デュアルウェア講習会課題 2

会津大学 講習会

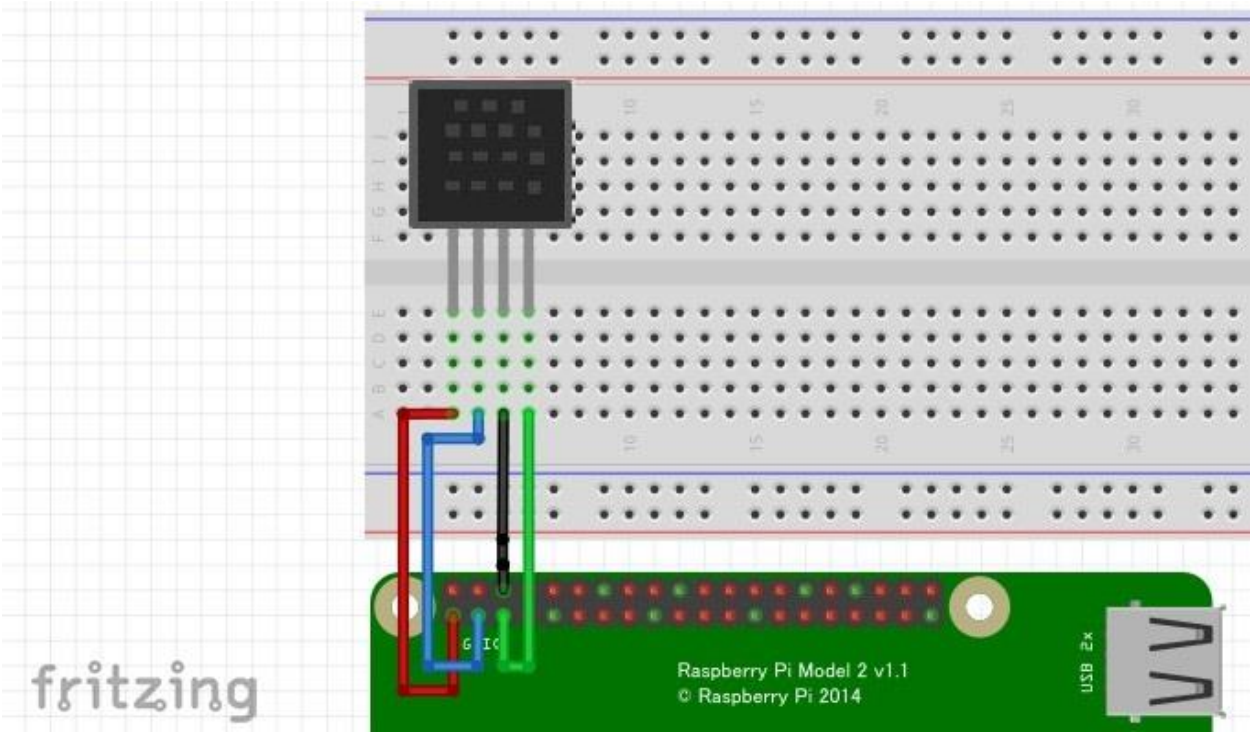
AM2320 を使い温度を取得する

目次

1 課題

1.1 課題説明

ブレッドボードに AM2320 を差し込み，Raspberry Pi と接続。温度を取得するプログラムを作成し温度を取得する。



2 機材説明

この課題は以下の機材を使用します。手元にあることを確認してください。

機材	個数
Raspberry Pi	1
ブレッドボード	1
AM2320	1
ジャンパー（オスーメス）	4

2.1 Raspberry Pi

課題 1 に記載

2.2 ブレッドボード

課題 1 に記載

2.3 AM2320

AM2320 は I2C で通信できる温度センサです。データシートの内容をまとめました。

データシート

http://www.kyohritsu.jp/eclib/OTHER/DATASHEET/SENSOR/am2320.pdf

特徴（表紙　Product Features を参照）

特徴

・超小型サイズ

・費用対効果が高い

・低電圧動作

・長期安定性に優れている

・標準 I2C およびシングルバス出力

通信方式が I2C であることを確認

電圧・電流（P.3 6. Electrical Characteristics を参照）

・P.3 Table 3: AM2320 DC Characteristics を確認

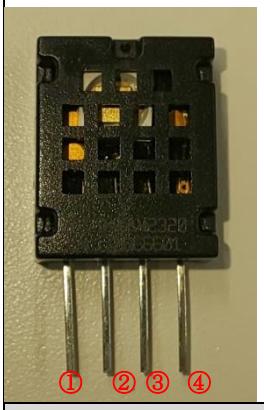
パラメータ	条件	最小	標準	最大	単位
電源電圧		3.1	5.0	5.5	V
消費電力	スリープ	8	10		μA

サンプリング期間		2.0			S

・電源電圧 が 3.1～5.5V

端子（P.2：ピン配置図を参照）

・P.2 4. Dimensions, P.7 7.1 AM2320 pin assignment 確認



No	端子名	説明
1	VDD	電力供給（3.1-5.5V）
2	SDA	I2C でデータの通信をするピン
3	GND	グランド
4	SCL	同期をとるためのピン

通信方式

特徴の所で I2C であることを確認

スレイブアドレス(P.8 Slave Address, P.10 8.2 AM2320 sensor I2C communication protocol を参照)

・P.8 Slave Address

デバイスアドレスは 7bit のスレイブアドレスと下位 1bit の Read/Write で構成される

・P.10 8.2 AM2320 sensor I2C communication protocol

デバイスアドレスは[0xB8]

・まとめ

下位 1bit は Write/Read を表す bit なので今回のプログラム上で使用するスレイブアドレスは[0xB8]の下位 1bit を消した値[0x5c]になる

