ATOMTEX®

ガンマ線・放射線測定器ユニット

BDKG-02

取扱説明書

RS485 通信仕様

目次

1　はじめに 5

1.1　検知器の取扱の注意点 5
1.2　保管 5
1.3　保証 6
1.4　機能と測定項目 7

2　測定値の読み方 8

1.2 測定時間と偏差の関係 9
1.3 測**定時間の目安と偏差** 10
1.4 放射線量が不安定な場合 11

3　仕様・付属品 12

1.5　付属品 12
1.6　仕様 BDKG-02 13

4　シリアル通信 14

1.7　回路図 14

5　RS485 シリアル通信 15

1.8　RS485 接続の基本 15
1.9　RS485 で使うケーブル 15
1.10　機器の接続 16
1.11　反射と終端抵抗 16
1.12　複数機器の接続 17
1.13　数珠つなぎが基本 19
1.14　例外 20
1.15　RS485-USB 通信機 (別売り) 20

6　通信プロトコル 21

1.16　ビットとバイト 21
1.17　シリアル接続の設定 21
1.18　機器アドレス 22
1.19　通信のやり取り 23
1.20　通信パケット 24
1.21　命令コマンド一覧 25

1.22 エンディアン 26
1.23 サンプルコードのダウンロード 27
1.24 誤り確認コードの計算方法 28
1.25 線量率の取得 0x03 31
1.26 浮動小数点 32
1.27 線量率の偏差（%）を取得する 0x1A 36
1.28 線量率の測定リセット 0x0A 37
7 その他 38
1.29 RS485 ケーブル 38
1.30 その他の通信機能のご紹介 39
8 Raspberry Pi との接続 40
1.31 配線図(SPI 接続の場合) 41
1.32 配線図(USB 接続の場合) 43
1.33 テスト環境 45
1.34 OS のインストール 45
1.35 起動 48
1.36 Raspberry Pi の設定等 48
1.37 SPI 接続の有効化 49
1.38 Emacs のインストール 51
1.39 Telnet の設定 51
1.40 フォルダを作る 51
1.41 Tera Term (Windows 版) 52
1.42 FTP サーバーの起動 53
1.43 WinSCP のインストール 53
1.44 SCI 接続の起動設定 55
1.45 その他 56
1.46 python の開発環境 56
1.47 Python サンプルプログラムの実行 56
1.48 SPI 接続 RS485-HAT 基板の説明書 57
9 パソコン(USB)との接続 58
1.49 C# サンプルアプリの起動例 59

1 はじめに

1.1　検知器の取扱の注意点

- 検知器は-40 度～+50 度の範囲でお使いください。
- 特に真夏に車の中に検知器を放置しないでください。高温の状態になると検出器は深刻なダメージを受けるため修理が必要になります。このような検出器の不具合は保証の対象外です。
- その他、落下や水没などにも注意してください。これらの原因による破損、動作不良は保証の対象外です。

1.2　保管

- 線量計は、10～35 度、湿度 80%（25℃）以下の環境で保管してください。
- 埃の少ない清潔な場所
- 酸・アルカリの揮発物質のない場所

1.3　保証

購入後 12 ヶ月間の保証期間があります。

保証期間内において、通常利用の範囲内での故障・不具合・初期不良の場合には、メーカーの判断によって無償修理を受けることができます。購入日付、保証期間は機器のシリアル番号で管理されており、保証書はありません。

保証期間内でも下記の場合には有償修理となります。

- 保証期間中に発生した故障でも保証期間後に修理を依頼された場合
- 取扱説明書などに記載のある使い方以外で発生した故障･損傷
- お買い上げ後の輸送･管理などが不適切で発生した故障･損傷
- 火災、地震、風水害、落雷、その他の天災、公害、塩害、異常電圧などによる故障･損傷
- 電池液漏れ、水没、落下による破損、改造、誤使用により発生した故障･損傷
- 他製品との接続などにより発生した故障･損傷
- 消耗品の摩耗、故障･損傷

本製品の故障またはその使用上生じたお客様の直接･間接の損害について当社はその責に任じません。故障によるデータの損失、修理･交換によるデータ損失に関しては、当社はその責に任じません。修理後の無償保証期間は、元の保証書の残存期間とさせていただきます。

校正点検は、保証期間内でも有償となります。

1.4　機能と測定項目

この製品はガンマ線・X 線の空間線量率を測定するための放射線測定ユニットです。

RS485 通信により、２つの値を取得することができます。

- 線量当量率 H˙*(10)（毎時シーベルト）
- 線量当量率 H˙*(10) の偏差（%）

測定値の読み出しには、ご自身でアプリを開発する必要があります。

動作テストを行うためのサンプルアプリ(C#, Phyton, Java 等)については、販売店にお問い合わせください。プログラム言語によっては場合によってはサンプルコードがない場合もあります。

