


デュアルウェア講習会 II 温度センサの使い方



温湿度センサ AM2320 を使い温度と湿度を取得する

目次

第 1 章	課題	3
1.1	課題目的	3
第 2 章	機材説明	4
2.1	使用機材	4
2.2	Raspberry Pi	4
2.3	ブレッドボード	6
2.4	温湿度センサ AM2320	6
2.5	ジャンパー	11
第 3 章	I2C	12
3.1	I2C(Inter Integrated Circuit ^{*1}) について	12
3.2	I2C の使い方	12
第 4 章	回路製作	14
4.1	AM2320 を差し込む	15
4.2	ブレッドボードにジャンパー (オス-メス) を差し込む	15
4.3	Raspberry Pi と接続	16
4.4	Raspberry Pi に電源を入れる	17
第 5 章	プログラム	18
5.1	温度取得プログラム	18
5.2	プログラムの解説	20
5.3	プログラムの実行	22
5.4	温度と湿度を取得するプログラム	23

1.1 課題目的

図 1.1 のように、ブレッドボードに温湿度センサ AM2320 を差し込み、Raspberry Pi と接続して、温度と湿度を取得するプログラムを作成できる。

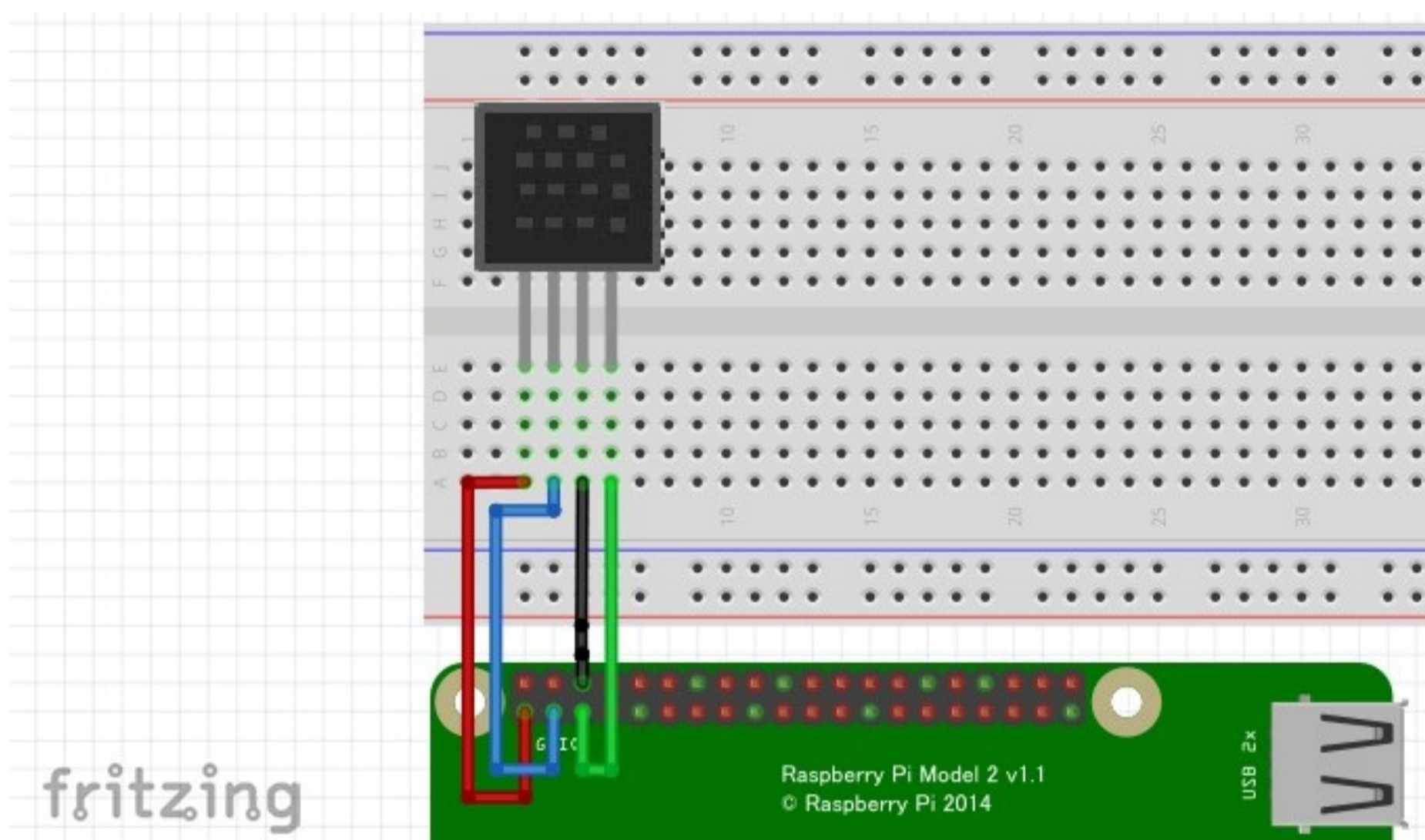


図 1.1: 温湿度センサ回路

2.1 使用機材

この課題は以下の機材を使用します (表 2.1).