16 進数	2 進数
0xB8	10111000
0x5C	1011100

※今回 Python プログラムで I2C を使う際はスレイブアドレスを使用しますが、別のプログラムの書き方ではデバイスアドレスを使用する場合があります。自分が何で使用するかで

レジスタアドレス(P.11 Communication I2C function code, P.11-12 C communication data area, P.12 Temperature output format 参照)

・P.11 Communication I2C function code

ファンクションコード	定義	操作

0x03	レジスタデータの読み取り	レジスタを読み込む
0x10	複数レジスタへの書き込み	複数のレジスタへデータを書き込み

センサへの動きを設定するためのコード。データを読み取るためには[0x03]を使用

・P.11-12 C communication data area,Temperature output format

レジスタ情報	アドレス	レジスタ情報	アドレス	レジスタ情報	アドレス
湿度上位 8 ビット	0x00	Retention	0x04	型番上位8ビット	0x08
湿度下位 8 ビット	0x01	Retention	0x05		
温度上位 8 ビット	0x02	Retention	0x06		
温度下位 8 ビット	0x03	Retention	0x07	Retention	0x1F

温度と湿度の値は上記のレジスタアドレスに格納されている。温度と湿度の値は 16bit の値になります。温度は 0bit～14bit までが温度を表し、16bit 目は正の値か負の値かを表します。1 の場合は負の値を表します。さらに最後は 10 で割る必要があります。

値の取得方法　(P.12 I2C Mod Bus Function Code Description 参照)

AM2320 から値を取得するにはホスト（Raspberry Pi）からセンサに読み取り指令を送ると温度の値が返答されます。

ホストからセンサへ読み取り指令を送るときのフォーマットは以下になります。

・送信フォーマット

通信項目	意味
センサアドレス	接続するスレイブアドレス
ファンクションコード	センサの動きを決めるコード
スタートレジスタ	読み取りを開始するレジスタアドレス
レジスタ数	読み込むレジスタの数

センサからホストに以下のフォーマットで返答します。

・返答フォーマット

通信項目	意味
センサアドレス	接続するスレイブアドレス
ファンクションコード	送信したときのファンクションコードを返します
レジスタ数	返答したレジスタ数。レジスタ数で決まります
返答のレジスタ	返答したレジスタ。送信のフォーマット時のレジスタ数によって数が変わる
CRC Code 下位 8bit	CRC Code の下位ビット
CRC Code 上位 8bit	CRC Code の上位ビット

