Welcome!

Thank you for purchasing our *AZ-Delivery D1 R32*. On the following pages, we will introduce you to how to use and set-up this handy device.

Have fun!

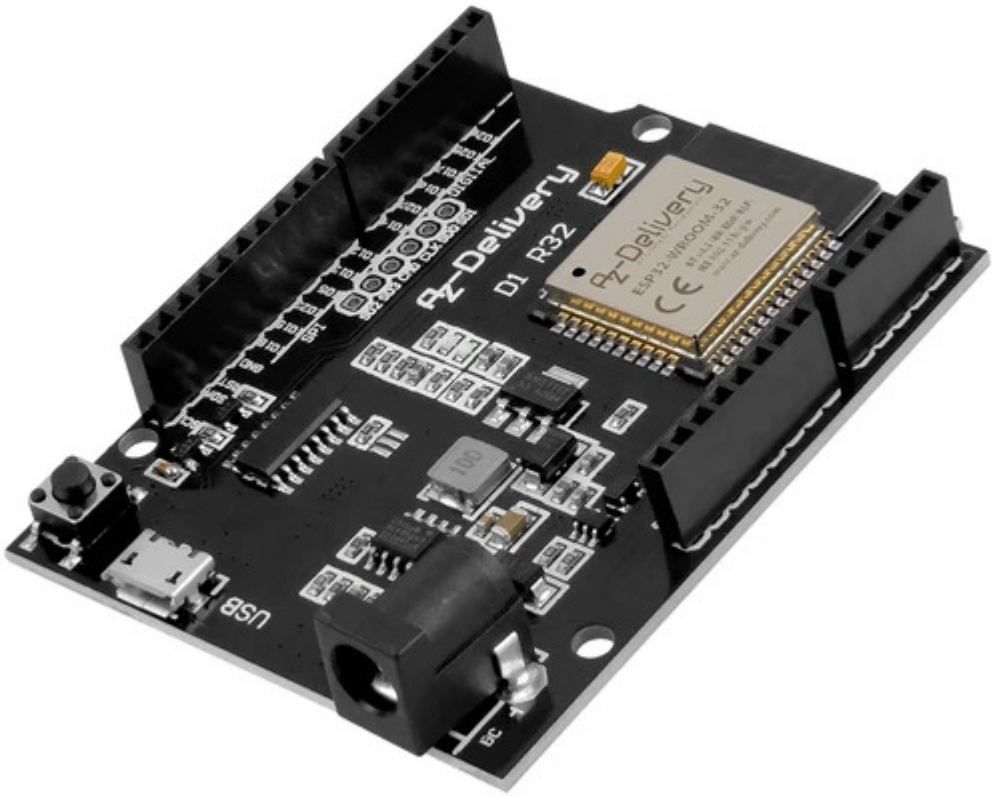


Table of Contents

Introduction

The D1 R32 is an (Arduino) Uno alike development board created around ESP32 WROOM-32 chip, containing voltage regulator, USB programmer circuit for ESP32 chip, and a many other features.

For appplication development there is a choice between Arduino IDE or ESP-IDF (Native platform). Most users choose the Arduino IDE because of its simplicity and compatibility. The Arduino user community is very active and supports platforms such as ESP32.

The D1 R32 comes with a pre-installed firmware which allows to work with the interpreted language, sending commands through the serial port (CH340 chip). The ESP32 boards are one of the most used platforms for Internet of Things (IoT) projects.

The D1 R32 board is designed to work with different peripherals and has compatiblity with some shields designed particularly for this board. It has a voltage regulator that allows it to feed directly from the USB port or the socket for a DC jack. The input/output pins work at 3.3V. The CH340 chip is responsible for USB to serial communication.

