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

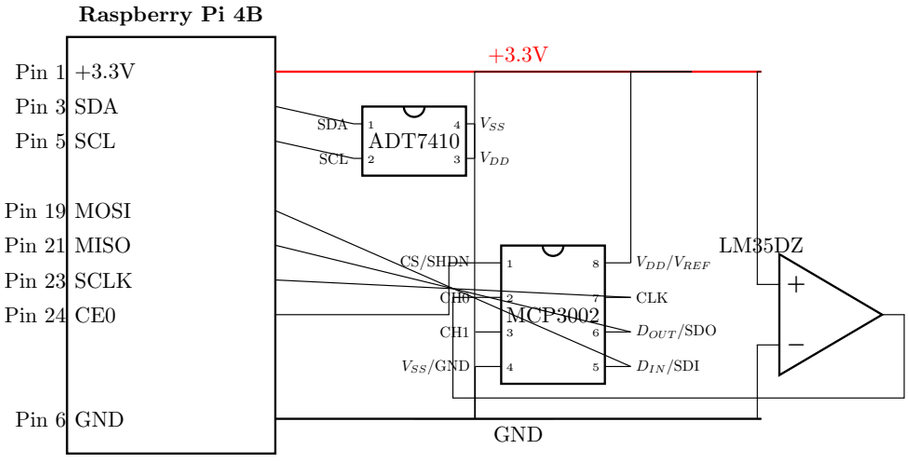

図 1　演習 (1) の回路図

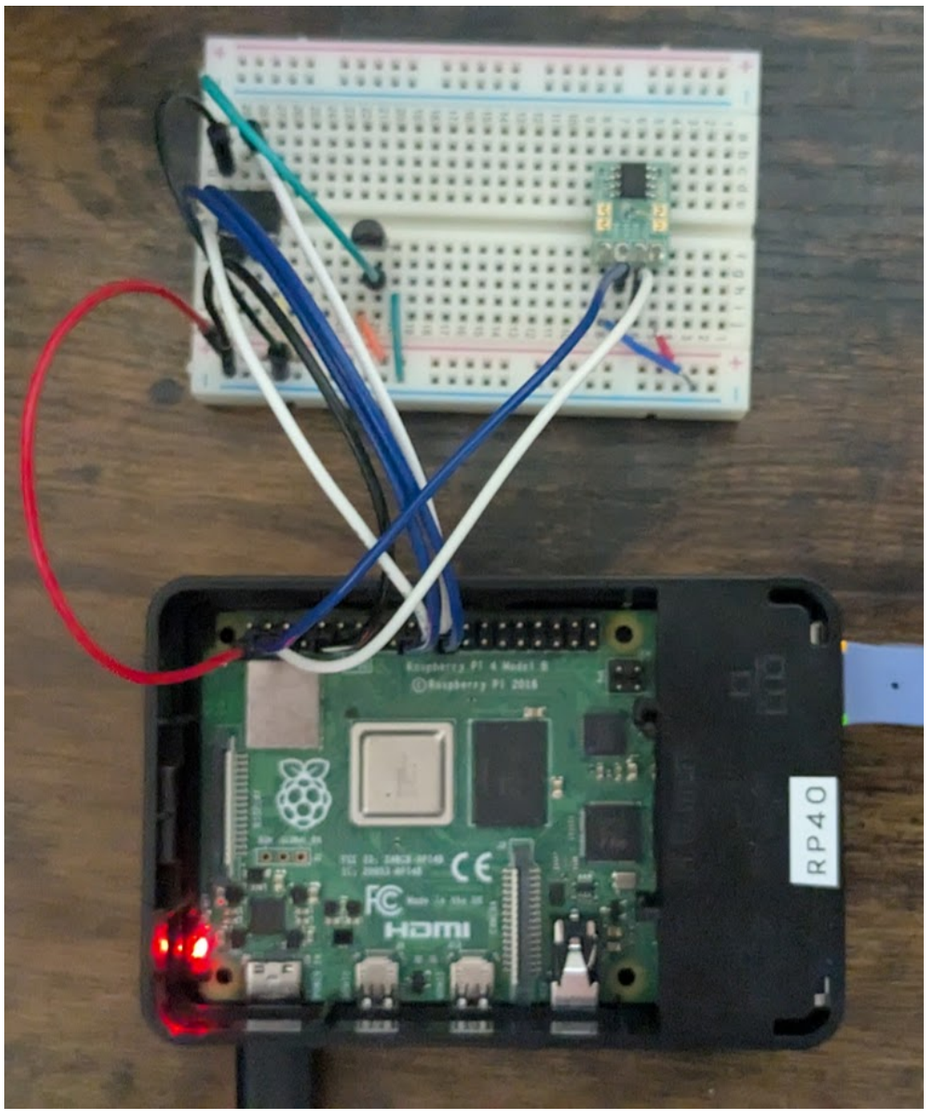

図 2　演習 (1) の回路写真

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

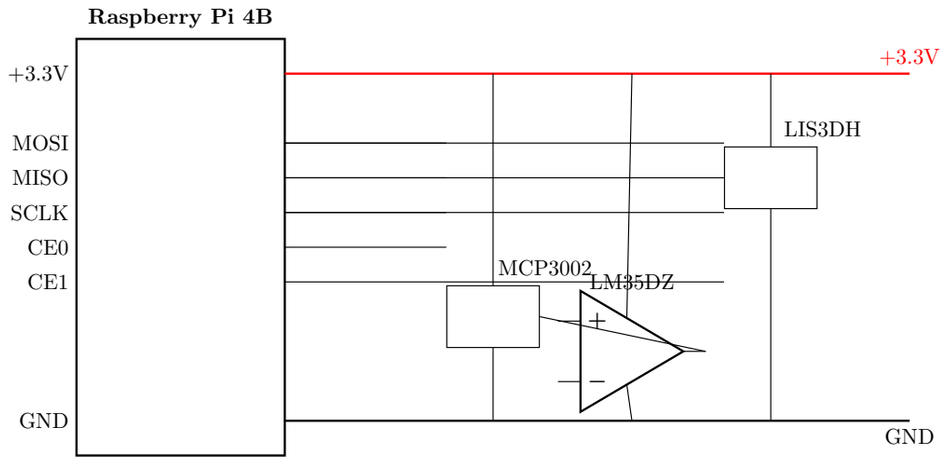

図 3　演習 (2) の回路図

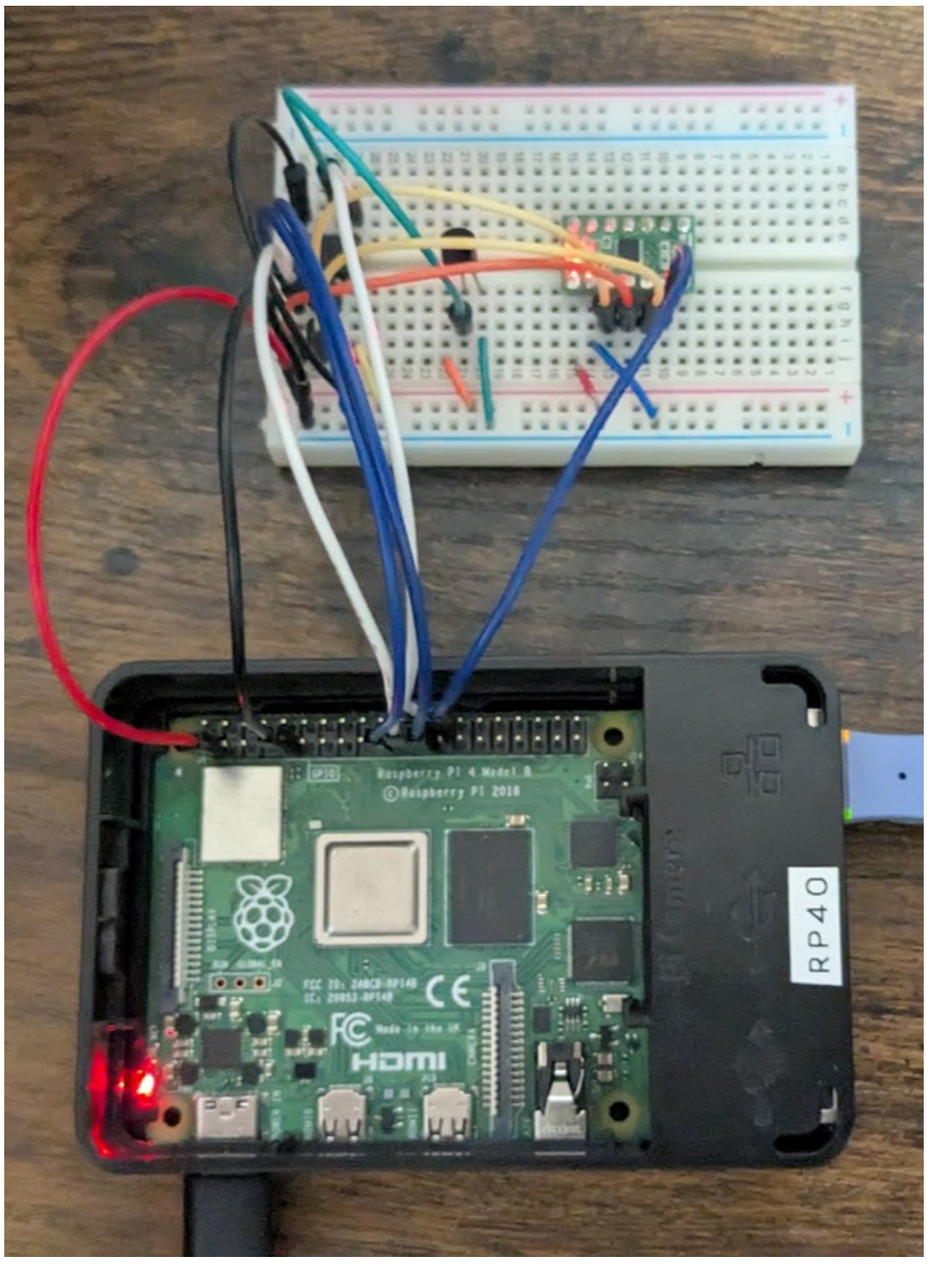

図 4　演習 (2) の回路写真

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

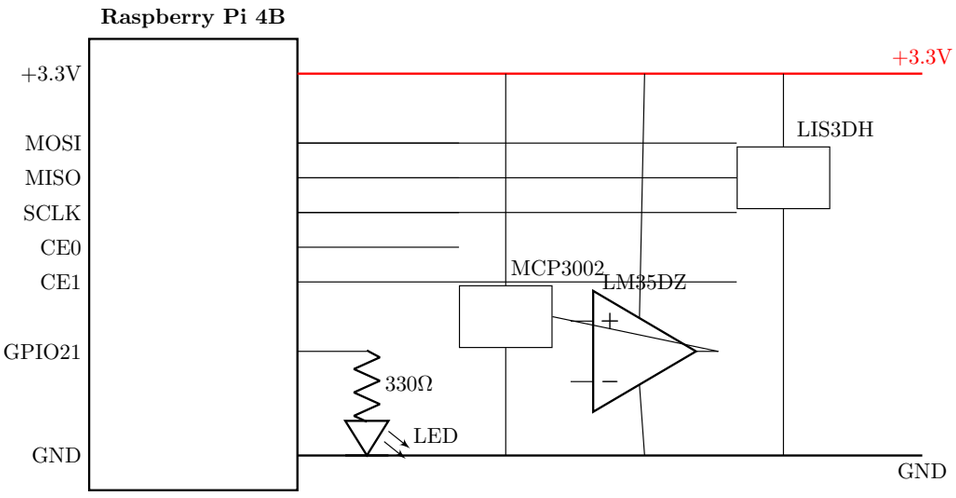

図 5　演習 (3) の回路図

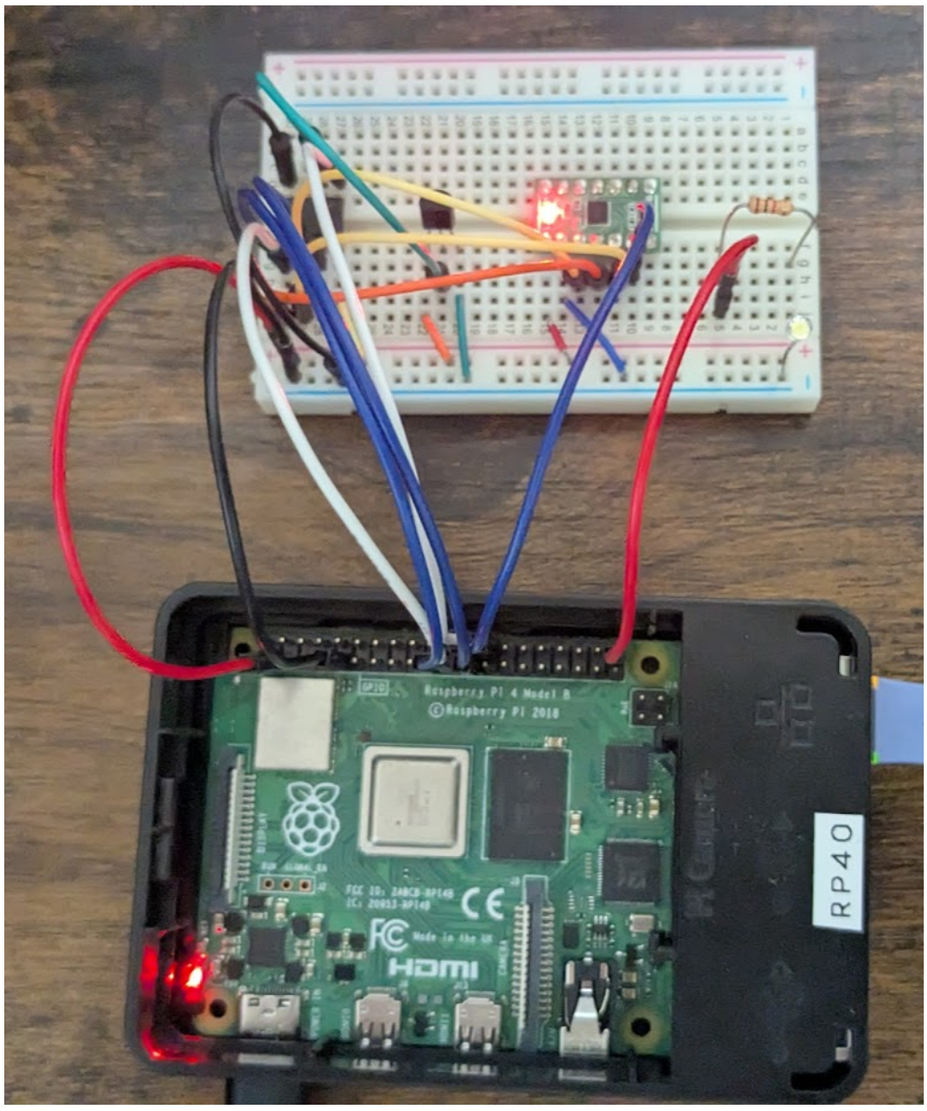

図 6　演習 (3) の回路写真

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

```
import smbus
import spidev
