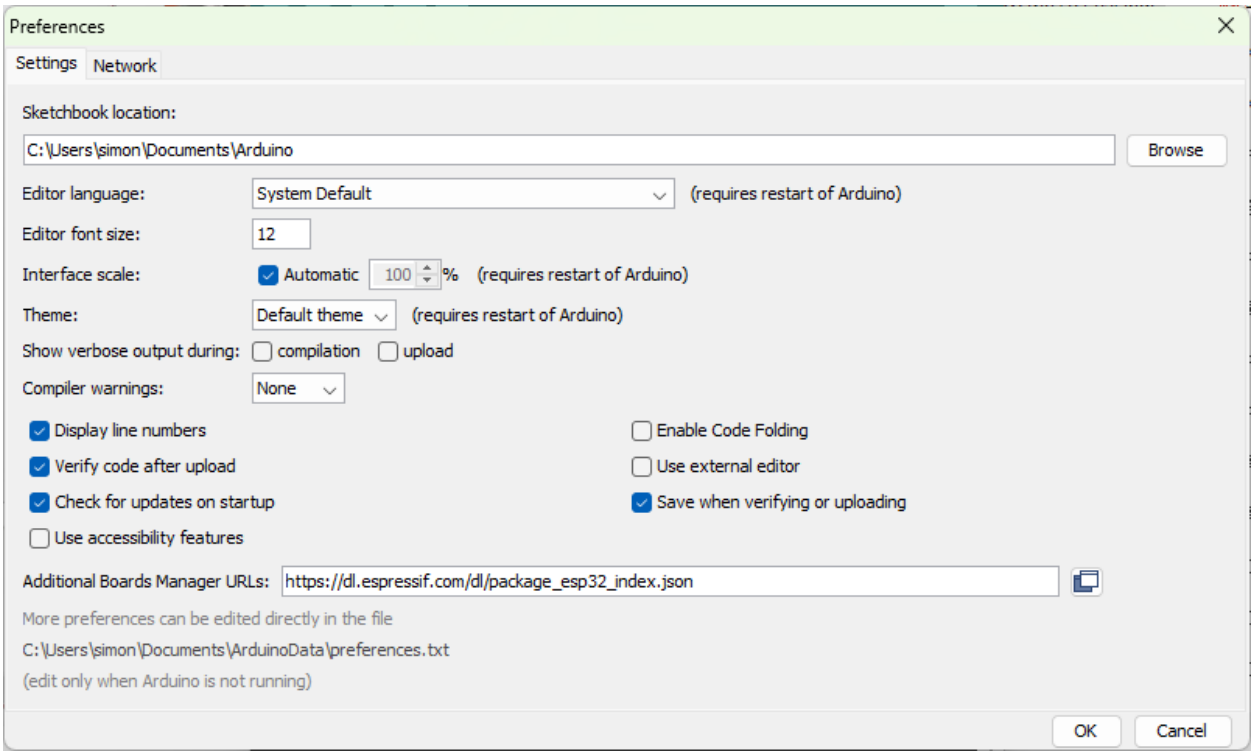
- **ESPDuino32 Board Schematic**
- **ESP32 Datasheet**

Setting Up[3]

1.) Before we plug into the Wemos R32 board, we need to install Arduino first. Get the Arduino from this link: https://www.arduino.cc/en/Guide/HomePage

2.) Afterwards, open the Arduino and obtain the ESP32 support first:

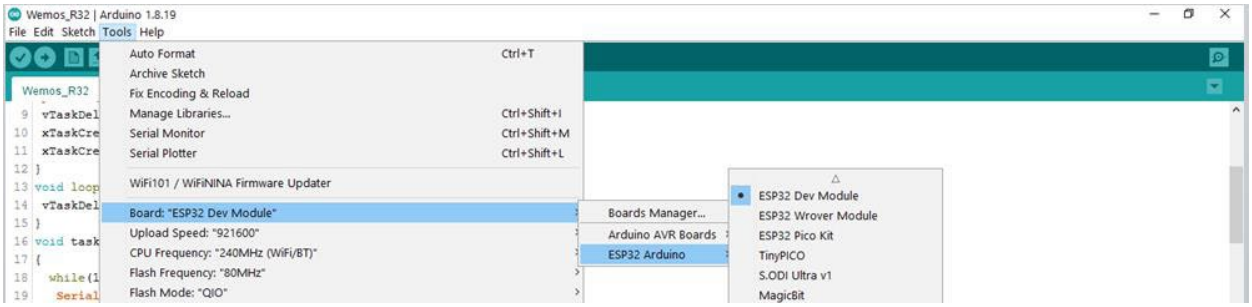
Select **File** -> **Preferences**, and then copy this link (*https://dl.espressif.com/dl/package_esp32_index.json*) into the **Additional Boards Manager URLs and click OK**:



3.) Go to **Tools** -> **Board** -> **Boards Manager**, then type in "ESP32". When you see this, press **Install**:

[2] https://artofcircuits.com/product/wemos-d1-r32-espduino32-4mb-wi-fi-and-bluetooth-board
[3] https://www.hackster.io/NYH-workshop/wemos-r32-with-arduino-startup-guide-7bc841#overview

4.) Once it's all done, go to **Tools** -> **Board** -> **Boards Manager**, and then select "**ESP32 Dev Module**":



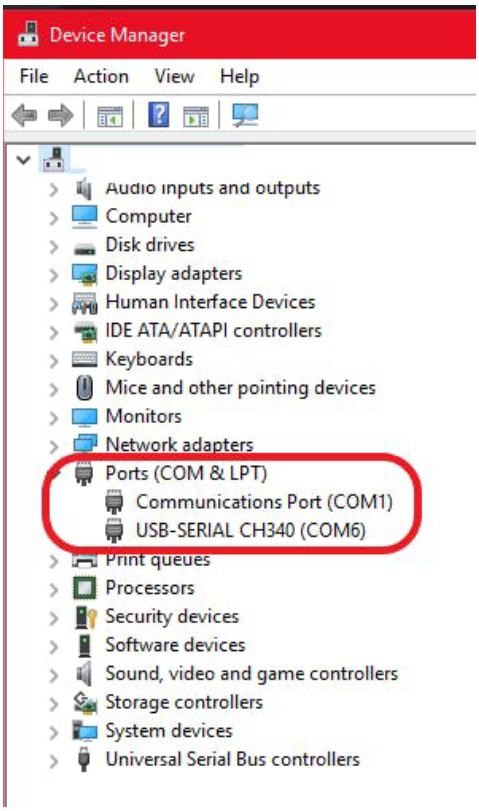
5.) Create a new project! Type this in:

```
void setup() {
 Serial.begin(112500);
 pinMode(2, OUTPUT);
 vTaskDelay(1000 / portTICK_PERIOD_MS);
 xTaskCreate(task1,"task1", 2048, NULL,1,NULL);
 xTaskCreate(task2,"task2", 2048, NULL,1,NULL);
}
void loop() {
 vTaskDelay(1000 / portTICK_PERIOD_MS);
}
void task1( void * parameter )
{
  while(1) {
   Serial.println("Hello World!");
   vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
void task2( void * parameter)
{
  while(1) {
     digitalWrite(2, HIGH);
     vTaskDelay(500 / portTICK_PERIOD_MS);
     digitalWrite(2, LOW);
```

```
    vTaskDelay(500 / portTICK_PERIOD_MS);
  }
}
```

6.) Save the project.

