n-Blocks

n-PRO-20

Table of Contents

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 µA, 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

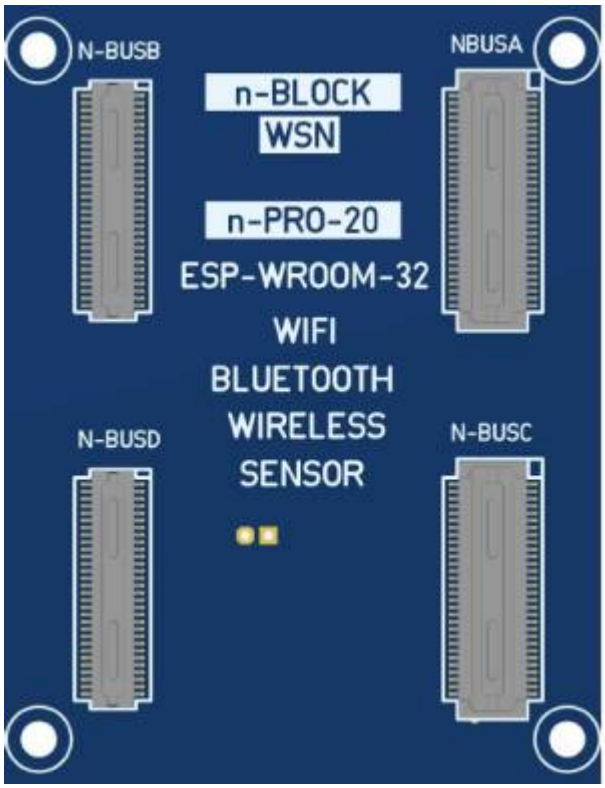
n-PRO-20 Features

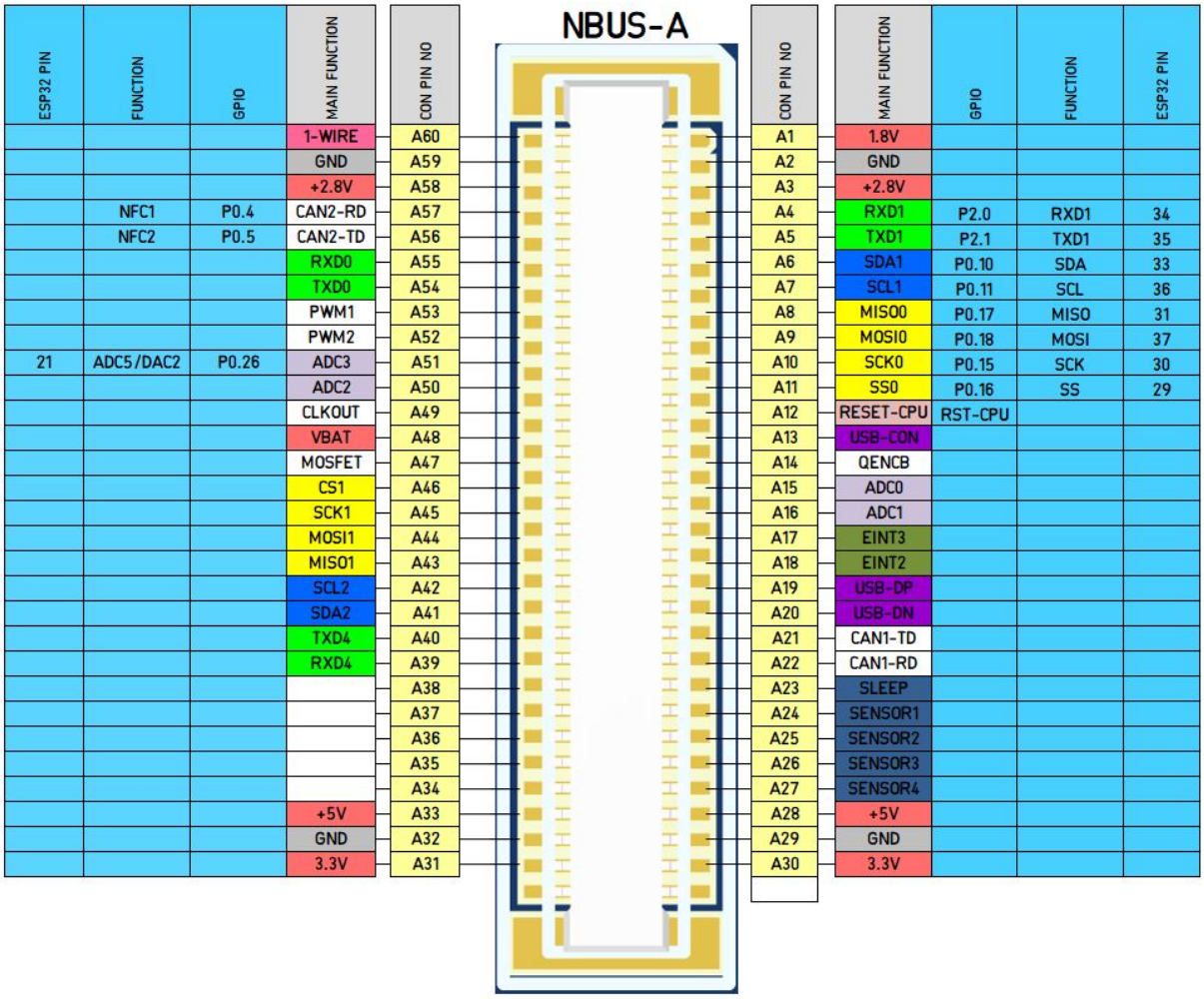
- Simple and Low Power
- Standard n-Blocks pinout
- Miniature Li-Ion battery connector (optional)