import time

# --- ADT7410 (I2C) 設定 ---
i2c = smbus.SMBus(1)
adt7410_address = 0x48 # ADT7410 のスレーブアドレス

# --- MCP3002 (SPI) 設定 ---
spi = spidev.SpiDev()
spi.open(0, 0)
spi.bits_per_word = 8
spi.max_speed_hz = 10000

start = 0b01000000
sgl = 0b00100000
ch0 = 0b00000000
ch1 = 0b00010000
msbf = 0b00001000

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

def read_adt7410():
    """ADT7410 から温度を読み取る関数"""
    try:
        # 上位バイトと下位バイトを読み込み
        word_data = i2c.read_word_data(adt7410_address, 0x00)
        # バイトオーダーをスワップ
        data = (word_data & 0xFF) << 8 | (word_data >> 8)
        # 13bit モードなので 3bit 右シフト
        data = data >> 3

        # 2の補数表現で負の値を判定
        if data & 0x1000: # 13bit 目で判定
            data -= 8192 # 2^13 = 8192

        # 温度に換算（分解能 0.0625℃）
        temperature = data * 0.0625
        return temperature
    except IOError:
        return None

def read_mcp9700a():
```

```
    """MCP9700A の値を MCP3002 経由で読み取り温度に変換する関数"""
    try:
        data = mcp3002(ch0)

        # デジタル値を電圧に変換（10bit, 3.3V）
        voltage = (data * 3.3) / 1023.0

        # 電圧を温度に変換（℃）
        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()

# 2025/06/19 18:20
# ADT7410 (Digital): 27.56 C, MCP9700A (Analog): 27.42 C
# ADT7410 (Digital): 27.50 C, MCP9700A (Analog): 27.10 C
# ADT7410 (Digital): 28.12 C, MCP9700A (Analog): 27.10 C
# ADT7410 (Digital): 27.94 C, MCP9700A (Analog): 27.42 C
# ADT7410 (Digital): 29.00 C, MCP9700A (Analog): 27.42 C

# 2025/06/19 19:20
# ADT7410 (Digital): 27.56 C, MCP9700A (Analog): 26.45 C
# ADT7410 (Digital): 26.94 C, MCP9700A (Analog): 26.45 C
# ADT7410 (Digital): 27.50 C, MCP9700A (Analog): 26.45 C
# ADT7410 (Digital): 27.50 C, MCP9700A (Analog): 26.45 C
# ADT7410 (Digital): 27.06 C, MCP9700A (Analog): 26.13 C
# ADT7410 (Digital): 26.75 C, MCP9700A (Analog): 26.45 C
```