Specifications

Power supply voltage (microUSB)	5VDC
DC input voltage	7-12V
Input/Output voltage	3.3V
Operating current required	min. 250mA
SoC	ESP32 WROOM-32
Clock frequency range	240MHz
RAM	512kB
External flash memory	4MB
Digital pins	20
Analog pins	6
Communication interfaces	SPI, I2C, I2S, IR, UART, PWM
Wi-Fi protocols	802.11 b/g/n/i (802.11n up to 150Mbps)
Wi-Fi frequency	2.4 GHz - 2.5 GHz
Wireless antenna	PCB
USB to Serial chip	CH340
Dimensions	70x55x13mm(2.7x2.1x0.5in)

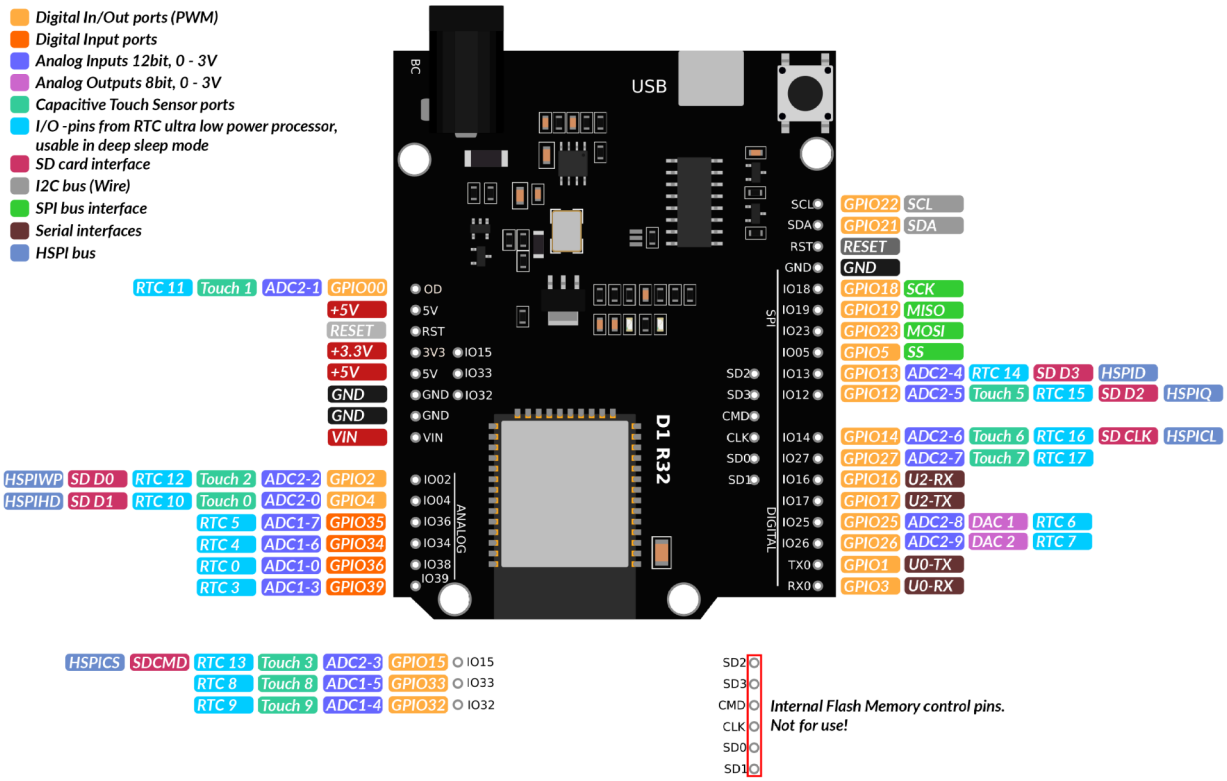
D1 R32

The ESP32 series of Wi-Fi chips is produced by Espressif Systems. ESP32 WROOM-32 is affordable Wi-Fi module suited for DIY projects in the Internet of Things (IoT) field. This module comes with many GPIOs and support for a variety of protocols like SPI, I2C, I2S, UART, and more. Remarkable is that it comes with wireless networking included, which makes it different to other micro controllers like the Arduino. This means that it can easily control and monitor devices remotely via Wi-Fi and Bluetooth® at an affordable price.