- Custom firmware development via SDK
- Download and write firmware via host
- User configuration via cloud server and Android/iOs App

Board Pinout

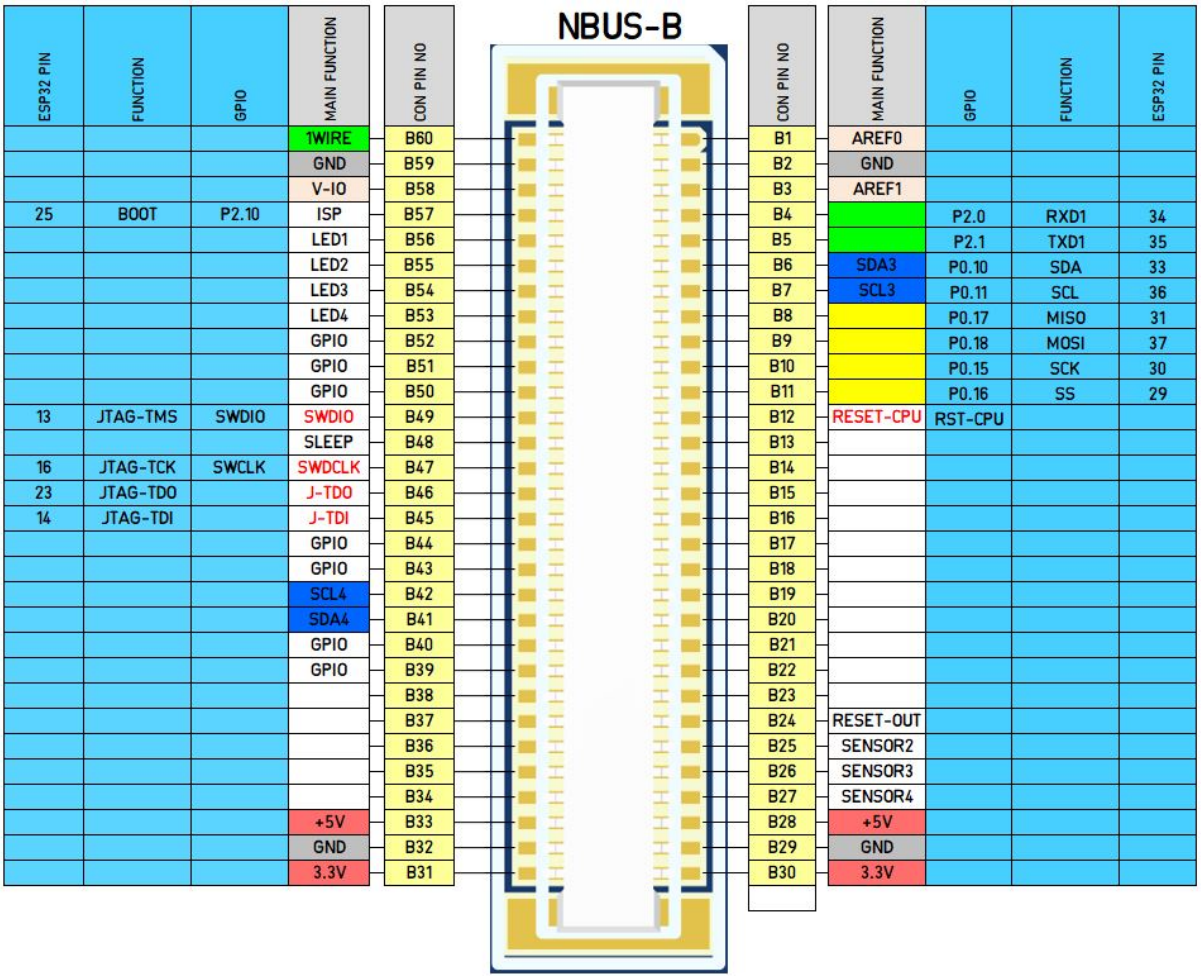
n-PRO-20 is a **HOST** board with four Hirose DF30-series 60-pin low profile connectors at bottom side, following the n-Blocks PRO form factor.



