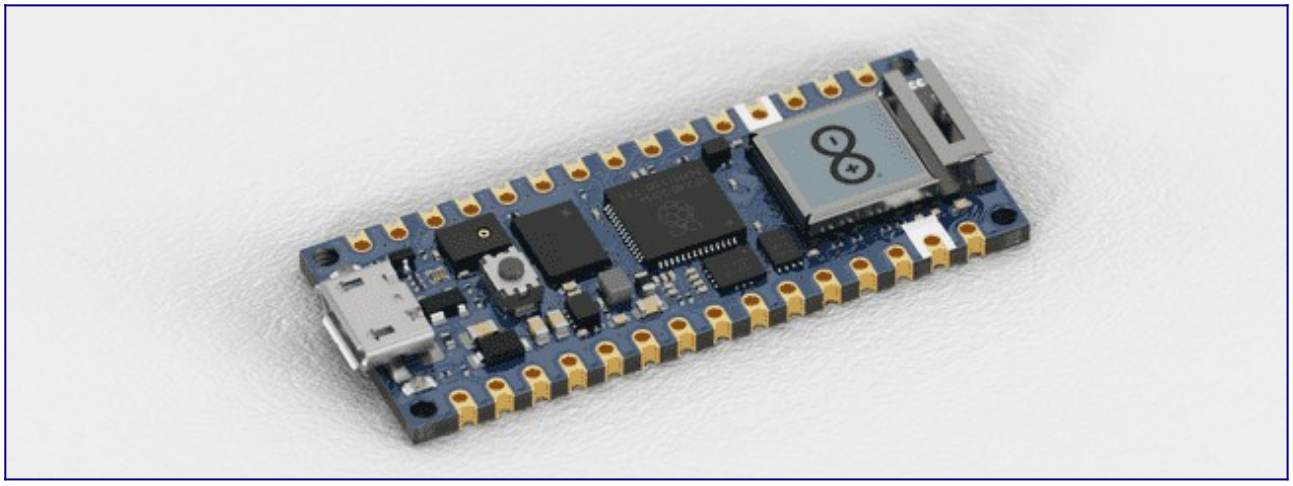
Part 01
-
Hardware

Version: 2021-06-01

The Arduino Nano RP2040 Connect



The Arduino® Nano RP2040 Connect is a development board in Nano format, based on the RP2040 microcontroller. It features a Wi-Fi & Bluetooth® module, a 6-axis IMU (Inertial Measurement Unit) with machine learning capabilities, a microphone and a built-in RGB LED.

Microcontroller	Raspberry Pi RP2040	
USB connector	Micro USB	
Pins	Built-in LED Pin	13
	Digital I/O Pins	20
	Analog input pins	8
	PWM pins	20 (except A6, A7)
	External interrupts	20 (except A6, A7)
Connectivity	Wi-Fi	Nina W102 uBlox module
	Bluetooth	Nina W102 uBlox module
	Secure element	ATECC608A-MAHDA-T Crypto IC
Sensors	IMU	LSM6DSOXTR (6-axis)
	Microphone	MP34DT05
Communication	UART	Yes
	I2C	Yes
	SPI	Yes
Power	Circuit operating voltage	3.3V
	Input voltage (VIN)	5-21V
	DC Current per I/O Pin	4 mA
Clock speed	Processor	133 MHz
Memory	AT25SF128A-MHB-T	16MB Flash IC
Dimensions	Nina W102 uBlox module	448 KB ROM, 520KB SRAM, 16MB Flash
	Weight	6 g
	Width	18 mm
	Length	45 mm

The Nano RP2040 Connect uses the Arduino Mbed OS Nano Boards core.

The Nano RP2040 Connect can be programmed through the Classic Arduino IDE 1.8.X. , Arduino IDE 2.0.X (beta) and the Web Editor.