表 2.1: 使用機材

機材	個数
Raspberry Pi	1
ブレッドボード	1
温湿度センサ AM2320	1
ジャンパー (オスメス)	4

2.2 Raspberry Pi

Raspberry Pi は小型のコンピュータです。ハードウェアにつなげやすく特徴があり、電子工作に利用されることが多いです。

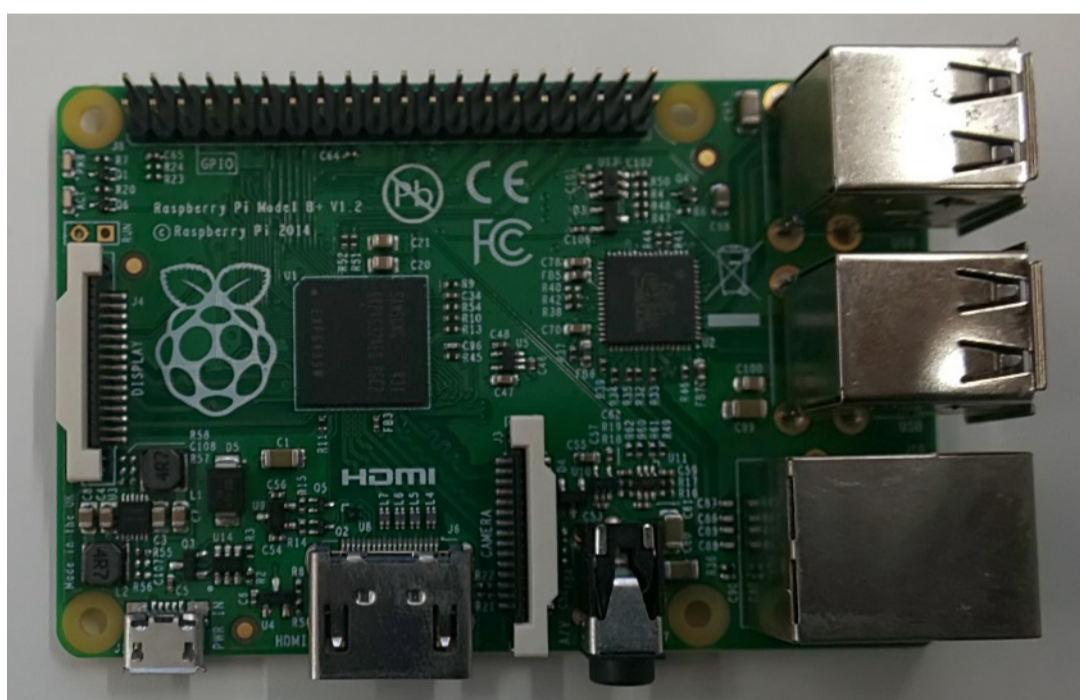


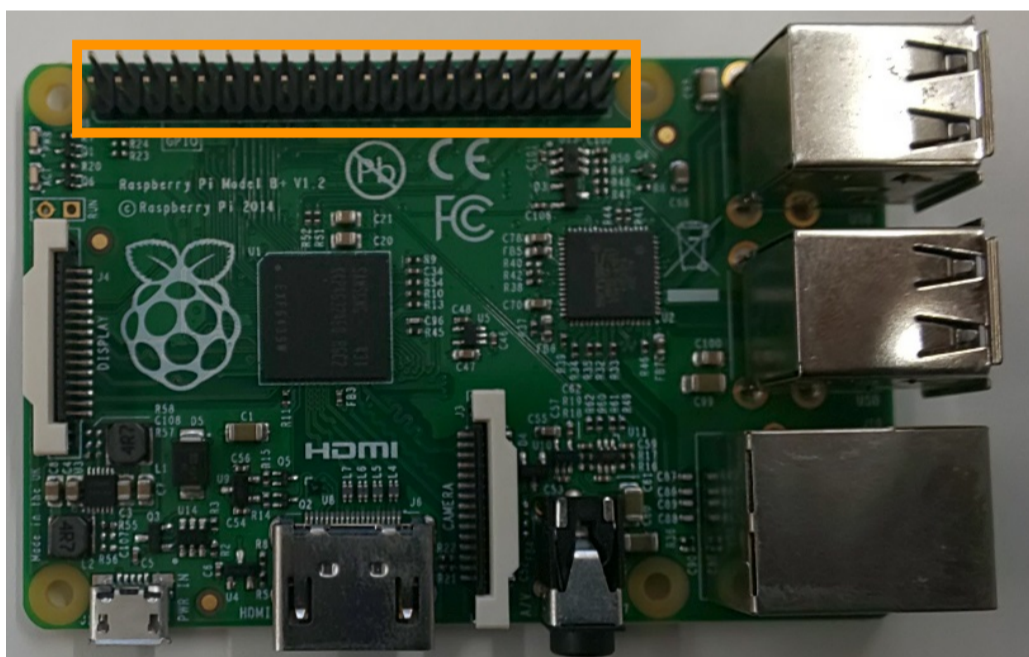
図 2.1: Raspberry Pi

今回使用する Raspberry Pi は [Raspberry pi model 3 B+] です。スペックを表 2.2 に示します。

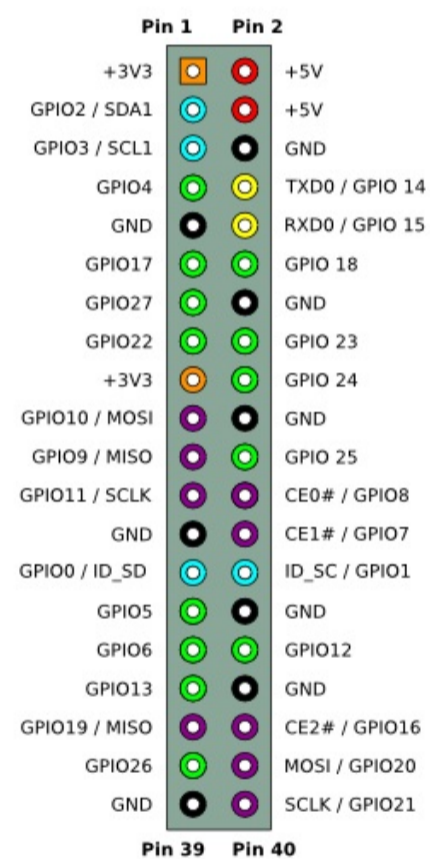
表 2.2: Raspberry Pi スペック

CPU	700 MHz / ARM1176JZF-S コア
メモリ	512 MB
USB ポート	4
ネットワーク	10/100Mbps イーサネット
ビデオ出力	HDMI, DSI
音声出力	3.5mm ジャック
低レベル出力	GPIO×40 (GPIO, UART, I2C, SPI, 3.3V, 5V, GND など)
必要電源	5V, Micro USB Micro-B
サイズ	86×56×18mm (約 45g)

Raspberry Pi には GPIO というインターフェースがあります。この GPIO を使用して電子回路を制御します。



(a) GPIO ピンの場所



(b) GPIO ピンの意味^{*1}

図 2.2: Raspberry PI GPIO ピン

- 3.3V, 5V : 電圧を常に供給するピン (ON/OFF はできない), 電池でいうプラスに相当する
- GND : 電圧が 0 になるピン, 電池でいうマイナスに相当する
- GPIO : デジタル入出力ができるピン
- I2C : I2C 通信に関わるピン, SDA と SCL が割り当てられている
- SPI : SPI 通信に関わるピン, MOSI, MISO, SCLK, CE が割り当てられている