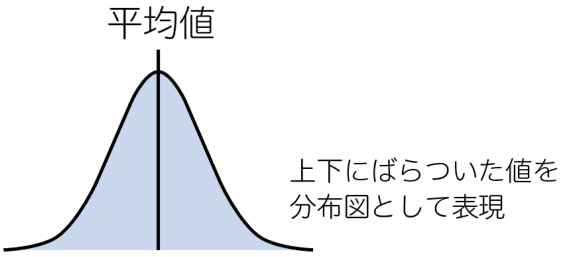
2　測定値の読み方

放射線測定器ユニット BDKG-02 は、線量率の測定値（Sv/h）と偏差（%）の２つの値を出力することができます。

$$1.00\ \mu Sv/h \qquad \pm 25\%$$

測定値　　偏差

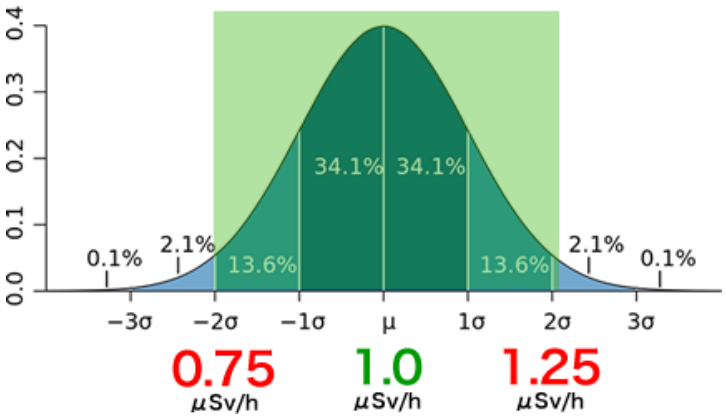
ここでは偏差（%）について解説いたします。一般的に偏差とは、個々の数値と平均値との差です。個々の数値データがばらついている様子は分布図を使って表現することができ、偏差は平均値からのばらつきの程度を示す数値になります。



放射線測定器の場合には、少し表現を変えて「短時間に測定した線量率の値（個々の測定値）」と「長期間に測定した平均の線量率の値（平均値）」の差を意味しています。

右図は、測定値 1.0 μSv/h 、偏差 25%の状態を示しています。

25%の部分は、1.0 μSv/h に対しての 25%と解釈して±0.25μSv/h の範囲となります。



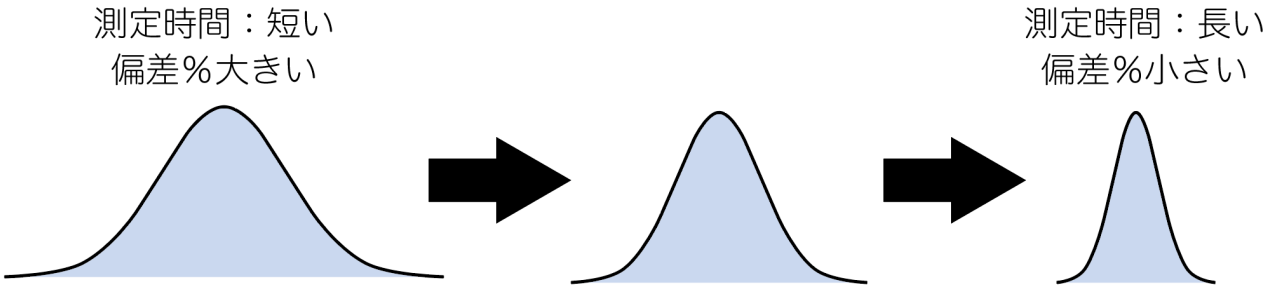
ここで平均値は 1.0μSv/h であり、偏差 25%の意味は平均値を中心として± 0.25μSv/h 範囲 (0.75 ～ 1.25 μSv/h の範囲) という意味になります。

放射線測定器の偏差は 95%の個々の測定値を含む範囲として定義されているため、0.75 ～ 1.25 μSv/h の範囲に、短時間で測定した個々の測定値の 95%が含まれるという意味にもなっています。

1.2 測定時間と偏差の関係

放射線は原子核から出てくる物体ですが、確率的に出たり出なかったりするためするため短時間だけ放射線を測定すると高い場合も出てくるし、低い場合も出てきます。

短時間での測定では、偏差が大きく分布は広がっているといえます。分布図でいえば横に広がった釣り鐘型の分布となります。分布が広いということは、偏差（%）の値も 60～90%といった大きな値になります。



放射線源と測定器の距離が一定の場合や、周辺一帯が放射線に汚染された場所では、放射線が安定した値と測定されます。このような場所で時間をかけて測定すると、安定した測定値が得られるため偏差の値は 90%、70%、50%、30%、20%といった具合に小さくなってきます。これは平均値に対して釣り鐘型の分布が細くなっていくことを意味しています。

長時間の測定を行い、偏差（%）＝ばらつきが小さい時の平均測定値について以下のことがいえるようになります。

- 偏差（ばらつき）が小さいので、
  平均値としての測定値は、正しい測定値である。

- 偏差が小さいということは、
  測定対象物の放射線量（線量率）が安定している状態である。

1.3 測定時間の目安と偏差

ここまでの解説で時間をかけて測定すれば、偏差（%）の値が小さくなってくることを見てきました。偏差の値は測定時間が長くなるほど小さい値になります。偏差の値が一定の値まで下がるまで待ってから測定することで、同じ精度での測定が可能となります。