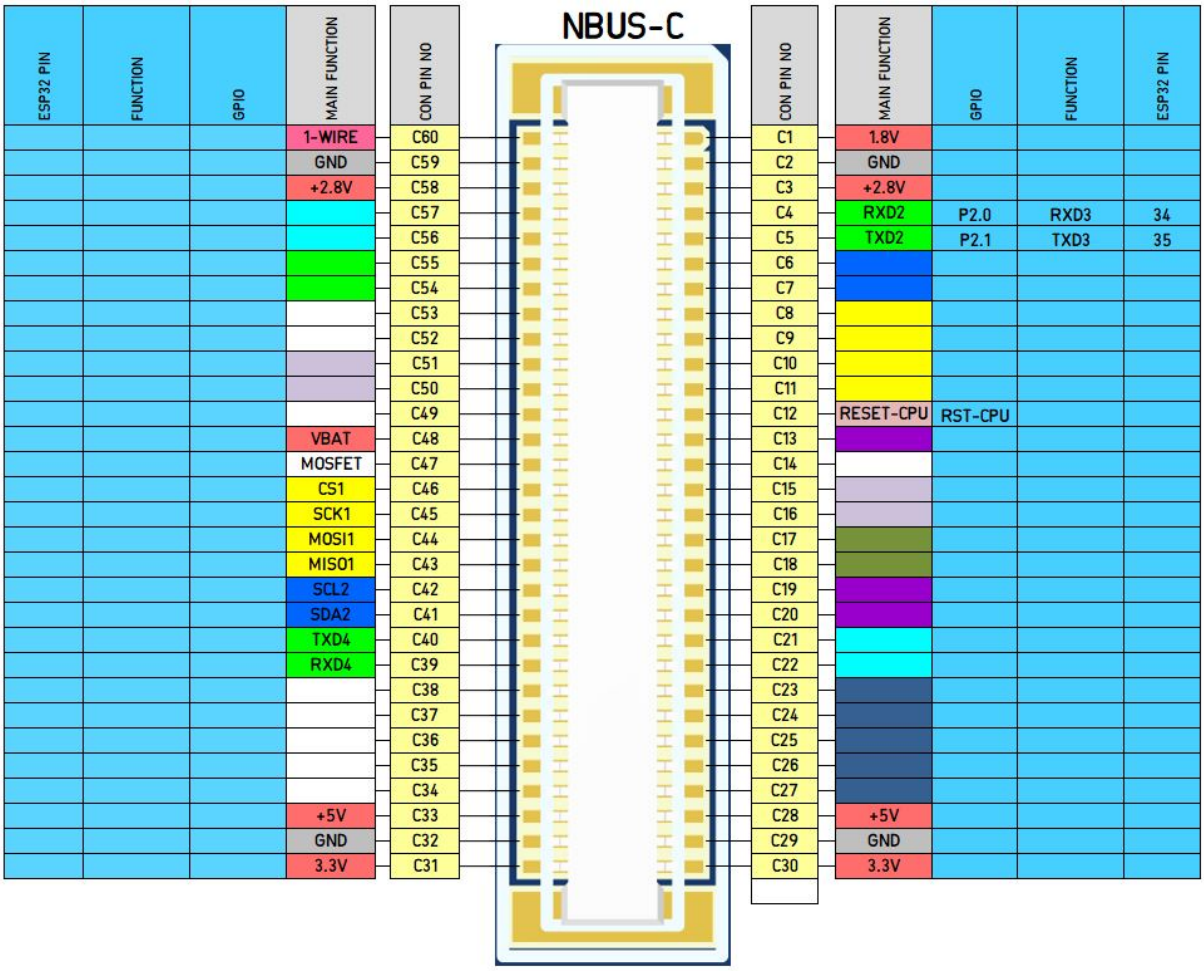


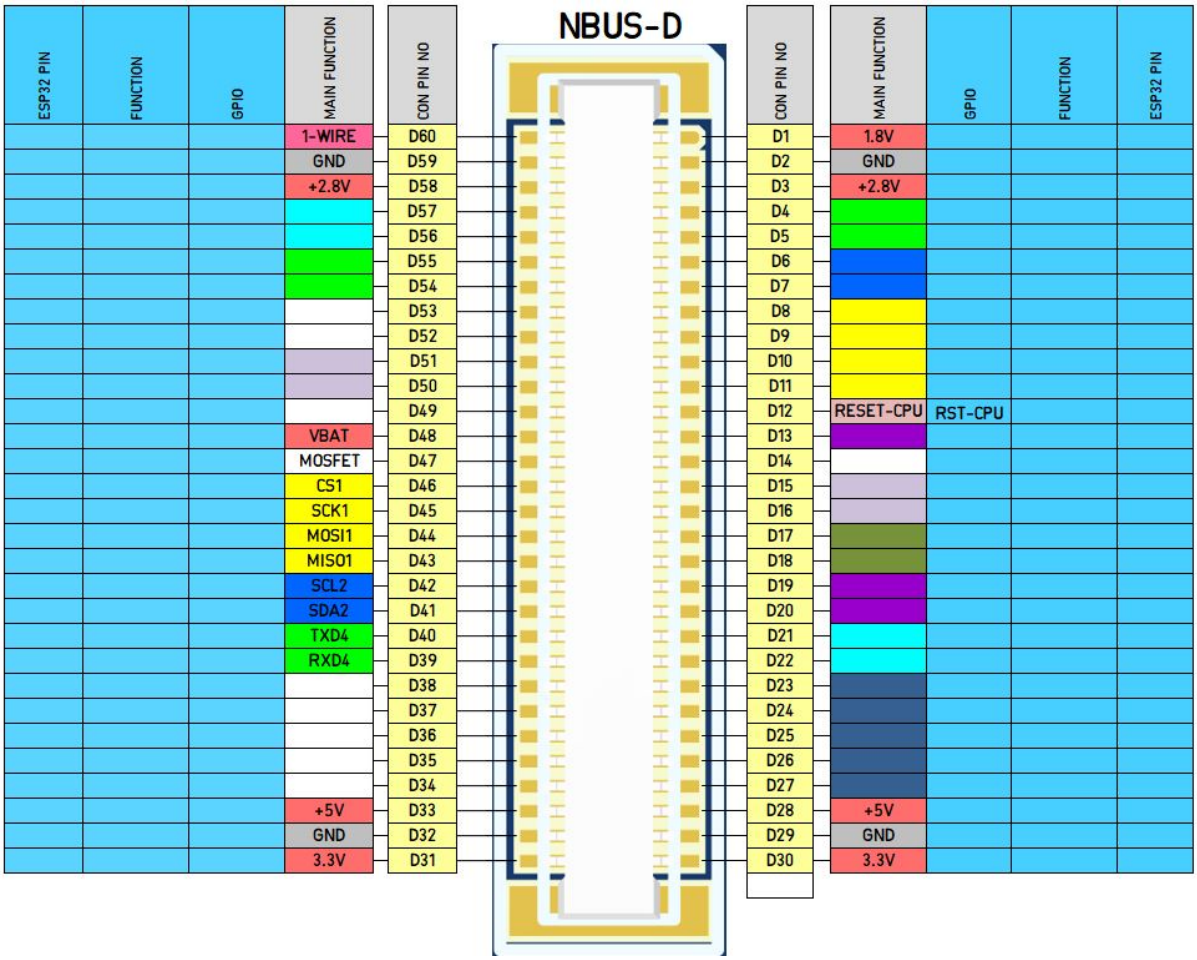
ESP32 PIN	FUNCTION	GPIO	MAIN FUNCTION	CON PIN NO
			1-WIRE	A60
			GND	A59
			+2.8V	A58
	NFC1	P0.4	CAN2-RD	A57
	NFC2	P0.5	CAN2-TD	A56
			RXD0	A55
			TXD0	A54
			PWM1	A53
			PWM2	A52
21	ADC5/DAC2	P0.26	ADC3	A51
			ADC2	A50
			CLKOUT	A49
			VBAT	A48
			MOSFET	A47
			CS1	A46
			SCK1	A45
			MOSI1	A44
			MISO1	A43
			SCL2	A42
			SDA2	A41
			TXD4	A40
			RXD4	A39
				A38
				A37
				A36
				A35
				A34
			+5V	A33
			GND	A32
			3.3V	A31