ESP32 WROOM-32 is a system-on-chip (SoC) integrating a 32-bit Tensilica microcontroller, standard digital peripheral interfaces, antenna switches, RF balun, power amplifier, low noise receiver amplifier, filters and power management modules into a small package. It provides 2.4GHz Wi-Fi (802.11 b/g/n, supporting speed up to 150MB/s), BLE and classic Bluetooth® wireless communication, Overall pins provide connectivity to GPIO pins supporting PWM (Pulse Width Modulation), GPI pins (input only), Capacitive Touch Sensors, I2C and I2S interfaces, ADC (analog to digital conversion), DAC (digital to analog conversion), SPI interface or UART on dedicated pins.

Pinout

The D1 R32 has 22 pins. The pinout is shown on the following image:



For detailed pinout description and I/O capabilities please refer to the datasheet which can be found on the following link.

Pins description

Just like a normal Arduino board, the D1 R32 has digital input/output pins (GPIO pins - General Purpose Input/Output pins). These digital input/outputs operate at 3.3V.

5V voltage must not be connected to any ESP32 chip pins!

The pins are not 5V tolerant, applying more than 3.3V on any pin will destroy the chip.

The GPIO pins 34 to 39 are GPIs – input only pins. These pins do not have internal pull-ups or pull-down resistors. They cannot be used as outputs, so use these pins only as inputs: GPIO34, GPIO35, GPIO36, GPIO39

There is an integrated SPI flash on the ESP-WROOM-32 chip. The pins GPIO6 to GPIO11 are exposed in certain ESP32 development boards. These pins: GPIO6(SCK/CLK), GPIO7(SDO/SD0),GPIO8(SDI/SD1), GPIO9(SHD/SD2), GPIO10(SWP/SD3), GPIO11(CSC/CMD) are connected to the integrated SPI flash on the chip and are not recommended for other uses.

Capacitive Touch sensor pins

The ESP32 has internal capacitive touch sensors. The capacitive touch pins can also be used to wake up the ESP32 from deep sleep.

The pins on D1 R32 with capacitive touch capability are:
GPIO4(Touch 0), GPIO00(Touch 1), GPIO2(Touch 2), GPIO12(Touch 5), GPIO14(Touch 6), GPIO27(Touch 7).

Analog to Digial converter pins

The ESP32 has 16x12 bits ADC(Analog to Digital converter) input channels (while the ESP8266 only has 1x10bits ADC). These are the GPIOs that can be used as ADC on the D1 R32:
GPIO36(ADC1-0), GPIO39(ADC1-3), GPIO32(ADC1-4), GPIO33(ADC1-5), GPIO35(ADC1-7), GPIO34(ADC1-6), GPIO04(ADC2-0), GPIO00(ADC2-1), GPIO2(ADC2-2), GPIO15(ADC2-3), GPIO13(ADC2-4), GPIO12(ADC2-5), GPIO14(ADC2-6), GPIO27(ADC2-7), GPIO25(ADC2-8), GPIO26(ADC2-9)

Digital to Analog converter pins

There are 2x8 bits DAC(Digital to Analog converter) channels on the ESP32 to convert digital signals into analog voltage signal outputs. These are the DAC channels:
DAC1(GPIO25), DAC2(GPIO26).

Real Time Clock GPIO pins

There is RTC(Real time clock) GPIO support on the ESP32. The GPIOs routed to the RTC low-power subsystem can be used when the ESP32 is in deep sleep. These RTC GPIOs can be used to wake up the ESP32 from deep sleep when the Ultra Low Power (ULP) co-processor is running. The following GPIOs can be used as an external wake up source:
GPIO36(RTC-0), GPIO39(RTC-3), GPIO34(RTC-4), GPIO35(RTC-5), GPIO25(RTC-6), GPIO26(RTC-7), GPIO33(RTC-8), GPIO32(RTC-9), GPIO04(RTC-10), GPIO00(RTC-11), GPIO2(RTC-12), GPIO15(RTC-13), GPIO13(RTC-14), GPIO12(RTC-15), GPIO14(RTC-16), GPIO27(RTC-17).

