2020/03/27 Discussion

0616014 楊政道

April 3, 2020

1 Discussion 1

1.1 Based on the requirements, set correct value in the sample code

- Output data rate: 100Hz
- ± 2g
- Convert LSB to G (by using SCALE_MULTIPLIER)

```
# set value
ADXL345_SCALE_MULTIPLIER= ???
ADXL345_BW_RATE_100HZ    = 0x???
ADXL345_MEASURE          = 0x???
```

1.1.1 ADXL345_SCALE_MULTIPLIER

The value of the `ADXL345_SCALE_MULTIPLIER` is `1/SENSITIVITY`. According to the datasheet, the value of the sensitivity is `256 LSB/g`. Therefore, the value of the `ADXL345_SCALE_MULTIPLIER` is `1/256=0.00390625`.

Parameter	Test Conditions	Min	Typ[1]	Max	Unit
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	230	256	282	LSB/*g*
	±2 *g*, 10-bit resolution	230	256	282	LSB/*g*

1.1.2 ADXL345_BW_RATE_100HZ

The value of the `ADXL345_BW_RATE_100Hz` is a 8-bit integer value. According to the datasheet, we can find the rate code is `1010` if we want the output data rate is `100Hz`. Therefore, the value of the `ADXL345_BW_RATE_100HZ` is `0x0A`.

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (μA)
3200	1600	1111	140
1600	800	1110	90
800	400	1101	140
400	200	1100	140
200	100	1011	140
100	50	1010	140
50	25	1001	90
25	12.5	1000	60

1.1.3 ADXL345_MEASURE

The value of the ADXL345_MEASURE is to set the ADXL345 into measure mode. According to the datasheet, we can set the POWER_CTL register into 00001000. Therefore, we will set the ADXL345 into measure mode.

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

1.2 Continuously measurement

```
    while True:
        getX()
        getY()
        getX()
        print value
        time.sleep(0.1)
```

The sample code can only output the value once. If we want to measure continuously, we can put the sample code into a while loop with a sleep statement.

1.3 Calibrate your sensor

Set all the offsets value into zero and output the value read from sensor.

```
pi@raspberrypi:~$ python3 1acc.py
x = -0.043 G,   y = -0.144 G,   z = 1.055 G
x = -0.043 G,   y = -0.144 G,   z = 1.055 G
x = -0.043 G,   y = -0.144 G,   z = 1.055 G
x = -0.043 G,   y = -0.144 G,   z = 1.055 G
x = -0.043 G,   y = -0.144 G,   z = 1.055 G
x = -0.043 G,   y = -0.144 G,   z = 1.055 G
```

The value read from the sensor should be (0, 0, 1) theoretically, so we need to set the offsets to make the value into correct ones.

```
        self.Xoffset = 0.043
        self.Yoffset = 0.144
        self.Zoffset = -0.055
```

After we set the offset value, the output will close to (0, 0, 1).

```
pi@raspberrypi:~$ python3 1acc.py
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
x = -0.001 G,  y = 0.001 G,  z = 1.000 G
```

2 Discussion 2

2.1 Based on the requirements, set correct value in the sample code

- Data rate: 100Hz, cut-off = 12.5
- Full Scale selection = 250 dps
- Set Sensitivity for 250 dps (by using SCALE_MULTIPLIER)

```
# set value
self.gain_std = ??     # dps/digit

self.write_byte(L3G4200D_CTRL_REG1, 0x??)
self.write_byte(L3G4200D_CTRL_REG4, 0x??)
```

2.1.1 gain_std

Since the sensitivity is 8.75 mdps/digit, the value of the `gain_std` will be `8.75`$\times 10^{-3}$ `= 0.00875`.

2.1.2 L3G4200D_CTRL_REG1

Table 20. CTRL_REG1 register

DR1	DR0	BW1	BW0	PD	Zen	Yen	Xen

Table 2. Operating mode selection

Operating mode	PD	Zen	Yen	Xen
Power down	0	-	-	-
Sleep	1	0	0	0
Normal mode	1	-	-	-

Table 3. Data rate configuration

DR <1:0>	BW <1:0>	ODR [Hz]	Cut-off LPF1 [Hz]	Cut-off LPF2 [Hz]
00	00	100		12.5
00	01	100	32	25
00	10	100		25
00	11	100		25
01	00	200	54	12.5
01	01	200		25
01	10	200		50
01	11	200		70
10	00	400	78	20
10	01	400		25
10	10	400		50
10	11	400		110
11	00	800	93	30
11	01	800		35
11	10	800		50
1	11	800		110

According to the datasheet, we need to set `L3G4200D_CTRL_REG1` into 0x0F so as to make data rate 100Hz, cut-off rate 12.5 and the sensor into normal mode.

