PageESP32-WROOM-32S 32d

Board

Full name

IdealSpark® ESP32 Development Board Integrated 2.9 cm ST7789 135x240 TFT LCD Display, WiFi+BL Wireless Module, CH340 Driver USB Type-C for Arduino Micropython

Datasheet

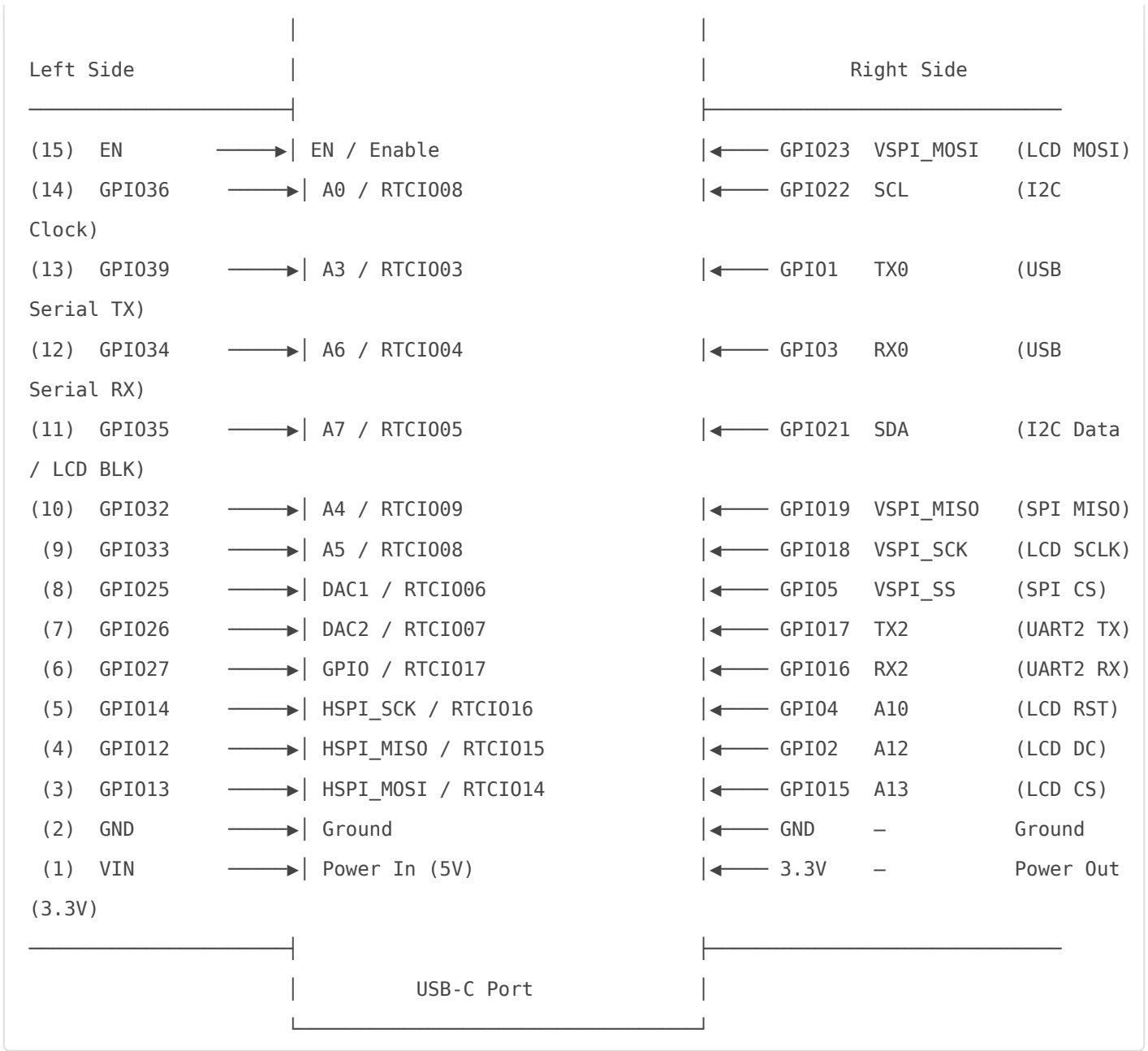
Board description

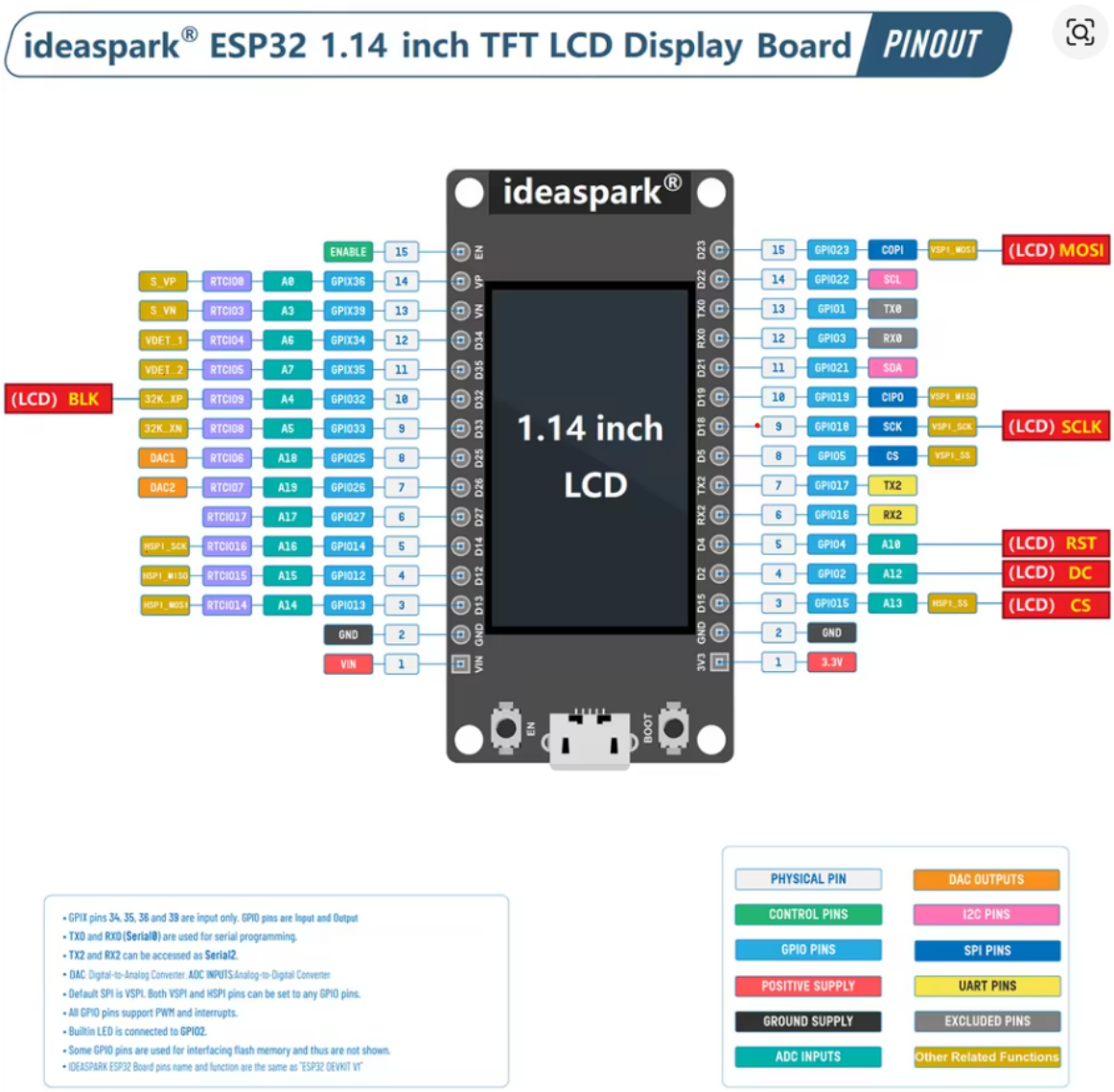
The Dual Core 30pin ESP32-WROOM-32S 32d is compatible with the Arduino IDE. To get started, you'll need to install the ESP32 board definitions through the Arduino Board Manager. Here’s how:

1. **Install the Board Package:**
   In the Arduino IDE, open *File* > *Preferences* and add the following URL to the “Additional Boards Manager URLs” field:
   `https://dl.espressif.com/dl/package_esp32_index.json`
2. **Use the Board Manager:**
   Go to *Tools* > *Board* > *Boards Manager*, search for “ESP32,” and install the package provided by Espressif.
3. **Select Your Board:**
   Once installed, select the appropriate ESP32 board from *Tools* > *Board*. Your Dual Core ESP32-WROOM-32S should appear in the list, allowing you to program it using the familiar Arduino environment.

With these steps, you can leverage the Arduino IDE to develop and upload code to your ESP32 board.

