組込システムⅠ
第９回 課題

提出日 2025/06/19

学籍番号 21T2166D
名前 渡辺 大樹

1　演習 (1)

1.1　課題

2 つの温度センサ、アナログの LM35DZ とデジタルの ADT7410 を同時に接続し、それぞれの計測値を摂氏 (°C) 単位で比較する。アナログセンサは A/D コンバータ MCP3002 を介して SPI 通信で、デジタルセンサは I2C 通信で読み取る。

1.2　使用部品

- Raspberry Pi 4B
- アナログ温度センサ (LM35DZ)
- デジタル温度センサ (ADT7410)
- A/D コンバータ (MCP3002)
- ブレッドボード
- ジャンパー線

1.3　回路図と回路写真

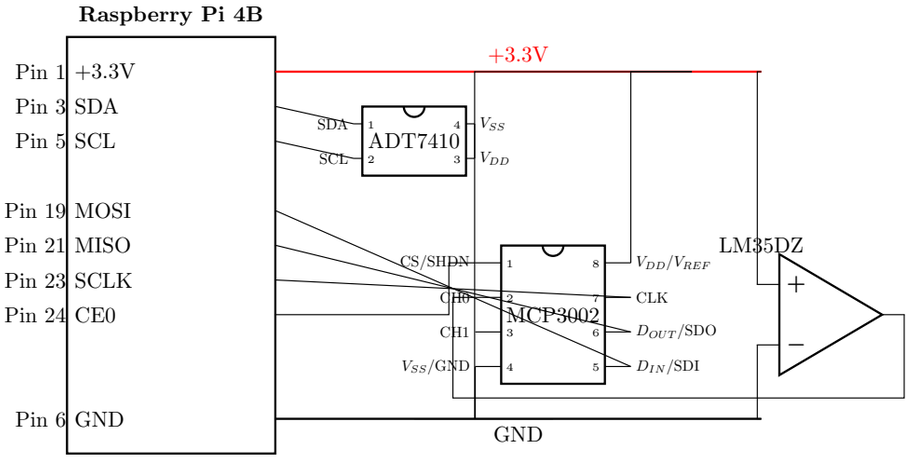

図 1　演習 (1) の回路図

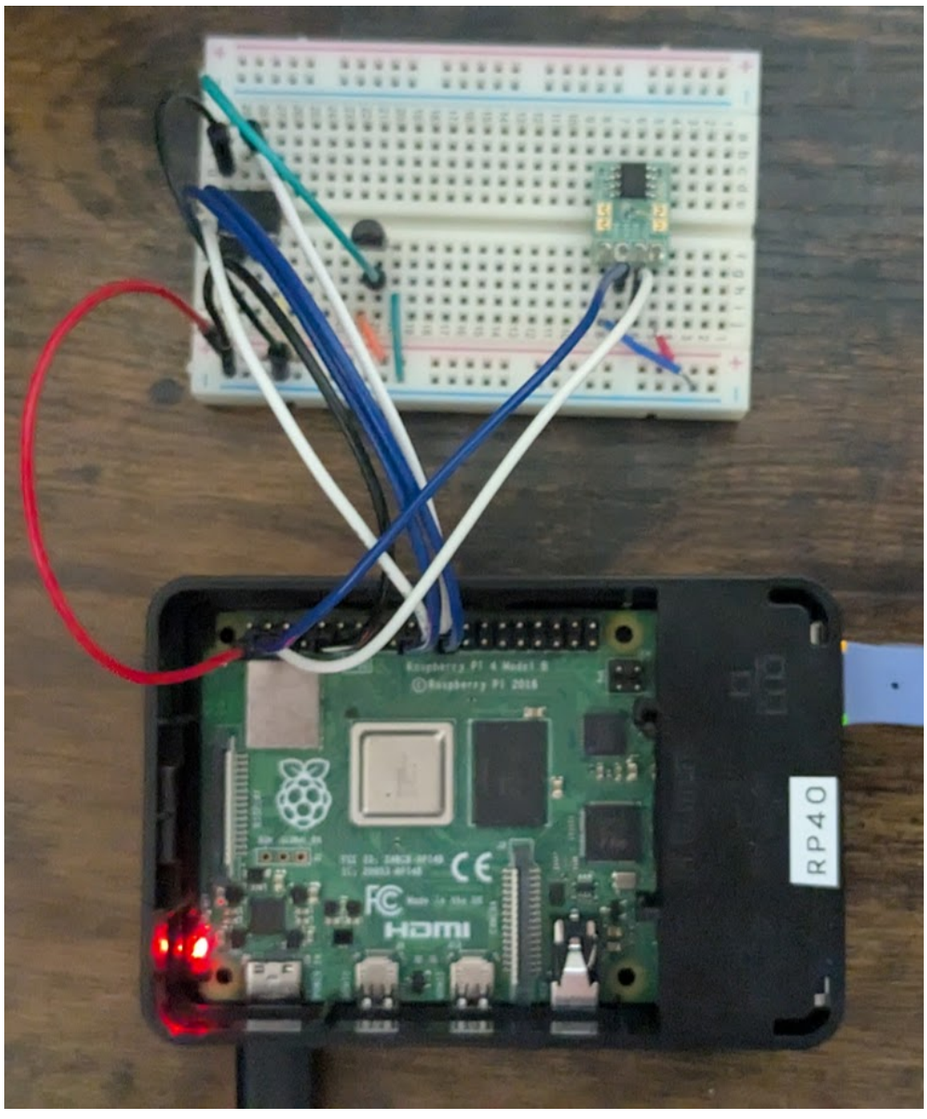

図 2　演習 (1) の回路写真

1.4　アルゴリズムの説明

1. I2C バスと SPI バスをそれぞれ初期化する。ADT7410 のアドレスは 0x48、MCP3002 は SPI デバイス 0(CE0) に設定する。
2. ‘read_adt7410‘関数で I2C 通信を行い、ADT7410 から温度データを読み取る。2 バイトのデータを結合し、13bit の 2 の補数表現として解釈した後、分解能 0.0625 を乗じて温度に変換する。
3. ‘read_mcp9700a‘関数で SPI 通信を行い、MCP3002 の CH0 から A/D 変換値を取得する。得られた 10bit のデジタル値を電圧に変換し、温度係数 (10.0mV/° C) を用いて温度に変換する。LM35DZ は 0° C で 0V を出力するため、オフセットの計算は不要である。
4. メインループ内で 1 秒ごとに両方のセンサから温度を取得し、コンソールに表示する。

1.5　結果

プログラムを実行し、2 つの温度センサの値を異なる時間帯で計測した。

```