温度の湿度を読み込むときは以下のようになります。

・送信

通信項目	値

センサアドレス	0x5c
ファンクションコード	0x03
スタートレジスタ	0x00
レジスタ数	0x04

・返答

通信項目	値
センサアドレス	0x5c
ファンクションコード	0x03
レジスタ数	0x04
返答のレジスタ 1	湿度の上位 8 ビットの値
返答のレジスタ 2	湿度の下位 8 ビットの値
返答のレジスタ 3	温度の上位 8 ビットの値
返答のレジスタ 4	温度の下位 8 ビットの値
CRC Code 下位 8bit	CRC Code の下位ビットの値
CRC Code 上位 8bit	CRC Code の上位ビットの値

実際にプログラムで読み取れるのは赤枠の値になります。

値の計算方法　(P.13 Numerical calculation 参照)

取得した上位 8bit と下位 8bit を足し 10 で割ります。

例）温度上位 8bit=0x01、温度下位 8bit=0x15

0x0115 = 1×256 +1×16 +5 = 277 => 温度 = 277÷10 = 27.7 ℃

手順　(P.16 I2C read and write timing decomposition 参照)

センサから値を取得するには以下の手順で行います。

1. センサを休止状態から起動
   センサは発熱からの誤差を抑えるため普段は動いいていない状態です。そのため使用する間にセンサを起動する指令を送る必要があります。スレイブアドレスを送信後、0.0008 秒～0.003 秒待ち次のステップに行きます。

2. 読み出し命令を送信
   センサを起動させましたら、読み取りたいデータの先頭レジスタアドレスとレジスタ長を送ります。読み取り指令を送信後、0.0015 秒待ち次のステップに行きます。

3. データの受信・値の計算
   データの読み取りをします。受信するデータは返答フォーマットの通りです。この中から実際に読み込むのはファンクションコード以降になります。読み込んだデータを計算して目的の値を求めます。

※データシートにはプルアップ抵抗を付ける様に記載されていますが、Raspberry Pi 内部にプルアップが入っているので今回は接続しません。

2.4 ジャンパー

課題 1 に記載

3 I2C

3.1 I2C(Inter Integrated Circuit; アイ スクエア シー)について

フィリップ社が開発した周辺デバイスとのシリアル通信の方式です。
SDA, SCL の 2 本信号線だけでデバイスを制御し、複数のデバイスに並列して接続し使用出来ます。

3.2 I2C の使い方

3.2.1 Python での使い方

・import
Python プログラムで I2C を使用するには[smbus]を import します。

```
import smbus # smbus をインポート
```

・宣言
バス番号を引数にして初期宣言をします。

```
i2c = sumbus.SMBus(1) #初期宣言 引数は BusNumber ([sudo i2cdetect 1]の 1)
```

今回の講習会では以下の関数を使用して値のやり取りを行います。
・read_byte(スレイブアドレス)
スレイブアドレスに書き込まれている値を読み取る

スレイブアドレス	接続している機器のアドレス

以下のように使用します。

```
 i2c.read_byte(0x5c)
```

・read_i2c_block_data(スレイブアドレス , 命令レジスタ番号, 1 つ以上の値)
命令レジスタ番号に書き込まれている値を読み取る

スレイブアドレス	接続している機器のアドレス
命令レジスタ番号	値が入っているレジスタの番号
1 つ以上の値	読み込むアドレス数

以下のように使用します。

```
value = i2c.read_i2c_block_data(0x5c, 0x00,0x04)
```

・write_i2c_block_data(スレイブアドレス , 命令レジスタ番号, 1 つ以上の値)
命令レジスタ番号を先頭に複数の命令レジスタ番号に値を書き込む