通常お使いの場合では、偏差が 30%以下になるまで待ってから線量率の測定値(平均値)を読んでください。

1.4 放射線量が不安定な場合

ここまでの解説では、周りの放射線量が安定していて変動が少ない状況を解説してきました。ですが歩きながら放射線量を測定するような場合には、周りの放射線量が高くなったり低くなったりと変動する可能性があります。

このような状況では場合には、放射線量が変動して高くなったり低くなったりしているため、いくら時間をかけても測定しても偏差（%）は下がりません。

時間をかけて測定したが偏差が下がらない場合には、測定値（Sv/h）と偏差（%）の両方を記録しておくことをおすすめします。

これによって測定値は誤差がある状況であり、周りの放射線量が変化していることを読み取ることができます。

偏差の値	意味合い
偏差（%）の値が大きい 30〜100%	測定時間が短いため、より長い時間測定してください。 周りの放射線量がふらふらと変動している 急に放射線量が変化した
偏差（%）の値が小さい 1〜30%	十分な測定時間、測定できているので表示される線量率を正しい値として読むことができる。 周りの放射線量の変動が少なく安定している。

3　仕様・付属品

1.5　付属品

以下の付属品があります。購入時にご確認ください。

内容	個数
BDKG-02 ガンマ線・放射線測定ユニット	1
ケーブル 1m 以上	1
AC アダプター(オプション)	1
取扱説明書（本書）	1
校正証明書	1

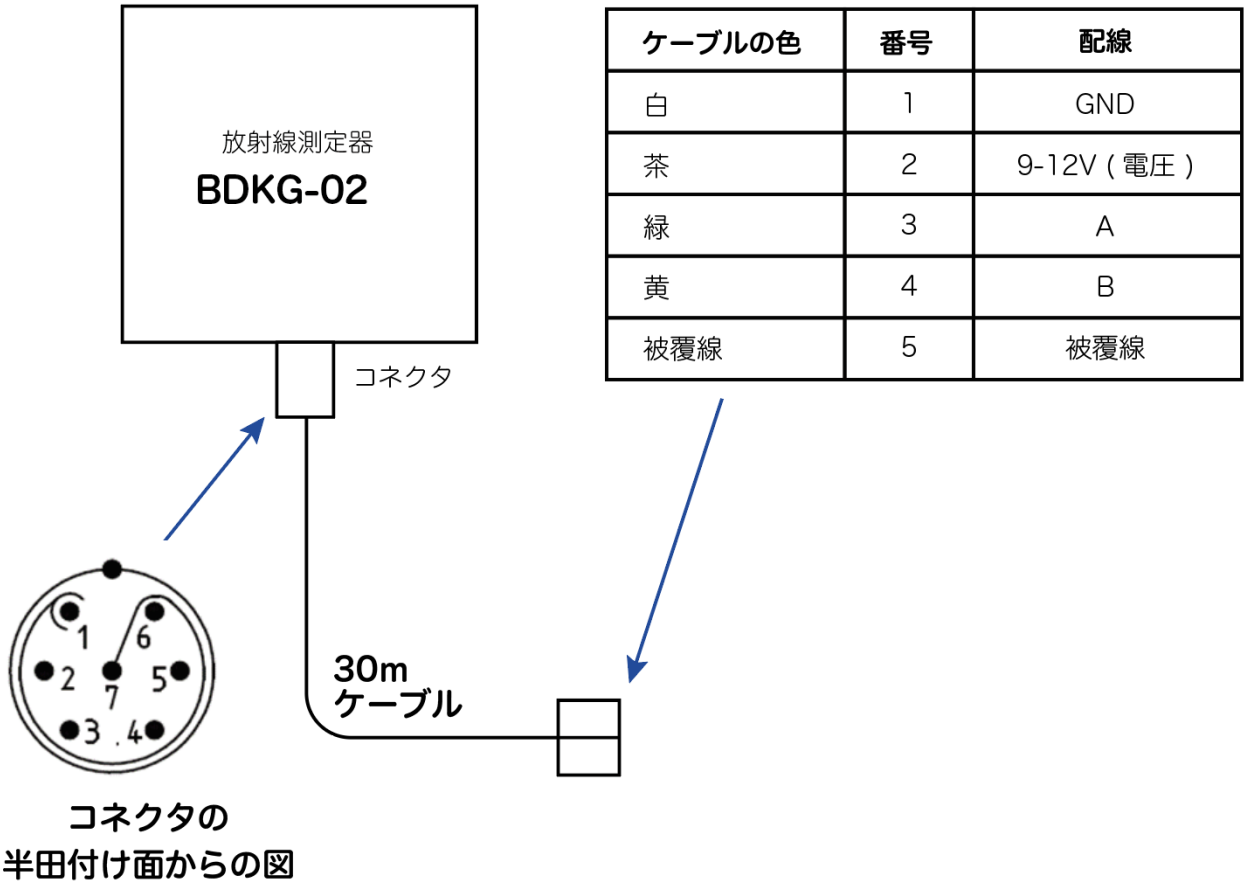
特注の場合には、RS485-USB 変換器を含みます。

1.6　仕様 BDKG-02

検出器	GM 管
線量率の測定	0.1 µSv/h ~ 10 Sv/h
積算線量の測定	BDKG-02 : 積算線量は測定できません。
線量率の測定の固有相対誤差	±15%
ガンマ線エネルギー範囲	60 keV～ 3 MeV
感度	4 cps/(µSv/h) - (^{137} Cs)
エネルギー依存性	-25 ~ +35 %
検出器寿命	100 Sv
外部デバイス接続インターフェース	BDKG-02 : RS485 シリアル通信
外部電源の電圧	BDKG-02 : DC 6 ~ 12 V
消費電力	1 W 以下
動作温度範囲	-40 ~ +50°C
湿度(35°C以下・結露なし)	95%まで
防水・防塵 (IEC 529:89)	IP 57
寸法（検出器のみ）	φ55 x 260 mm
重さ（検出器のみ）	500 g