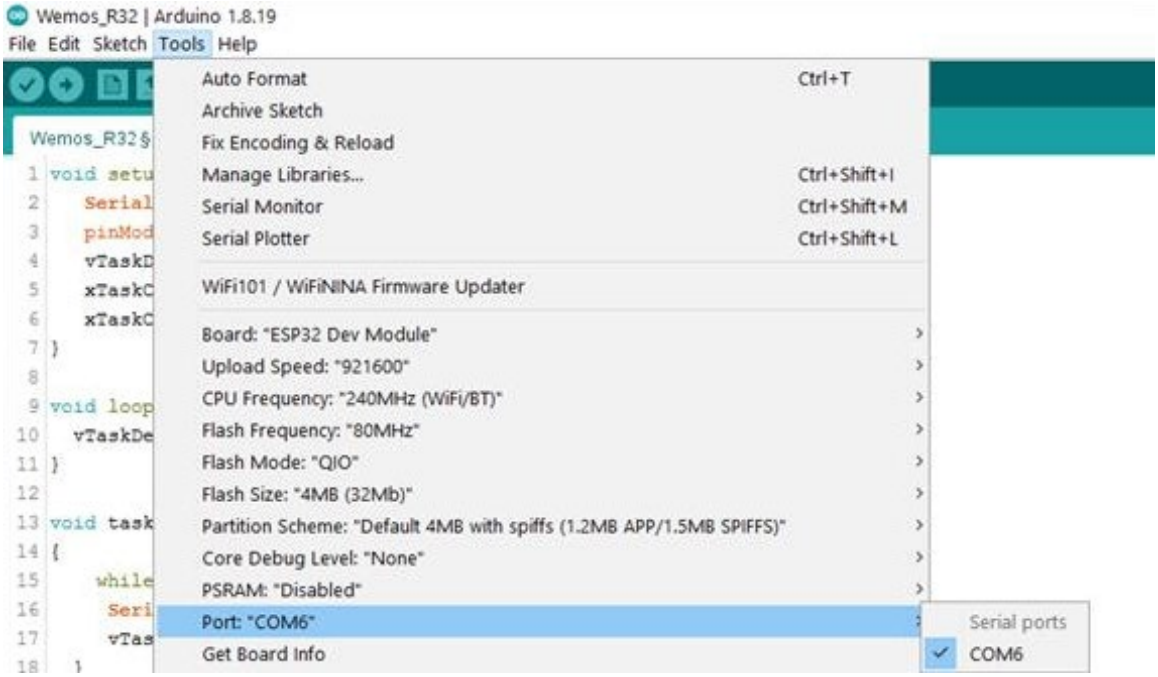
7.) Before you upload, connect the R32 board to your PC. Find the COM port that is with the R32 board through Device Manager. The board is equipped with the CH340, so take note of the COM number:



In this example it's COM6, so it will be different on your PC.

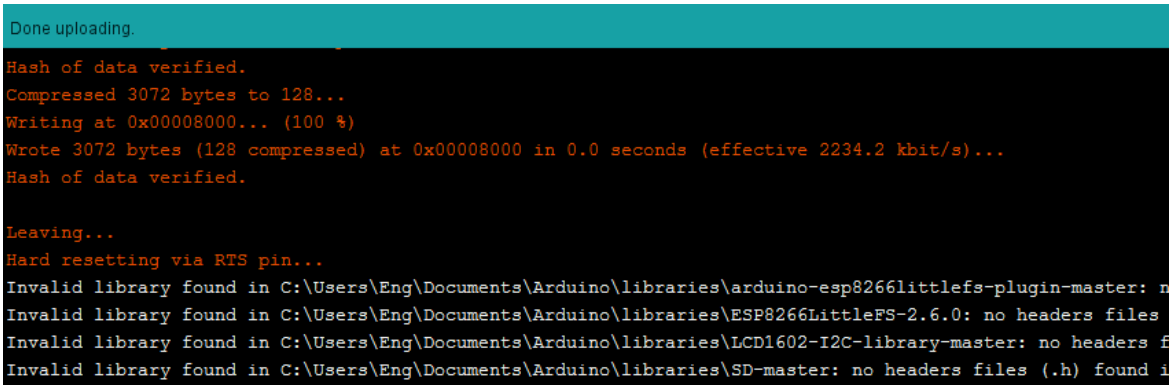
Note: If you couldn't find this "USB-SERIAL CH340", the drivers are probably not installed. You can refer to the CH340 drivers installation guide there: https://sparks.gogo.co.nz/ch340.html

8.) Back to your Arduino, select your COM port that is connected to the board:



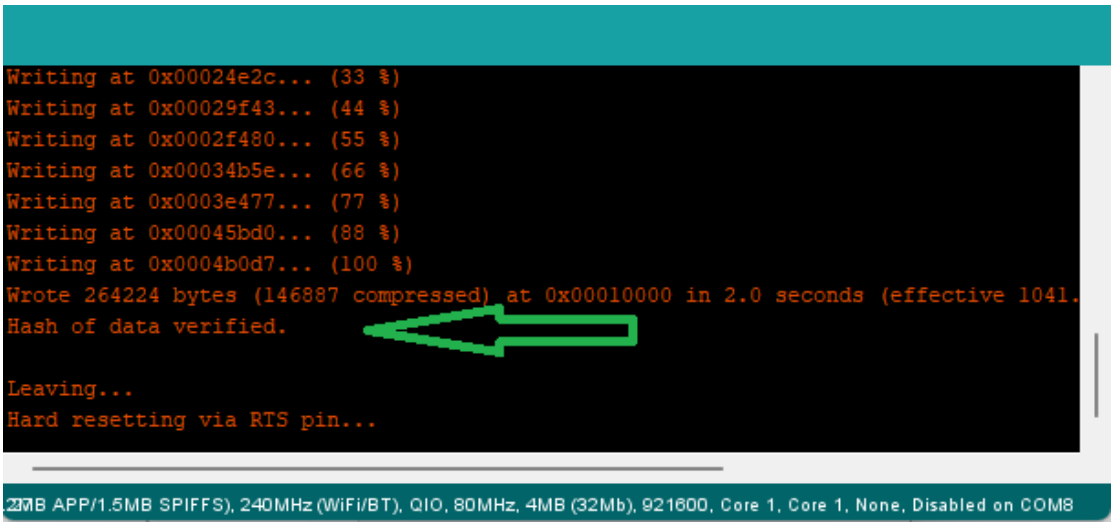
9.) Press "**Upload**".

When you see the "Connecting...", press the Reset button for about 4 seconds, and then release it:

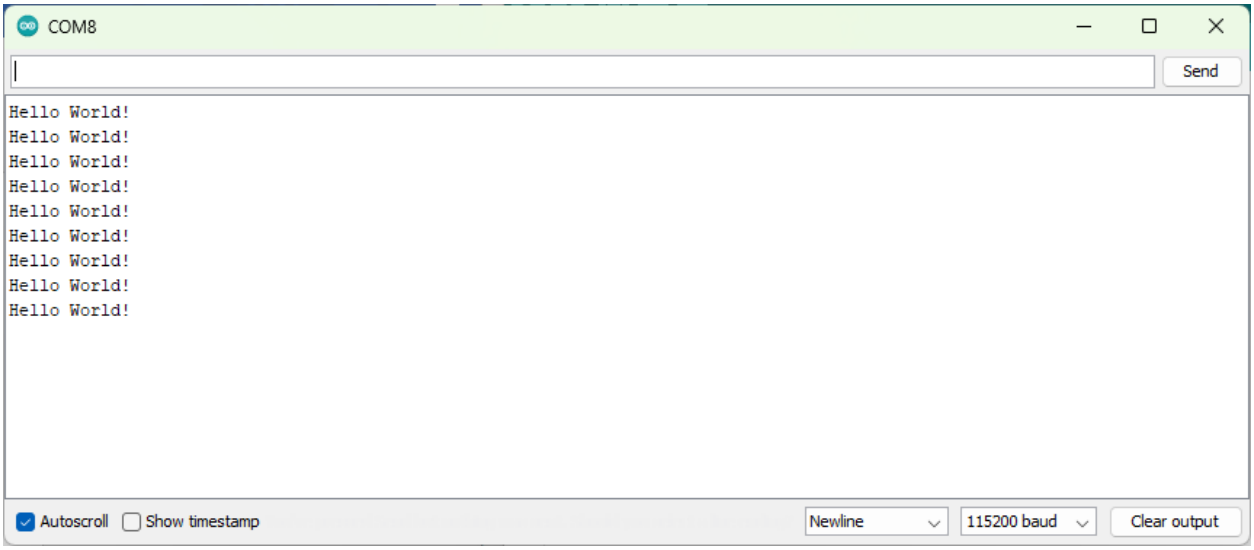


You will see a lot of messages scrolling downwards. It is in the process of uploading the binary to the ESP32