^{*1} <https://elinux.org/File:Pi-GPIO-header.png>

2.3 ブレッドボード

電子回路を作成するためのボード。穴がありその中に LED やジャンパーを差し込み，回路を作製します (図 2.3)。

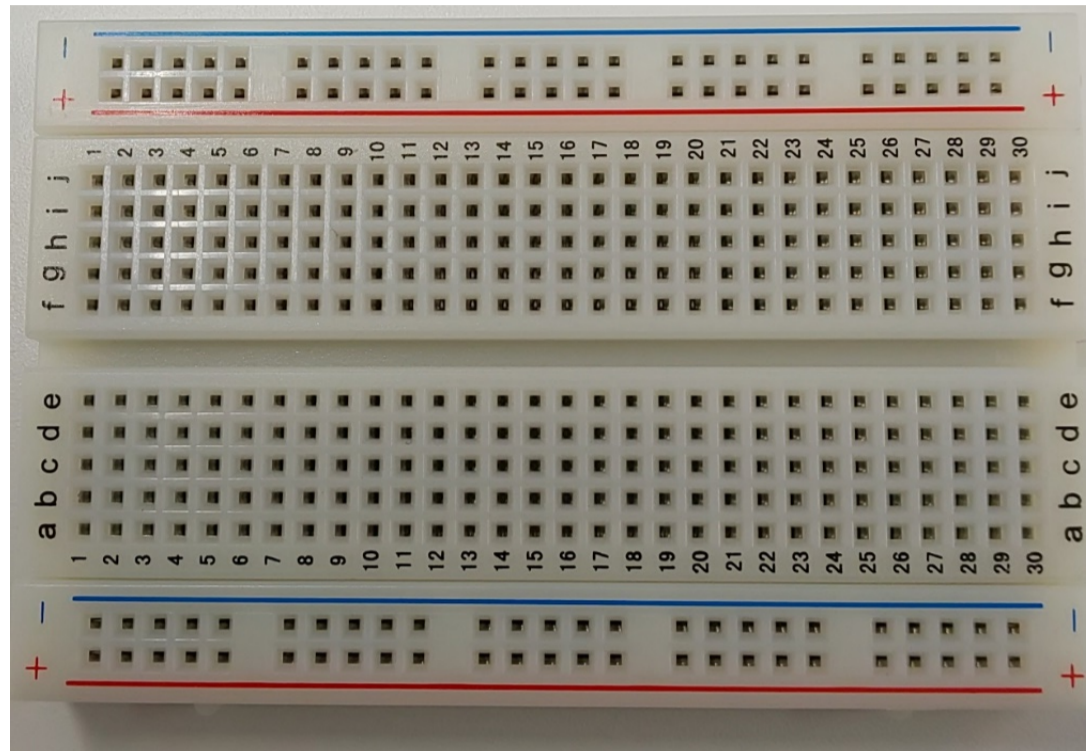


図 2.3: ブレッドボード

2.4 温湿度センサ AM2320

温湿度 AM2320 は I2C で通信できる温度センサです。データシート^{*2}の内容を以下にまとめました。

特徴 (表紙 Product Features を参照) は以下の通りです。

- 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

この特徴から，通信方式が I2C であることが判明する。

^{*2} <http://www.kyohritsu.jp/eclib/OTHER/DATASHEET/SENSOR/am2320.pdf>

電圧・電流 (Page 3 6.Electrical Characteristics を参照)^{*3}は以下の通りです (表 2.3).

表 2.3: 電圧・電流特性

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

この特徴から、電源電圧が 3.1~5.5V であることが判明する。

端子 (Page 2 : ピン配置図を参照)^{*4}は以下の通りです (図 2.4, 表 2.4).

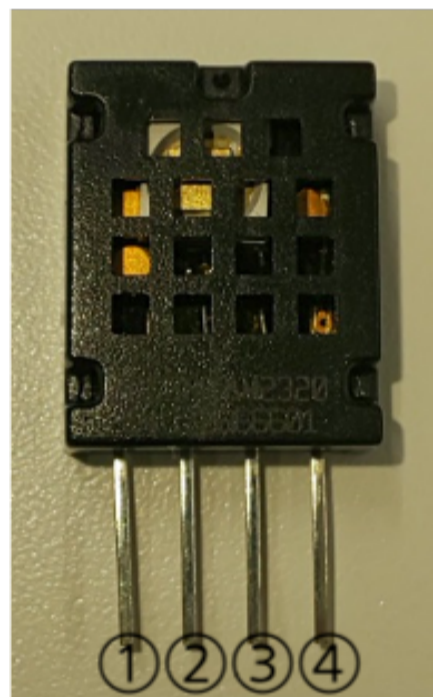


図 2.4: AM2320 ピン

表 2.4: AM2320 ピン

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	グラウンド
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

^{*3} Page 3 Table 3: AM2320 DC Characteristics を確認

^{*4} Page 2 4.Dimensions, Page 7 7.1.AM2320 pin assignment を確認

スレイブアドレス (Page 8 Slave Address, Page 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 (表 2.5)

表 2.5: スレイブアドレス

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決めることになります.

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 :
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 を使用する (表 2.6)

表 2.6: 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

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

温度と湿度の値はレジスタアドレスに格納されている (表 2.7)。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。

表 2.7: レジスタ情報

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

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 (表 2.8)。

表 2.8: 送信フォーマット

通信項目	意味
センサアドレス	接続するスレーブアドレス
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 (表 2.9)。

表 2.9: 返答フォーマット

通信項目	意味
センサアドレス	接続するスレーブアドレス
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
レジスタ数	返答したレジスタ数 (レジスタ数依存)
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依存)
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・湿度を読み込むときは以下ようになります (表 2.10, 2.11).

表 2.10: 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

表 2.11: 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ピンクセルの部分になります。

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

取得した上位 8bit と下位 8bit を足して、10 で割ることで、値が求まる。温度上位 8bit : 0x01, 温度下位 8bit : 0x15 の場合, $0x0115 = 1 * 256 + 1 * 16 + 5 = 277$ から, $277 \div 10 = 27.7^{\circ}\text{C}$ となる。

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、次のステップに移ります。

2. 読み出し命令を送信 :

センサを起動させた後、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、次のステップに移ります。

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よう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。

2.5 ジャンパー

ブレッドボード上や電子部品を接続させるための導線のこと。この講習会では以下のジャンパー（オス-メス）を使用します（図 2.5）。



図 2.5: ジャンパー（オス-メス）

3.1 I2C(Inter Integrated Circuit^{*1}) 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 本の信号線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できます。

3.2 I2C の使い方

3.2.1 Python 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。

```
1 import smbus
```

- 宣言 :

バス番号を引数にして宣言・初期化をします^{*2}。

```
1 i2c = smbus.SMBus(1) # 引数はBusNumber (sudo i2cdetect 1の1)
```

- `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 (表 3.1)。

```
1 i2c.read_byte(0x5c) # 使用例
```

表 3.1: `read_byte` メソッド

スレイブアドレス	接続している機器のアドレス
----------	---------------

^{*1} アイ・スクエアド・シーが正式な読み方とされている。

^{*2} コマンド `sudo i2cdetect 1の1`

- `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 (表 3.2).

```
1 value = i2c.read_i2c_block_data(0x5c, 0x00, 0x04) # 使用例
```