4　シリアル通信

1.7　回路図



ケーブルの色	番号	配線
白	1	GND
茶	2	9-12V (電圧)
緑	3	A
黄	4	B
被覆線	5	被覆線

測定器 BDKG-02 に接続されるコネクタのピン配置は、はんだを行う面から見た図＝コネクタの内部から見た図です。製品となったコネクタを外側から見る場合には、逆転することになるので注意してください。

- RS485 接続を行う場合には、GND, A,B の 3 線を使います。
- 電源には AC/DC アダプターを利用して DC 6~12 V の電圧をかけてください。

5　RS485 シリアル通信

1.8　RS485 接続の基本

RS485 接続を始めて行う場合には、書籍、WEB などで学習することをお勧めします。こちらでは簡単に RS485 ネットワークについて解説いたします。

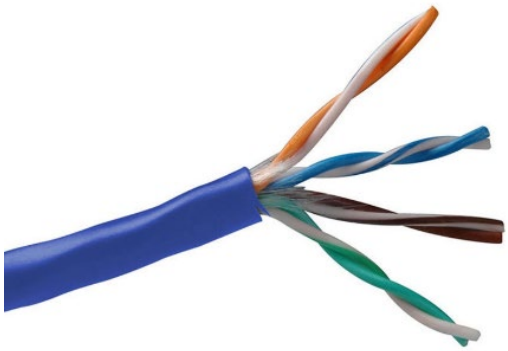
RS485 は半二重通信です。半二重通信とは２本のケーブルをひとつの伝送路として送信、受信を交互に使うという意味です。

つまり通信ケーブルへデータを送信しているときは同時には受信できません。逆に受信している時は送信できません。これが半二重通信です。

プログラムを作って通信を始めるときには、このルールに従ってプログラムを開発してください。つまり送信したら、受信するまで時間的に待ちます。受信ができたら、次の送信を行います。

1.9　RS485 で使うケーブル

RS485 ケーブルは GND, A, B の３本の信号線で成り立っています。このうち２本 (A,B)は「より線」（ねじってあるケーブル）が必要です。



市販の LAN ケーブルは「より線」が使われていますので、これを利用するのが一番、安価な方法です。

お勧めの LAN ケーブルは、1.29　RS485 ケーブル (p.38)を見てください。

1.10　機器の接続

RS485 接続の基本形は、こちらの形です。

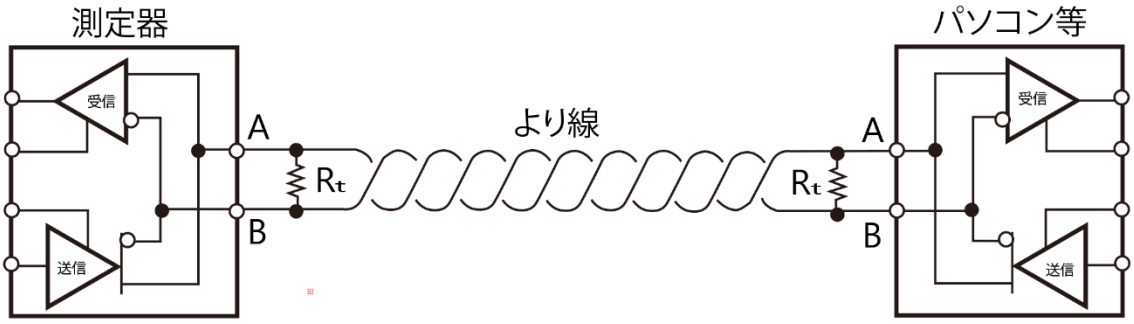


図 5-1

左側に放射線測定器、右側にパソコンの 2 台で RS485 通信する最もシンプルな形です。A,B のケーブルは、より線のイメージになっています。それぞれの左右にあるデバイスには、送信機と受信機があります。送受信が同じ場所に接続されているため、同時には通信できないケーブル（半二重通信）になっています。

左側のパソコンから右側の放射線測定器に命令を送ると、放射線測定器は応答をパソコン側に返すことになります。

1.11　反射と終端抵抗

ネットワークに送信された信号は、ケーブルの端まで来ると跳ね返って、再び逆走して戻っていきます。これを反射と呼びます。

反射が起こると、次に送信する信号や測定器からの応答信号と重なってしまい正しいデータを通信できない状況になります。これを防止するには、ケーブルの両端に終端抵抗 R_T を入れることで解決できます。図 5-1 では、AB の「より線」をまたぐ形で、ケーブルの両端（測定器とパソコンの近く）に終端抵抗 R_T (120Ω) が入っています。

