

n-Blocks

n-PRO-20

Table of Contents

Overview	1
MCU Features	2
n-PRO-20 Features	3
Board Pinout	3
Getting started	7
Blink with Eclipse IDE	8
Blink with Arduino IDE	11
Flash Memory Programming	13
Programming with sudo-bed	13
References	13
Related articles in this Wiki	13

n-PRO-20

n-PRO-20 is an ultra low-power, high performance and secure development board from the n-Blocks family. It is available in the [n-Blocks PRO form factor](#), designed for Internet-of-Things gateway applications.



n-PRO-20



ESP32-WROOM in n-PRO modular form factor

License	GPL 2.0
Status	Tested
Buy at:	
Categories	
Hardware repo	Bitbucket
Firmware repo	

Overview

n-PRO-20 is a development board based on Espressif ESP32-WROOM. The WiFi, Bluetooth Classic and BLE make it a great choice to build anything connected. The Wi-Fi allows a large physical range and direct connection to the Internet through a Wi-Fi router. While using Bluetooth, the user can conveniently connect to the phone or broadcast low energy beacons for its detection.

The built-in hardware accelerator enables secure code storage and securely connecting to the Internet with TLS (SSL). Data rate of up to 150 Mbps are supported, and 20 dBm output power at the antenna ensures the widest physical range. The sleep current is less than 5 μ A, which makes it suitable for battery powered and wearable electronics applications.

MCU Features

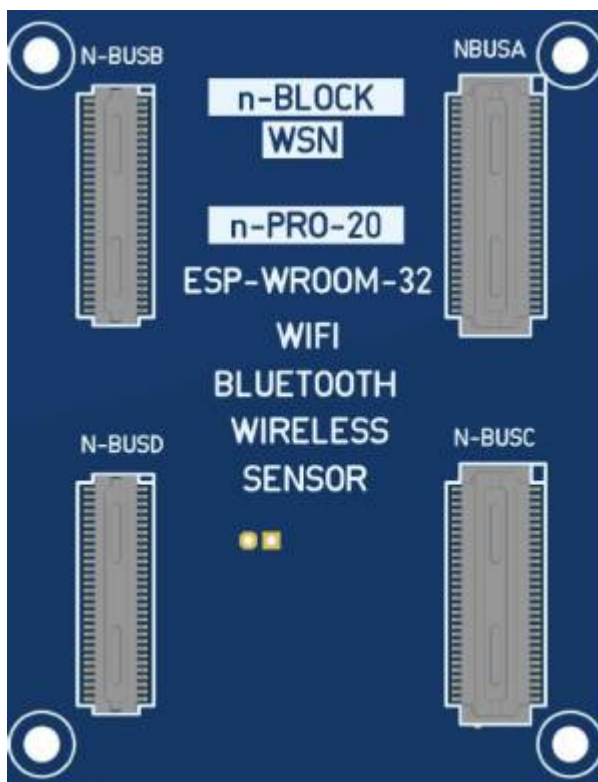
- Hybrid Wi-Fi & Bluetooth
 - Ultra-low power management
 - 240 MHz dual core Tensilica LX6 microcontroller with 600DMIPS
 - 4 MB Flash
 - 2.2V to 3.6V operating voltage
 - On-board PCB antenna
 - Integrated 520 KB SRAM
 - Integrated 802.11b/g/n HT40 Wi-Fi transceiver, base-band, stack and LWIP
 - Integrated dual mode Bluetooth (classic and BLE)
 - 12-bit SAR ADC up to 18 channels, 2 × 8-bit DACs
 - 10 × touch sensors (capacitive sensing GPIOs)
 - 4 × SPI, 3 × UART
 - 2 × I²S interfaces, 2 × I²C interfaces
 - SD/SDIO/CE-ATA/MMC/eMMC host controller, SDIO/SPI slave controller
 - Ethernet MAC interface with dedicated DMA and IEEE 1588 Precision Time Protocol support
-
- CAN bus 2.0
 - Infrared remote controller (TX/RX, up to 8 channels)
 - Motor PWM and LED PWM (up to 16 channels)
 - Hall effect sensor
 - Ultra low power analog pre-amplifier
 - IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
 - Secure boot
 - Flash encryption
 - 1024-bit OTP, up to 768-bit for customers
 - Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)
 - Internal low-dropout regulator
 - Individual power domain for RTC
 - 5µA deep sleep current
 - Wake up from GPIO interrupt, timer, ADC measurements, capacitive touch sensor interrupt