表 3.2: `read_i2c_block_data` メソッド

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

- `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 (表 3.3).

```
1 i2c.write_i2c_block_data(0x5c, 0x03, [0x00, 0x04]) # 使用例
```

表 3.3: `write_i2c_block_data` メソッド

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

ブレッドボードに AM2320 を差し回路を製作します。完成イメージを図 4.1 に示します。※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。

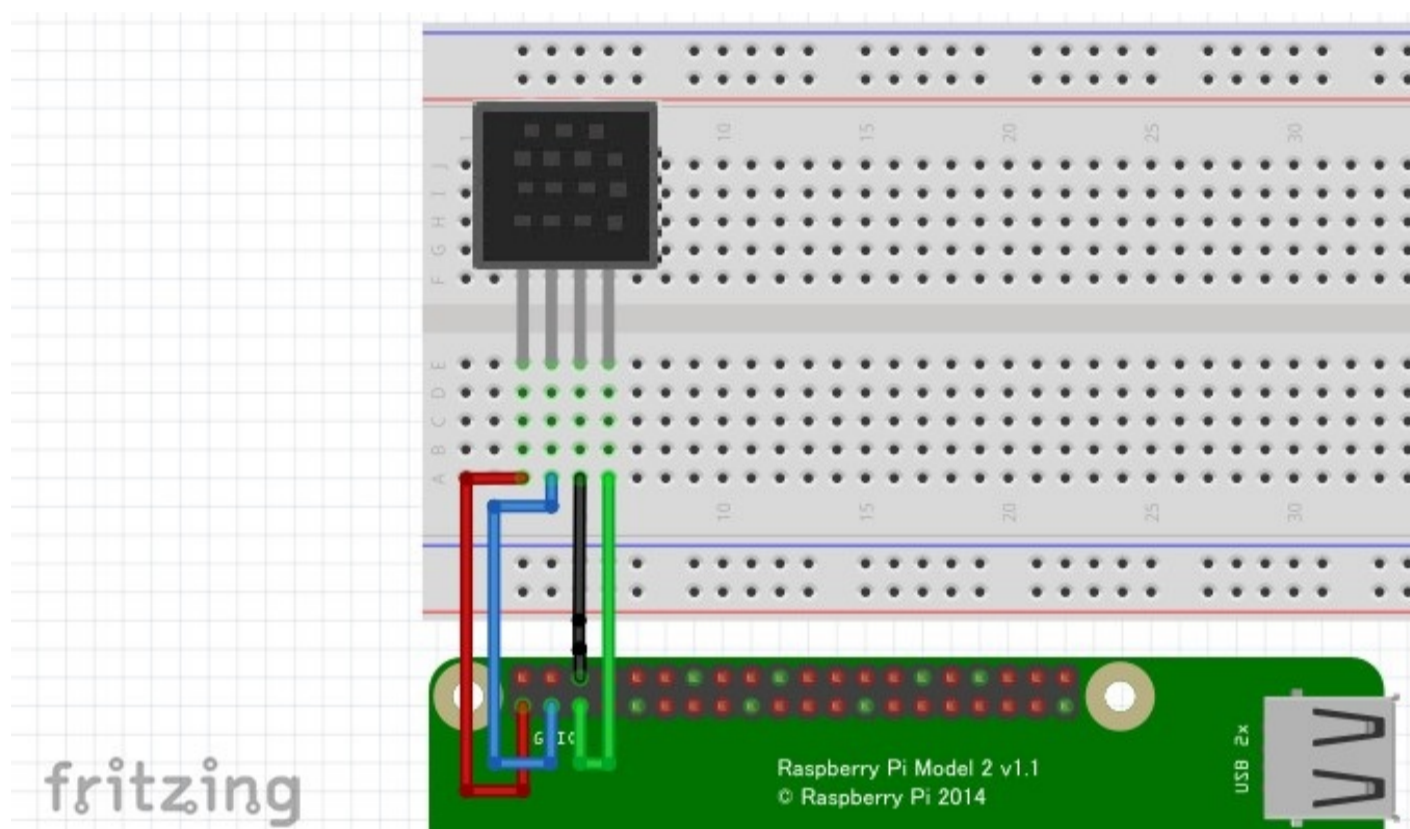


