AHRS Library Documentation

Release 1.0

Mark Komus

Dec 07, 2020

Contents

AHRS library for CircuitPython

This library contains right now one alogrithm for AHRS - Attitude and Heading Reference System. It is used to combine multiple sensor values to give a heading, pitch and roll value such as used by aircraft.

See the Wiki for a more complete description.

CHAPTER 1

Dependencies

This library depends on:

- Adafruit CircuitPython

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Usage Example

Create the filter, set the parameters and start feeding it sensor data

```
filter = mahony.Mahony(Kp, Ki, sample_frequency)

while True:
        filter.update(gx, gy, gz, ax, ay, az, mx, my, mz)
```

CHAPTER 3

Caution

The calculations are very processor intensive. I have tested this on an Adafruit Feather M4 Express. Mahony was able to do about 300 samples/sec Madgwick was only able to about 15 samples/sec

Also be careful which values you feed the filter and the orientation of your sensor. I turned the gryoscope/accelerometer off to make sure magnetic fields were correct and then turned on only the gyroscope/accelerometer to ensure they were correct.

CHAPTER 4

Contributing

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

CHAPTER 5

Documentation

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/mahony_simpletest.py

```python
##
## This test file has calibration values for my device
## Anyone else will have to calibrate and set their own values
##
## Only calibrates for the gryo and hardiron offsets
## Set to run on the Adafruit LSM9DS1 over I2C cause that is what I have
##

import time
import board
import busio
import adafruit_lsm9ds1
import mahony

MAG_MIN = [-0.5764, 0.0097, -0.5362]
MAG_MAX = [0.4725, 0.9919, 0.4743]

## Used to calibrate the magenetic sensor
def map_range(x, in_min, in_max, out_min, out_max):
    """
    Maps a number from one range to another.
    :return: Returns value mapped to new range
    :rtype: float
    """
    mapped = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
    if out_min <= out_max:
        return max(min(mapped, out_max), out_min)
```

(continues on next page)

(continued from previous page)

```

    return min(max(mapped, out_max), out_min)


## create the ahrs_filter
ahrs_filter = mahony.Mahony(50, 5, 100)

# create the sensor
i2c = busio.I2C(board.A3, board.A2)
sensor = adafruit_lsm9ds1.LSM9DS1_I2C(i2c)

count = 0  # used to count how often we are feeding the ahrs_filter
lastPrint = time.monotonic()  # last time we printed the yaw/pitch/roll values
timestamp = time.monotonic_ns()  # used to tune the frequency to approx 100 Hz

while True:
    # on an Feather M4 approx time to wait between readings
    if (time.monotonic_ns() - timestamp) > 6500000:

        # read the magenetic sensor
        mx, my, mz = sensor.magnetic

        # adjust for magnetic calibration - hardiron only
        # calibration varies per device and physical location
        mx = map_range(mx, MAG_MIN[0], MAG_MAX[0], -1, 1)
        my = map_range(my, MAG_MIN[1], MAG_MAX[1], -1, 1)
        mz = map_range(mz, MAG_MIN[2], MAG_MAX[2], -1, 1)

        # read the gyroscope
        gx, gy, gz = sensor.gyro
        # adjust for my gyro calibration values
        # calibration varies per device and physical location
        gx -= 1.1250
        gy -= 3.8732
        gz += 1.2834

        # read the accelerometer
        ax, ay, az = sensor.acceleration

        # update the ahrs_filter with the values
        # gz and my are negative based on my installation
        ahrs_filter.update(gx, gy, -gz, ax, ay, az, mx, -my, mz)

        count += 1
        timestamp = time.monotonic_ns()

    # every 0.1 seconds print the ahrs_filter values
    if time.monotonic() > lastPrint + 0.1:
        # ahrs_filter values are in radians/sec multiply by 57.20578 to get degrees/sec
        yaw = ahrs_filter.yaw * 57.20578
        if yaw < 0:  # adjust yaw to be between 0 and 360
            yaw += 360
        print(
            "Orientation: ",
            yaw,
            ", ",
```

(continues on next page)

(continued from previous page)

```
            ahrs_filter.pitch * 57.29578,
            ", ",
            ahrs_filter.roll * 57.29578,
        )
        print(
            "Quaternion: ",
            ahrs_filter.q0,
            ", ",
            ahrs_filter.q1,
            ", ",
            ahrs_filter.q2,
            ", ",
            ahrs_filter.q3,
        )

        # print("Count: ", count)    # optionally print out frequency
        count = 0  # reset count
        lastPrint = time.monotonic()
```

6.2 `mahony`

AHRS library for CircuitPython Mahony Algorithm

Madgwick's implementation of Mayhony's AHRS algorithm. See: http:##www.x-io.co.uk/open-source-imu-and-ahrs-algorithms/

From the x-io website "Open-source resources available on this website are provided under the GNU General Public Licence unless an alternative licence is provided in source."

Original Information Date Author Notes 29/09/2011 SOH Madgwick Initial release 02/10/2011 SOH Madgwick Optimised for reduced CPU load Algorithm paper: http:##ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4608934&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%

This version based upon AdaFruit AHRS https://github.com/adafruit/Adafruit_AHRS

- Author(s): Mark Komus

6.2.1 Implementation Notes

Hardware:

Any 9DOF sensor

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

class `gamblor21_ahrs.mahony.`**`Mahony`**(*Kp=0.5*, *Ki=0.0*, *sample_freq=100*)

AHRS Mahony algorithm.

Ki

The current Ki value (Integral gain).

Kp

The current Kp value (Proportional gain).

compute_angles()

Compute all the angles if there have been new samples (internal use)

pitch

Current pitch (y-axis) value in radians/sec. (read-only)

roll

Current roll (x-axis) value in radians/sec. (read-only)

sample_freq

The current sample frequency value in Hertz.

update(*gx*, *gy*, *gz*, *ax*, *ay*, *az*, *mx*, *my*, *mz*)

Call this function sample_freq times a second with values from your sensor The units of the accelerometer and magnetometer do not matter for this alogirthm The gryoscope must be in degrees/sec

Parameters

- **gy, gz** (*gx,*) – Gyroscope values in degrees/sec
- **ay, az** (*ax,*) – Accelerometer values
- **my, mz** (*mx,*) – Magnetometer values

update_IMU(*gx*, *gy*, *gz*, *ax*, *ay*, *az*)

Called is was have no mag reading (internal use)

yaw

Current yaw (z-axis) value in radians/sec. (read-only)

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index