Arduino Nicla Sense ME Cheat Sheet

Learn how to set up the Arduino Nicla Sense ME and get a quick overview of the components. Obtain information regarding pins and how to use the different sensors.

Author • Sebastian Romero

Last revision • 07/17/2024

Core

The Nicla Sense ME uses the Arduino Mbed OS Nicla Boards core.

Installation

Arduino IDE 1.8.X

The Nicla Sense ME can be programmed through the **Classic Arduino IDE 1.8.X**. To install your board, you can check out the guide below:

- Installing the Arduino Mbed OS Nicla Boards core

Arduino IDE 2

The Nicla Sense ME can be programmed through the

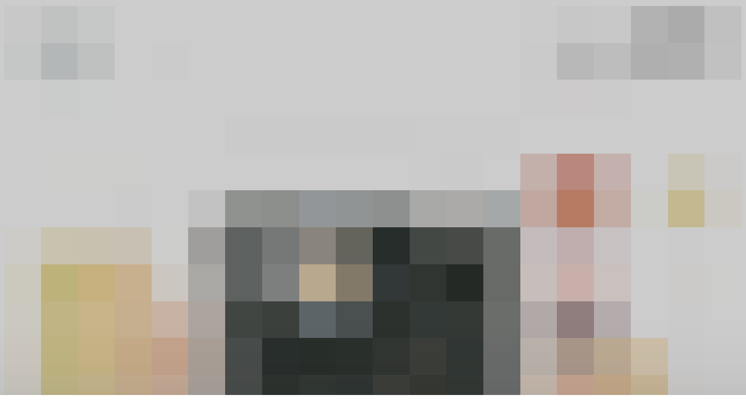
Search on Docs

Board Not Detected

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. Reconnect the board to your computer.
3. If it still doesn't show up, double-tap the reset button, to activate the bootloader mode.

Pins



Search on Docs

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

PWM Pins

Most of the digital and analog pins can be used as PWM (Pulse Width Modulation) pins. Check the full pinout in the resources section of the **Nicla Sense ME product page** to see which pins can be used.

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

Digital Pins

There are a total of 10 digital pins, whereas the 2 analog pins can also be used as digital pins.

To use them, they need to be configured. Usually this is

Search on Docs

To write a state to a digital pin:

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

RGB LED

The RGB LED is located in the rounded corner

The Nicla Sense ME features a built-in RGB that can be utilized as a feedback component for applications. The

We use cookies

Our websites use cookies (also from third parties) for functional and analytical purposes. You can adjust this in **Cookie Settings** or learn more by reading our **cookie policy**.

ONLY REQUIRED

ACCEPT ALL

Search on Docs

```
nicla::begin();
nicla::leds.begin();
```

The LED can be set to the desired RGB value using red, green and blue components or by using one of the following predefined colors:

- off
- red
- green
- blue
- yellow
- magenta
- cyan

To set the LED to a predefined color (e.g. green or blue):

Search on Docs

```
void loop() {
  int red = 234;
  int green = 72;
  int blue = 122;

  nicla::leds.setColor(red, green, blue)
  delay(1000);
  nicla::leds.setColor(off);
  delay(1000);
}
```

This is a complete example code to blink the built-in I2C LED:

```
#include "Nicla_System.h"

void setup() {
  nicla::begin();
  nicla::leds.begin();
}
```

2. Read sensor values through Bluetooth® Low Energy
3. Read sensor values through I2C by connecting an ESLOV cable

To read from the sensors in any of these mode, you need to install the **Arduino_BHY2** and **Arduino_BHY2Host** libraries. These can be found in the library manager using the Arduino IDE. To do so in the IDE, select **Tools->Manage Libraries...**, now search for **Arduino_BHY2** and **Arduino_BHY2Host** in the new window that opened and click on the install button.

To use the sensors in your sketches, you need to know the sensors ID. You can find them in the section "Sensor IDs" of this article. They can also be found in the header file here. Additionally, there is an example sketch in the library that will print all available sensors in the Serial Monitor of the Arduino IDE. This example sketch can be found in **File > Examples > Arduino_BHY2 > ShowSensorList** in the Arduino IDE.