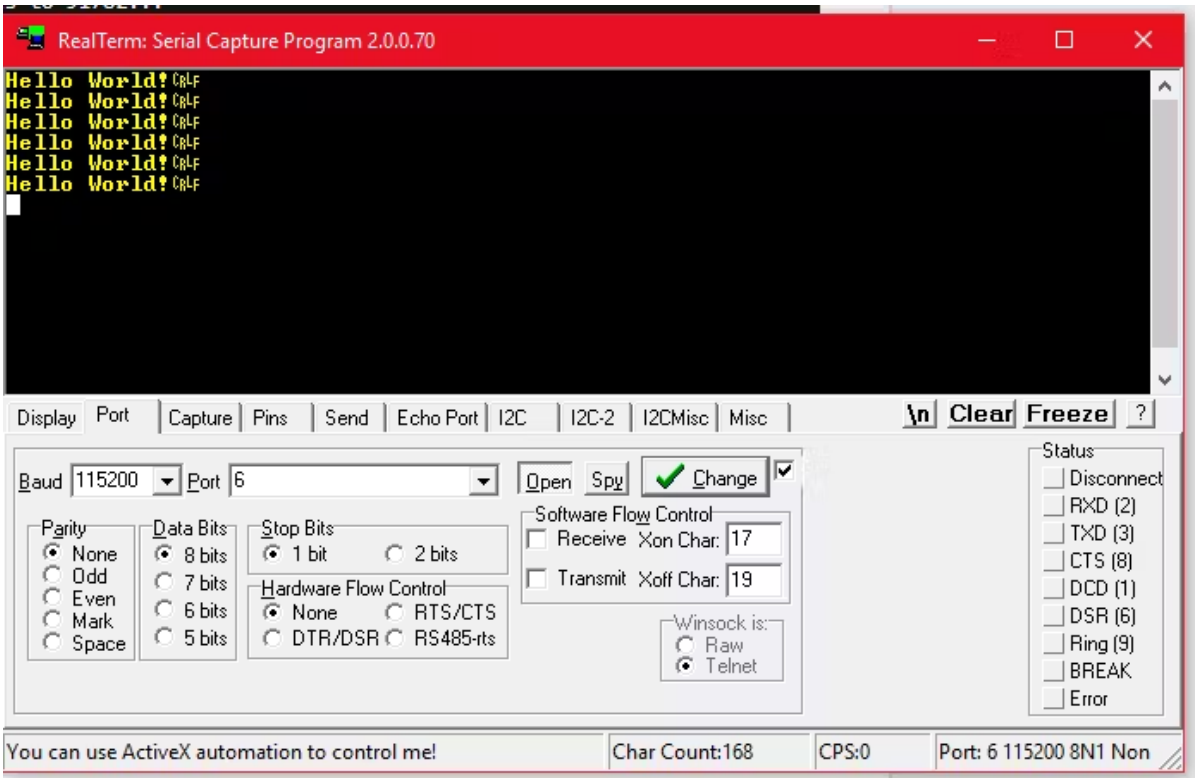
The upload is successful when it says : "**Hash of data verified. Leaving...**" at the end of the process.



10.) Open a terminal program and you will see "Hello World!" printed on it each second, and the LED on the board toggles at the rate of 500ms (115200 baud):



Or using “RealTerm: Serial/TCP Terminal” program
Download from here: https://osdn.net/projects/sfnet_realterm/downloads/Realterm/2.0.0.70/Realterm_2.0.0.70_setup.exe/



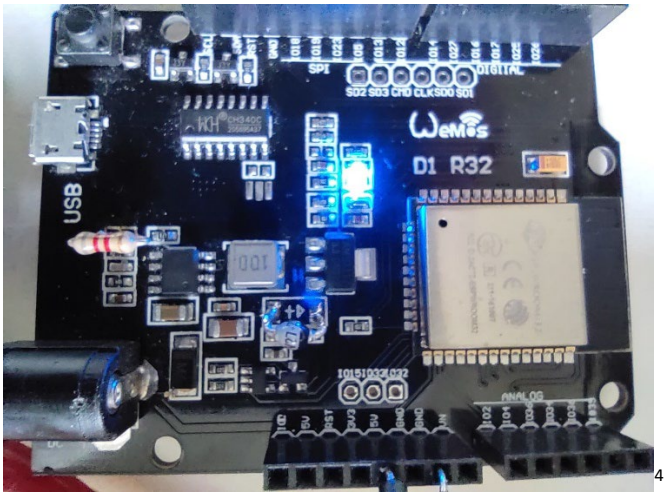
You are done here!