終端抵抗を入れると、ネットワーク内に送信された信号は反対側の機器の受信部にたどり着いた後、終端抵抗に吸収されてなくなります。このように終端抵抗は、反射をなくすという重要な役割があります。終端抵抗をいれるのが RS485 通信の基本なので入れるようにしてください。

1.12　複数機器の接続

ネットワーク上に複数の機器を接続することもできます。たとえばパソコン 1 台で、２本(以上)の放射線測定器を RS485 ケーブルに接続できます。

複数台数を接続する場合には、下図のような接続になります。この場合でも終端抵抗は両側にのみあることを注意してください。

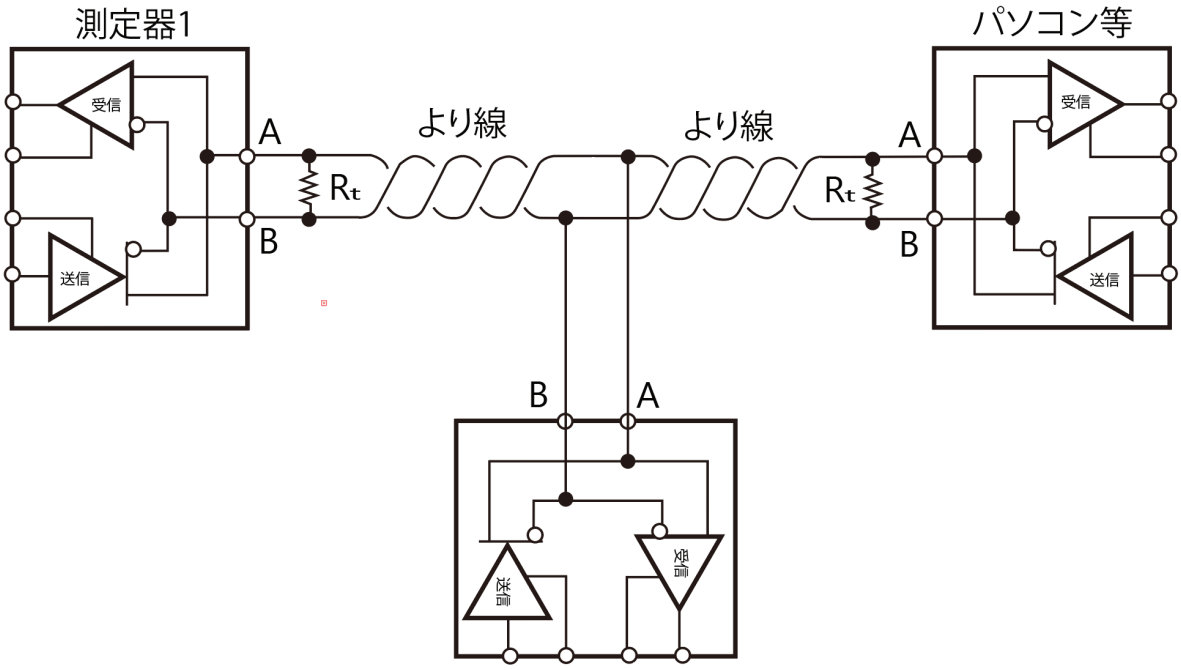


図 5-2

こちらのガイド（英語）が参考になります。

https://www.go4b.com/usa/technical-support/product-manuals/t500-hotbus/rs485-wiring-guide.pdf