# 2025/06/19 18:20
# ADT7410 (Digital): 27.56 C, LM35DZ (Analog): 27.42 C
# ADT7410 (Digital): 27.50 C, LM35DZ (Analog): 27.10 C
```

```
# ADT7410 (Digital): 28.12 C, LM35DZ (Analog): 27.10 C
# ADT7410 (Digital): 27.94 C, LM35DZ (Analog): 27.42 C
# ADT7410 (Digital): 29.00 C, LM35DZ (Analog): 27.42 C

# 2025/06/19 19:20
# ADT7410 (Digital): 27.56 C, LM35DZ (Analog): 26.45 C
# ADT7410 (Digital): 26.94 C, LM35DZ (Analog): 26.45 C
# ADT7410 (Digital): 27.50 C, LM35DZ (Analog): 26.45 C
# ADT7410 (Digital): 27.50 C, LM35DZ (Analog): 26.45 C
# ADT7410 (Digital): 27.06 C, LM35DZ (Analog): 26.13 C
# ADT7410 (Digital): 26.75 C, LM35DZ (Analog): 26.45 C
```

1.6 考察

18:20 の計測では、指でセンサを温めたため、両方のセンサで温度上昇が観測された。ADT7410 の方が LM35DZ よりも値の変動が大きく、より敏感に反応しているように見える。19:20 の計測では、室温が安定しているため両者の値は近くなったが、ADT7410 の方が若干高い値を示した。これは、ADT7410 の精度が± 0.5° C であるのに対し、LM35DZ は代表的な精度が± 1° C 程度であるためと考えられる。デジタルセンサはノイズに強く、より正確な値を提供する傾向があることが確認できた。

2 演習 (2)