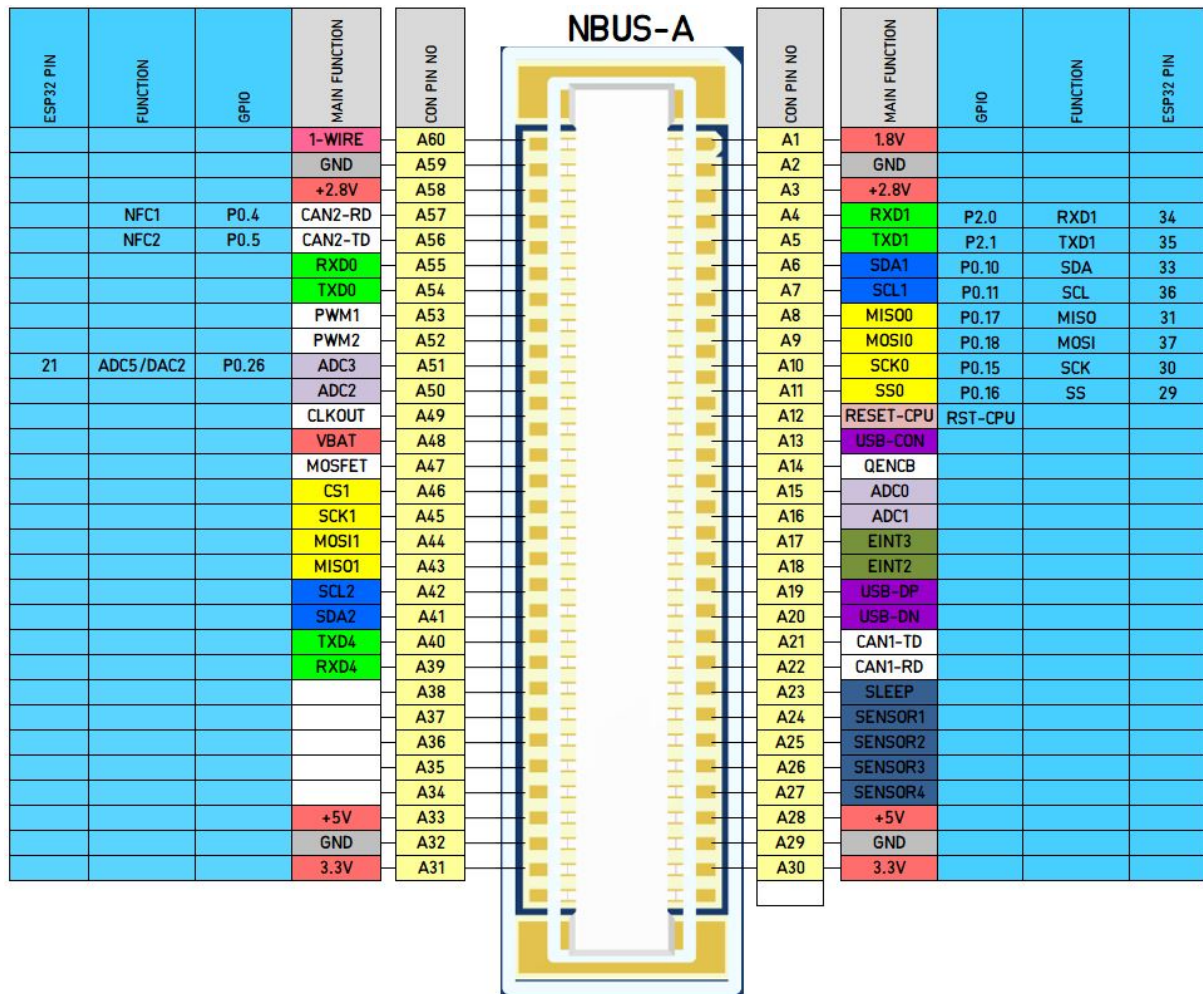
n-PRO-20 Features

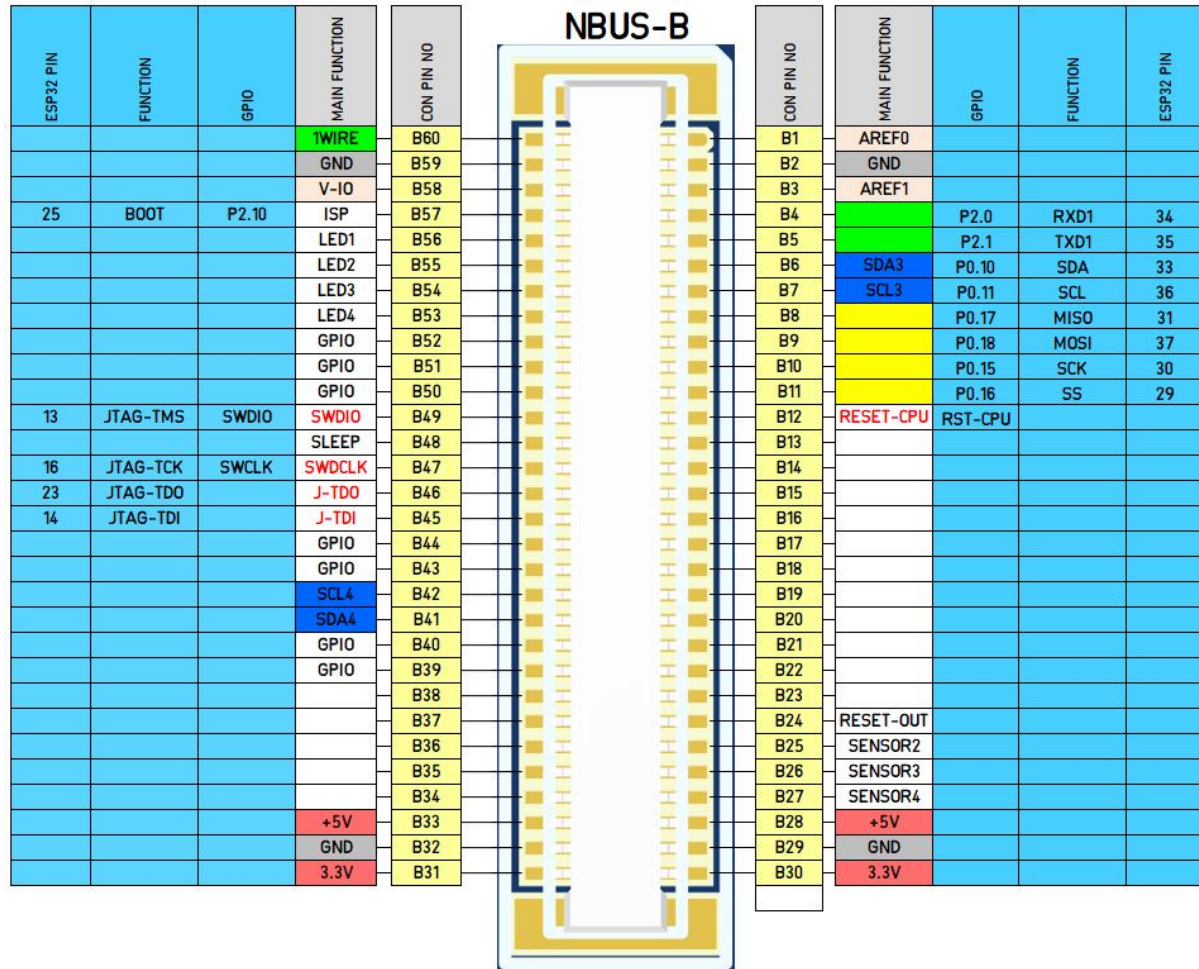
- Simple and Low Power
- Standard [n-Blocks pinout](#)
- Miniature Li-Ion battery connector (optional)
- Custom firmware development via SDK
- Download and write firmware via host
- User configuration via cloud server and Android/iOs App