Sketch examples

Blinking LED

int ledPin = 2;

```
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
```



[4]

PWM - Pulse Width Modulation

```
#define LEDC_CHANNEL_0 0
#define LEDC_TIMER_13_BIT 13
#define LEDC_BASE_FREQ 5000
#define LED_PIN 2
int brightness = 0;
int fadeAmount = 5;
void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255) {
uint32_t duty = (8191 / valueMax) * min(value, valueMax);
ledcWrite(channel, duty);
}
void setup() {
ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
```

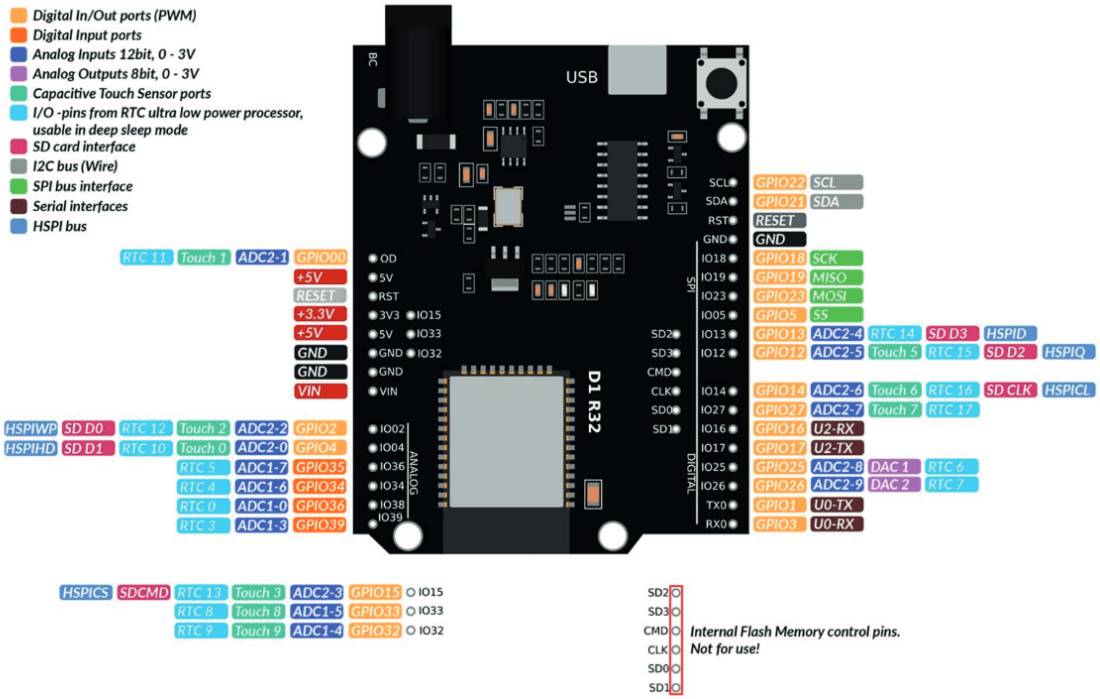
[4] https://community.platformio.org/uploads/default/original/3X/2/a/2a8c98b91d520915677620572c95251b9e3a766e.jpeg

```
ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}
void loop() {
ledcAnalogWrite(LEDC_CHANNEL_0, brightness);
brightness = brightness + fadeAmount;
if (brightness <= 0 || brightness >= 255) {
fadeAmount = -fadeAmount;
}
delay(30);
}
```

That's not all, if you want to learn how to use the FreeRTOS in this platform, refer to the external links and FreeRTOS' documentation for more info:

- What is FreeRTOS? https://techtutorialsx.com/2017/05/06/esp32-arduino-using-freertos-functions/
- Creating a task in FreeRTOS: https://techtutorialsx.com/2017/05/06/esp32-arduino-creating-a-task/
- FreeRTOS manual: https://docs.aws.amazon.com/freertos-kernel/latest/ref/welcome.html

D1 R32 Board Pinout



-

Arduino boards & libraries:
ESP32 based board support
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json