2.1 課題

SPI 通信のチップセレクト機能を利用して、2 つの SPI デバイス (A/D コンバータ MCP3002 と 3 軸加速度センサ LIS3DH) を同時に制御する。温度と加速度の値を同時に取得し、表示する。

2.2 使用部品

- Raspberry Pi 4B
- アナログ温度センサ (LM35DZ)
- 3 軸加速度センサ (LIS3DH)
- A/D コンバータ (MCP3002)
- ブレッドボード
- ジャンパー線

2.3　回路図と回路写真

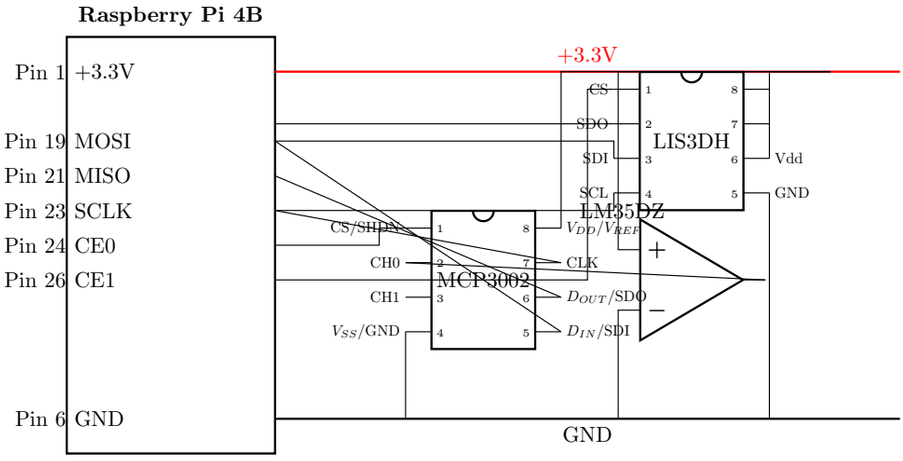

図 3　演習 (2) の回路図

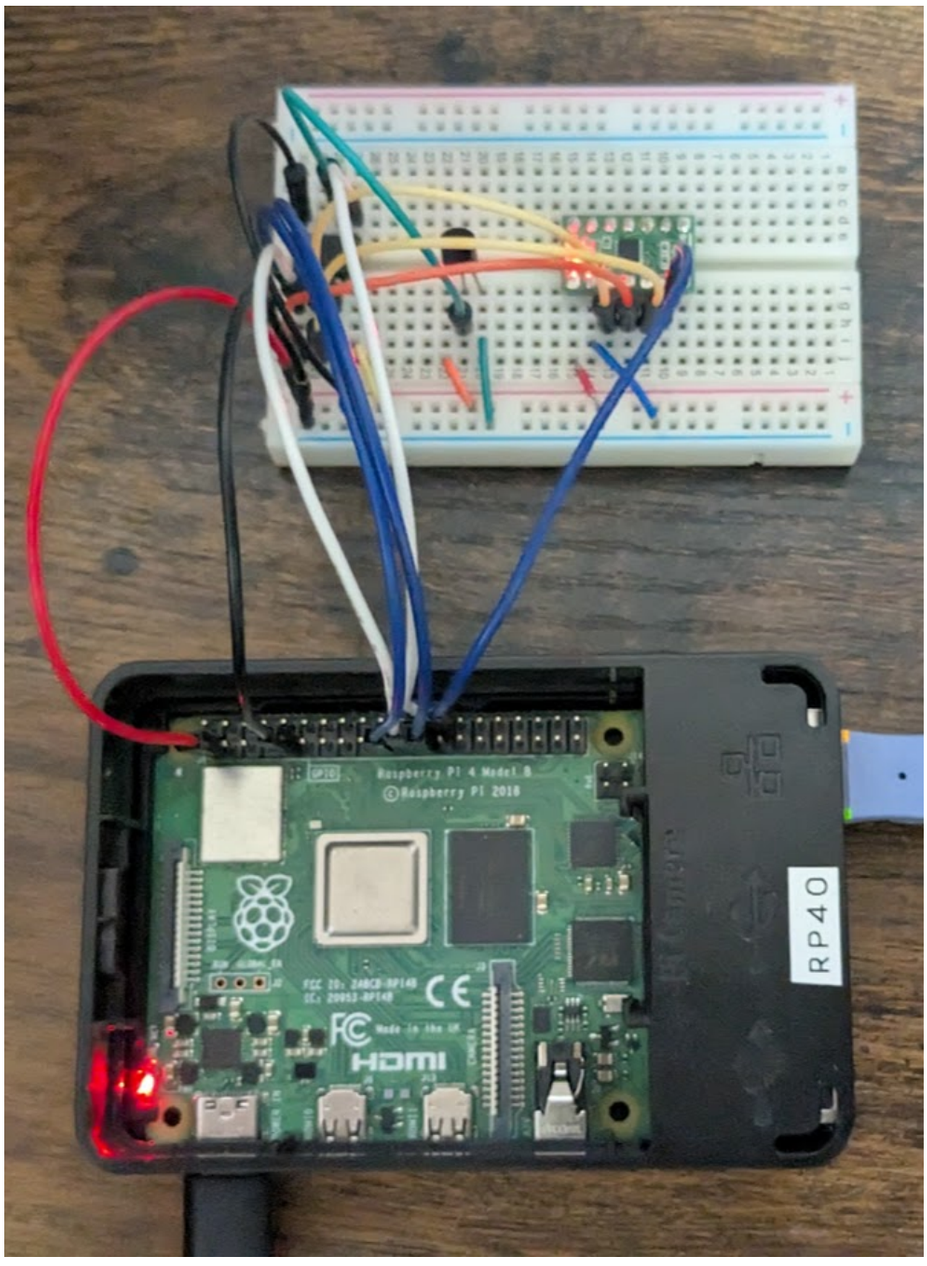

図 4　演習 (2) の回路写真

2.4　アルゴリズムの説明

1. ‘spidev‘のインスタンスを 2 つ生成し、それぞれデバイス 0(CE0) とデバイス 1(CE1) に割り当てる。
2. ‘setup_lis3dh‘関数で、LIS3DH の CTRL_REG1(0x20) に 0x27 を書き込み、データレート 10Hz、XYZ 軸有効のノーマルモードに設定する。
3. ‘read_lis3dh_axis_g‘関数で、指定された軸のレジスタから 2 バイト読み取り、12bit の 2 の補数データに変換後、g 単位の加速度に変換する。
4. メインループ内で 10 秒ごとに、CE0 に接続された MCP3002 から温度を、CE1 に接続された LIS3DH から 3 軸の加速度を読み取り、コンソールに表示する。

2.5　結果

プログラムを実行し、温度と加速度を同時にモニタリングした。センサを静止させた状態では Z 軸が約 1g を示し、傾けると重力加速度が各軸に分散される様子が確認できた。