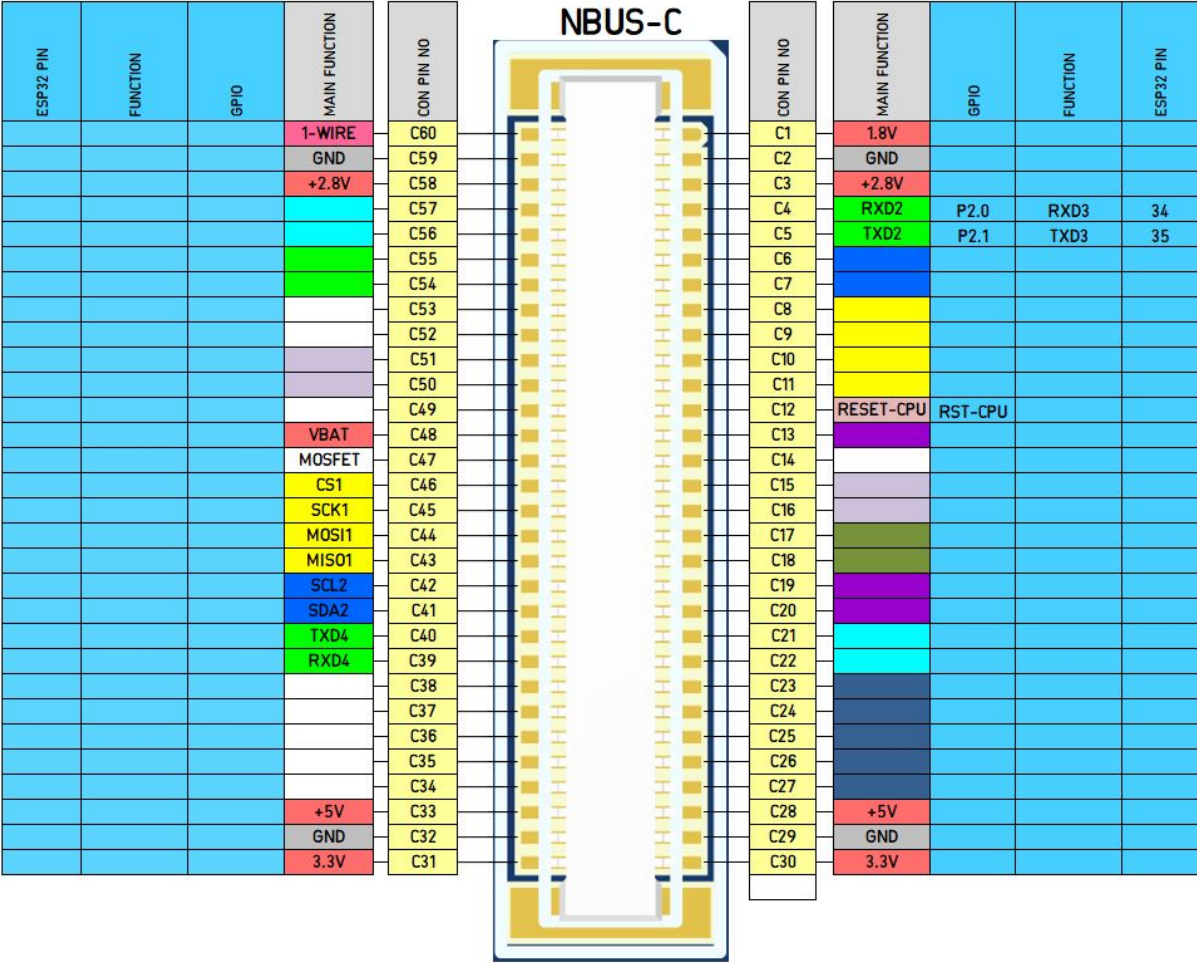
Board Pinout

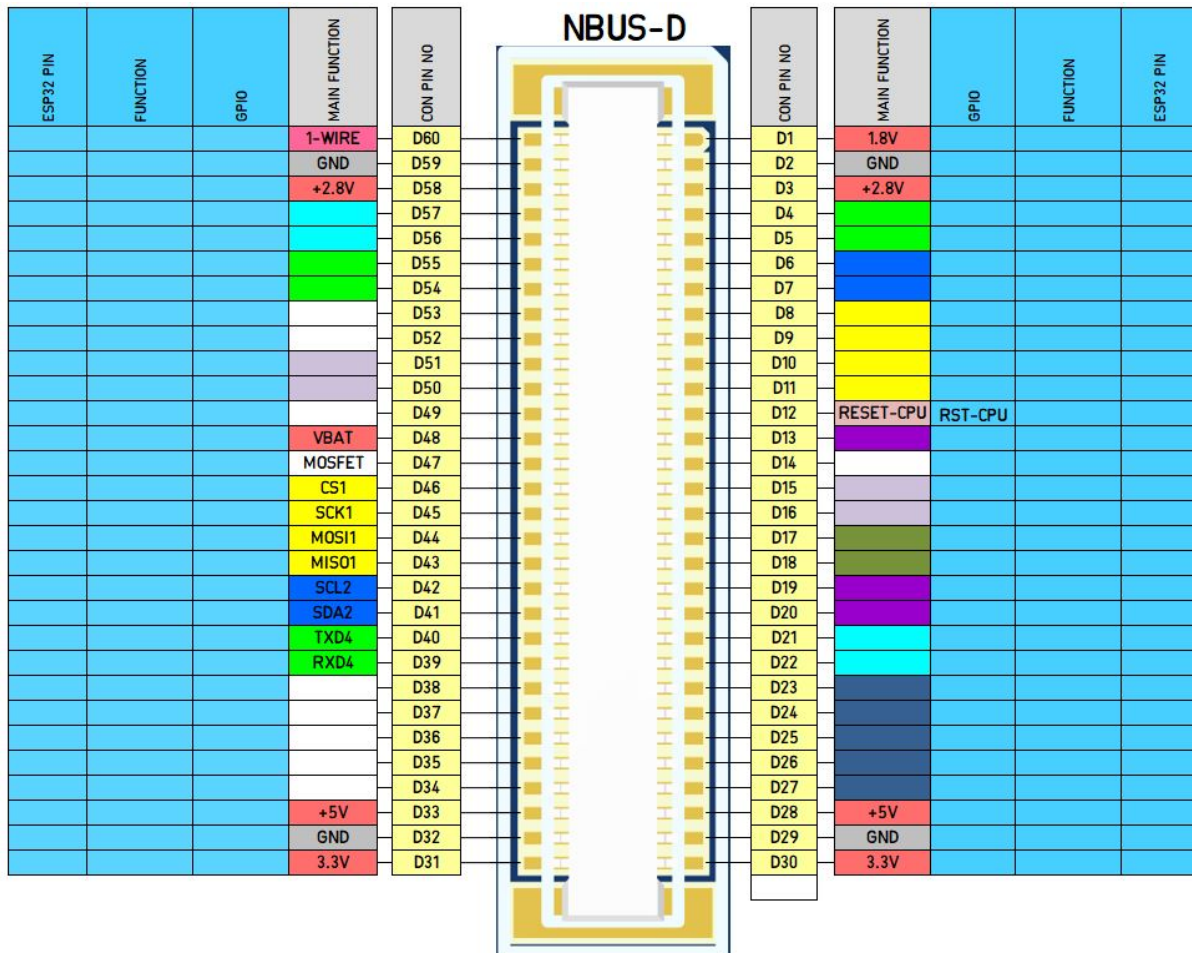
n-PRO-20 is a **HOST** board with four Hirose DF30-series 60-pin low profile connectors at bottom side, following the [n-Blocks PRO form factor](#).











Getting started

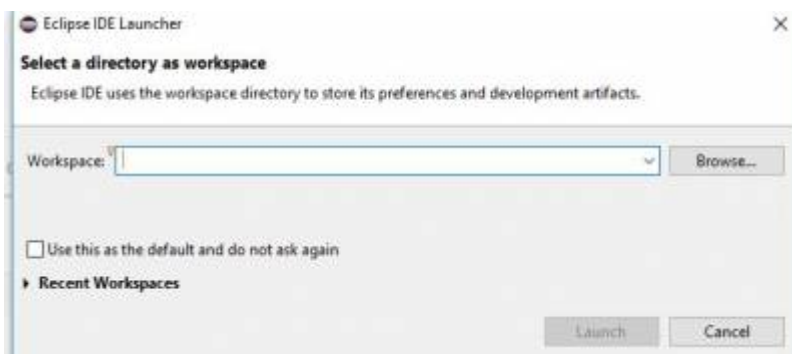
To get started with the board you need:

- PC loaded with either Windows, Linux or Mac operating system
- Toolchain to build the Application for ESP32
- ESP-IDF that essentially contains API for ESP32 and scripts to operate the Toolchain
- A text editor to write programs (Projects) in C, e.g. Eclipse
- The n-PRO-20 board itself and a USB cable to connect it to the PC
- Preparation of development environment consists of following steps:
 - [Setup of Toolchain](#)
 - [Getting of ESP-IDF from GitHub](#)
 - [Set Environment variables](#)
 - [Install the Required Python Packages](#)
 - [Installation and configuration of Eclipse](#) (You may skip this step, if you prefer to use different editor)

