Adafruit BNO055 Library Documentation

Release 1.0

Radomir Dopieralski

Apr 29, 2021

Contents

1 Dependencies **3**

2 Installing from PyPI **5**

3 Usage Notes **7**

4 Contributing **9**

5 Documentation **11**

6 Table of Contents **13**

6.1 Simple test 13

6.2 Raspberry PI I2C GPIO Simpletest 14

6.3 `adafruit_bno055` 15

7 Indices and tables **21**

Python Module Index **23**

Index **25**

CHAPTER 1

Dependencies

This driver depends on the Register and Bus Device libraries. Please ensure they are also available on the CircuitPython filesystem. This is easily achieved by downloading a library and driver bundle.

CHAPTER 2

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-bno055
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-bno055
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-bno055
```

CHAPTER 3

Usage Notes

Of course, you must import the library to use it:

```
import adafruit_bno055
```

This driver takes an instantiated and active I2C object as an argument to its constructor. The way to create an I2C object depends on the board you are using. For boards with labeled SCL and SDA pins, you can:

```
import board

i2c = board.I2C()
```

Once you have the I2C object, you can create the sensor object:

```
sensor = adafruit_bno055.BNO055_I2C(i2c)
```

And then you can start reading the measurements:

```
print(sensor.temperature)
print(sensor.euler)
print(sensor.gravity)
```

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/bno055_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_bno055


i2c = board.I2C()
sensor = adafruit_bno055.BNO055_I2C(i2c)

# If you are going to use UART uncomment these lines
# uart = board.UART()
# sensor = adafruit_bno055.BNO055_UART(uart)

last_val = 0xFFFF


def temperature():
    global last_val  # pylint: disable=global-statement
    result = sensor.temperature
    if abs(result - last_val) == 128:
        result = sensor.temperature
        if abs(result - last_val) == 128:
            return 0b00111111 & result
    last_val = result
    return result
```

(continues on next page)

(continued from previous page)

```


while True:
    print("Temperature: {} degrees C".format(sensor.temperature))
    """
    print(
        "Temperature: {} degrees C".format(temperature())
    )  # Uncomment if using a Raspberry Pi
    """
    print("Accelerometer (m/s^2): {}".format(sensor.acceleration))
    print("Magnetometer (microteslas): {}".format(sensor.magnetic))
    print("Gyroscope (rad/sec): {}".format(sensor.gyro))
    print("Euler angle: {}".format(sensor.euler))
    print("Quaternion: {}".format(sensor.quaternion))
    print("Linear acceleration (m/s^2): {}".format(sensor.linear_acceleration))
    print("Gravity (m/s^2): {}".format(sensor.gravity))
    print()

    time.sleep(1)
```

6.2 Raspberry PI I2C GPIO Simpletest

This example demonstrates how to instantiate the Adafruit BNO055 Sensor using this library and just the I2C bus number.

Listing 2: examples/bno055_i2c-gpio_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
This example demonstrates how to instantiate the
Adafruit BNO055 Sensor using this library and just
the I2C bus number.
This example will only work on a Raspberry Pi
and does require the i2c-gpio kernel module to be
installed and enabled. Most Raspberry Pis will
already have it installed, however most do not
have it enabled. You will have to manually enable it
"""

import time
from adafruit_extended_bus import ExtendedI2C as I2C
import adafruit_bno055

# To enable i2c-gpio, add the line `dtoverlay=i2c-gpio` to /boot/config.txt
# Then reboot the pi

# Create library object using our Extended Bus I2C port
# Use `ls /dev/i2c*` to find out what i2c devices are connected
i2c = I2C(1)  # Device is /dev/i2c-1
sensor = adafruit_bno055.BNO055_I2C(i2c)

last_val = 0xFFFF
```

(continues on next page)

(continued from previous page)