```
# Temp: 27.42 C | Accel X: 0.007, Y: 0.035, Z: 1.059
# Temp: 27.42 C | Accel X: 0.007, Y: 0.027, Z: 1.066
#（センサを X 軸方向に傾けた場合）
# Temp: 26.45 C | Accel X: -0.781, Y: 0.121, Z: 0.680
#（センサを Y 軸方向に傾けた場合）
# Temp: 26.77 C | Accel X: -0.050, Y: 1.047, Z: 0.058
```

2.6 考察

SPI バスを共有しながら、CE0 と CE1 を切り替えることで 2 つの異なるデバイスを独立して制御できることを確認した。‘spidev‘ライブラリがこの切り替えを抽象化しているため、プログラム上では異なるインスタンスを操作するだけで簡単に実現できた。これにより、少ないピン数で多くのデバイスを接続できる SPI の利点を実感した。加速度センサの値を g 単位に変換する計算では、データシートから分解能やデータ形式を正確に読み取ることの重要性を学んだ。

3 演習 (3)

3.1 課題

演習 (2) のシステムに LED を追加し、温度または加速度が設定した閾値を超えた場合に、LED を点滅させて異常を通知するシステムを構築する。

3.2 使用部品

- Raspberry Pi 4B
- アナログ温度センサ (LM35DZ)
- 3 軸加速度センサ (LIS3DH)
- A/D コンバータ (MCP3002)
- LED (赤色)
- 抵抗 (330 Ω)
- ブレッドボード
- ジャンパー線