PWM (Pulse Width Modulation) pins

The ESP32 PWM (Pulse width modulation) controller has 16 independent channels that can be configured to generate PWM signals with different properties. All pins that can act as outputs can be used as PWM pins (GPIOs 34 to 39 cannot generate PWM). To set a PWM signal, parameters in the code has to be defined: Signal's frequency, Duty cycle, PWM channel, GPIO where to output the signal.

The I2C interface pins

The ESP32 has two I2C channels and any pin can be set as SDA or SCL. When using the ESP32 with the Arduino IDE, the default I2C pins are: GPIO21(SDA), GPIO22(SCL).

SPI inteface pins

By default, the pin mapping for SPI pins is:

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO23	GPIO19	GPIO18	GPIO5
HSPI	GPIO13	GPIO12	GPIO14	GPIO15

Strapping Pins

Following pins are used to put the ESP32 into bootloader or flashing mode: GPIO0, GPIO2, GPIO4, GPIO5 (must be HIGH during boot), GPIO12 (must be LOW during boot), GPIO15 (must be HIGH during boot).

Most development boards put pins in the right state for flashing or boot mode. If peripherals are connected to the strapping pins and the IDE is unable to upload the code or flash the ESP32, it may be because those peripherals are preventing the ESP32 to enter the right mode. After resetting, flashing, or booting, those pins work as expected. There is Boot Mode Selection documentation guide on the following link. Further and more extensive explanations are not in the scope of this eBook so please, refer to the datasheet.

Pins HIGH at Boot

Some GPIOs change their state to HIGH or output PWM signals at boot or reset. This means that if outputs are connected to these GPIOs this may get unexpected results when the ESP32 resets or boots.

GPIO1, GPIO3, GPIO5, GPIO6 to GPIO11 (connected to the ESP32 integrated SPI flash memory), GPIO14, GPIO15.

USB to Serial communication

The D1 R32 has a microUSB connection port. It is made around CH340 chip which allows USB to UART serial communication. The chip has the virtual COM port (VCP) feature that appears as COM port in PC applications. The CH340 UART interface implements all RS-232 signals, including control and handshaking signals, so existing system firmware does not need to be modified. To be able to use the D1 R32 the respective driver has to be installed on the computer/operating system.

WiFi Communication

The D1 R32 has integrated Wi-Fi communication interface and can operate in three different modes: Wi-Fi station, Wi-Fi access point, and both at the same time. It supports the following features:

- 802.11b and 802.11g data-rates
- 802.11n MCS0-7 in both 20MHz and 40MHz bandwidth
- 802.11n MCS32
- 802.11n 0.4µS guard-interval
- Data-rate up to 150Mbps
- Receiving STBC 2x1
- Up to 21 dBm transmitting power
- Adjustable transmitting power
- Antenna diversity and selection (software-managed hardware)

Bluetooth Communication

The D1 R32 has an integrated Bluetooth Radio and supports following features:

• Class-1, class-2 and class-3 transmit output powers and over 30 dB dynamic control range
• π/4 DQPSK and 8 DPSK modulation
• High performance in NZIF receiver sensitivity with over 98 dB dynamic range
• Class-1 operation without external PA
• Internal SRAM allows full speed data transfer, mixed voice and data, and full piconet operation
• Logic for forward error correction, header error control, access code correlation, CRC, demodulation, encryption bit stream generation, whitening and transmit pulse shaping
• ACL, SCO, eSCO and AFH
• A-law, μ-law and CVSD digital audio CODEC in PCM interface
• SBC audio CODEC
• Power management for low power applications
• SMP with 128-bit AES

Also, the Bluetooth Radio has support for the following communication interface protocols:

• UART HCI interface, up to 4Mbps
• SDIO/SPI HCI interface
• I2C interface
• PCM/I2S audio interface.