In the following section, you can see how these ID's are used in an Arduino sketch.

- **SensorXYZ**: This class handles sensors with the XYZ format, like the accelerometer and the gyroscope. It contains `x` `y` and `z` values
- **SensorQuaternion**: Can be used to handle sensors with the quaternion format, can be used to calculate rotation vector, game rotation vector and geomagnetic rotation vector. You can access the `x` , `y` , `z` and `w` property using this class.
- **SensorActivity**: Use this class to handle sensors with the activity format. The activity is encoded as ID and can be retrieved from the `value` property. Use `getActivity` to get a human readable version of the activity e.g. "Walking activity started".
- **SensorBSEC**: BSEC stands for Bosch Sensortec Environmental Cluster, basically you can access the air quality (IAQ) level and it contains the following data:

Function	Description	Data type

Search on Docs

`comp_h()`	compensated humidity	float
`comp_g()`	compensated gas resistance (Ohms)	unsigned 32bit
`accuracy()`	accuracy level: [0-3]	unsigned 8bit

Further down you can see how these objects are used in an Arduino sketch to get readings from the sensors.

Sensor IDs

The IDs to address the sensors both through ESLOV and WebBLE are as follows:

ID	Description	SENSOR_ID MACRO
1	Accelerometer passthrough	SENSOR_ID_ACC_

Search on Docs

	passthrough	
12	Gyroscope uncalibrated	SENSOR_ID_GYRC
13	Gyroscope corrected	SENSOR_ID_GY
14	Gyroscope offset	SENSOR_ID_GYRC
15	Gyroscope wake up	SENSOR_ID_GYR(
16	Gyroscope uncalibrated wake up	SENSOR_ID_GYRO_I
19	Magnetometer passthrough	SENSOR_ID_MAG
21	Magnetometer uncalibrated	SENSOR_ID_MAG
22	Magnetometer corrected	SENSOR_ID_M
23	Magnetometer offset	SENSOR_ID_MAG

Search on Docs

34	Rotation vector	SENSOR_ID_F
35	Rotation vector wake up	SENSOR_ID_RV_
37	Game rotation vector	SENSOR_ID_GAN
38	Game rotation vector wake up	SENSOR_ID_GAME
40	Geomagnetic rotation vector	SENSOR_ID_GE
41	Geomagnetic rotation vector wake up	SENSOR_ID_GEOF
43	Orientation	SENSOR_ID_C
44	Orientation wake up	SENSOR_ID_ORI
48	Tilt detector	SENSOR_ID_TILT_DE

67	Wrist tilt gesture	SENSOR_ID_WRIST_TIL
69	Device orientation	SENSOR_ID_DEVIC
70	Device orientation wake up	SENSOR_ID_DEVICE_
75	Stationary detect	SENSOR_ID_STATION
77	Motion detect	SENSOR_ID_MOTIC
91	Accelerometer offset wake up	SENSOR_ID_ACC_B
92	Gyroscope offset wake up	SENSOR_ID_GYRO_E
93	Magnetometer offset wake up	SENSOR_ID_MAG_B
94	Step detector wake up	SENSOR_ID_STD
115	BSEC data	SENSOR_ID_BS

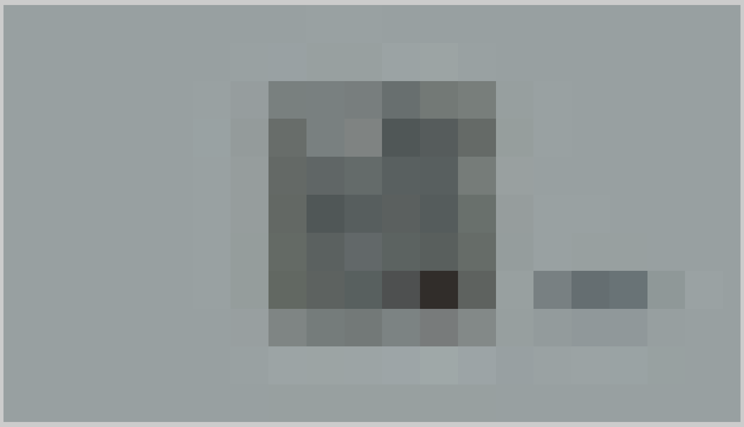
Search on Docs

	Step counter	
137	Hardware Step detector	SENSOR_ID_STD
138	Hardware Significant motion	SENSOR_ID_SIG
139	Hardware Step counter wake up	SENSOR_ID_STC_H
140	Hardware Step detector wake up	SENSOR_ID_STD_H
141	Hardware Significant motion wake up	SENSOR_ID_SIG_H
142	Any motion	SENSOR_ID_ANY_M
143	Any motion	SENSOR_ID_ANY_MO