3.3 回路図と回路写真

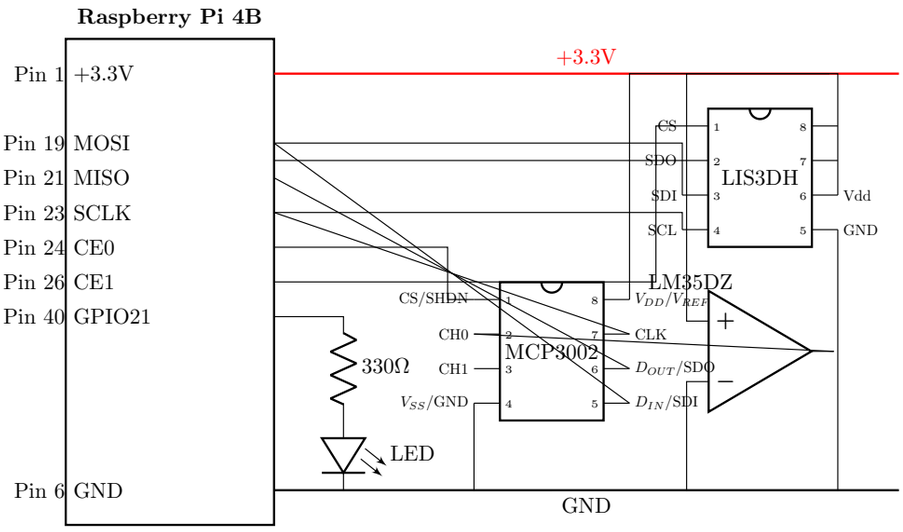

図 5　演習 (3) の回路図

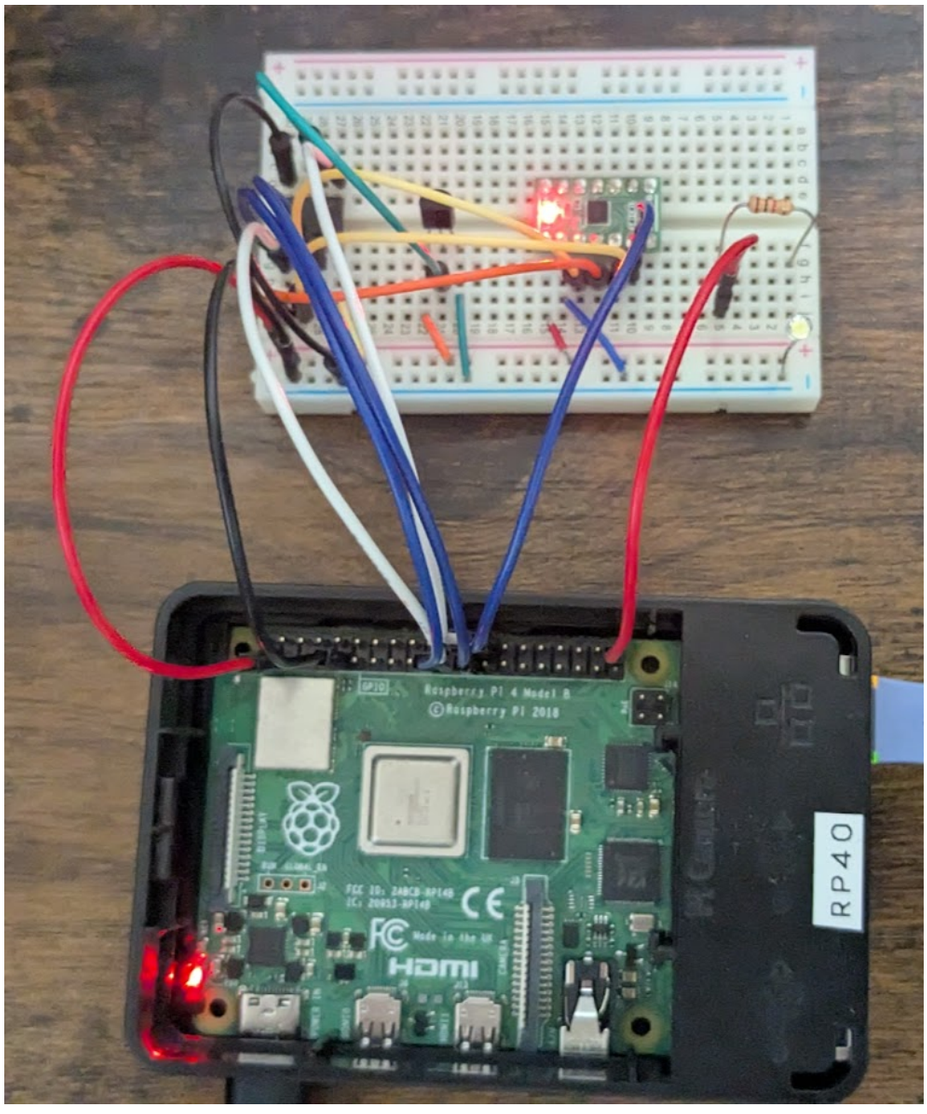

図 6　演習 (3) の回路写真

3.4 アルゴリズムの説明

1. 演習 (2) のコードをベースに、‘RPi.GPIO‘ライブラリを追加し、LED を接続した GPIO21 を出力モードに設定する。
2. 温度の閾値 (30.0° C) と加速度の閾値 (1.5g) を定数として定義する。
3. メインループの周期を 0.5 秒に短縮し、よりリアルタイムに異常を検知できるようにする。
4. 温度が閾値を超えた場合、「TEMPERATURE ALERT」と表示し、LED を 0.1 秒間隔で 5 回点滅させる。
5. 加速度のいずれかの軸の絶対値が閾値を超えた場合、「ACCELERATION ALERT」と表示し、LED を 0.3 秒間隔で 3 回点滅させる。
6. 正常時は LED を消灯状態に保つ。
7. KeyboardInterrupt 発生時に、GPIO と SPI のクリーンアップ処理を行う。

3.5 結果

プログラムを実行し、センサを手で温めたり、振ったりして異常状態を発生させた。

```
#（温度を 30 ℃以上に上げた場合）
# Temp: 30.32 C | Accel X: 0.003, Y: 0.042, Z: 1.066
!!! TEMPERATURE ALERT !!!
#（LED が速く 5 回点滅）

#（加速度を 1.5g 以上に上げた場合）
# Temp: 27.42 C | Accel X: 0.429, Y: 0.062, Z: 1.993
!!! ACCELERATION ALERT !!!
#（LED が遅く 3 回点滅）
```

結果として、温度と加速度の異常をそれぞれ異なる点滅パターンで正しく通知できることを確認した。

3.6 考察