CON PIN NO	MAIN FUNCTION	GPIO	FUNCTION	ESP32 PIN
A1	1.8V			
A2	GND			
A3	+2.8V			
A4	RXD1	P2.0	RXD1	34
A5	TXD1	P2.1	TXD1	35
A6	SDA1	P0.10	SDA	33
A7	SCL1	P0.11	SCL	36
A8	MISO0	P0.17	MISO	31
A9	MOSI0	P0.18	MOSI	37
A10	SCK0	P0.15	SCK	30
A11	SS0	P0.16	SS	29
A12	RESET-CPU	RST-CPU		
A13	USB-CON			
A14	QENCB			
A15	ADC0			
A16	ADC1			
A17	EINT3			
A18	EINT2			
A19	USB-DP			
A20	USB-DN			
A21	CAN1-TD			
A22	CAN1-RD			
A23	SLEEP			
A24	SENSOR1			
A25	SENSOR2			
A26	SENSOR3			
A27	SENSOR4			
A28	+5V			
A29	GND			
A30	3.3V			



ESP32 PIN	FUNCTION	GPIO	MAIN FUNCTION	CON PIN NO	CON PIN NO	MAIN FUNCTION	GPIO	FUNCTION	ESP32 PIN
			1WIRE	B60	B1	AREF0			
			GND	B59	B2	GND			
			V-IO	B58	B3	AREF1			
25	BOOT	P2.10	ISP	B57	B4		P2.0	RXD1	34
			LED1	B56	B5		P2.1	TXD1	35
			LED2	B55	B6	SDA3	P0.10	SDA	33
			LED3	B54	B7	SCL3	P0.11	SCL	36
			LED4	B53	B8		P0.17	MISO	31
			GPIO	B52	B9		P0.18	MOSI	37
			GPIO	B51	B10		P0.15	SCK	30
			GPIO	B50	B11		P0.16	SS	29
13	JTAG-TMS	SWDIO	SWDIO	B49	B12	RESET-CPU	RST-CPU		
			SLEEP	B48	B13				
16	JTAG-TCK	SWCLK	SWDCLK	B47	B14				
23	JTAG-TDO		J-TDO	B46	B15				
14	JTAG-TDI		J-TDI	B45	B16				
			GPIO	B44	B17				
			GPIO	B43	B18				
			SCL4	B42	B19				
			SDA4	B41	B20				
			GPIO	B40	B21				
			GPIO	B39	B22				
				B38	B23				
				B37	B24	RESET-OUT			
				B36	B25	SENSOR2			
				B35	B26	SENSOR3			
				B34	B27	SENSOR4			
			+5V	B33	B28	+5V			
			GND	B32	B29	GND			
			3.3V	B31	B30	3.3V			