図 4.1: 回路完成図

4.1 AM2320 を差し込む

AM2320 をブレッドボードに差し込みます (図 4.2)。AM2320 の網の方を手前に向けて差し込み、両端が e3 と e6 になるように差し込みます。

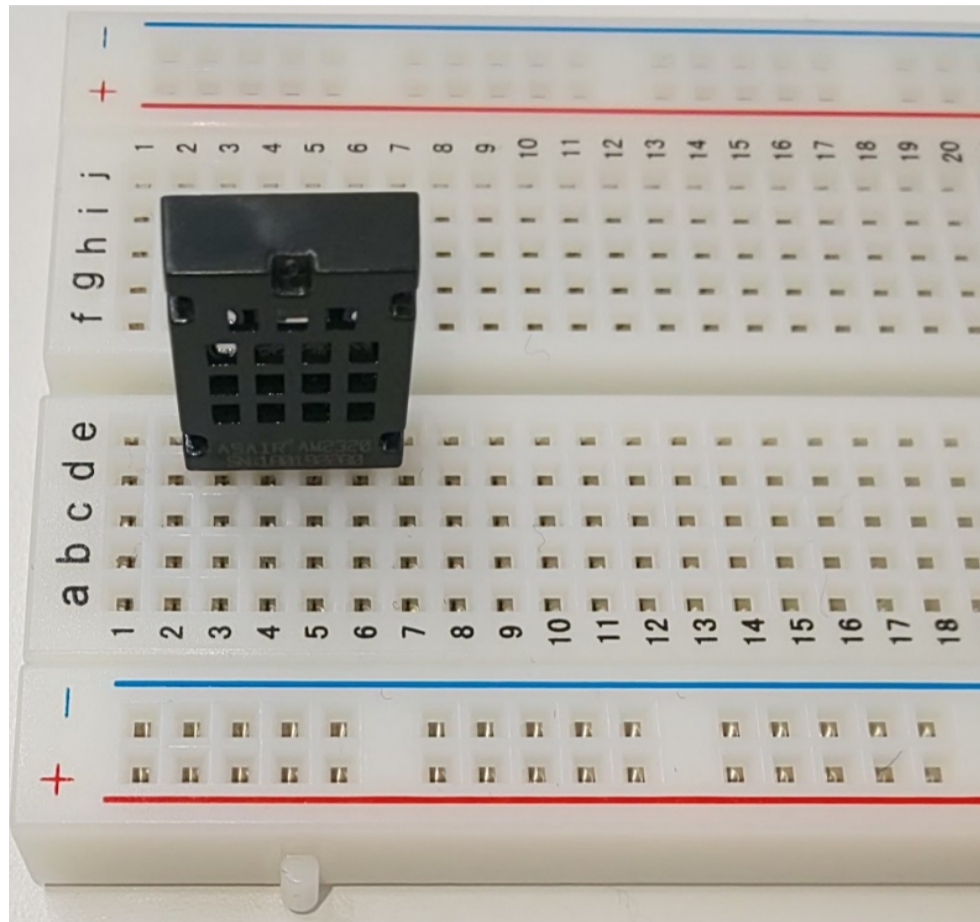


図 4.2: AM2320 差し込み

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 (図 4.3, 表 4.1)。

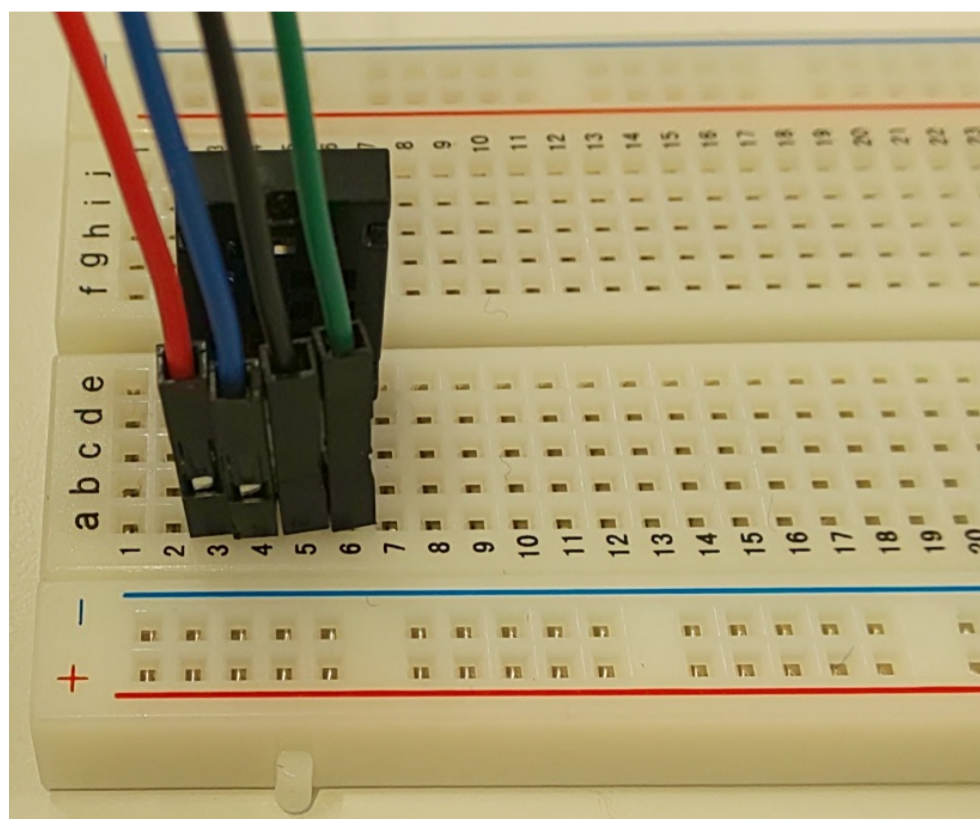


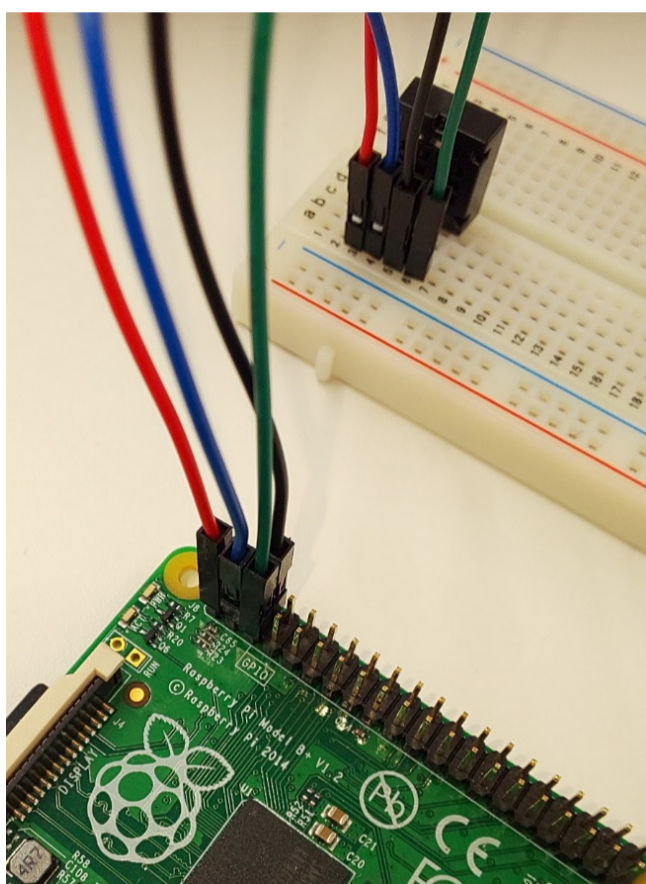
図 4.3: ジャンパー差し込み

表 4.1: ジャンパー差し込み

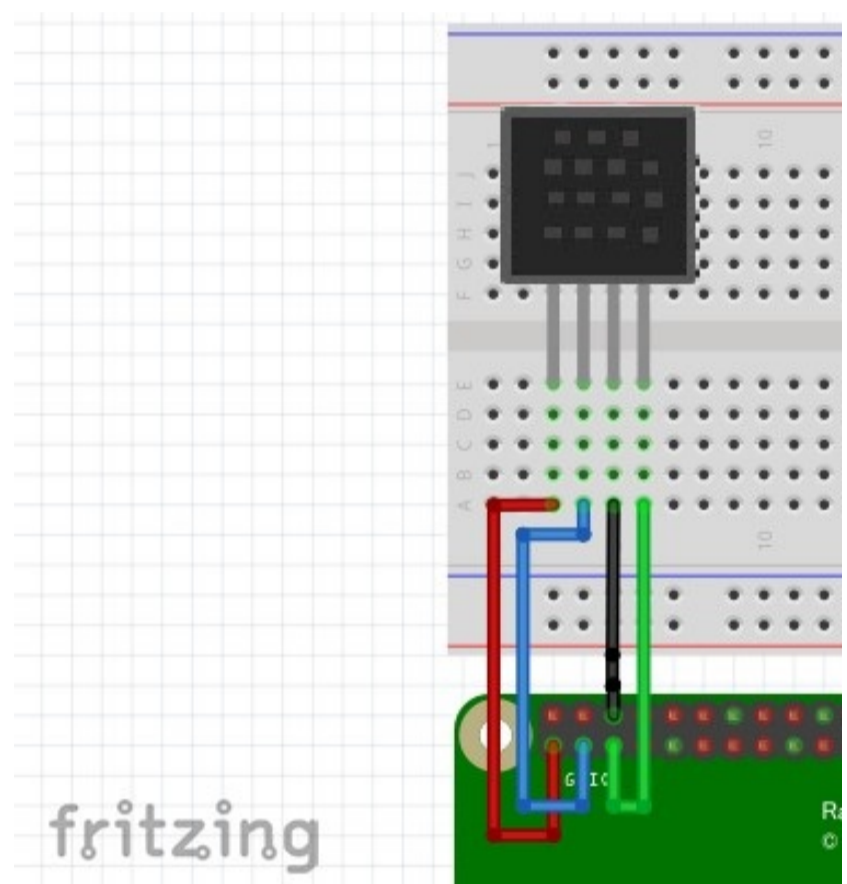
ジャンパー線色	差し込み場所
赤 (VDD)	a3
青 (SDA)	a4
黒 (GND)	a5
緑 (SCL)	a6