終端抵抗はケーブルの端／端のみに入れてください。中間の機器（図 5-2 の測定器２）に終端抵抗を入れるとそれより向こうの機器は通信できなくなります。

機器の個数がさら増えた場合には、こちらのような接続になります。

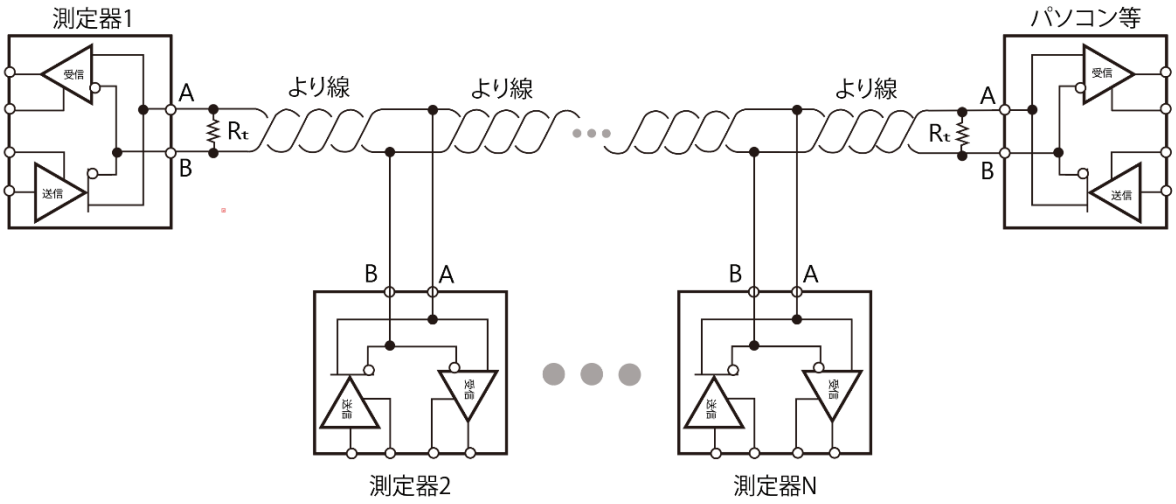
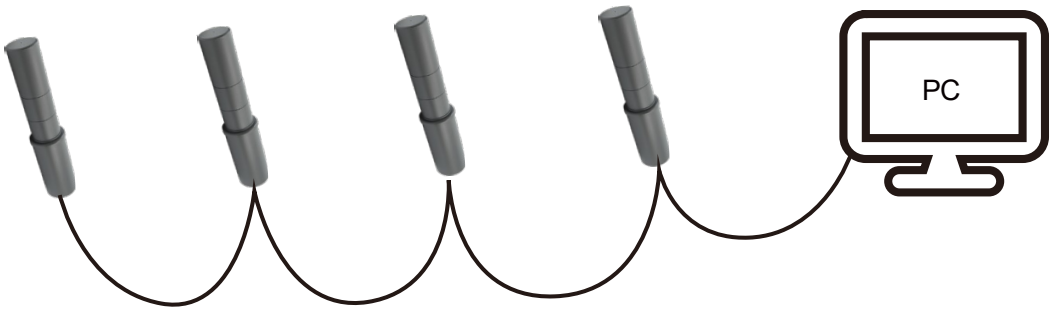


図 5-3

1.13　数珠つなぎが基本

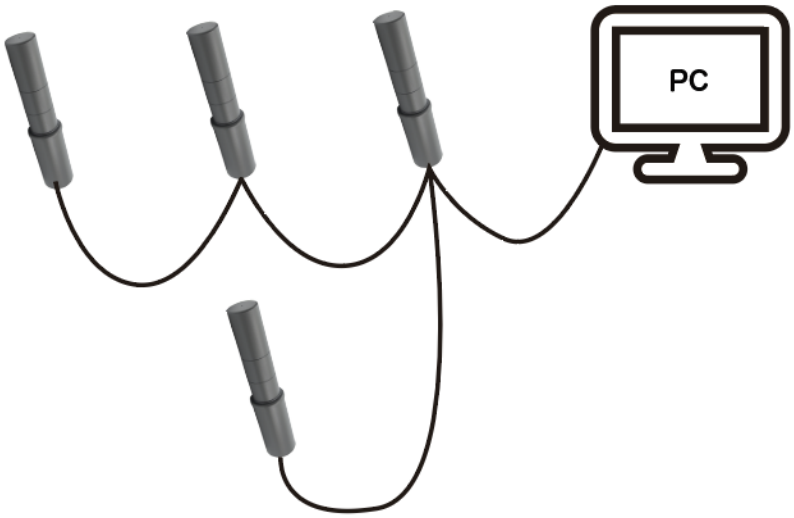
RS485 接続は数珠つなぎが基本です。

正しいつなぎ方



間違ったつなぎ方

RS485 は途中で分岐した場合、通信できなくなります。
必ず数珠つなぎで複数の測定器や、パソコンなどのデバイスをつなげてください。



1.14 例外

例外として以下の条件の場合には、終端抵抗がなくてもうまく通信できます。

- ケーブルが短い場合（10 m 以内）
- 短いケーブルの両端に通信機器（２台）があるだけのシンプルな構成

このような場合には終端抵抗を入れなくても通信できます。ですが RS485 接続の基本仕様では終端抵抗があることが前提です。終端抵抗を使わない場合には、将来、ケーブルを長くした場合にうまく通信できない状況になることがあります。そのため設計段階から終端抵抗をつけることをお勧めします。

1.15 RS485-USB 通信機 (別売り)

オプションの RS485--USB 変換器です。この機器の場合には、終端抵抗をあり・なしを簡単に切り替えられるようになっています。

終端抵抗のあり・なしは、機器の裏蓋をドライバーで外して、ジャンパースイッチで切り替えることができます。

- ショート（短絡）状態＝終端抵抗あり（ON）
- 取り外す（解放）状態＝終端抵抗なし（OFF）

6　通信プロトコル

1.16　ビットとバイト

こちらの解説では、ビット、バイトといった言葉を使います。

8 ビットは、 0， 1 のデータが 8 個まとまった単位で、8 ビット＝1 バイトと呼びます。バイトを表す記法として、0x01, 0x02, … 0xFF といった表記を使います。このあたりが分からない場合には、インターネット等で学習してください。

1.17　シリアル接続の設定

放射線測定器 BDKG-02 のシリアル接続には、以下の情報でシリアル通信機器の設定を行ってください。

ボーレート *	9600 (初期値)　　(9600, 1200 のいずれかのみ利用可能)
パリティ	なし
ストップビット	1
データビット	8
ハンドシェイク	なし

この情報は、プログラム言語でシリアルポートを開く時に必要となる情報です。どんなプログラム言語でも、 serial.open(9600,なし,1,8,なし) といった命令文で設定できるはずです。