スレイブアドレス	接続している機器のアドレス

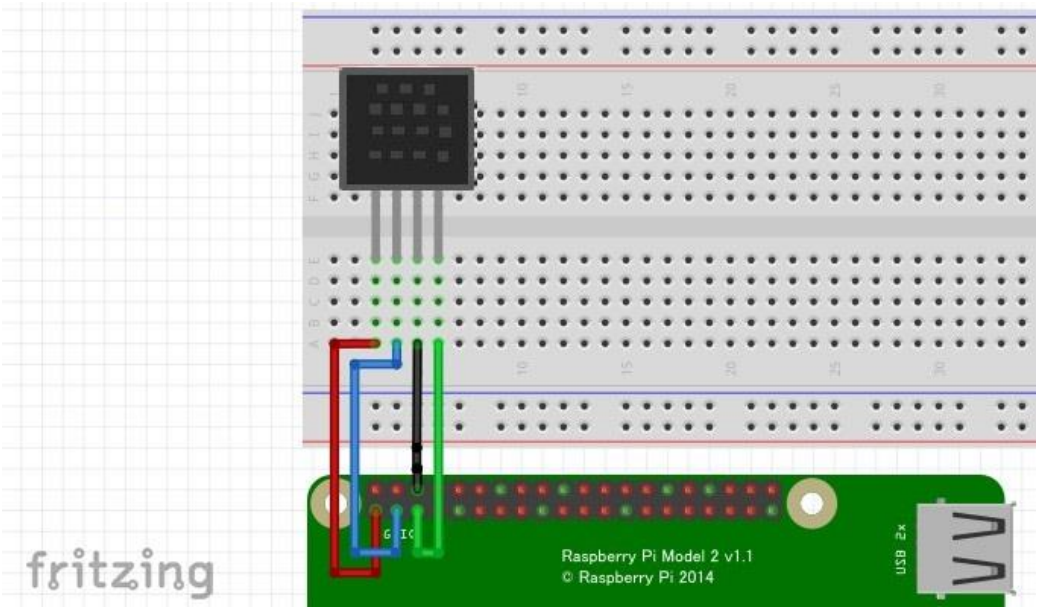
命令レジスタ番号	値が入っているレジスタの番号
1 つ以上の値	複数の値を書き込むことによって複数の命令レジスタに値を書き込める。

以下のように使用します。

```
i2c. write_i2c_block_data(0x5c, 0x03, [0x00,0x04])
```

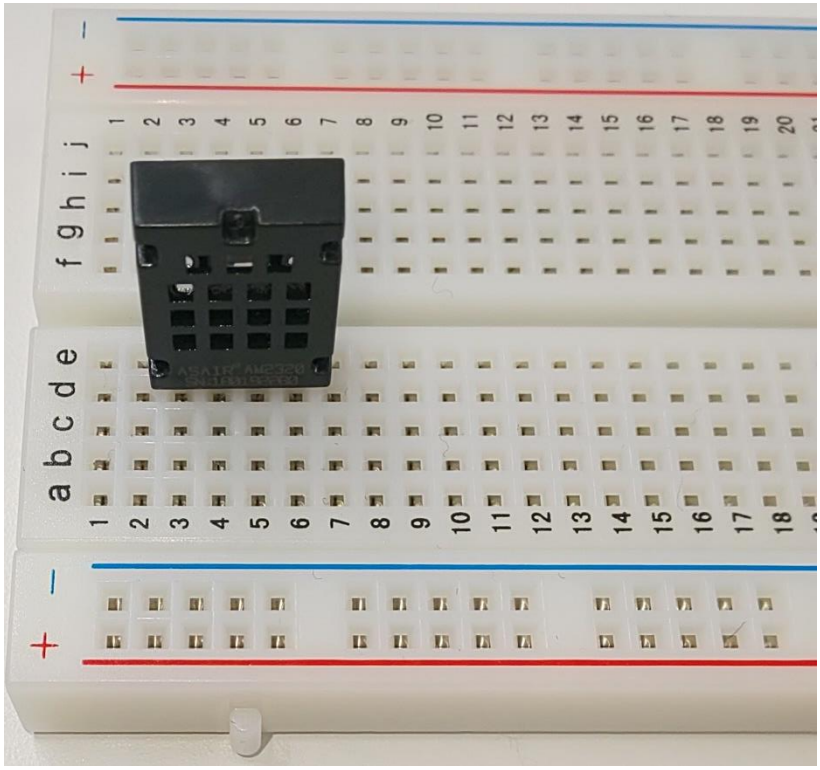
4 回路作成

ブレッドボードに AM2320 を差し回路を作成します。
完成イメージ図は以下になります。
※接続作業は必ず Raspberry Pi の電源を切ってから行ってください