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

# --- SPI デバイスの準備 ---
```

```
# デバイス 0 (CE0) for ADC (MCP3002)
adc_spi = spidev.SpiDev()
adc_spi.open(0, 0) # SPI バス 0, デバイス 0 (CE0)
adc_spi.max_speed_hz = 10000

# デバイス 1 (CE1) for Accelerometer (LIS3DH)
lis_spi = spidev.SpiDev()
lis_spi.open(0, 1) # SPI バス 0, デバイス 1 (CE1)
lis_spi.max_speed_hz = 10000

# --- MCP3002 制御用定数 ---
start = 0b01000000
sgl = 0b00100000
ch0 = 0b00000000
msbf = 0b00001000

# --- LIS3DH レジスタアドレス等 ---
CTRL_REG1 = 0x20
OUT_X_L = 0x28
OUT_Y_L = 0x2A
OUT_Z_L = 0x2C

def mcp3002(ch):
    """MCP3002 から指定チャンネルの値を読み取る (ADC 用 SPI を使用)"""
    rcv = adc_spi.xfer2([(start + sgl + ch + msbf), 0x00])
    ad = (((rcv[0] & 0x03) << 8) + rcv[1])
    return ad

def read_mcp9700a():
    """MCP9700A の値を MCP3002 経由で読み取り温度に変換する関数"""
    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():
    """LIS3DH を初期化する (加速度センサ用SPI を使用)"""
    # CTRL_REG1 (0x20) に 0x27 を書き込み
    # ODR=10Hz, Normal mode, X/Y/Z 軸有効
    lis_spi.xfer2([CTRL_REG1, 0x27])

def read_lis3dh_axis_g(reg_l):
    """LIS3DH の指定された軸のデータを読み取り、g 値に変換して返す"""
    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]