1.18 機器アドレス

測定器には機器アドレス（1 バイト）が割り振られています。工場出荷時は、機器アドレスは 0x01 に割り振られています。

機器アドレス　*	0x01 (初期値)

＊の項目は、通信プロトコルを使って変更可能です。

放射線測定器は、自分の割り当てアドレスに対しての命令だけ応答するようになっています。

1 本の RS485 ネットワーク内に複数の放射線測定器を配置する場合には、接続した放射線測定器のアドレスを 0x01, 0x02, 0x03 と分けておくことで、1 本の RS485 ケーブルで複数の機器を個別に制御することができます。アドレスを変更するコマンドが用意されています。

1.19 通信のやり取り

ここではパソコンと放射線測定器との 1 対 1 の通信を例に紹介します。

1. パソコン側から RS485 ネットワークを通じて、放射線測定器に命令コマンドを送ります。

2. 放射線測定器は、これを受信してから測定値などの情報をケーブルに送り出します。

3. パソコンは測定値を受信できます。

4. 放射線測定器は、命令を受けて初めて測定値を返します。毎秒ごとに測定値が必要な場合には、パソコン側から毎秒ごとに命令を送り、測定値を受信する必要があります。

5. これを繰り返すことで連続した測定となります。

1.20 通信パケット

測定器側に送信するシリアル通信のパケット構成は、緑色＋黄色の部分です。
白色部分は、パケット前後にある無通信時間です。

開始 無送信時間	機器 アドレス	命令バイト	データバイト数 ＝N	データ領域	誤り確認 コード	終了 無送信時間
T3~T5以上	1バイト	1バイト	1バイト	Nバイト	2バイト	T3~T5以上

ここで T は 1 バイトを送信するためにかかる時間です。

パケット（緑色＋黄色部分）は順番のバイトから構成されています。

1. 最初に、放射線測定器のアドレス（初期値 0x01 ）
2. 続いて、命令コマンド (0x02, 0x03.. 等)
3. 続いて、データ領域のバイト数(=サイズ)を示す 1 バイト
   (0x01~0xFF)
4. 続いて、送信・受信するデータ本体のバイト列
5. 最後に、誤り訂正コード（2 バイト）(黄色部分)の構成です。

RS485 ネットワークに送信する通信パケット（緑色）は、前後に 1 バイトの送信にかかる時間の 3-5 倍以上の時間の無送信時間(白部分)が必要です。たくさんの命令を送る場合には、パケット同士を時間的に分離して時間的に区切りながら送信してください。このあたりは自分でプログラムを組む際に制御する必要があります。

1.21　命令コマンド一覧

BDKG-02 の命令コマンドはこちらです。この日本語解説書では基本コマンドのみをご紹介します。

命令バイト	命令の内容
0x03	線量率の取得(単位 nSv/h)
0x1A	線量率測定の偏差の取得 (単位 %)
0x0A	線量率の測定をリセットします。最初から測定しなおします。 これにより偏差が99%となり平均されている数値が破棄されたことを意味します。

これ以外にも英語版の通信プロトコルマニュアルにはすべてのコマンドについての解説があります。

1.22 エンディアン

複数バイトにわたって一つの値を保存する場合、２タイプの保存方法があります。たとえば、0x000A という２bytes の値（１６ビット整数等 0x00 , 0x0A）を保存する場合、２通りの保存方法があります。

ビック・エンディアン	リトル・エンディアン
0x00 , 0x0A	0x0A, 0x00
通信時には、0x00 を先に送信してから次に 0x0A を送信します。受け取りもこの順になります。	通信時には、0x0A を先に送信してから次に 0x00 を送信します。受け取りもこの順になります。

参考：Google ：エンディアンについて

https://www.google.com/search?q=%E3%82%A8%E3%83%B3%E3%83%87%E3%82%A3%E3%82%A2%E3%83%B3

通信プロトコルでは、２つのエンディアンが以下のように混ざっています。通信プロトコルに記載があれば、それに従ってください。

ビック・エンディアン	リトル・エンディアン
測定データ（線量率、積算線量） 線量率の偏差（%）など 測定器から返ってくるすべての測定データは、ビック・エンディアンで格納されています。	通信パケットの最後に 2bytes として追加されている誤り確認コードは、リトル・エンディアンになっています。

世の中にあるほとんどのプログラム言語(C#, Java,,)は、byte 配列から浮動小数点などに変換する関数があります。これは「リトル・エンディアン」だけを対象としています。「ビック・エンディアン」の byte 配列を変換する場合には、バイト列の順番を逆にしてから、変換する必要があります。

1.23 サンプルコードのダウンロード

３タイプのサンプルコードを用意しています。

実行環境	言語	接続方法	サンプルコードのダウンロード先
Raspberry Pi	Python	SPI 接続	下記１
Raspberry Pi	Python	USB 接続	下記２
Windows パソコン	C#	USB 接続	下記３

1) Phython-SPI
https://taroumaru.jp/app/webroot/download/samplecode/bdkg02/python-spi/python-spi.zip

2) Python-USB
https://taroumaru.jp/app/webroot/download/samplecode/bdkg02/python-usb/python-usb.zip