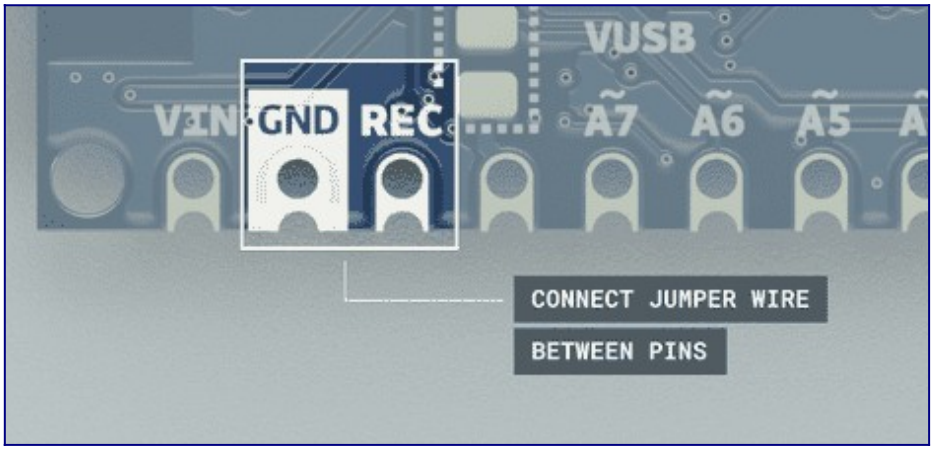
It is compatible with the Arduino IoT Cloud, a cloud service that allows you to create IoT applications in just minutes.

Bootloader
Since our upload procedure relies on the Raspberry's bootloader using a mass storage device, if your computer is fast enough during an upload, it can notify you about an USB removable being plugged.
When this occurs, we can force the ROM bootloader mode, which will enable mass storage, allowing us to upload UF2 images like CircuitPython / MicroPython or a regular Arduino sketch.
When a sketch is uploaded successfully, the mass storage of the Nano RP2040 Connect may be visible in the operating system. The mass storage should only appear for a few seconds, then it will automatically close.

There is a risk that the uploading process gets stuck during an upload. If this happens, we can double-tap the reset button, to forcefully trigger the bootloader.

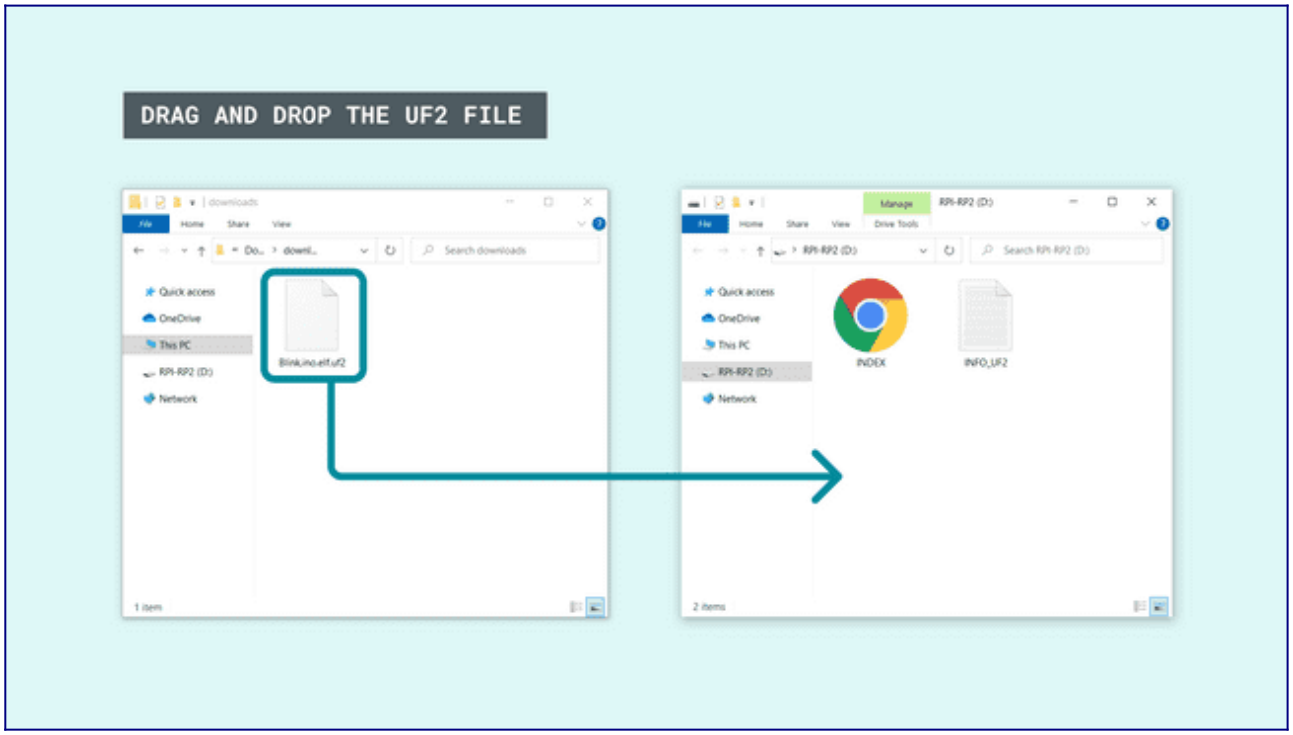
Sometimes the board is not detected even when the board is connected to your computer. This can be solved through the following steps:

1. Disconnect the board from your computer by removing the USB cable.
2. Place a jumper wire between the REC and GND pins on the board, then connect the board to your computer.



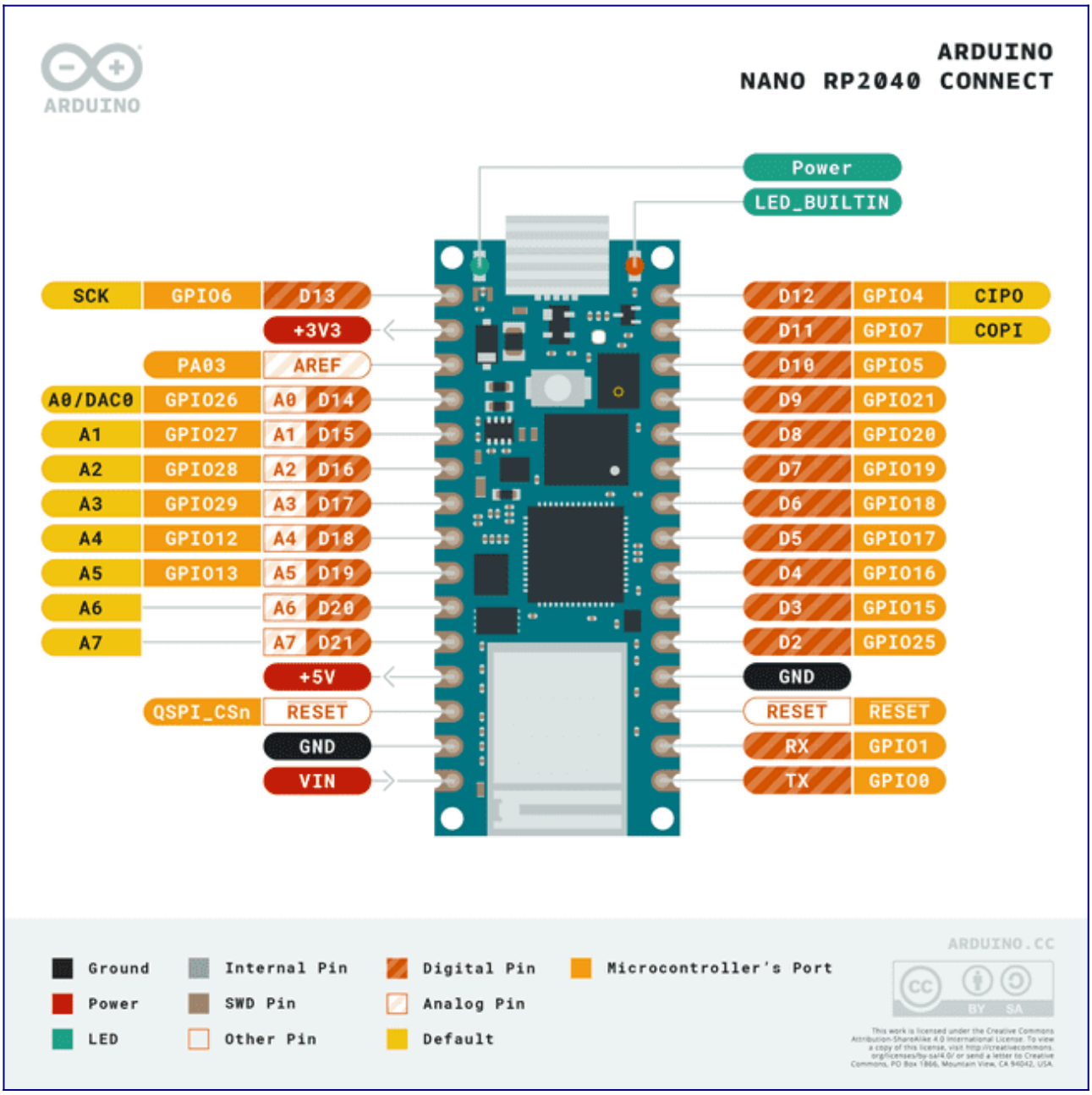
3. When the board is connected, it will open the mass storage device. You can now remove the jumper wire.
4. Upload a basic sketch, such as the blink example to the board (even though it is not visible in the port selection).
5. When it has finished uploading, your board should be visible in the board/port selection, and your board's built-in LED should be blinking. This means it is successful!