```


def temperature():
    global last_val  # pylint: disable=global-statement
    result = sensor.temperature
    if abs(result - last_val) == 128:
        result = sensor.temperature
        if abs(result - last_val) == 128:
            return 0b00111111 & result
    last_val = result
    return result


while True:
    print("Temperature: {} degrees C".format(temperature()))
    print("Accelerometer (m/s^2): {}".format(sensor.acceleration))
    print("Magnetometer (microteslas): {}".format(sensor.magnetic))
    print("Gyroscope (rad/sec): {}".format(sensor.gyro))
    print("Euler angle: {}".format(sensor.euler))
    print("Quaternion: {}".format(sensor.quaternion))
    print("Linear acceleration (m/s^2): {}".format(sensor.linear_acceleration))
    print("Gravity (m/s^2): {}".format(sensor.gravity))
    print()

    time.sleep(1)
```

6.3 adafruit_bno055

This is a CircuitPython driver for the Bosch BNO055 nine degree of freedom inertial measurement unit module with sensor fusion.

- Author(s): Radomir Dopieralski

Hardware:

- Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055 (Product ID: 4646)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice
- Adafruit's Register library: https://github.com/adafruit/Adafruit_CircuitPython_Register

class adafruit_bno055.**BNO055**

Base class for the BNO055 9DOF IMU sensor.

Quickstart: Importing and using the device

Here is an example of using the *BNO055* class. First you will need to import the libraries to use the sensor

```
import board
import adafruit_bno055
```

Once this is done you can define your `board.I2C` object and define your sensor object

```
i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = adafruit_bno055.BNO055_I2C(i2c)
```

Now you have access to the `acceleration` attribute among others