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 (図 4.4).



(a) 接続図



(b) 回路図

図 4.4: Raspberry Pi と接続

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 (表 4.2).

表 4.2: 接続対応表

ジャンパー線色 (AM2320)	Raspberry Pi
赤 (VDD)	ピン番号 1(3.3V)
青 (SDA)	ピン番号 3(SDA)
黒 (GND)	ピン番号 6(GND)
緑 (SCL)	ピン番号 5(SCL)

4.4 Raspberry Pi に電源を入れる

Raspberry Pi にモバイルバッテリーを接続し、Raspberry Pi に電源を供給します (図 4.5).

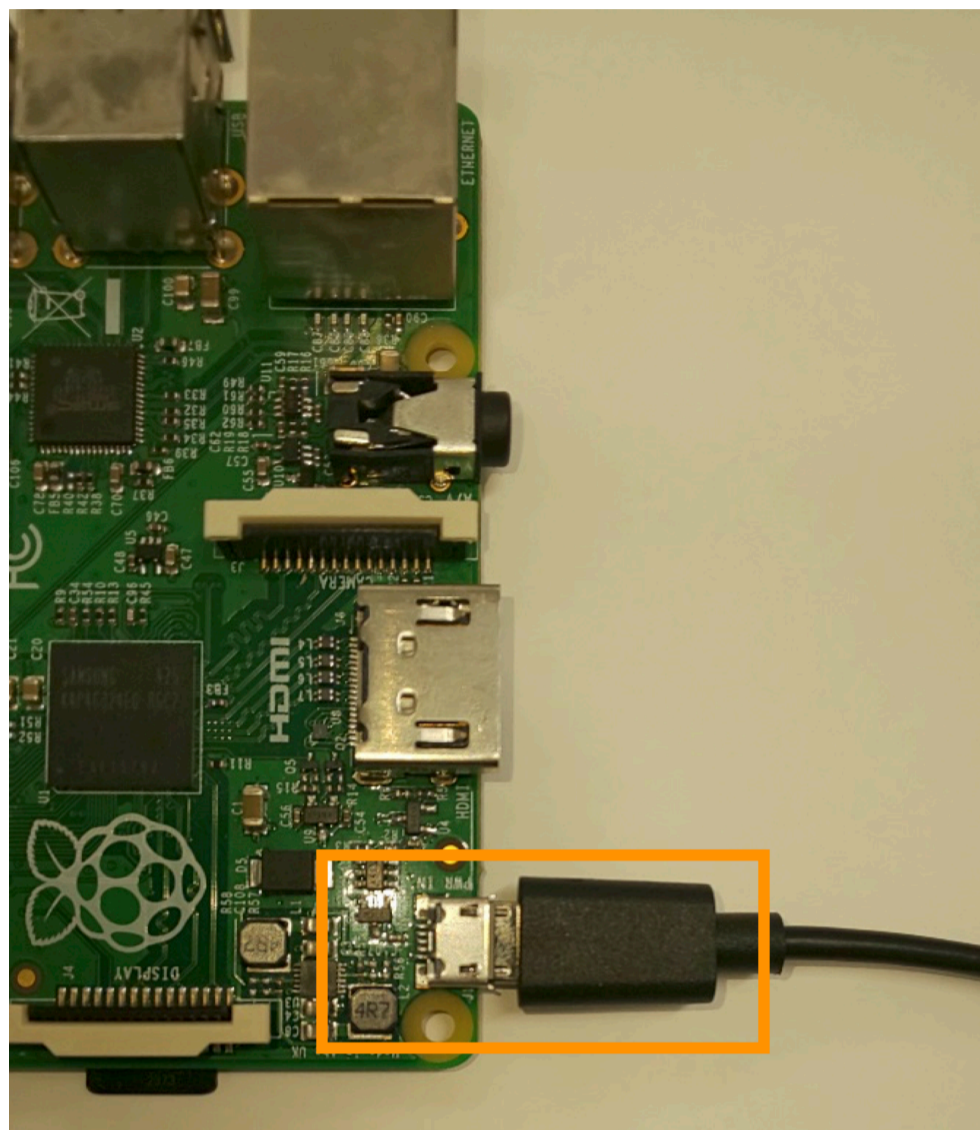


図 4.5: Raspberry Pi 電源接続

温湿度センサ AM2320 を使用して、温度を取得するプログラムを作成します。

5.1 温度取得プログラム

5.1.1 温度を取得するプログラム

以下のプログラムで、温湿度センサ 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。

```
1 # -*- coding: utf-8 -*-
2 import smbus
3 import time
4
5 AM2320_ADDR =#TODO:スレイブアドレス#
6
7 i2c=smbus.SMBus(1)
8
9 # AM2320を起動
10 try:
11     i2c.read_byte(AM2320_ADDR)
12 except:
13     pass
14 time.sleep(0.003)
15
16 #読み取りたいデータの指定
17 read_function=0x03
18 start_reg=#TODO:スタートレジスタ#
19 num_reg=#TODO:レジスタ長#
20 #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
21 i2c.write_i2c_block_data(AM2320_ADDR, read_function, [ start_reg,
    num_reg])
22 time.sleep(0.015)
23
```



```
24 #センサからの情報を取得・温度を計算して表示
25 read_num=4
26 data=i2c.read_i2c_block_data(AM2320_ADDR, read_function, read_num
   )
27 temp=float(data[2]<<8|data[3])/10
28
29 print"temp:",temp," °C"
```