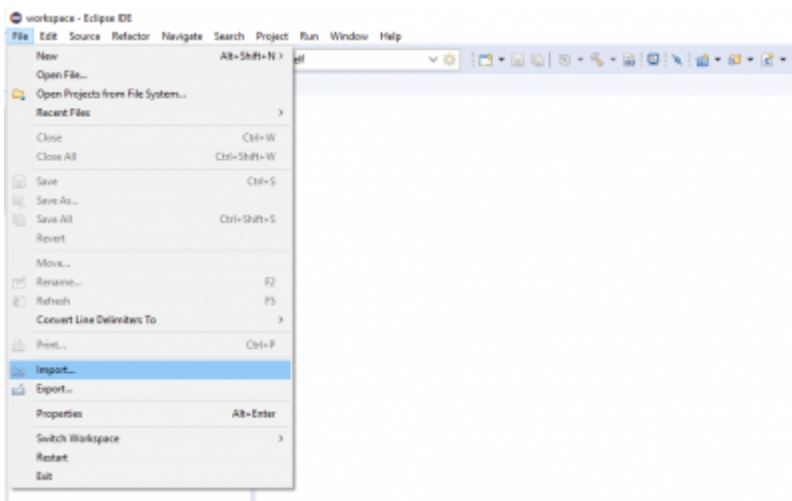
- [Start a Project](#)
- [Connect Your Device](#)
- [Configure](#)
- [Build and Flash](#)
- [Monitor](#)
- The reference APIs can be accessed from [API Reference](#) and [API Guide](#)

Blink with Eclipse IDE

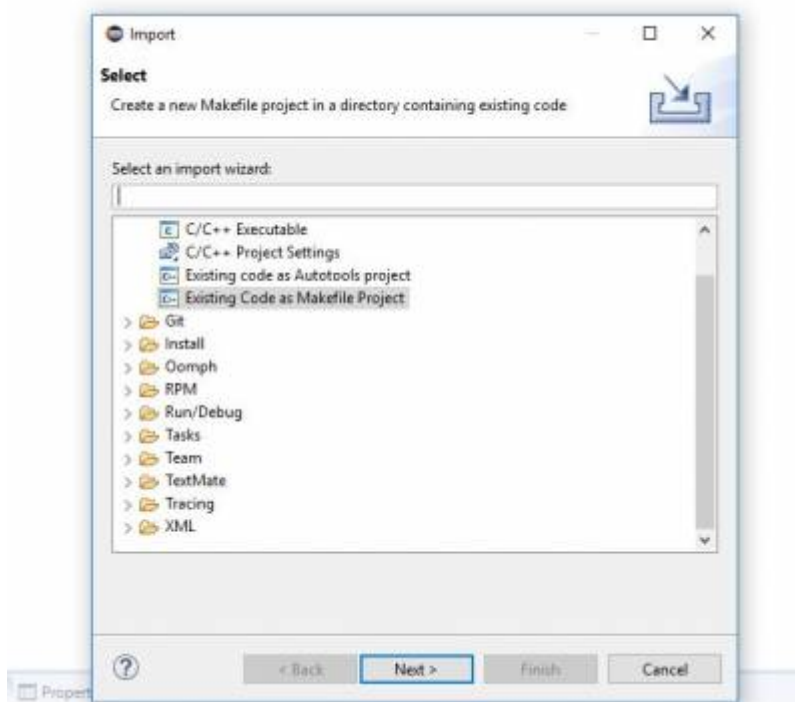
- Open Eclipse, select a workspace directory for e.g. C:\Users\Documents\esp32\workspace and launch.