2.1.3 L3G4200D_CTRL_REG4

CTRL_REG4 (23h)

Table 30. CTRL_REG4 register

BDU	BLE	FS1	FS0	-	ST1	ST0	SIM

Table 31. CTRL_REG4 description

BDU	Block Data Update. Default value: 0 (0: continous update; 1: output registers not updated until MSB and LSB reading)
BLE	Big/Little Endian Data Selection. Default value 0. (0: Data LSB @ lower address; 1: Data MSB @ lower address)
FS1-FS0	Full Scale selection. Default value: 00 (00: 250 dps; 01: 500 dps; 10: 2000 dps; 11: 2000 dps)
ST1-ST0	Self Test Enable. Default value: 00 (00: Self Test Disabled; Other: See *Table*)
SIM	SPI Serial Interface Mode selection. Default value: 0 (0: 4-wire interface; 1: 3-wire interface).

According to the datasheet, we need to set all the bits, except the reserved one, into zero. Therefore, we can match the requirements mentioned above. (There is a typo in the equation at page `36` of the slide, `0x80 = 0000 1000`. It might be 0x08).

2.2 Continuously measurement

```
while True:
    getX()
    getY()
    getX()
    print value
    time.sleep(0.1)
```

The sample code can only output the value once. If we want to measure continuously, we can put the sample code into a while loop with a sleep statement.

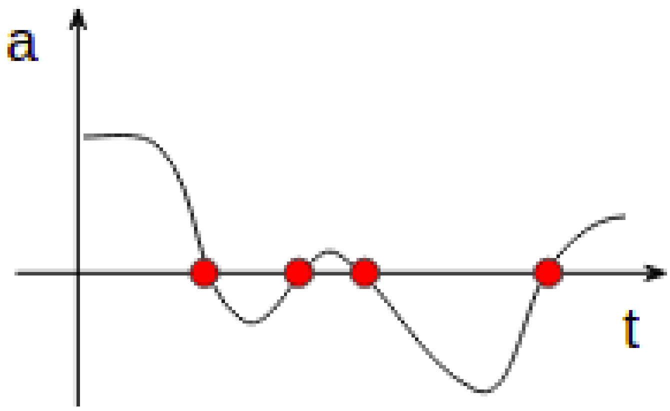
3 Discussion 3

3.1 We can obtain ax/ay/az from accelerometer. How to calculate the distance?

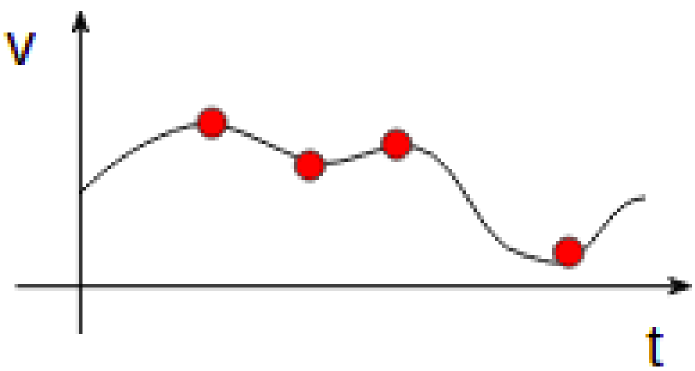
First, we can calculate the distance in 3 dimensions separetely, and then combine this 3 data into a real distance by the below equation.

$$D = \sqrt{{D_x}^2 + {D_y}^2 + {D_z}^2}$$

We can draw the a-t plot first.



Then, we can construct the v-t plot. The value `v(t)` at `t` on the v-t plot is the area from zero to `t` on the a-t plot, as known as integration.



Therefore, if we want to calculate the distance, we can calculate the area on the v-t plot. The equation wil be like below one.

$$D_x = \int_0^t (\int_0^t a_x(t)dt) + v(0)dt$$

Similar to the D_y and D_z

$$D_y = \int_0^t (\int_0^t a_y(t)dt) + v(0)dt$$

$$D_z = \int_0^t (\int_0^t a_z(t)dt) + v(0)dt$$

For the implementation, we will split the area into many pieces. For each piece, we can treat it as a uniform accelerated motion and use the formula below to calculate the distance.

$$s = v_0 t + \frac{1}{2}at^2$$

So the result will be

$$D_x = \sum_{i=0}^{n-1} v_x(\frac{it}{n})\frac{t}{n} + \frac{1}{2}a_x(\frac{it}{n})(\frac{t}{n})^2$$

Similar to the D_y and D_z

$$D_y = \sum_{i=0}^{n-1} v_y(\frac{it}{n})\frac{t}{n} + \frac{1}{2}a_y(\frac{it}{n})(\frac{t}{n})^2$$

$$D_z = \sum_{i=0}^{n-1} v_z(\frac{it}{n})\frac{t}{n} + \frac{1}{2}a_z(\frac{it}{n})(\frac{t}{n})^2$$