テキストエディタにコピーして、「01_AM2320temp.py」のファイル名で保存してください。

5.1.2 Raspberry Pi へプログラムを転送

プログラム作成後、Raspberry Pi に転送します。今回は「TeraTerm」を利用します。TeraTerm をダブルクリックして起動してください。

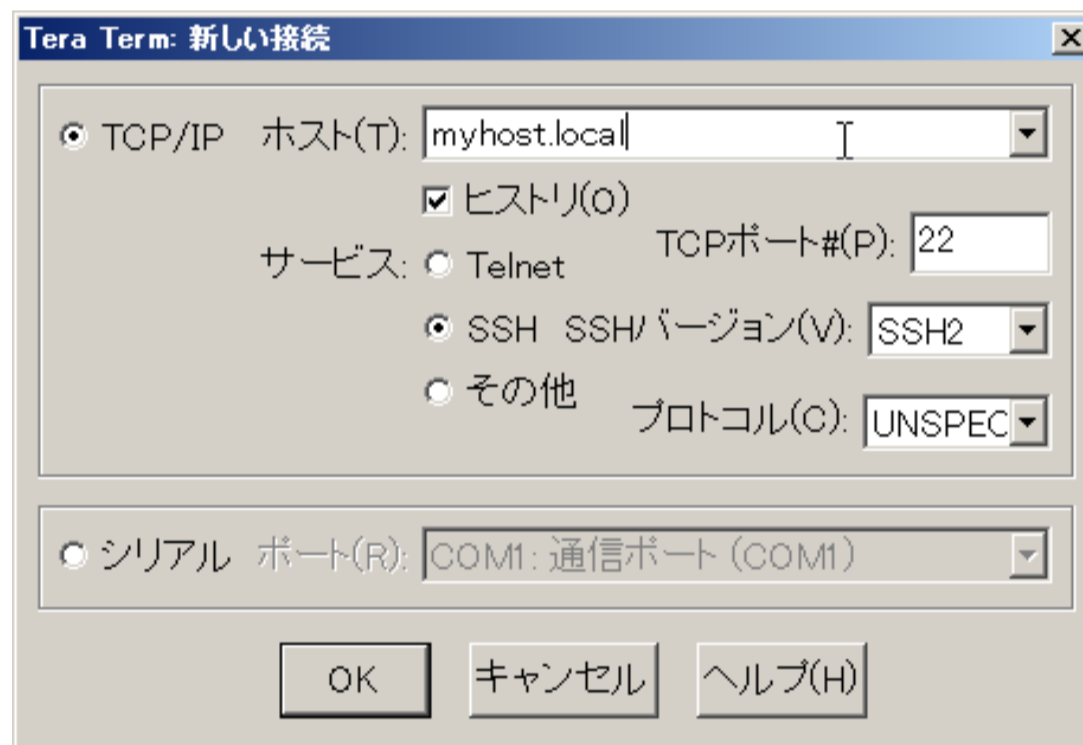
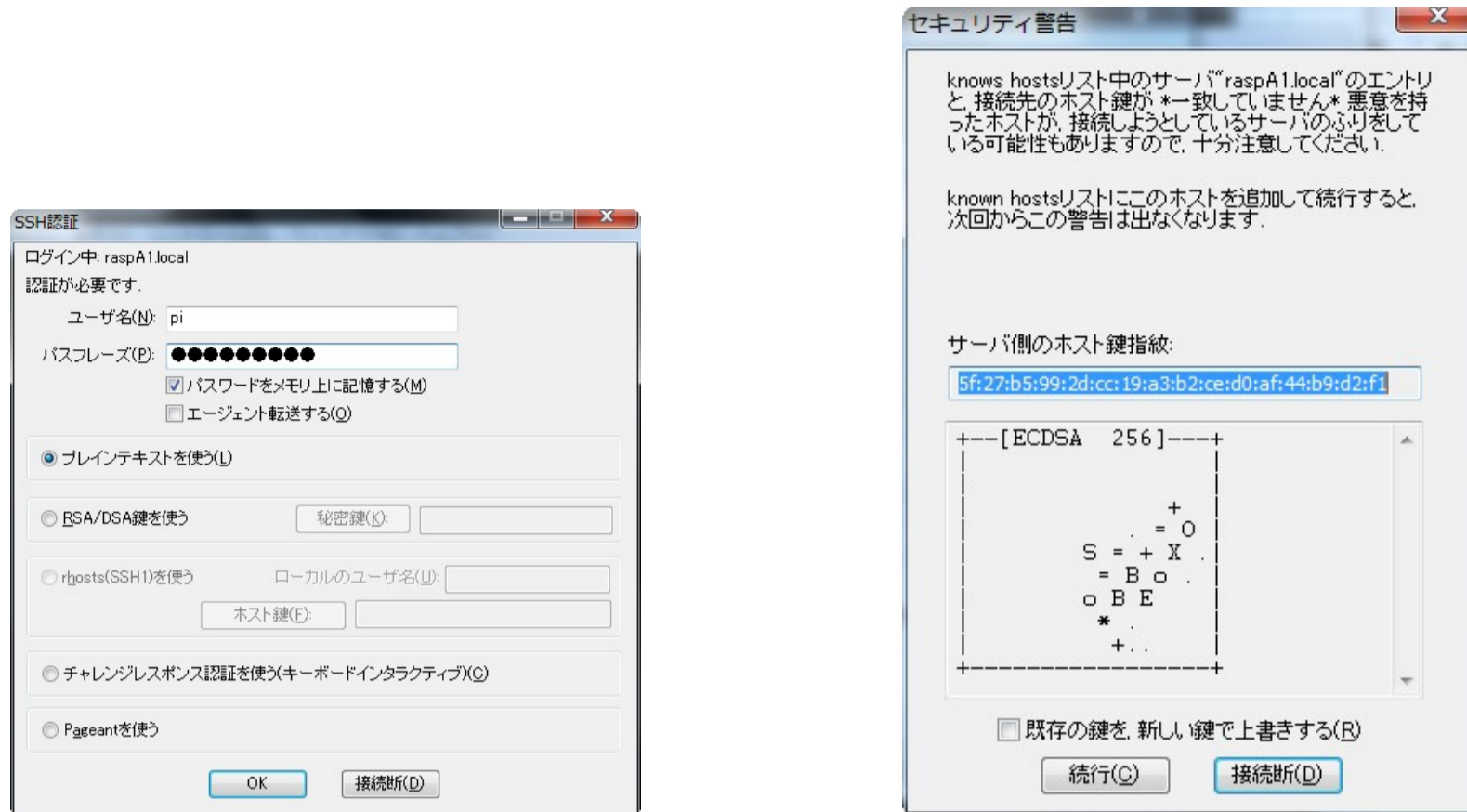


図 5.1: TeraTerm 接続 1

TeraTerm を起動すると接続ダイアログが現れますので (図 5.1), 接続する Raspberry Pi のホスト名.local, または, Raspberry Pi の IP アドレスを「ホスト」のテキストボックスに入力し, OK をクリックします。今回の講習会では, ホスト名は箱の付箋に記載されています。

SSH 認証 (図 5.2(a)) が出現しましたらユーザ名: pi, パスフレーズ: raspberry を入力してログインします*1.