本演習では、センサからの入力に応じて出力を制御する、組込システムの基本的なフィードバックループを実装した。センサの値に基づいて条件分岐を行い、GPIO を操作することで、単なるデータロガーから一歩進んだ実用的なアプリケーションを構築できた。異常の種類によって LED の点滅パターンを変えることで、ユーザーは視覚的にどのセンサが異常を検知したかを判断できる。これは、より高度な HMI(Human Machine Interface) の基礎となる考え方である。ポーリング周期を 0.5 秒に設定したが、これは検知のリアルタイム性と CPU 負荷のトレードオフであり、実際の製品開発では要求仕様に応じて最適な値を決定する必要があると感じた。

4 問いの解答

問い：演習（1）において、氷点下 10 度のときの AD 変換器の出力値（10bit）と、ADT7410 の出力値（13bit）を計算で求めなさい.

解答：

4.1 LM35DZ と MCP3002 (A/D 変換器)

LM35DZ の仕様より、温度係数は 10.0mV/° C であり、0° C での出力電圧は 0mV である。-10° C のときの出力電圧 V_{out} は、

$$V_{out} = 0\text{mV} + (10.0\text{mV/}^\circ\ \text{C} \times -10^\circ\ \text{C}) = -100\text{mV} = -0.1\text{V} \tag{1}$$

しかし、MCP3002 の入力電圧範囲は GND から Vdd(3.3V) まででであり、負の電圧を直接測定することはできない。したがって、回路が負電圧を扱えるように設計されていない限り、-10° C のときの入力電圧は GND レベル (0V) として読み取られる。

$$D_{ADC} = \frac{0\mathrm{V}}{3.3\mathrm{V}} \times 1023 = 0 \tag{2}$$

したがって、この回路構成における AD 変換器の出力値は **0** となる。

4.2 ADT7410 (デジタル温度センサ)

ADT7410 の仕様より、13bit モードでの温度分解能は 0.0625° C である。-10° C のときのデジタル出力値 D_{ADT} は、

$$D_{ADT} = \frac{-10^\circ\ \mathrm{C}}{0.0625^\circ\ \mathrm{C/LSB}} = -160 \tag{3}$$