NBUS-C
ESP32 PIN
FUNCTION
GPIO
MAIN FUNCTION
CON PIN NO
1-WIRE
GND
+2.8V
VBAT
MOSFET
CS1
SCK1
MOSI1
MISO1
SCL2
SDA2
TXD4
RXD4
+5V
GND
3.3V
C60 C59 C58 C57 C56 C55 C54 C53 C52 C51 C50 C49 C48 C47 C46 C45 C44 C43 C42 C41 C40 C39 C38 C37 C36 C35 C34 C33 C32 C31
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30
1.8V
GND
+2.8V
RXD2
TXD2
RESET-CPU
+5V
GND
3.3V
GPIO
FUNCTION
ESP32 PIN
P2.0
RXD3
34
P2.1
TXD3
35
RST-CPU



Getting started

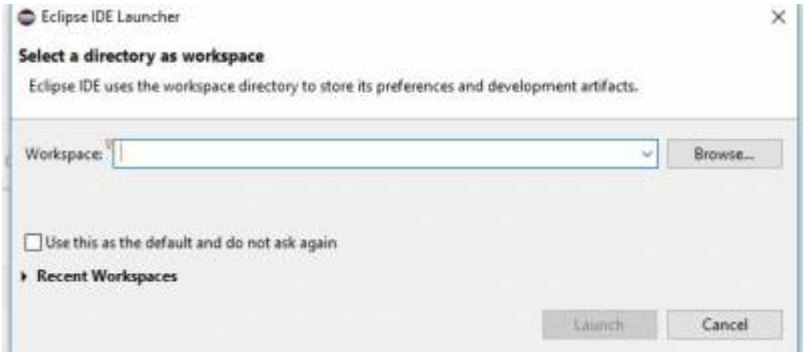
To get started with the board you need:

- 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)

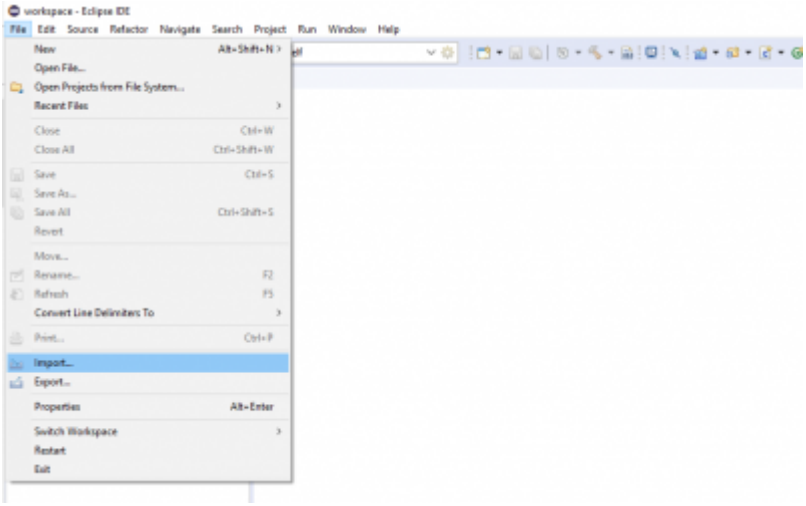
    - Start a Project
    - Connect Your Device
    - Configure
    - Build and Flash
    - Monitor
- The reference APIs can be accessed fromAPI Reference and API Guide