Alternatively, you can choose to factory-reset the board by dragging the blink.ino.elf.uf2 file into the mass storage. You can download the file from the link below: https://content.arduino.cc/assets/Blink.ino.elf.uf2



After dragging the U2F file, the board will be flashed with a program that blinks the built-in LED, and shifts between the red, green and blue pixels.

PinOut


ARDUINO
ARDUINO
NANO RP2040 CONNECT
Power
LED_BUILTIN
SCK GPIO6 D13
+3V3
PA03 AREF
A0/DAC0 GPIO26 A0 D14
A1 GPIO27 A1 D15
A2 GPIO28 A2 D16
A3 GPIO29 A3 D17
A4 GPIO12 A4 D18
A5 GPIO13 A5 D19
A6 A6 D20
A7 A7 D21
+5V
QSPI_CSn RESET
GND
VIN
D12 GPIO4 CIPO
D11 GPIO7 COPI
D10 GPIO5
D9 GPIO21
D8 GPIO20
D7 GPIO19
D6 GPIO18
D5 GPIO17
D4 GPIO16
D3 GPIO15
D2 GPIO25
GND
RESET RESET
RX GPIO1
TX GPIO0
Ground
Power
LED
Internal Pin
SWD Pin
Other Pin
Digital Pin
Analog Pin
Default
Microcontroller's Port
ARDUINO.CC
BY SA

Analog pins
The Nano RP2040 Connect has 8 analog pins, that can be used through the `analogRead()` function.

```
value = analogRead(pin, value);
```

Note: pin A4 and A5 are I2C only, while pin A6 and A7 are input only and do not support PWM.

PWM pins
Most of the digital & analog pins can be used as PWM (Pulse Width Modulation) pins, the exception being the following pins: A4, A5, A6, A7

```
analogWrite(pin, value);
```

Digital pins
There are a total of 14 digital pins, wheras the 8 analog pins can also be used as digital pins.

Note: A4 and A5 are I2C only, while A6 and A7 can only be used as inputs.

To use them, we first need to define them inside the `setup()` function of our sketch.

Note: digital pin 3 cannot be configured as INPUT_PULLUP

```
pinMode(pin, INPUT); //configured as an input
pinMode(pin, OUTPUT); //configured as an output
pinMode(pin, INPUT_PULLUP); //uses the internal 10k ohm resistor
```