    # 2バイトを結合して 16bit のデータにする
```

```
58      value = (high_byte << 8) | low_byte
59
60      # 2の補数表現で負の値を判定
61      if value > 32767:
62          value -= 65536
63
64      # 12bit データなので 4bit 右シフトして raw 値を取得
65      raw_value = value >> 4
66
67      # raw 値を g 値に変換（± 2g モードの場合）
68      # 12bit の最大値(2047)が +2g に相当
69      g_value = (raw_value / 2047.0) * 2.0
70      return g_value
71
72  # --- メイン処理 ---
73  try:
74      setup_lis3dh()
75
76      while True:
77          temp = read_mcp9700a()
78
79          x_val = read_lis3dh_axis_g(OUT_X_L)
80          y_val = read_lis3dh_axis_g(OUT_Y_L)
81          z_val = read_lis3dh_axis_g(OUT_Z_L)
82
83          if temp is not None:
84              print(f"# Temp: {temp:.2f} C | Accel X: {x_val}, Y: {y_val}, Z: {z_val}")
85          else:
86              print("Failed to read temperature.")
87
88          time.sleep(10)
89
90  except KeyboardInterrupt:
91      print("\nProgram stopped.")
92  finally:
93      adc_spi.close()
94      lis_spi.close()
95
96  # Temp: 27.42 C | Accel X: 0.007816316560820713, Y: 0.03517342452369321, Z:
        1.0591108939912066
97  # Temp: 27.42 C | Accel X: 0.007816316560820713, Y: 0.027357107962872496, Z:
        1.0669272105520273
98  # Temp: 26.45 C | Accel X: -0.7816316560820713, Y: 0.12115290669272105, Z:
        0.680019540791402
99  # Temp: 26.77 C | Accel X: -0.9965803615046409, Y: 0.05471421592574499, Z:
        0.1250610649731314
100 # Temp: 26.77 C | Accel X: -0.050806057645334635, Y: 1.0473864191499755, Z:
        0.05862237420615535
101 # Temp: 26.45 C | Accel X: -0.0039081582804103565, Y: 1.051294577430386, Z:
        -0.03126526624328285
102 # Temp: 27.10 C | Accel X: 0.0039081582804103565, Y: 0.039081582804103565, Z:
        1.063019052271617
```

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

# --- 異常値の閾値とLED ピンを定義 ---
TEMP_THRESHOLD = 30.0 # 温度の異常値 (℃)
ACCEL_THRESHOLD = 1.5 # 加速度の異常値 (g)
LED_PIN = 21 # LED を接続する GPIO ピン

# --- GPIO のセットアップ ---
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

# --- SPI デバイスの準備 ---
# デバイス 0 (CE0) for ADC (MCP3002)
adc_spi = spidev.SpiDev()
adc_spi.open(0, 0) # SPI バス 0, デバイス 0 (CE0)
adc_spi.max_speed_hz = 10000

# デバイス 1 (CE1) for Accelerometer (LIS3DH)
lis_spi = spidev.SpiDev()
lis_spi.open(0, 1) # SPI バス 0, デバイス 1 (CE1)
lis_spi.max_speed_hz = 10000

# --- MCP3002 制御用定数 ---
start = 0b01000000
sgl = 0b00100000
ch0 = 0b00000000
msbf = 0b00001000

# --- LIS3DH レジスタアドレス等 ---
CTRL_REG1 = 0x20
OUT_X_L = 0x28
OUT_Y_L = 0x2A
OUT_Z_L = 0x2C

def mcp3002(ch):
    """MCP3002 から指定チャンネルの値を読み取る (ADC 用 SPI を使用)"""
    rcv = adc_spi.xfer2([(start + sgl + ch + msbf), 0x00])
    ad = (((rcv[0] & 0x03) << 8) + rcv[1])
    return ad

def read_mcp9700a():
    """MCP9700A の値を MCP3002 経由で読み取り温度に変換する関数"""
    try:
        data = mcp3002(ch0)
        voltage = (data * 3.3) / 1023.0