Blink with Eclipse IDE

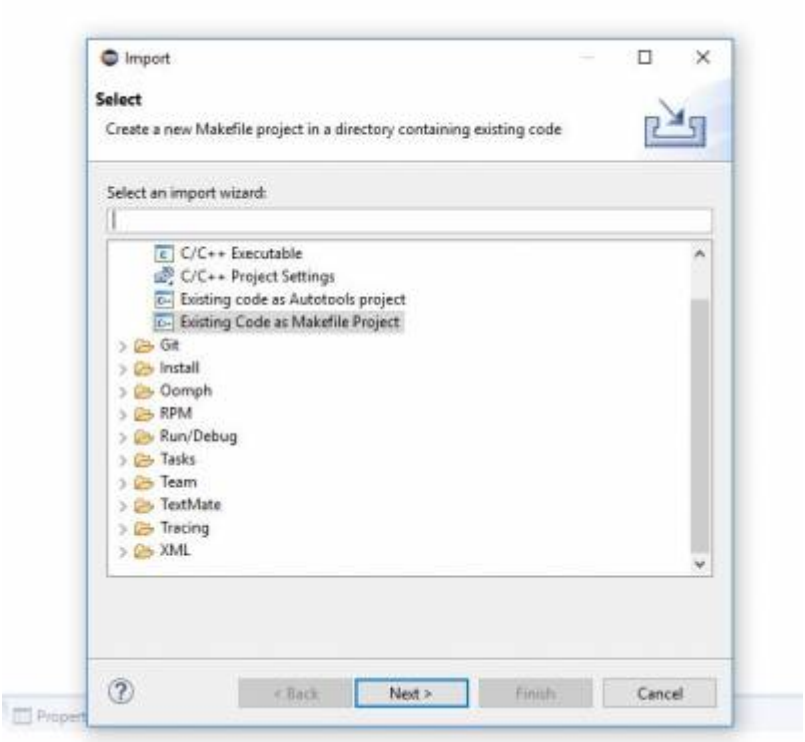
- Open Eclipse, select a workspace directory for e.g. C:\Users\Documents\esp32\workspace and launch.



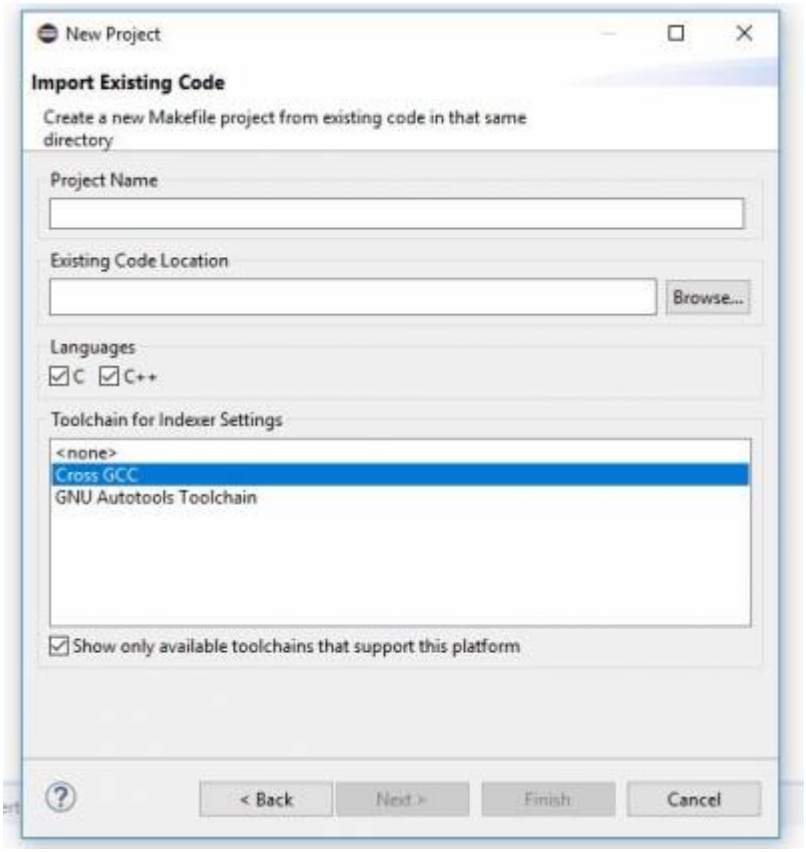
- On the top left corner, click on File. In the drop down menu click on Import.