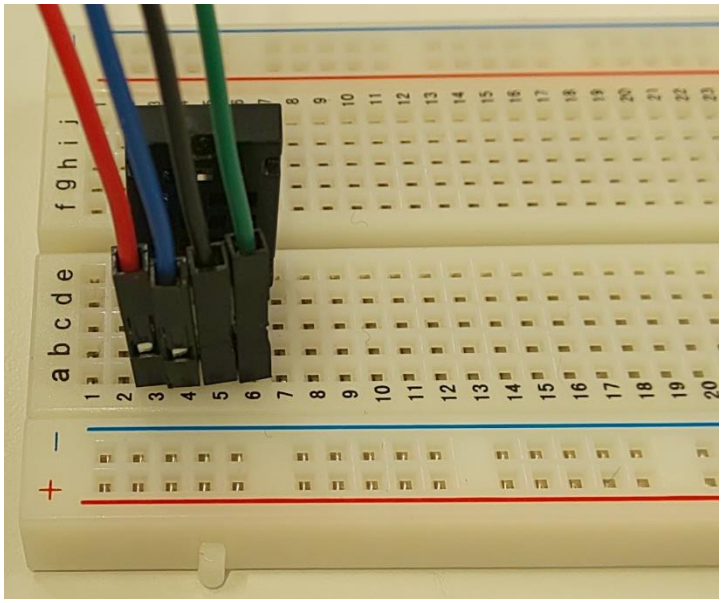
4.1 AM2320 を差し込む

AM2320 をブレッドボードに差し込みます。AM2320 の網の方を手前に向けて差し込みます。
両端が E3 と E6 になるように差し込みます。



4.2 ブレッドボードにジャンパー（オス-メス）を差し込む

ブレッドボードにジャンパー（オス-メス）を差し込みます



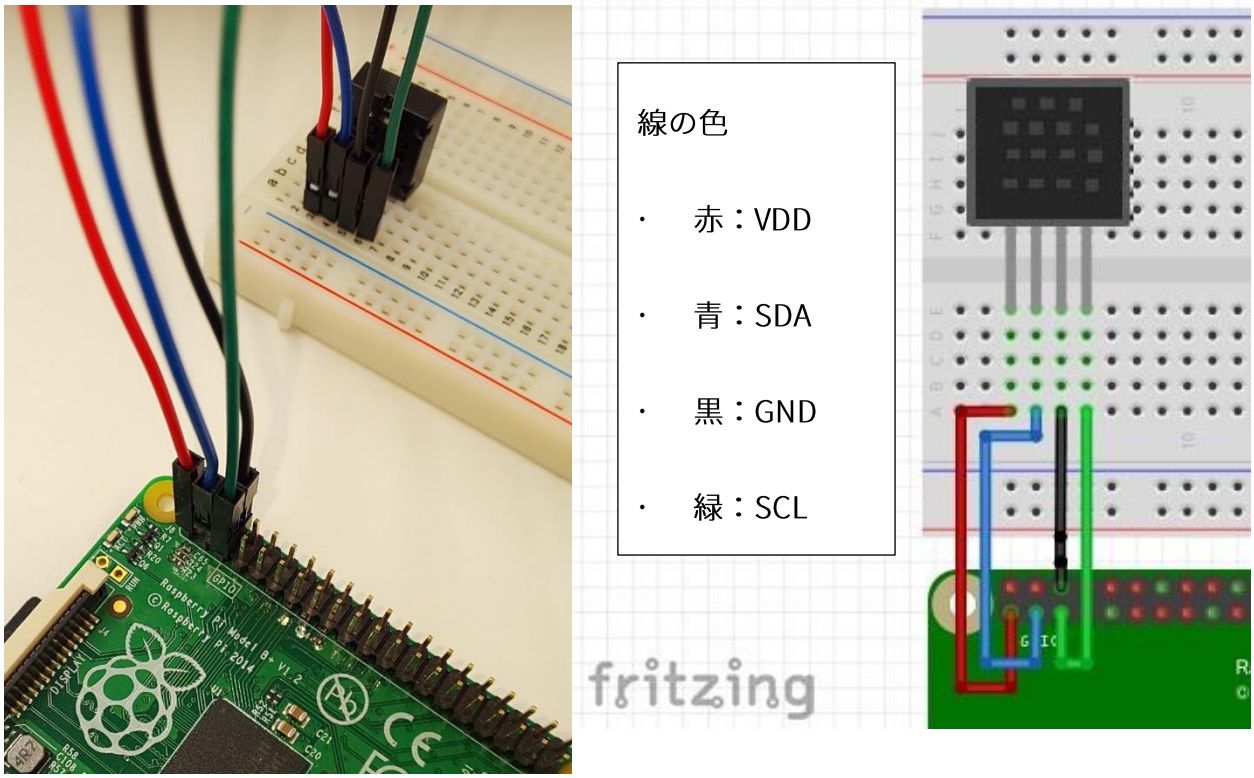
線の色

- 赤：VDD
- 青：SDA
- 黒：GND
- 緑：SCL

それぞれ A3、A4、A5、A6 に差し込みます。

4.3 Raspberry Pi と接続

Raspberry Pi とブレッドボードの回路を接続します。
A3 の線をピン番号 1(3.3V)、A4 の線をピン番号 3(SDA)、A5 をピン番号 6(GND)、A6 をピン番号 5(SCL)に差し込みます。



AM2320	Raspberry Pi
VDD	3.3V (PIN 番号 1)
SDA	SDA1 (PIN 番号 3)
GND	GND(PIN 番号 6)
SCL	SCL1(PIN 番号 5)

5 プログラム

AM2320 を使用して温度を取得するプログラムを書きます。

5.1 温度取得のプログラム

5.1.1 温度を取得するプログラム

以下のプログラムで AM2320 から温度を取得します。

一部「TODO:」とコメントアウトしていますので、その部分の処理を作成してください。

```
# -*- coding: utf-8 -*-
import smbus
import time

AM2320_ADDR =#TODO:スレイブアドレス#

i2c=smbus.SMBus(1)

while(True):
        # AM2320 を起動
        try:
                i2c.read_byte(AM2320_ADDR)
        except:
                pass
        time.sleep(0.003)