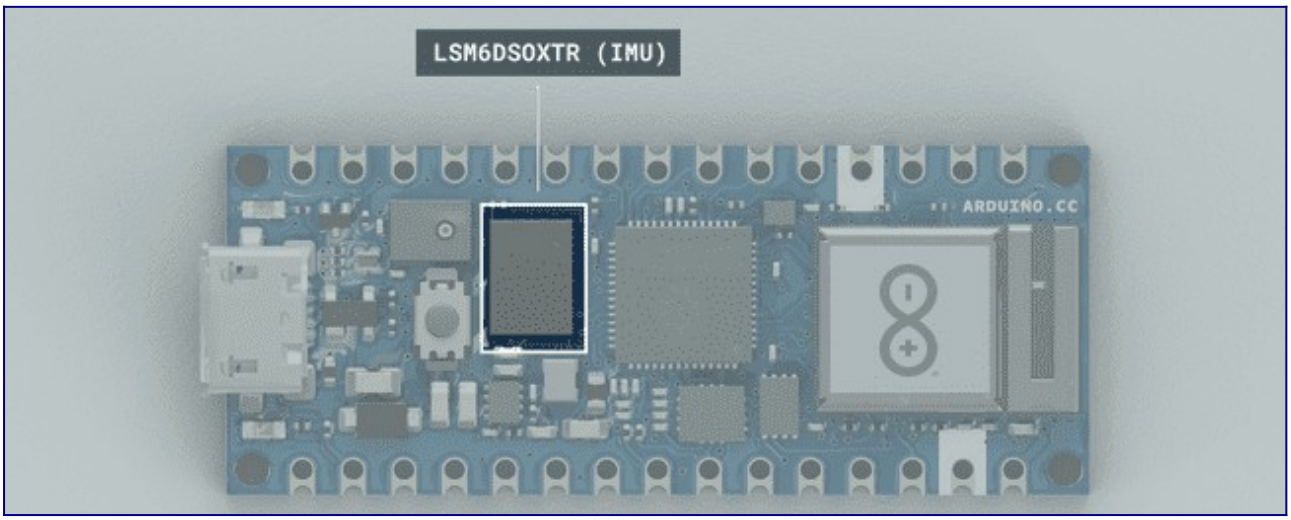
To read the state of a digital pin:

```
state = digitalRead(pin);
```

To write a state to a digital pin:

```
digitalWrite(pin, HIGH);
```

IMU (Inertial Measurement Unit)



The LSM6DSOXTR from STM is an Inertial Measurement Unit that features a 3D digital accelerometer and a 3D digital gyroscope. It features among many other things, a machine learning core, which is useful for any motion detection projects, such as free fall, step detector, step counter, pedometer.

To access the data from the LSM6DSOX module, we need to install the LSM6DSOX library, which comes with examples that can be used directly with the Nano RP2040 Connect.
It can be installed directly from the library manager through the IDE of your choice. To use it, we need to include at the top of the sketch:

```
#include <Arduino_LSM6DSOX.h>
```

And to initialize the library, we can use the following command inside `setup()`
.

```
if (!IMU.begin())
{
  Serial.println("Failed to initialize IMU!");
  while (1);
}
```

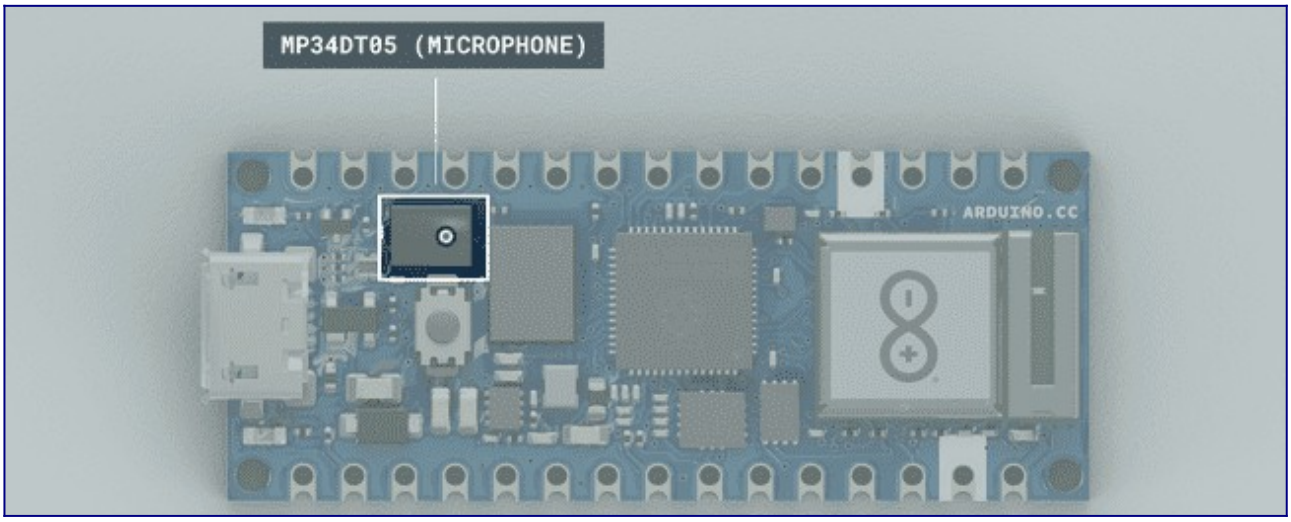
Accelerometer
The accelerometer data can be accessed through the following commands:

```
float x, y, z;
if (IMU.accelerationAvailable())
{
  IMU.readAcceleration(x, y, z);
}
```