この値は 13bit の 2 の補数で表現される。正の 160 は、13bit バイナリで ‘000 1010 0000‘ となる。-160 を求めるには、全ビットを反転して 1 を加える。

1. 全ビット反転: ‘111 0101 1111‘
2. 1 を加算: ‘111 0110 0000‘

したがって、ADT7410 の出力値は 16 進数で **0x1D80**、2 進数で **‘11101100000‘** となる。

5　付録：ソースコード

5.1　演習（1）のソースコード

ソースコード 1　exam9-1.py

```
1  import smbus
2  import spidev
3  import time
4
5  # --- ADT7410 (I2C) 設定 ---
6  i2c = smbus.SMBus(1)
7  adt7410_address = 0x48 # ADT7410 のスレーブアドレス
8
9  # --- MCP3002 (SPI) 設定 ---
10 spi = spidev.SpiDev()
11 spi.open(0, 0)
12 spi.bits_per_word = 8
13 spi.max_speed_hz = 10000
14
15 start = 0b01000000
16 sgl = 0b00100000
17 ch0 = 0b00000000
18 ch1 = 0b00010000
19 msbf = 0b00001000
20
21 def mcp3002(ch):
22     rcv = spi.xfer2([(start + sgl + ch + msbf ), 0x00 ] )
23     ad = ((( rcv[0] & 0x03 ) << 8 ) + rcv[1] )
24     return ad
25
26 def read_adt7410():
27     try:
28         word_data = i2c.read_word_data(adt7410_address, 0x00)
29         data = (word_data & 0xFF) << 8 | (word_data >> 8)
30         data = data >> 3
31
32         if data & 0x1000:
33             data -= 8192
34
35         temperature = data * 0.0625
36         return temperature
37     except IOError:
38         return None
39
40 def read_mcp9700a():
41     try:
42         data = mcp3002(ch0)
43
44         voltage = (data * 3.3) / 1023.0
45
46         temperature = (voltage / 0.01)
```

```
        return temperature
    except Exception as e:
        print(f"MCP9700A read error: {e}")
        return None

try:
    while True:
        temp_adt = read_adt7410()
        temp_mcp = read_mcp9700a()

        print(f"# ADT7410 (Digital): {temp_adt:.2f} C, MCP9700A (Analog): {temp_mcp:.2f} C")

        time.sleep(1)

except KeyboardInterrupt:
    print("\nProgram stopped.")
finally:
    spi.close()
    i2c.close()
```

5.2 演習（2）のソースコード

ソースコード 2　exam9-2.py

```
import spidev
import time

adc_spi = spidev.SpiDev()
adc_spi.open(0, 0)
adc_spi.max_speed_hz = 10000

lis_spi = spidev.SpiDev()
lis_spi.open(0, 1)
lis_spi.max_speed_hz = 10000

start = 0b01000000
sgl = 0b00100000
ch0 = 0b00000000
msbf = 0b00001000

CTRL_REG1 = 0x20
OUT_X_L = 0x28
OUT_Y_L = 0x2A
OUT_Z_L = 0x2C

def mcp3002(ch):
    rcv = adc_spi.xfer2([(start + sgl + ch + msbf), 0x00])
    ad = (((rcv[0] & 0x03) << 8) + rcv[1])
    return ad

def read_mcp9700a():
    try:
```

```
        data = mcp3002(ch0)
        voltage = (data * 3.3) / 1023.0

        temperature = (voltage / 0.01)
        return temperature
    except Exception as e:
        print(f"MCP9700A read error: {e}")
        return None

def setup_lis3dh():
    lis_spi.xfer2([CTRL_REG1, 0x27])

def read_lis3dh_axis_g(reg_l):
    reg_h = reg_l + 1
    low_byte = lis_spi.xfer2([reg_l | 0x80, 0x00])[1]
    high_byte = lis_spi.xfer2([reg_h | 0x80, 0x00])[1]

    value = (high_byte << 8) | low_byte

    if value > 32767:
        value -= 65536

    raw_value = value >> 4

    g_value = (raw_value / 2047.0) * 2.0
    return g_value

# --- メイン処理 ---
try:
    setup_lis3dh()

    while True:
        temp = read_mcp9700a()

        x_val = read_lis3dh_axis_g(OUT_X_L)
        y_val = read_lis3dh_axis_g(OUT_Y_L)
        z_val = read_lis3dh_axis_g(OUT_Z_L)

        if temp is not None:
            print(f"# Temp: {temp:.2f} C | Accel X: {x_val}, Y: {y_val}, Z: {z_val}")
        else:
            print("Failed to read temperature.")

        time.sleep(10)

except KeyboardInterrupt:
    print("\nProgram stopped.")
finally:
    adc_spi.close()
    lis_spi.close()
```