        #読み取りたいデータの指定
        read_function=0x03
        start_reg=#TODO:スタートレジスタ#
        num_reg=#TODO:レジスタ長#
        #読み取りたいデータのスタートレジスタとレジスタ長を指定
        i2c.write_i2c_block_data(AM2320_ADDR, read_function,[ start_reg, num_reg])
        time.sleep(0.015)

        #センサからの情報を取得・温度を計算して表示
        read_num=4
        data=i2c.read_i2c_block_data(AM2320_ADDR, read_function, read_num)
        temp=float(data[2]<<8|data[3])/10

        print"temperature:",temp,"℃"
        time.sleep(1)
```

テキストにコピーして「02_AM2320.py」のファイル名で保存してください。

コピーが出来ない方は以下 URL からダウンロードしてください。

https://rtc-fukushima.jp/wp/wp-content/uploads/2019/01/02_AM2320.zip

※温度取得の課題が終わり時間の余った方は，湿度の表示にも挑戦してみてください！！！

5.2 プログラムの解説

5.2.1 プログラムの流れ

1 秒ごとに温度を表示するプログラムは以下のような処理になります。

1. AM2320 を起動
   AM2320 は待機状態なので信号を送り起動させる。
2. 読み取りたいデータの指定
   ファンクションコード 0x03 を送り、データテーブルから読み取りたいデータの範囲を指定する。
3. センサからの情報を取得・温度を計算して表示
   センサから返答の中から必要なデータの範囲を指定
   データは上位ビット下位ビットに分かれているので、計算して温度の値に直し表示