The syntax to instantiate a sensor object is

Search on Docs

IMU


The IMU sensor

Follow these steps to use the library to read the sensor values.

Include the library's header file:

```
#include "Arduino_BHY2.h"
```

Search on Docs

```
void setup(){
  Serial.begin(115200);
  BHY2.begin();
```

The `begin()` function starts the sensor by calling the `configure()` with default parameters, making it easy to start and use on-board sensors. The parameters in the `configure()` function are sample rate and latency. If specific parameters are needed, then simply call `configure()` with your preferred values. E.g.: `configure(10, 1)`. In this case, the sample rate would be set to `10 Hz` and the latency would be `1ms`.

- **Sample rate** is used also to enable/disable the sensor. 0 to disable, > 0 to enable.
- **Latency** is the longest delay, in milliseconds, before the host is notified of a new value from a sensor. Even though the latency parameter is a 32-bit integer, only the least significant 24 bits of it are actually used. A latency of 0 means that the host is notified immediately when a new value is available.

Reading the sensor values:

accelerometer and the gyroscope as follows:

```
short accX = accelerometer.x();
short accY = accelerometer.y();
short accZ = accelerometer.z();

short gyroX = gyroscope.x();
short gyroY = gyroscope.y();
short gyroZ = gyroscope.z();
```

Temperature

To read the temperature in standalone mode, you also need to use the Arduino_BHY2 library as described in the section above.

Follow these steps to use the library to read the sensor values.

Include the library's header file:

Search on Docs

```
void setup(){
  Serial.begin(115200);
  BHY2.begin();
  temperature.begin();
}
```

Reading the sensor value:

```
void loop(){
  static auto lastCheck= millis();
  BHY2.update();

  // Check sensor values every second
  if (millis() - lastCheck >= 1000) {
    lastCheck = millis();
    Serial.println(String("temperature:
  }
}
```

Gas

Search on Docs

```
Sensor gas(SENSOR_ID_GAS);
```

Activating the sensor:

```
void setup() {
  Serial.begin(115200);
  BHY2.begin();
  gas.begin();
}
```

Reading the sensor value:

```
void loop(){
  static auto lastCheck= millis();
  BHY2.update();

  // Check sensor values every second
```

We use cookies

Our websites use cookies (also from third parties) for functional and analytical purposes. You can adjust this in **Cookie Settings** or learn more by reading our **cookie policy**.

ONLY REQUIRED

ACCEPT ALL

Search on Docs

```
#include "Arduino_BHY2.h"
```

Define the sensor object:

```
Sensor pressure(SENSOR_ID_BARO);
```

Activating the sensor:

```
void setup() {
  Serial.begin(115200);
  BHY2.begin();
  pressure.begin();
}
```

To get readings from the IMU in a quaternion format in standalone mode, you also need to use the Arduino_BHY2 library as described in the section above.

Follow these steps to use the library to read the sensor values.

Include the library's header file:

```
#include "Arduino_BHY2.h"
```

Define the sensor object:

```
SensorQuaternion rotation(SENSOR_ID_RV);
```

Activating the sensor:

```
void loop(){
  static auto lastCheck = millis();
  BHY2.update();

  if (millis() - lastCheck >= 1000) {
    lastCheck = millis();
```

Activity

To get activity status in standalone mode, you also need to use the Arduino_BHY2 library as described in the section above.

Follow these steps to use the library to read the sensor values.