- 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);
}
```

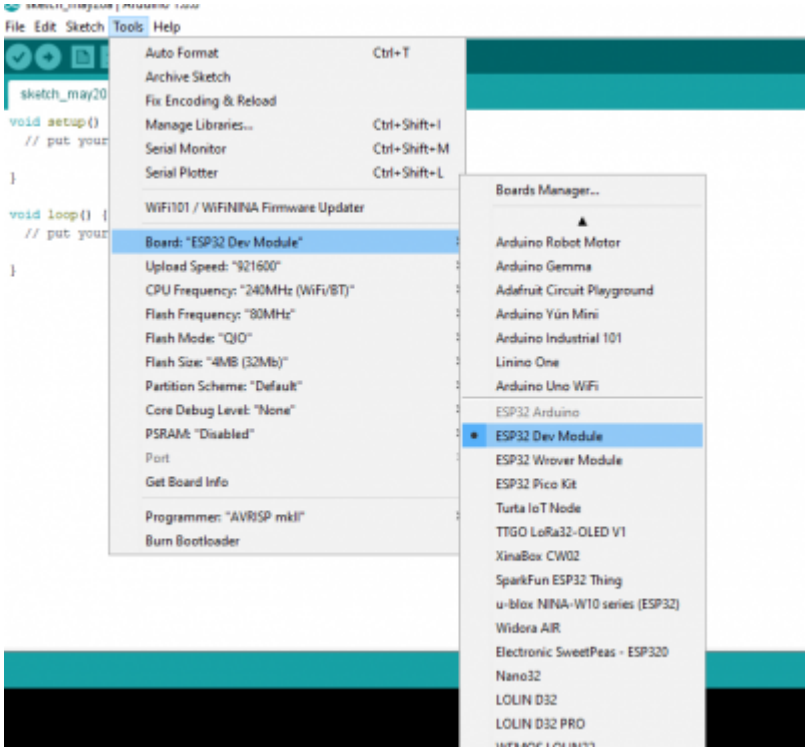
- Build the project.
- Flash the target (Right Click on Blink>Build Targets>Create>Target

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

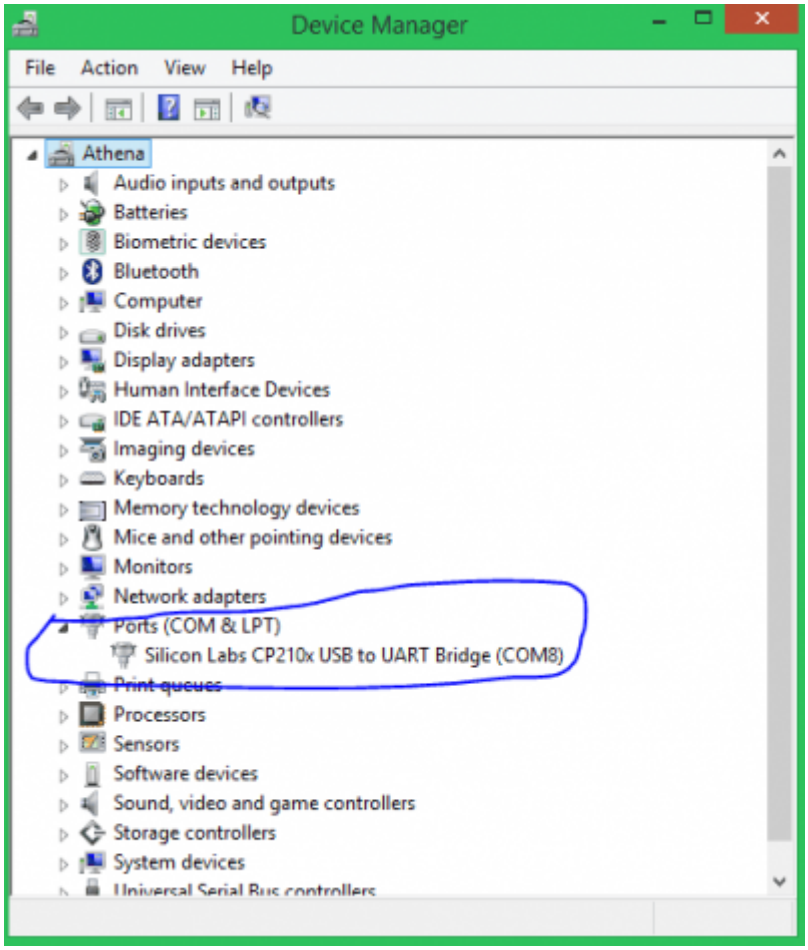
NOTE: You might have to hold the Boot button during build to avoid error.

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

Address: Cork Institute of Technology Campus, Bishopstown, Cork

Phone: (021) 433 5560