5.2.1 AM2320 を起動

AM2320 を待機状態から起動状態にするために信号を送ります。送る信号はスレイブアドレスのみになります。
信号の送り方は以下になります。

```
i2c.read_byte(0x5c)
```

待機状態から起動状態へするために信号を送っているだけなので、値は返ってきません。そのためにエラーにならないようにするために try と except を使用しています。

5.2.2 読み取りたいデータの指定

データを読み取るときは、ファンクションコード[0x03]と読み取りたいデータの最初のアドレスとそこから何アドレス読み込むかを指定します。
以下の表の湿度と温度のデータを読み取りたいときは以下のようになります。

レジスタ情報	アドレス	レジスタ情報	アドレス	レジスタ情報	アドレス
湿度上位 8 ビット	0x00	Retention	0x04	型番	0x08
湿度下位 8 ビット	0x01	Retention	0x05		
温度上位 8 ビット	0x02	Retention	0x06		
温度下位 8 ビット	0x03	Retention	0x07	Retention	0x1F

```
i2c.write_i2c_block_data(0x5c,0x03,[0x00,0x04])
```

5.2.3 センサからの情報を取得・温度を計算して表示

読み取りの指令を送った後、結果が返ってきますのでその値を確認します。

湿度と温度のデータの読み取り指令を送った場合結果は以下になります。

通信項目	値
センサアドレス	0x5c
ファンクションコード	0x03
レジスタ数	0x04
返答のレジスタ 1	湿度の上位 8 ビットの値
返答のレジスタ 2	湿度の下位 8 ビットの値
返答のレジスタ 3	温度の上位 8 ビットの値
返答のレジスタ 4	温度の下位 8 ビットの値
CRC Code 下位 8bit	CRC Code の下位ビットの値
CRC Code 上位 8bit	CRC Code の上位ビットの値

実際にプログラムで読み込むのは赤枠の値になります。読み込むときはファンクションコードからどこまで読み込むかを指定します。
ファンクションコードから返答レジスタ 4 まで読み込むときは以下のようになります。

```
data=i2c.read_i2c_block_data(AM2320_ADDR,0x03,6)
```

読み込んだ結果が data に配列で格納されます。

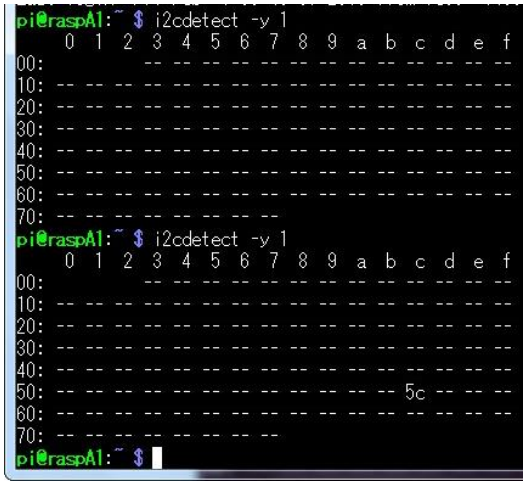
※本来 read_i2c_block_data、write_i2c_block_data は第 2 引数に対象となるレジスタアドレスを指定して使います。
たとえば今回のですと read_i2c_block_data は 0x03 から 6 バイト分読み込むという意味になり、write_i2c_block_data は 0x03 に 0x00 を 0x04 に 0x04 の値を書き込むという意味になります。
しかし AM2320 は特殊なデバイスなので、上記で示した手順で値の取得を行います。

5.3 プログラムの実行

Raspberry Pi の電源を入れてファイルを転送します。
転送後、AM2320 が接続されているか確認します。以下のコマンド入力してください。

```
$ i2cdetect -y 1
```

接続されていれば以下の様に[0x5c]と表示されます。一回目は待機状態のため表示されませんので素早く 2 回入れて表示させてください。



表示されましたらプログラムを実行してください。

```
$ python 02_AM2320.py
```

以下のように温度の値を取得できれば成功です。