Other features

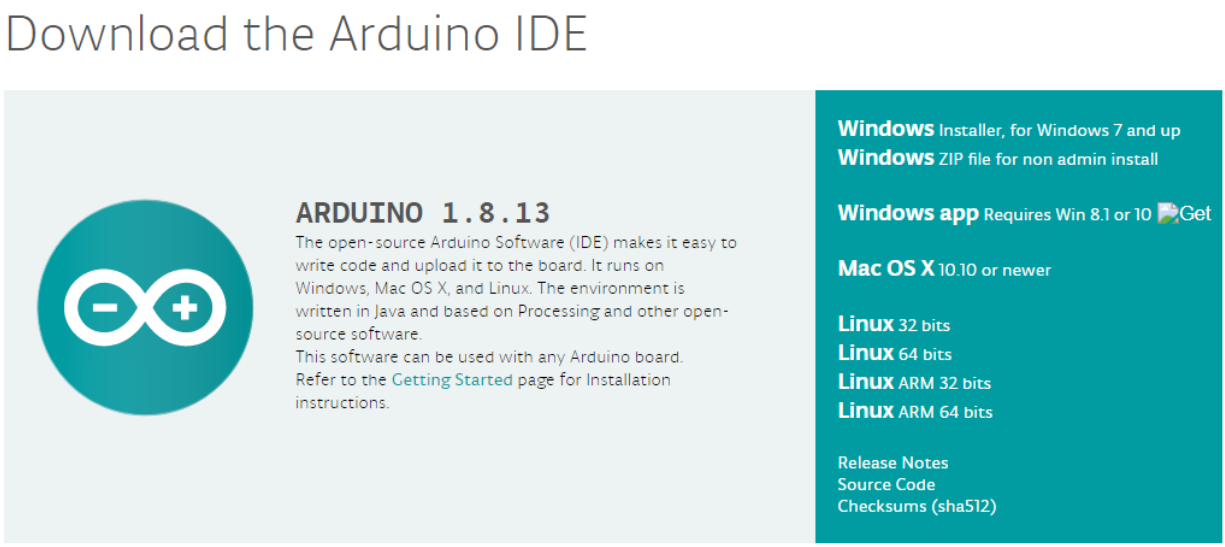
The ESP32-WROOM 32 chip has an integrated Hall Effect Sensor that detects changes in the magnetic field in its surroundings.

The Hall sensor is based on an N-carrier resistor. When the chip is in the magnetic field, the Hall sensor develops a small voltage on the resistor, which can be directly measured by the analog-digital converter (ADC), or amplified by the ultra low noise analog pre-amplifier and then measured by the ADC.

The temperature sensor generates a voltage that varies with temperature. The voltage is internally converted via an analog-to-digital converter into a digital code. The temperature sensor has a range of -40°C to 125°C. As the offset of the temperature sensor varies from chip to chip due to process variation, together with the heat generated by the Wi-Fi circuitry itself (which affects measurements), the internal temperature sensor is only suitable for applications that detect temperature changes instead of absolute temperatures and for calibration purposes as well. However, if the user calibrates the temperature sensor and uses the device in a minimally powered-on application, the results could be accurate enough.

How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice. The Arduino IDE version used for this eBook is **1.8.13.**



For *Windows* users, double click on the downloaded *.exe* file and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

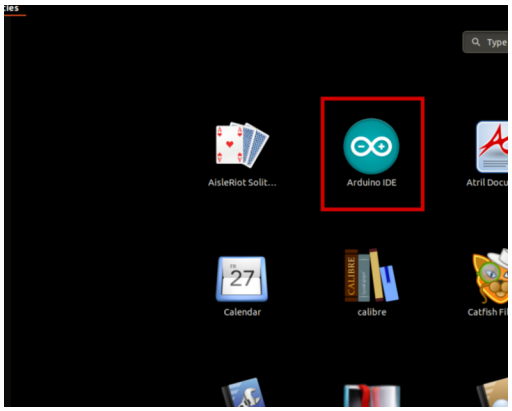
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

