Adafruit L3GD20 Library Documentation

Release 1.0

Michael McWethy

Jul 02, 2020

Contents

Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - L3GD20 Driver

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Register

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit 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-l3gd20
```

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

```
sudo pip3 install adafruit-circuitpython-l3gd20
```

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-l3gd20
```

CHAPTER 3

Usage Example

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

```
import adafruit_l3gd20
```

This driver takes an instantiated and active I2C object (from the `busio` or the `bitbangio` library) 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:

```
from busio import I2C
from board import SDA, SCL

i2c = I2C(SCL, SDA)
```

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

```
sensor = adafruit_l3gd20.L3GD20_I2C(i2c)
```

And then you can start reading the measurements:

```
print(sensor.gyro)
```

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

For I2C or SPI communications, ensure your device works with this simple test.

Listing 1: examples/l3gd20_simpletest.py

```
import time
import board
import busio
import adafruit_l3gd20

# Hardware I2C setup:
I2C = busio.I2C(board.SCL, board.SDA)
# Initializes L3GD20 object using default range, 250dps
SENSOR = adafruit_l3gd20.L3GD20_I2C(I2C)
# Initialize L3GD20 object using a custom range and output data rate (ODR).
# SENSOR = adafruit_l3gd20.L3GD20_I2C(
#     I2C, rng=adafruit_l3gd20.L3DS20_RANGE_500DPS, rate=adafruit_l3gd20.L3DS20_RATE_
↪200HZ
# )

# Possible values for rng are:
# adafruit_l3gd20.L3DS20_Range_250DPS, 250 degrees per second. Default range
# adafruit_l3gd20.L3DS20_Range_500DPS, 500 degrees per second
# adafruit_l3gd20.L3DS20_Range_2000DPS, 2000 degrees per second

# Possible values for rate are:
# adafruit_l3gd20.L3DS20_RATE_100HZ, 100Hz data rate. Default data rate
# adafruit_l3gd20.L3DS20_RATE_200HZ, 200Hz data rate
# adafruit_l3gd20.L3DS20_RATE_400HZ, 400Hz data rate
# adafruit_l3gd20.L3DS20_RATE_800HZ, 800Hz data rate

# Hardware SPI setup:
```

(continues on next page)

(continued from previous page)

```
# import digitalio
# CS = digitalio.DigitalInOut(board.D5)
# SPIB = busio.SPI(board.SCK, board.MOSI, board.MISO)
# SENSOR = adafruit_l3gd20.L3GD20_SPI(SPIB, CS)
# SENSOR = adafruit_l3gd20.L3GD20_I2C(
#     SPIB,
#     CS,
#     rng=adafruit_l3gd20.L3DS20_RANGE_500DPS,
#     rate=adafruit_l3gd20.L3DS20_RATE_200HZ,
# )

while True:
    print("Angular Momentum (rad/s): {}".format(SENSOR.gyro))
    print()
    time.sleep(1)
```

6.2 adafruit_l3gd20

Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - L3GD20

This is a CircuitPython driver for the Bosch L3GD20 nine degree of freedom inertial measurement unit module with sensor fusion.

- Author(s): Michael McWethy

6.2.1 Implementation Notes

Hardware:

- L3GD20H Triple-Axis Gyro Breakout Board

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases
- Adafruit's Register library: https://github.com/adafruit/Adafruit_CircuitPython_Register

class adafruit_l3gd20.**L3GD20**(*rng=0*, *rate=0*)

Driver for the L3GD20 3-axis Gyroscope sensor.

Parameters

- **rng** (*int*) – a range value one of L3DS20_RANGE_250DPS (default), L3DS20_RANGE_500DPS, or L3DS20_RANGE_2000DPS
- **rate** (*int*) – a rate value one of L3DS20_RATE_100HZ (default), L3DS20_RATE_200HZ, L3DS20_RATE_400HZ, or L3DS20_RATE_800HZ

gyro

x, y, z angular momentum tuple floats, rescaled appropriately for range selected

class adafruit_l3gd20.**L3GD20_I2C**(*i2c*, *rng=0*, *address=107*, *rate=0*)

Driver for L3GD20 Gyroscope using I2C communications

Parameters

- **i2c** (*I2C*) – initialized busio I2C class

- **rng** (*int*) – the optional range value: L3DS20_RANGE_250DPS(default), L3DS20_RANGE_500DPS, or L3DS20_RANGE_2000DPS
- **address** – the optional device address, 0x68 is the default address

gyro_raw
Gives the raw gyro readings, in units of rad/s.

read_register(*register*)
Returns a byte value from a register

Parameters **register** – the register to read a byte

write_register(*register*, *value*)
Update a register with a byte value

Parameters

- **register** (*int*) – which device register to write
- **value** – a byte to write

class adafruit_l3gd20.**L3GD20_SPI**(*spi_busio*, *cs*, *rng=0*, *baudrate=100000*, *rate=0*)
Driver for L3GD20 Gyroscope using SPI communications

Parameters

- **spi_busio** (*SPI*) – initialized busio SPI class
- **cs** (*DigitalInOut*) – digital in/out to use as chip select signal
- **rng** (*int*) – the optional range value: L3DS20_RANGE_250DPS(default), L3DS20_RANGE_500DPS, or L3DS20_RANGE_2000DPS
- **baudrate** – spi baud rate default is 100000

gyro_raw
Gives the raw gyro readings, in units of rad/s.

read_bytes(*register*, *buffer*)
Low level register stream reading over SPI, returns a list of values

Parameters

- **register** – the register to read bytes
- **buffer** (*bytearray*) – buffer to fill with data from stream

read_register(*register*)
Low level register reading over SPI, returns a list of values

Parameters **register** – the register to read a byte

write_register(*register*, *value*)
Low level register writing over SPI, writes one 8-bit value

Parameters

- **register** (*int*) – which device register to write
- **value** – a byte to write

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index