```

```
49          temperature = (voltage / 0.01)
50          return temperature
51      except Exception as e:
52          print(f"MCP9700A read error: {e}")
53          return None
54
55  def setup_lis3dh():
56      """LIS3DH を初期化する（加速度センサ用SPI を使用)"""
57      # CTRL_REG1 (0x20) に 0x27 を書き込み
58      # ODR=10Hz, Normal mode, X/Y/Z 軸有効
59      lis_spi.xfer2([CTRL_REG1, 0x27])
60
61  def read_lis3dh_axis_g(reg_l):
62      """LIS3DH の指定された軸のデータを読み取り、g 値に変換して返す"""
63      reg_h = reg_l + 1
64      low_byte = lis_spi.xfer2([reg_l | 0x80, 0x00])[1]
65      high_byte = lis_spi.xfer2([reg_h | 0x80, 0x00])[1]
66
67      # 2バイトを結合して 16bit のデータにする
68      value = (high_byte << 8) | low_byte
69
70      # 2の補数表現で負の値を判定
71      if value > 32767:
72          value -= 65536
73
74      # 12bit データなので 4bit 右シフトして raw 値を取得
75      raw_value = value >> 4
76
77      # raw 値を g 値に変換（± 2g モードの場合）
78      # 12bit の最大値(2047)が +2g に相当
79      g_value = (raw_value / 2047.0) * 2.0
80      return g_value
81
82
83  # --- メイン処理 ---
84  try:
85      setup_lis3dh()
86
87      while True:
88          temp = read_mcp9700a()
89
90          x_val = read_lis3dh_axis_g(OUT_X_L)
91          y_val = read_lis3dh_axis_g(OUT_Y_L)
92          z_val = read_lis3dh_axis_g(OUT_Z_L)
93
94          if temp is not None:
95              print(f"# Temp: {temp:.2f} C | Accel X: {x_val}, Y: {y_val}, Z: {z_val}")
96              if temp is not None and temp > TEMP_THRESHOLD:
97                  # 温度異常:速い点滅
98                  print("\n!!! TEMPERATURE ALERT !!!")
99                  for _ in range(5):
100                     GPIO.output(LED_PIN, GPIO.HIGH)
101                     time.sleep(0.1)
```

```
102                     GPIO.output(LED_PIN, GPIO.LOW)
103                     time.sleep(0.1)
104             elif abs(x_val) > ACCEL_THRESHOLD or abs(y_val) > ACCEL_THRESHOLD or abs(z_val) >
                    ACCEL_THRESHOLD:
105                 # 加速度異常：遅い点滅
106                 print("\n!!! ACCELERATION ALERT !!!")
107                 for _ in range(3):
108                     GPIO.output(LED_PIN, GPIO.HIGH)
109                     time.sleep(0.3)
110                     GPIO.output(LED_PIN, GPIO.LOW)
111                     time.sleep(0.3)
112             else:
113                 # 正常時はLED オフ
114                 GPIO.output(LED_PIN, GPIO.LOW)
115         else:
116             print("Failed to read temperature.")
117 
118         time.sleep(0.5)
119 
120 except KeyboardInterrupt:
121     print("\nProgram stopped.")
122 finally:
123     adc_spi.close()
124     lis_spi.close()
125 
126 # Temp: 27.10 C | Accel X: 0.01172447484123107, Y: 0.027357107962872496, Z:
        1.0669272105520273
127 # Temp: 27.10 C | Accel X: 0.0, Y: 0.039081582804103565, Z: 1.063019052271617
128 # Temp: 27.10 C | Accel X: 0.03517342452369321, Y: 0.03126526624328285, Z:
        1.1216414264777723
129 # Temp: 27.10 C | Accel X: -0.050806057645334635, Y: 0.06643869076697606, Z:
        1.0825598436736688
130 # Temp: 27.10 C | Accel X: -0.01172447484123107, Y: 0.019540791402051783, Z:
        1.051294577430386
131 # Temp: 27.10 C | Accel X: 0.007816316560820713, Y: 0.019540791402051783, Z:
        1.0591108939912066
132 # Temp: 27.10 C | Accel X: 0.007816316560820713, Y: 0.027357107962872496, Z:
        1.063019052271617
133 # Temp: 27.10 C | Accel X: 0.01172447484123107, Y: 0.05862237420615535, Z:
        1.0669272105520273
134 # Temp: 27.42 C | Accel X: 0.4298974108451392, Y: 0.0625305324865657, Z:
        1.9931607230092818
135 
136 # !!! ACCELERATION ALERT !!!
137 # Temp: 27.10 C | Accel X: 0.0039081582804103565, Y: 0.039081582804103565, Z:
        1.0317537860283341
138 # Temp: 27.10 C | Accel X: 0.0, Y: 0.027357107962872496, Z: 1.0552027357107963
139 # Temp: 27.10 C | Accel X: 0.0, Y: 0.03126526624328285, Z: 1.063019052271617
140 # Temp: 27.10 C | Accel X: -0.0039081582804103565, Y: 0.03126526624328285, Z:
        1.0473864191499755
141 # Temp: 27.10 C | Accel X: -0.007816316560820713, Y: 0.04298974108451392, Z:
        1.051294577430386
```

```
142 # Temp: 27.42 C | Accel X: 0.01172447484123107, Y: 0.027357107962872496, Z:
        1.0669272105520273
143 # Temp: 27.74 C | Accel X: 0.0039081582804103565, Y: 0.039081582804103565, Z:
        1.0786516853932584
144 # Temp: 28.39 C | Accel X: 0.0039081582804103565, Y: 0.03517342452369321, Z:
        1.0669272105520273
145 # Temp: 29.03 C | Accel X: 0.0039081582804103565, Y: 0.04689789936492428, Z:
        1.0591108939912066
146 # Temp: 29.35 C | Accel X: -0.0039081582804103565, Y: 0.03517342452369321, Z:
        1.0591108939912066
147 # Temp: 29.68 C | Accel X: 0.0, Y: 0.04689789936492428, Z: 1.0591108939912066
148 # Temp: 30.00 C | Accel X: 0.0039081582804103565, Y: 0.039081582804103565, Z:
        1.0669272105520273
149 # Temp: 30.32 C | Accel X: 0.0039081582804103565, Y: 0.04298974108451392, Z:
        1.0669272105520273
150
151 # !!! TEMPERATURE ALERT !!!
152 # Temp: 30.65 C | Accel X: 0.0039081582804103565, Y: 0.02344894968246214, Z:
        1.0552027357107963
153
154 # !!! TEMPERATURE ALERT !!!
155 # Temp: 30.32 C | Accel X: 0.0039081582804103565, Y: 0.03126526624328285, Z:
        1.074743527112848
```