Modulenaam in Arduino *IDE: ESP32 Dev Module*


Left Side
Right Side
(15) EN → EN / Enable
(14) GPIO36 → A0 / RTCIO08
(13) GPIO39 → A3 / RTCIO03
(12) GPIO34 → A6 / RTCIO04
(11) GPIO35 → A7 / RTCIO05
(10) GPIO32 → A4 / RTCIO09
(9) GPIO33 → A5 / RTCIO08
(8) GPIO25 → DAC1 / RTCIO06
(7) GPIO26 → DAC2 / RTCIO07
(6) GPIO27 → GPIO / RTCIO17
(5) GPIO14 → HSPI_SCK / RTCIO16
(4) GPIO12 → HSPI_MISO / RTCIO15
(3) GPIO13 → HSPI_MOSI / RTCIO14
(2) GND → Ground
(1) VIN → Power In (5V)
GPIO23 VSPI_MOSI (LCD MOSI)
GPIO22 SCL (I2C Clock)
GPIO1 TX0 (USB Serial TX)
GPIO3 RX0 (USB Serial RX)
GPIO21 SDA (I2C Data / LCD BLK)
GPIO19 VSPI_MISO (SPI MISO)
GPIO18 VSPI_SCK (LCD SCLK)
GPIO5 VSPI_SS (SPI CS)
GPIO17 TX2 (UART2 TX)
GPIO16 RX2 (UART2 RX)
GPIO4 A10 (LCD RST)
GPIO2 A12 (LCD DC)
GPIO15 A13 (LCD CS)
GND — Ground
3.3V — Power Out (3.3V)
USB-C Port