(a) SSH 認証

(b) セキュリティ警告

図 5.2: TeraTerm 接続 2

プログラムの転送は TeraTerm の「SSH SCP …」を使用し、ファイルをコピーします。TeraTerm の「ファイルメニュー」→「SSH SCP …」を選択します。From に 01_AM2320temp.py を選択, To に~/か/home/pi/と入力し, Send ボタンをクリックします (図 5.3)。

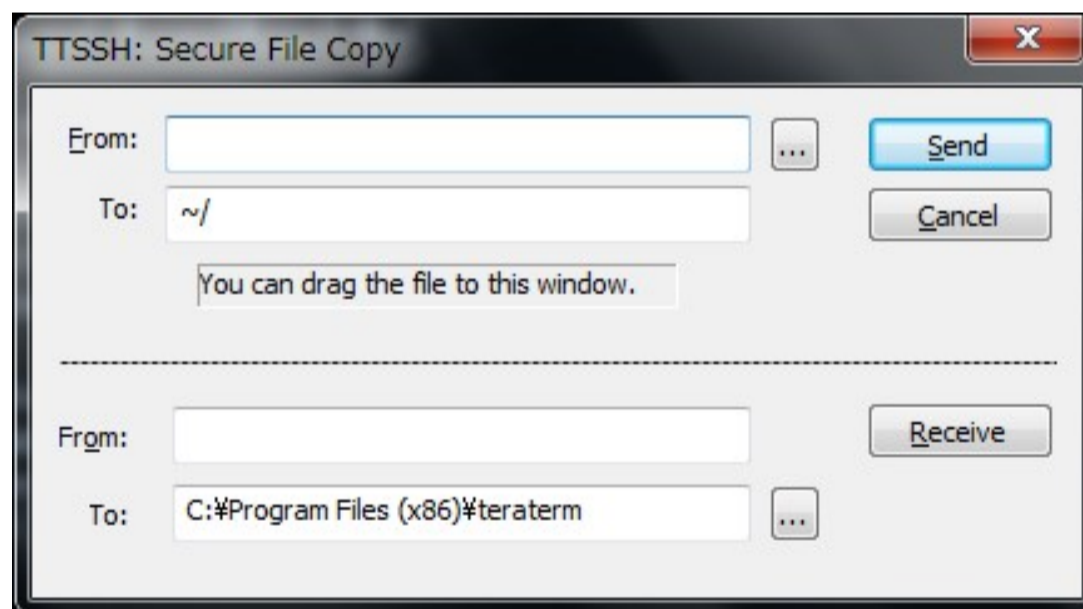


図 5.3: TeraTerm 接続 3

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, 以下のような処理となります。

*1 セキュリティ警告 (図 5.2(b)) が出現した場合, 続行ボタンを押してください。

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：

センサからの返答の中から必要なデータの範囲を指定する。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す。

5.2.2 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。

```
1 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。

5.2.3 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。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になります (表 5.1)。

表 5.1: レジスタ情報

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

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

5.2.4 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（表 5.2）。

表 5.2: 受信情報

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ピンクセルの値になります。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。

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

読み込んだ結果がリスト data 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用します。たとえば、今回 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。

```
1 $ i2cdetect -y 1
```

接続されていれば、以下のように 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（図 5.4）。

0x5c が表示されたら、プログラムを実行してください。

```
1 $ python 01_AM2320temp.py
```

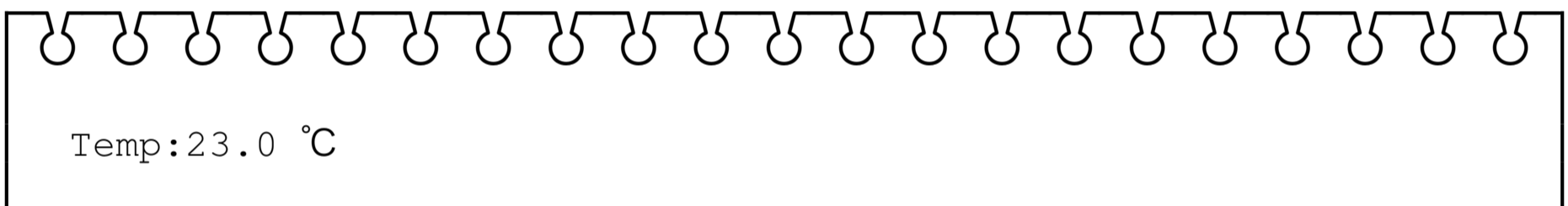
以下のように、温度の値を取得できれば成功です。

```

pi@raspA1:~$ i2cdetect -y 1
    0 1 2 3 4 5 6 7 8 9 a b c d e f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspA1:~$ i2cdetect -y 1
    0 1 2 3 4 5 6 7 8 9 a b c d e f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  -- 5c --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspA1:~$

```

図 5.4: AM2320 接続結果



5.4 温度と湿度を取得するプログラム

以下のプログラムで AM2320 から温度と湿度を取得します。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。

```

1 # -*- coding: utf-8 -*-
2 import smbus
3 import time
4
5 AM2320_ADDR =#TODO: スレイブアドレス#
6
7 i2c=smbus.SMBus(1)
8 # AM2320を起動
9 try:
10     i2c.read_byte(AM2320_ADDR)
11 except:
12     pass
13 time.sleep(0.003)

```



```
14 #読み取りたいデータの指定
15 read_function=0x03
16 start_reg=#TODO:スタートレジスタ#
17 num_reg=#TODO:レジスタ長#
18 #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
19 i2c.write_i2c_block_data(AM2320_ADDR, read_function, [ start_reg,
    num_reg])
20 time.sleep(0.015)
21 #センサからの情報を取得・温度を計算して表示
22
23 read_num=6
24 data=i2c.read_i2c_block_data(AM2320_ADDR, read_function, read_num
    )
25 humi=float(data[2]<<8|data[3])/10
26 temp=float(data[4]<<8|data[5])/10
27
28 print"temp:",temp," °C", "humi:",humi," %"
```

テキストにコピーして、01_AM2320temphumi.pyのファイル名で保存してください。