Include the library's header file:

```
#include "Arduino_BHY2.h"
```

Define the sensor object:

Reading the sensor value:

```
void loop(){
  static auto lastCheck = millis();
  BHY2.update();

  if (millis() - lastCheck >= 1000) {
    printTime = millis();
    Serial.println(String("Activity info:
  }
}
```

BSEC Data

To get readings from the BME sensor in standalone mode, you also need to use the Arduino_BHY2 library as described in the section above.

Follow these steps to use the library to read the sensor values.

Search on Docs

```
void setup(){
  Serial.begin(115200);

  BHY2.begin();
  bsec.begin();
}
```

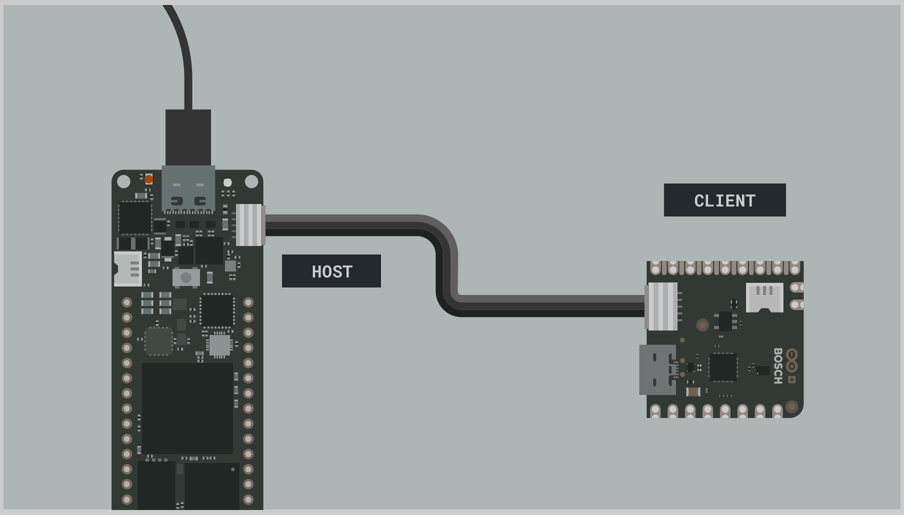
Reading the sensor value:

```
void loop(){
  static auto lastCheck = millis();
  BHY2.update();

  if (millis() - lastCheck >= 1000) {
    printTime = millis();
    Serial.println(String("BSEC info: ")
  }
}
```

Search on Docs

board you need to connect the boards with an ESLOV cable. The Nicla Sense ME could for example be connected to an Arduino Portenta H7.



1 To have the Nicla Sense ME pass the sensor data through ESLOV, you need to upload the **App** sketch. You can find it in the Examples menu in the IDE under **Arduino_BH2 > App**. After you upload the sketch, you can disconnect the Nicla Sense ME from the USB cable. It can be powered through the ESLOV connection.

If you like, you can build the tool yourself (e.g. if you are on macOS). For that, if you haven't installed **Go** yet, please do so by following there instructions. From the terminal execute this command to start the build: `go build`

4. Use the bhy command as follows:

```
# list available serial ports
./bhy list

# read available sensor data
./bhy sensor read -p /dev/ttyACM2

# continuously read sensor data when ava
./bhy sensor read -live -p /dev/ttyACM2

# configure sensor 10 with a sample rate
./bhy sensor config -p /dev/ttyACM2 -sen

# disable sensor 10
./bhy sensor config -p /dev/ttyACM2 -sen
```

```
$ ./bhy list
Found port: /dev/cu.usbmodem142301
```

Then execute a config command to enable the desired sensor. Please refer to the **Sensor IDs** section to find the desired sensor ID. The output of this command should look similar to this:

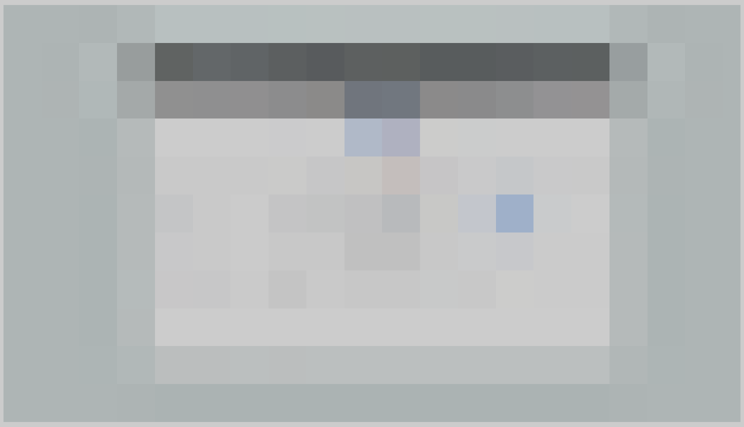
```
$ ./bhy sensor config -p /dev/cu.usbmodem
Connected - port: /dev/cu.usbmodem142303
Sending configuration: sensor 10    rate
Sensor configuration correctly sent!
```

Then you can retrieve sensor data from the sensor using the read command. Here is an example output:

```
$ ./bhy sensor read -live -p /dev/cu.usbm
Connected - port: /dev/cu.usbmodem142303
Sensor id: 10   name: GYRO_PASS   values:
```

Search on Docs

Then execute the following command to start the webserver: `./bhy webserver`. When the server has started, you can open the landing page in your browser: http://localhost:8000/. Click on "Open sensor page".


Sensor page in the browser

Then click the "Connect" button and pair your computer with the Nicla Sense ME board.

For this feature to work, make sure that Web Bluetooth® Low Energy is both supported and enabled! In Google Chrome go to chrome://flags and

Configured sensors

BSX Sensor Fusion Software

The BHI260AP sensor runs a customizable firmware based on the BSX Sensor Fusion library. It provides a complete 9-axis fusion solution, which combines the measurements from 3-axis gyroscope, 3-axis geomagnetic sensor and a 3-axis accelerometer, to provide a robust absolute orientation vector. The algorithm fuses the sensor raw data from the accelerometer, geomagnetic sensor and gyroscope in an intelligent way to improve each sensor’s output.

Go to this site or take a look at the BHI260AP's datasheet for more information.

Communication

Search on Docs

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

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

Inside `void setup()` you need to initialize the library.

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

And to write to the device:

```
digitalWrite(chipSelectPin, LOW); //pull

  SPI.transfer(address); // address for d
  SPI.transfer(value); // value to write

  digitalWrite(chipSelectPin, HIGH); // p
```

We use cookies

Our websites use cookies (also from third parties) for functional and analytical purposes. You can adjust this in **Cookie Settings** or learn more by reading our **cookie policy**.

ONLY REQUIRED

ACCEPT ALL

Search on Docs

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

Inside `void setup()` you need to initialize the library.

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

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

```
Wire.beginTransmission(1); //begin transm
  Wire.write(byte(0x00)); //send instruct
  Wire.write(val); //send a value
  Wire.endTransmission(); //stop transmit
```

Search on Docs

To read incoming data, you 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, you can use the following command:

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

Bluetooth® Low Energy

Search on Docs

```
#include "Arduino_BHY2Host.h"

Sensor temperature(SENSOR_ID_TEMP);
int lastCheck = 0;

void setup(){
  Serial.begin(115200);
  BHY2Host.begin(false, NICLA_VIA_BLE);
  temperature.begin();
}

void loop(){
  static auto lastCheck= millis();
  BHY2Host.update();

  // Check sensor values every second
  if (millis() - lastCheck >= 1000) {
    lastCheck = millis();
    Serial.println(String("temperature:
  }
}
```

The parameters of `BHY2Host::begin` are: data pass

We use cookies

Our websites use cookies (also from third parties) for functional and analytical purposes. You can adjust this in **Cookie Settings** or learn more by reading our **cookie policy**.

ONLY REQUIRED

ACCEPT ALL

Search on Docs

Include the Nicla System header at the top of your sketch:

```
#include "Nicla_System.h"
```

In the setup() function add:

```
nicla::begin();
```

Here is an example of how to use the Bluetooth® Low Energy library to advertise a byte characteristic that can be used for example to toggle an LED.

Include the library header it at the top of your sketch:

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

Start advertising:

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

Listen for BLE peripherals to connect:

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

Conclusion

This cheat sheet is written as a quick reference mainly to look up the features of this product. For a more in-depth walk though experience, please have a look at the other tutorials.