sh arduino-linux-setup.sh user_name

user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

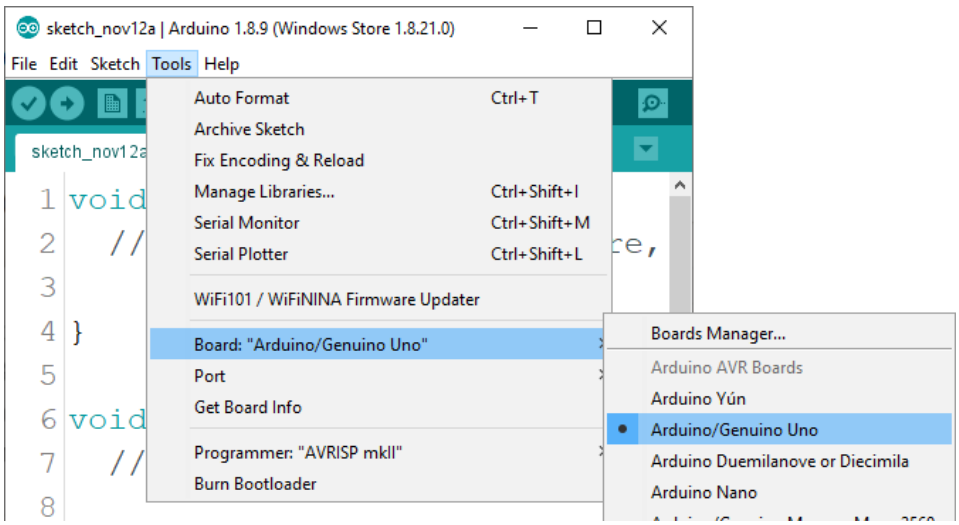
The second script, called *install.sh*, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



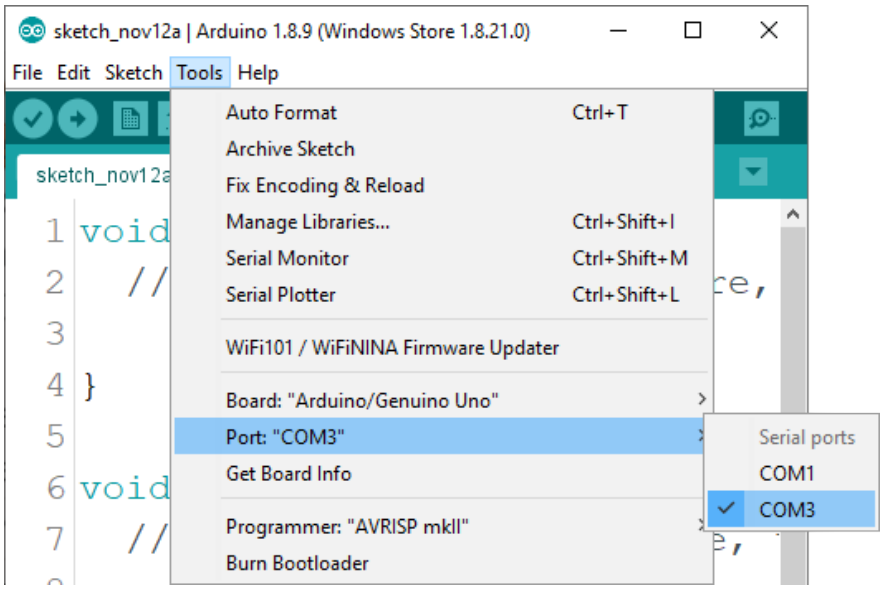
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check, if your PC can detect an Arduino board. Open freshly installed Arduino IDE, and go to:
Tools > Board > {your board name here}
{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:



The port to which the Arduino board is connected has to be selected. Go to:
Tools > Port > {port name goes here}
and when the Arduino board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:



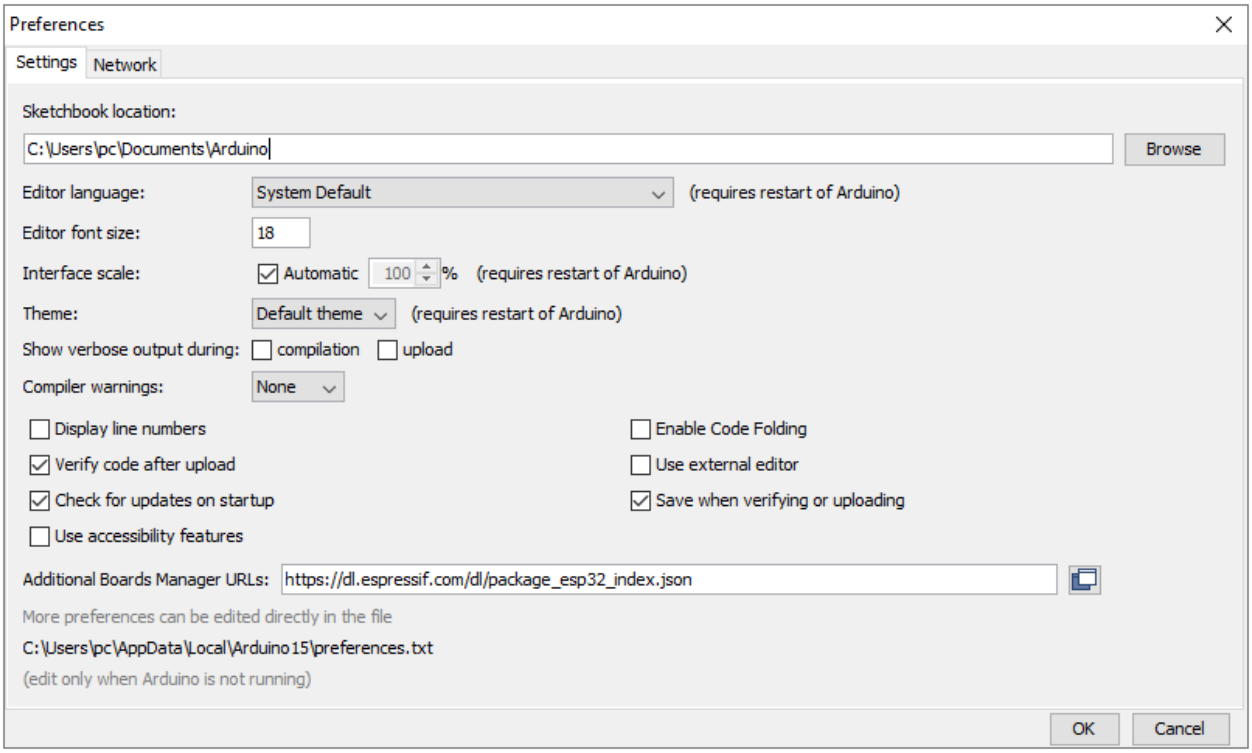
For *Linux* users, for example port name is */dev/ttyUSBx*, where *x* represents integer number between *0* and *9*.

Additional setup

In order to use D1 R32 with Arduino IDE, follow few easy steps. Before setting the Arduino IDE, the driver for the USB to Serial communication has to be installed. If the driver is not installed automatically, there is a support page that contains the drivers for Windows/Mac or Linux and can be chosen depending on which one is used. Drivers can be downloaded from the following link.

Next, to install support for the ESP32 platform, open Arduino IDE and go to: *File* > *Preferences*, and find Additional URLs field.

Then copy the following URL:
https://dl.espressif.com/dl/package_esp32_index.json



Paste this link in the Additional URLs field. If one or more links are inside this field, just add one comma after the last link, paste new link after comma and click the *OK* button.

Close the Arduino IDE and open it again, then go to:

Tools > Board > Boards Manager

When new window opens, type *esp32* in the search box and install the board called *esp32* made by *Espressif Systems*, as shown on the following image:

To select ESP32 board, go to:

Tools > Board > ESP32 Arduino > ESP32 Dev Module

To upload the sketch code to the ESP32 board, first select port to which the board is connected. Go to: *Tools > Port > {port name}*

Sketch examples

Blinking LED

```
int ledPin = 2;

void setup() {
  pinMode(ledPin, OUTPUT);
}

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

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);
  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);
}
```

Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the Internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us