Gyroscope
The gyroscope data can be accessed through the following commands:

```
float x, y, z;
if (IMU.gyroscopeAvailable())
{
  IMU.readGyroscope(x, y, z);
}
```

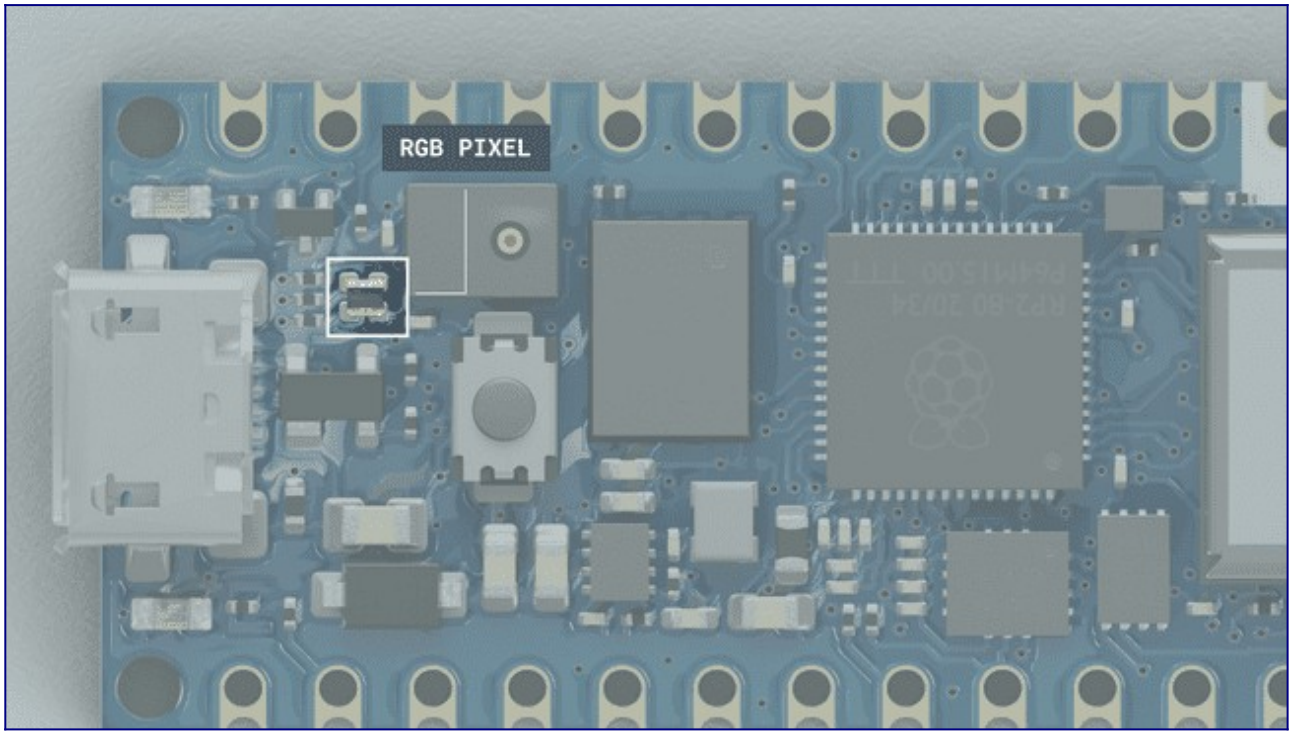
Microphone



The MP34DT05 is a compact, low-power omnidirectional digital MEMS microphone with an IC interface. It has a 64dB signal-to-noise ratio, is capable of sensing acoustic waves and can operate in temperatures of -40 °C to +85 °C.

To access the data from the MP34DT05, we need to use the PDM library that is included in the Arduino Mbed OS Nano Boards core. If the core is installed, you will find an example that works by browsing `File > Examples > PDM > PDMSerialPlotter`.