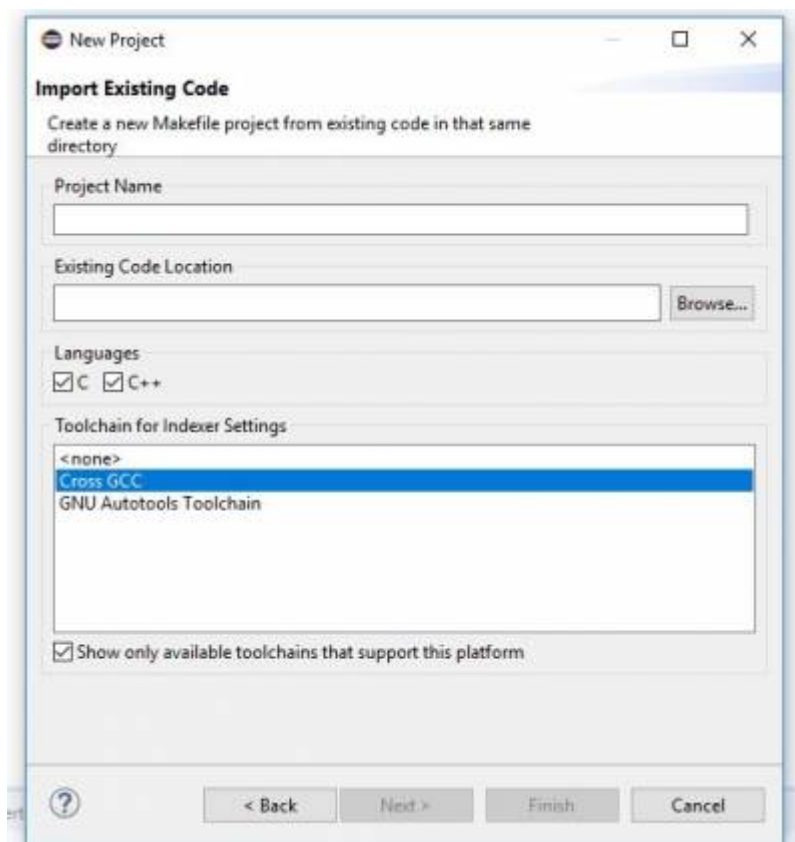
- On the top left corner, click on File. In the drop down menu click on Import.



- Select Existing projects from workspace and click next.



- Locate the examples folder and select blink. Make sure Cross GCC is selected. Click Finish.



- Blink will appear in project explorer.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "sdkconfig.h"

/* Can run 'make menuconfig' to choose the GPIO to blink,
   or you can edit the following line and set a number here.
*/
#define BLINK_GPIO CONFIG_BLINK_GPIO

void blink_task(void *pvParameter)
{
    /* Configure the IOMUX register for pad BLINK_GPIO (some pads
are
    muxed to GPIO on reset already, but some default to other
    functions and need to be switched to GPIO. Consult the
    Technical Reference for a list of pads and their default
    functions.)
    */
    gpio_pad_select_gpio(BLINK_GPIO);
    /* Set the GPIO as a push/pull output */
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);
    while(1) {
        /* Blink off (output low) */

        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        /* Blink on (output high) */
        printf("LED ON");
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        printf("LED OFF");
    }
}