```
sensor = accelerometer.acceleration
```

accel_bandwidth
Switch the accelerometer bandwidth and return the new bandwidth. Default value: 62.5 Hz See table 3-8 in the datasheet.

accel_mode
Switch the accelerometer mode and return the new mode. Default value: Normal See table 3-8 in the datasheet.

accel_range
Switch the accelerometer range and return the new range. Default value: +/- 4g See table 3-8 in the datasheet.

acceleration
Gives the raw accelerometer readings, in m/s. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

axis_remap
Return a tuple with the axis remap register values.

This will return 6 values with the following meaning:

- **X axis remap (a value of AXIS_REMAP_X, AXIS_REMAP_Y, or AXIS_REMAP_Z.** which indicates that the physical X axis of the chip is remapped to a different axis)
- Y axis remap (see above)
- Z axis remap (see above)
- **X axis sign (a value of AXIS_REMAP_POSITIVE or AXIS_REMAP_NEGATIVE** which indicates if the X axis values should be positive/ normal or negative/inverted. The default is positive.)
- Y axis sign (see above)
- Z axis sign (see above)

Note that the default value, per the datasheet, is NOT P0, but rather P1 ()

calibrated
Boolean indicating calibration status.

calibration_status
Tuple containing sys, gyro, accel, and mag calibration data.

euler
Gives the calculated orientation angles, in degrees. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

external_crystal
Switches the use of external crystal on or off.

gravity
Returns the gravity vector, without acceleration in m/s. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

gyro
Gives the raw gyroscope reading in radians per second. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

gyro_bandwidth
Switch the gyroscope bandwidth and return the new bandwidth. Default value: 32 Hz See table 3-9 in the datasheet.

gyro_mode
Switch the gyroscope mode and return the new mode. Default value: Normal See table 3-9 in the datasheet.

gyro_range
Switch the gyroscope range and return the new range. Default value: 2000 dps See table 3-9 in the datasheet.

linear_acceleration
Returns the linear acceleration, without gravity, in m/s. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

magnet_mode
Switch the magnetometer power mode and return the new mode. Default value: Forced See table 3-10 in the datasheet.

magnet_operation_mode
Switch the magnetometer operation mode and return the new mode. Default value: Regular See table 3-10 in the datasheet.

magnet_rate
Switch the magnetometer data output rate and return the new rate. Default value: 20Hz See table 3-10 in the datasheet.

magnetic
Gives the raw magnetometer readings in microteslas. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

mode
legend: x=on, -=off (see Table 3-3 in datasheet)

Mode	Accel	Compass (Mag)	Gyro	Fusion Absolute	Fusion Relative
CONFIG_MODE	•	•	•	•	•
ACCONLY_MODE	X	•	•	•	•
MAGONLY_MODE	•	X	•	•	•
GYRONLY_MODE	•	•	X	•	•
ACCMAG_MODE	X	X	•	•	•
ACCGYRO_MODE	X	•	X	•	•
MAGGYRO_MODE	•	X	X	•	•
AMG_MODE	X	X	X	•	•
IMUPLUS_MODE	X	•	X	•	X
COMPASS_MODE	X	X	•	X	•
M4G_MODE	X	X	•	•	X
NDOF_FMC_OFF_MODE	X	X	X	X	•
NDOF_MODE	X	X	X	X	•

The default mode is *NDOF_MODE*.

You can set the mode using the line below:
`sensor.mode = adafruit_bno055.ACCONLY_MODE`
replacing *ACCONLY_MODE* with the mode you want to use

CONFIG_MODE
This mode is used to configure BNO, wherein all output data is reset to zero and sensor fusion is halted.

ACCONLY_MODE
In this mode, the BNO055 behaves like a stand-alone acceleration sensor. In this mode the other sensors (magnetometer, gyro) are suspended to lower the power consumption.

MAGONLY_MODE
In MAGONLY mode, the BNO055 behaves like a stand-alone magnetometer, with acceleration sensor and gyroscope being suspended.

GYRONLY_MODE
In GYROONLY mode, the BNO055 behaves like a stand-alone gyroscope, with acceleration sensor and magnetometer being suspended.

ACCMAG_MODE
Both accelerometer and magnetometer are switched on, the user can read the data from these two sensors.

ACCGYRO_MODE
Both accelerometer and gyroscope are switched on; the user can read the data from these two sensors.

MAGGYRO_MODE
Both magnetometer and gyroscope are switched on, the user can read the data from these two sensors.

AMG_MODE
All three sensors accelerometer, magnetometer and gyroscope are switched on.

IMUPLUS_MODE
In the IMU mode the relative orientation of the BNO055 in space is calculated from the accelerometer and gyroscope data. The calculation is fast (i.e. high output data rate).