RGB LED



The Nano RP2040 Connect features a built-in RGB that can be utilized as a feedback component for applications. The RGB is connected through the W-102 module, so the WiFiNINA library needs to be installed and included at the top of your sketch to work.

```
#include <WiFiNINA.h>
```

The pins needs to be defined inside `setup()` as outputs:

```
pinMode(LEDR, OUTPUT);
pinMode(LEDG, OUTPUT);
pinMode(LEDB, OUTPUT);
```

To turn ON the pixels, write a HIGH state to the LED:

```
digitalWrite(LEDR, HIGH); //RED
digitalWrite(LEDG, HIGH); //GREEN
digitalWrite(LEDB, HIGH); //BLUE
```

To turn OFF the pixels, write a LOW state to the LED:

```
digitalWrite(LEDR, LOW); //RED
digitalWrite(LEDG, LOW); //GREEN
digitalWrite(LEDB, LOW); //BLUE
```

We can also choose a value between 255 - 0 to write to the LED:

```
analogWrite(LEDR, 72);  //GREEN
analogWrite(LEDG, 122); //BLUE
analogWrite(LEDB, 234); //RED
```

SPI

The pins used for SPI (Serial Peripheral Interface) on the Nano RP2040 Connect are the following:

- (CIPO) - D11
- (COPI) - D12
- (SCK) - D13
- (CS/SS) - Any GPIO (except for A6/A7)

To use SPI, we first need to include the SPI library.

```
#include <SPI.h>
```

Inside `setup()` we need to initialize the library.

```
SPI.begin();
```

And to write to the device:

```
digitalWrite(chipSelectPin, LOW); //pull down the CS pin
SPI.transfer(address); // address for device, for example 0x00
SPI.transfer(value); // value to write
digitalWrite(chipSelectPin, HIGH); // pull up the CS pin
```

I2C

The pins used for I2C (Inter-Integrated Circuit) on the Nano RP2040 Connect are the following:

- (SDA) - A4
- (SCL) – A5

To use I2C, we can use the Wire library, which we need to include at the top of our sketch.

```
#include <Wire.h>
```

Inside `setup()` we need to initialize the library.

```
Wire.begin();
```

And to write something to a device connected via I2C, we can use the following commands:

```
Wire.beginTransmission(1); //begin transmit to device 1
Wire.write(byte(0x00));    //send instruction byte
Wire.write(val);           //send a value
Wire.endTransmission();    //stop transmit5
```

UART

The pins used for UART (Universal asynchronous receiver-transmitter) are the following:

- (Rx) - D0
- (Tx) – D1

To send and receive data through UART, we will first need to set the baud rate inside `setup()`

```
Serial1.begin(9600);
```

To read incoming data, we can use a while loop() to read each individual character and add it to a string.

```
while(Serial1.available())
{
  delay(2);
  char c = Serial1.read();
  incoming += c;
}
```

And to write something, we can use the following command:

```
Serial1.write("Hello world!");
```

Wi-Fi

To use the Wi-Fi module on the Nano RP2040 Connect, we will need to install the WiFiNINA library, and include it at the top of our sketch:

```
#include <WiFiNINA.h>
```

To connect to a Wi-Fi network, we can use the following command:

```
WiFi.begin(ssid, pass);
```

The WiFiNINA library can be used to make GET & POST requests, while connected to the server. The command below is used to connect to www.google.com and return the results of searching for the keyword "arduino"

```
if (client.connect(server, port))
{
  client.println("GET /search?q=arduino HTTP/1.1");
  client.println("Host: www.google.com");
  client.println("Connection: close");
  client.println();
}
```

Bluetooth®

To enable Bluetooth® on the Nano RP2040 Connect, we can use the ArduinoBLE library, and include it at the top of our sketch:

```
#include <ArduinoBLE.h>
```

Set the service and characteristic:

```
BLEService ledService("180A"); // BLE LED Service
BLEByteCharacteristic switchCharacteristic("2A57", BLERead | BLEWrite);
```

Set advertised name and service:

```
BLE.setLocalName("Nano RP2040 Connect");
BLE.setAdvertisedService(ledService);
```

Start advertising:

```
BLE.advertise();
```

Listen for BLE peripherals to connect:

```
BLEDevice central = BLE.central();
```