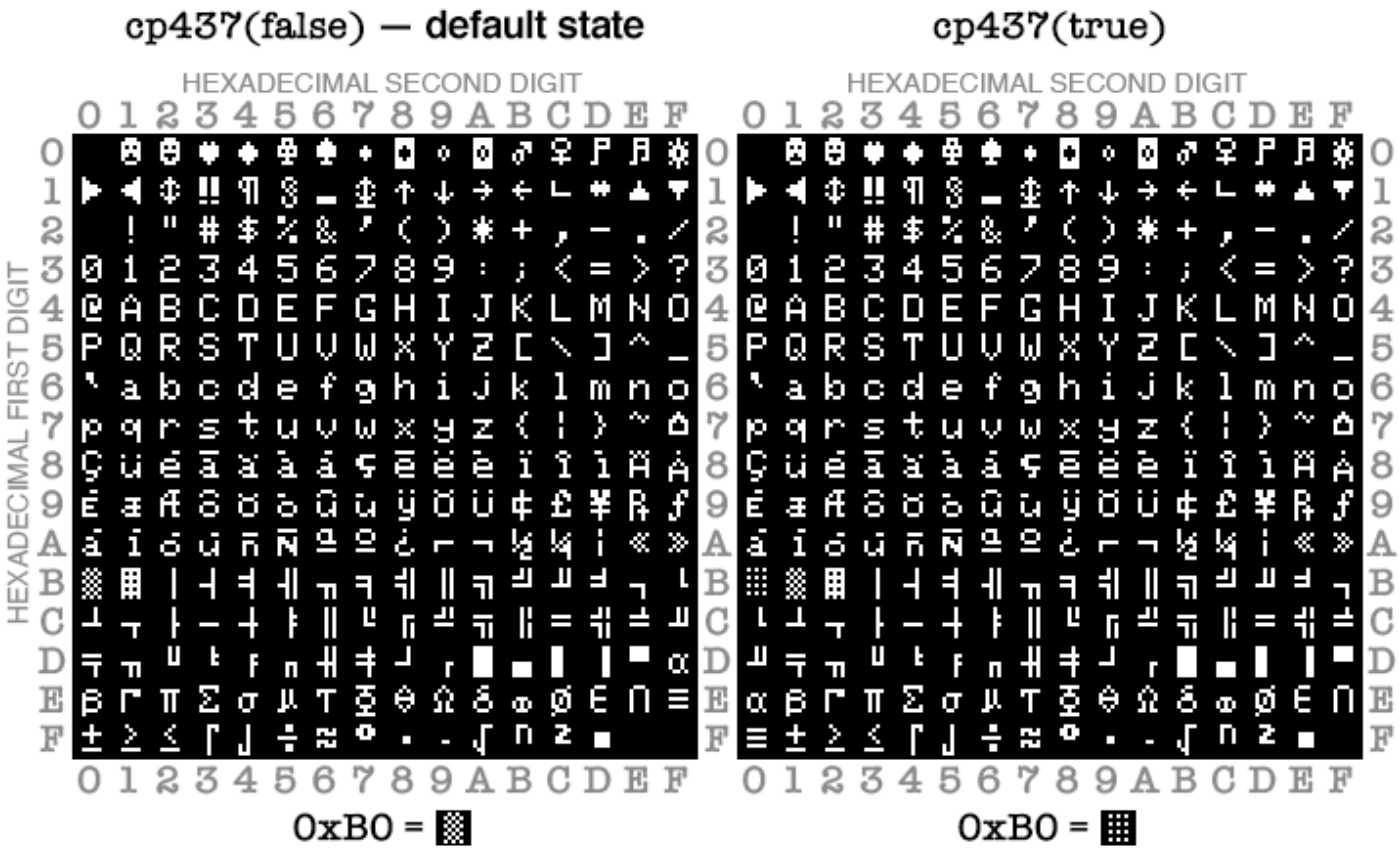


Graphics

Graphic LIbrary

https://learn.adafruit.com/adafruit-gfx-graphics-library?view=all

Char Set



Text size

Text Size (`setTextSize(n)`)	Character Width (px)	Height ?	Characters per Line (240 px / width)
1	6 px (5 + 1 spacing)	8 px	**40 chars**
2	12 px	16 px	**20 chars**
3	18 px	24 px	**13 chars**
4	24 px	32 px	**10 chars**
5	30 px	40 px	**8 chars**
6	36 px	48 px	**6 chars**

The height is Text Size X 8 (?)

Sleep and Wake up

Sleep types

- **Light sleep** → handig bij korte pauzes, reageert snel op beweging.
- **Deep sleep** → goed als er langere tijd geen beweging is (bespaart veel stroom).
- **Hibernation** → enkel nuttig als je op batterij werkt en uren/dagen wilt slapen; je moet dan *alles* opnieuw initialiseren bij wake (display, DHT, Wi-Fi, etc.).

Slaapmodus	CPU actief?	RTC actief?	RAM behouden?	Typisch verbruik
Light sleep	Nee	Ja	Ja	~1–5 mA
Deep sleep	Nee	Ja	Nee (RAM weg)	~0.1 mA
Hibernation	Nee	Nee	Nee	~5–10 µA

Functie	Deep Sleep	Hibernation
Wake via timer		
Wake via GPIO EXT0		
Wake via GPIO EXT1		
ULP / touch wake		
RTC memory bewaren		
Verbruik	~100 µA	~10 µA

(RTC memory: Klein stukje geheugen dat optioneel behouden blijft tijdens deep sleep.)

Hoe het werkt

- De ESP32 blijft tijdens **deep sleep** bijna volledig uitgeschakeld.
- Alleen het **RTC-subsysteem (Real-Time Clock)** blijft actief.
- Dat subsysteem bewaakt een klein aantal speciale **RTC GPIO's** (zoals GPIO 0, 2, 4, 12–15, 25–27, 32–39).
- Via **EXT0** kun je precies één van die pinnen gebruiken om wakker te worden.

EXT0/EXT1

- **EXT0:** slechts **één** pin, **level-triggered** (blijft hoog → blijft wakker).

- **EXT1:** kan **meerdere pins** combineren (OR/AND), **edge-like gedrag** (gaat van low naar high).

```
# EXT0
esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, 1);  // Word wakker als pin 13 HIGH wordt

# EXT1
esp_sleep_enable_ext1_wakeup(1ULL << 12 | 1ULL << 13, ESP_EXT1_WAKEUP_ANY_HIGH);
```

Hibernation

```
# je bereikt hibernation door in deep sleep alle power domains uit te zetten.
esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, 1);  // PIR HIGH wekt op
esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH, ESP_PD_OPTION_OFF); // RTC peripherals uit
esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_SLOW_MEM, ESP_PD_OPTION_OFF);
esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_FAST_MEM, ESP_PD_OPTION_OFF);

esp_deep_sleep_start();  // gaat in hibernation omdat alles uit is
```

Deep Sleep wake op PIR (ANY_HIGH)

```
#include <esp_sleep.h>

#define PIR_PIN 13  // RTC-capable pin

void setup() {
  pinMode(PIR_PIN, INPUT);            // of INPUT_PULLDOWN als je die wilt gebruiken
  // Wake bij HIGH op deze pin (EXT1, multiple pins mogelijk)
  esp_sleep_enable_ext1_wakeup(1ULL << PIR_PIN, ESP_EXT1_WAKEUP_ANY_HIGH);

  // ... event. nog iets loggen

  esp_deep_sleep_start();
}

void loop() {}
```

Light Sleep wake op PIR

```
#include <esp_sleep.h>

#define PIR_PIN 13

void setup() {
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT);
  esp_sleep_enable_ext1_wakeup(1ULL << PIR_PIN, ESP_EXT1_WAKEUP_ANY_HIGH);
}

void loop() {
  // Doe je werk...
  delay(50);

  // Ga naar light sleep tot PIR HIGH wordt
  esp_light_sleep_start();

  // Wakker! (PIR hield de lijn even hoog; verwerk event)
  Serial.println("Motion detected (woke from light sleep)");
  // kleine debounce
  delay(200);
}
```

Revision #17
Created 2025-03-16 19:56:52 UTC by Max
Updated 2025-12-03 19:47:28 UTC by Max