5.3 演習（3）のソースコード

ソースコード 3　exam9-3.py

```
import RPi.GPIO as GPIO
import spidev
import time

TEMP_THRESHOLD = 30.0
ACCEL_THRESHOLD = 1.5
LED_PIN = 21

GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

adc_spi = spidev.SpiDev()
adc_spi.open(0, 0)
adc_spi.max_speed_hz = 10000

lis_spi = spidev.SpiDev()
lis_spi.open(0, 1)
lis_spi.max_speed_hz = 10000

start = 0b01000000
sgl = 0b00100000
ch0 = 0b00000000
msbf = 0b00001000

CTRL_REG1 = 0x20
OUT_X_L = 0x28
OUT_Y_L = 0x2A
OUT_Z_L = 0x2C

def mcp3002(ch):
    rcv = adc_spi.xfer2([(start + sgl + ch + msbf), 0x00])
    ad = (((rcv[0] & 0x03) << 8) + rcv[1])
    return ad

def read_mcp9700a():
    try:
        data = mcp3002(ch0)
        voltage = (data * 3.3) / 1023.0

        temperature = (voltage / 0.01)
        return temperature
    except Exception as e:
        print(f"MCP9700A read error: {e}")
        return None

def setup_lis3dh():
    lis_spi.xfer2([CTRL_REG1, 0x27])

```

```
def read_lis3dh_axis_g(reg_l):
    reg_h = reg_l + 1
    low_byte = lis_spi.xfer2([reg_l | 0x80, 0x00])[1]
    high_byte = lis_spi.xfer2([reg_h | 0x80, 0x00])[1]

    value = (high_byte << 8) | low_byte

    if value > 32767:
        value -= 65536

    raw_value = value >> 4

    g_value = (raw_value / 2047.0) * 2.0
    return g_value

try:
    setup_lis3dh()

    while True:
        temp = read_mcp9700a()

        x_val = read_lis3dh_axis_g(OUT_X_L)
        y_val = read_lis3dh_axis_g(OUT_Y_L)
        z_val = read_lis3dh_axis_g(OUT_Z_L)

        if temp is not None:
            print(f"# Temp: {temp:.2f} C | Accel X: {x_val}, Y: {y_val}, Z: {z_val}")
            if temp is not None and temp > TEMP_THRESHOLD:
                print("\n!!! TEMPERATURE ALERT !!!")
                for _ in range(5):
                    GPIO.output(LED_PIN, GPIO.HIGH)
                    time.sleep(0.1)
                    GPIO.output(LED_PIN, GPIO.LOW)
                    time.sleep(0.1)
            elif abs(x_val) > ACCEL_THRESHOLD or abs(y_val) > ACCEL_THRESHOLD or abs(z_val) >
                ACCEL_THRESHOLD:
                print("\n!!! ACCELERATION ALERT !!!")
                for _ in range(3):
                    GPIO.output(LED_PIN, GPIO.HIGH)
                    time.sleep(0.3)
                    GPIO.output(LED_PIN, GPIO.LOW)
                    time.sleep(0.3)
            else:
                GPIO.output(LED_PIN, GPIO.LOW)
        else:
            print("Failed to read temperature.")

        time.sleep(0.5)

except KeyboardInterrupt:
    print("\nProgram stopped.")
finally:
    adc_spi.close()
```

```
101     lis_spi.close()
```