void app_main()
{
    xTaskCreate(&blink_task, "blink_task",
configMINIMAL_STACK_SIZE, NULL, 5, NULL);
}
```

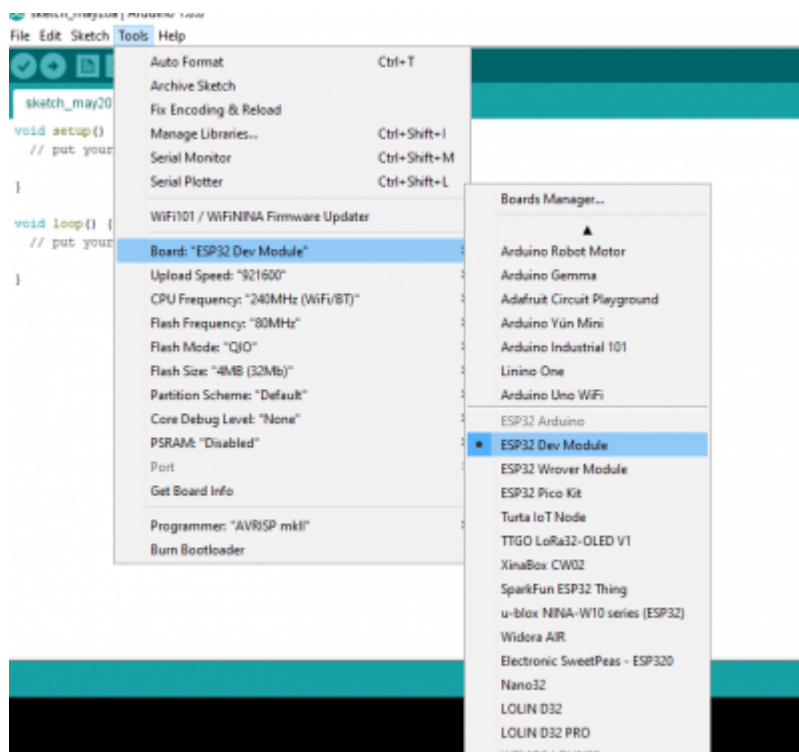
- Build the project.
- Flash the target (Right Click on Blink>Build Targets>Create>Target)

name:flash>OK>Select flash>build).

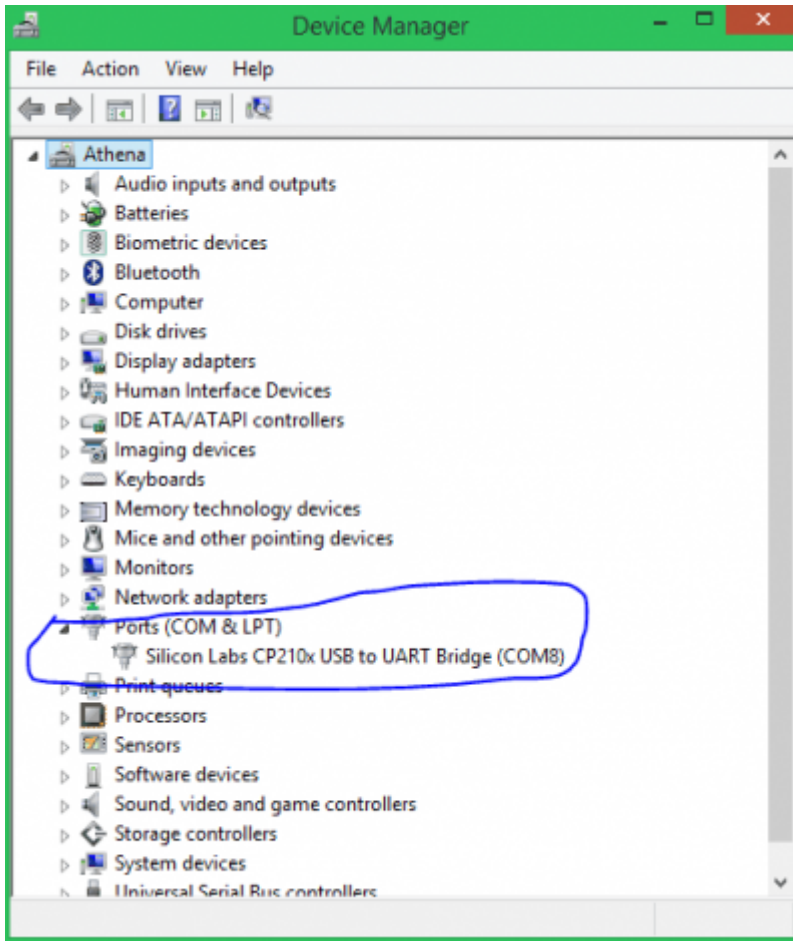
NOTE: You might have to hold the Boot button during build to avoid error.

Blink with Arduino IDE

- Connect your ESP32 board to your computer through the micro-USB cable. Make sure the red LED goes high on the module to ensure power supply.
- Launch the Arduino IDE and navigate to Tools → Boards and select ESP32Dev board as shown below.



- Open device manager and check to which com port your ESP32 is connected to and make sure the same port is selected under Tools>Port in IDE.



- Click on File>New and create a blinky sketch as shown below.

```
const int ledPin = 5;
void setup() {
  // setup pin 5 as a digital output pin
  pinMode (ledPin, OUTPUT);
}
void loop() {
  digitalWrite (ledPin, HIGH); // turn on the LED
  delay(500); // wait for half a second or 500 milliseconds
  digitalWrite (ledPin, LOW); // turn off the LED
  delay(500); // wait for half a second or 500 milliseconds
}
```

- Save the sketch. Verify and Upload.

NOTE: You might have to hold the Boot button during upload to avoid error.

Flash Memory Programming

to be updated

Programming with sudo-bed

References

- [ESP32 Datasheet](#)
- <https://docs.espressif.com/projects/esp-idf/en/latest/get-started/>

Related articles in this Wiki

- [n-PRO-20](#)

[RF](#), [CPU](#), [nblock](#), [BLE](#), [nsensorRF](#)

IMPORTANT NOTICE - PLEASE READ CAREFULLY

Nimbus Centre reserve the right to make changes, corrections, enhancements, modifications, and improvements to Nimbus Centre products and/or to this document at any time without notice.

All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.



Address: Cork Institute of Technology
Campus, Bishopstown, Cork

Phone: (021) 433 5560

© 2019 Nimbus Centre - All rights reserved