COMPASS_MODE
The COMPASS mode is intended to measure the magnetic earth field and calculate the geographic direction.

M4G_MODE
The M4G mode is similar to the IMU mode, but instead of using the gyroscope signal to detect rotation, the changing orientation of the magnetometer in the magnetic field is used.

NDOF_FMC_OFF_MODE
This fusion mode is same as NDOF mode, but with the Fast Magnetometer Calibration turned 'OFF'.

NDOF_MODE
This is a fusion mode with 9 degrees of freedom where the fused absolute orientation data is calculated from accelerometer, gyroscope and the magnetometer.

quaternion
Gives the calculated orientation as a quaternion. Returns an empty tuple of length 3 when this property has been disabled by the current mode.

temperature
Measures the temperature of the chip in degrees Celsius.

use_external_crystal
Switches the use of external crystal on or off.

class adafruit_bno055.**BNO055_I2C**(*i2c*, *address=40*)
Driver for the BNO055 9DOF IMU sensor via I2C.

offsets_accelerometer
Calibration offsets for the accelerometer

offsets_gyroscope
Calibration offsets for the gyroscope

offsets_magnetometer
Calibration offsets for the magnetometer

radius_accelerometer
Radius for accelerometer (cm?)

radius_magnetometer
Radius for magnetometer (cm?)

class adafruit_bno055.**BNO055_UART**(*uart*)
Driver for the BNO055 9DOF IMU sensor via UART.

offsets_accelerometer
Calibration offsets for the accelerometer

offsets_gyroscope
Calibration offsets for the gyroscope

offsets_magnetometer
Calibration offsets for the magnetometer

radius_accelerometer
Radius for accelerometer (cm?)

radius_magnetometer
Radius for magnetometer (cm?)

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

adafruit_bno055, 15

Index

A

accel_bandwidth (*adafruit_bno055.BNO055 attribute*), 16
accel_mode (*adafruit_bno055.BNO055 attribute*), 16
accel_range (*adafruit_bno055.BNO055 attribute*), 16
acceleration (*adafruit_bno055.BNO055 attribute*), 16
adafruit_bno055 (*module*), 15
axis_remap (*adafruit_bno055.BNO055 attribute*), 16

B

BNO055 (*class in adafruit_bno055*), 15
BNO055.ACCGYRO_MODE (*in module adafruit_bno055*), 19
BNO055.ACCMAG_MODE (*in module adafruit_bno055*), 19
BNO055.ACCONLY_MODE (*in module adafruit_bno055*), 18
BNO055.AMG_MODE (*in module adafruit_bno055*), 19
BNO055.COMPASS_MODE (*in module adafruit_bno055*), 19
BNO055.CONFIG_MODE (*in module adafruit_bno055*), 18
BNO055.GYRONLY_MODE (*in module adafruit_bno055*), 19
BNO055.IMUPLUS_MODE (*in module adafruit_bno055*), 19
BNO055.M4G_MODE (*in module adafruit_bno055*), 19
BNO055.MAGGYRO_MODE (*in module adafruit_bno055*), 19
BNO055.MAGONLY_MODE (*in module adafruit_bno055*), 19
BNO055.NDOF_FMC_OFF_MODE (*in module adafruit_bno055*), 19
BNO055.NDOF_MODE (*in module adafruit_bno055*), 19
BNO055_I2C (*class in adafruit_bno055*), 19
BNO055_UART (*class in adafruit_bno055*), 20

C

calibrated (*adafruit_bno055.BNO055 attribute*), 16
calibration_status (*adafruit_bno055.BNO055 attribute*), 16

E

euler (*adafruit_bno055.BNO055 attribute*), 16
external_crystal (*adafruit_bno055.BNO055 attribute*), 16

G

gravity (*adafruit_bno055.BNO055 attribute*), 16
gyro (*adafruit_bno055.BNO055 attribute*), 16
gyro_bandwidth (*adafruit_bno055.BNO055 attribute*), 17
gyro_mode (*adafruit_bno055.BNO055 attribute*), 17
gyro_range (*adafruit_bno055.BNO055 attribute*), 17

L

linear_acceleration (*adafruit_bno055.BNO055 attribute*), 17

M

magnet_mode (*adafruit_bno055.BNO055 attribute*), 17
magnet_operation_mode (*adafruit_bno055.BNO055 attribute*), 17
magnet_rate (*adafruit_bno055.BNO055 attribute*), 17
magnetic (*adafruit_bno055.BNO055 attribute*), 17
mode (*adafruit_bno055.BNO055 attribute*), 17

O

offsets_accelerometer (*adafruit_bno055.BNO055_I2C attribute*), 19
offsets_accelerometer (*adafruit_bno055.BNO055_UART attribute*), 20

offsets_gyroscope (*adafruit_bno055.BNO055_I2C attribute*), 19
offsets_gyroscope (*adafruit_bno055.BNO055_UART attribute*), 20
offsets_magnetometer (*adafruit_bno055.BNO055_I2C attribute*), 19
offsets_magnetometer (*adafruit_bno055.BNO055_UART attribute*), 20

Q

quaternion (*adafruit_bno055.BNO055 attribute*), 19

R

radius_accelerometer (*adafruit_bno055.BNO055_I2C attribute*), 20
radius_accelerometer (*adafruit_bno055.BNO055_UART attribute*), 20
radius_magnetometer (*adafruit_bno055.BNO055_I2C attribute*), 20
radius_magnetometer (*adafruit_bno055.BNO055_UART attribute*), 20

T

temperature (*adafruit_bno055.BNO055 attribute*), 19

U

use_external_crystal (*adafruit_bno055.BNO055 attribute*), 19