3) Windows-USB (C#)
https://taroumaru.jp/app/webroot/download/samplecode/bdkg02/cs-window-usb/BDKG02_RS485_CS.zip

1.24　誤り確認コードの計算方法

通信パケットの最後２バイトは誤り確認コード（下図の黄色部分）です。

BDKG-02 の誤り訂正符号はとても単純な形になっており、先頭１バイトを除く残りのバイトを足し合わせた数値が２バイト（リトル・エンディアン）で格納されています。

開始 無送信時間	機器 アドレス	命令バイト	データバイト数 ＝N	データ領域	誤り確認 コード	終了 無送信時間
T3~T5以上	1バイト	1バイト	1バイト	Nバイト	2バイト	T3~T5以上

1.24.1　python のサンプルコードです。

先頭１バイトを除く残りのバイトを足し合わせた数値を２バイトにいれています。

```
    # 受信したバイト列に入っている CRC 誤り符号が正しいか確認します。
    # 正しければ、true, 間違っていれば false
    # 正しければ、その次の受信処理に進めます。
    def makecrc_spk(self,buf, len):
        crc = 0
        for pos in range(1, len):
            crc += buf[pos] & 0xFF
        b = bytearray(2)
        b[0] = crc >> 0 & 0xFF
        b[1] = crc >> 8 & 0xFF
        return b
```

1.24.2　java のサンプルコードです。

先頭 1 バイトを除く残りのバイトを足し合わせた数値を 2 バイトにいれています。

```
  // BDKG02 の誤り訂正符号は、1 バイト目から足し上げた数という単純なものです。
  public byte[] makecrc_spk(byte[] buf, int len) {
    int crc = 0;
    for (int pos = 1; pos < len; pos++) {
      crc += buf[pos] & 0xFF; // http://masao6739.blog89.fc2.com/blog-entry-21.html より
    }

    byte[] b = new byte[2];
    b[0] = (byte) (crc >>> 0);
    b[1] = (byte) (crc >>> 8);
    return b;
  }
```

1.24.3　C# のサンプルコードです。

先頭 1 バイトを除く残りのバイトを足し合わせた数値を 2 バイトにいれています。

```
        public byte[] makecrc_spk(byte[] buf, int len)
        {
            int crc = 0;
            for (int pos = 1; pos < len; pos++)
            {
                crc += buf[pos];
            }
            byte[] b = new byte[2];
            b[0] = (byte)( (crc >> 0) & 0xFF );
            b[1] = (byte)( (crc >> 8) & 0xFF );
            return b;
        }
```

1.25　線量率の取得 0x03

線量率（単位 nSv/h） の動作例です。

送信で５バイトの命令文を送ると、放射線測定器は線量率を含んだ９バイトのデータを返してきます。

送信: 01-03-00-03-00

受信: 01-03-04-47-98-43-00-29-01

線量率　　: 0.076130859375 [uSv/h]

送信パケット

項目	バイト
機器アドレス	0x01
コマンド	0x03
データ数	0x00
誤り訂正符号(Lo)	0x03
誤り訂正符号 Hi)	0x00

受信パケット

項目	バイト
機器アドレス	0x01
コマンド	0x03
データ数	0x04
3 バイト浮動小数点による線量率の値	0x47 0x98 0x43
ステータスバイト (使わない)	0x00
誤り確認(Lo)	0x29
誤り確認(Hi)	0x01

意味の解説: データ数 0x04＝4 バイトのデータが入っていることを示しています。

ここで誤り訂正符号は 03-00 = 0x0003 ですが、これは先頭を除くバイトを足し合わせた数になっています。格納方法はリトルエンディアンです。受信命令の誤り訂正符号を確認することで、正しいデータだけを画面に表示することができます。

1.26 浮動小数点

通常、浮動小数点というと４バイト= 32 bit ですが、BDKG-02 の浮動小数点データは、少し変わった形式で３バイト=24 bit で保存されています。

この３バイトを使い、以下の演算をすることで線量率 (nSv/h)の値を得ることができます。

３バイトを　X2 , X1, S　とする場合

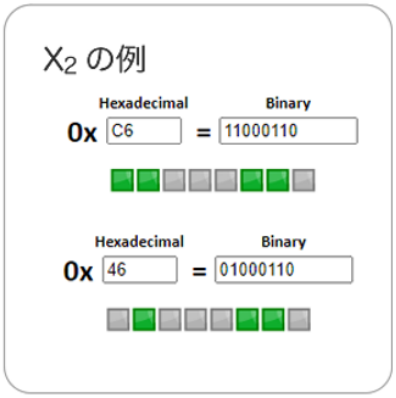
$$y = \frac{S \cdot X_1}{2^{\{16-(X_2 - 0x40)\}}}$$

ここで

X_1：2バイトで表現された符号なし整数

S ：符号（１または－１）
符号は X2 の上位８ビット目が１の場合マイナス
X2の上位 8ビット目が0の場合プラス

X_2：1バイトの数
符号 S がマイナスの場合、
８ビット目を除いた値とする。
例 0xC6 は 0x46 に変換する



例:　ビックエンディアン
0x44 , 0xA0, 0x00

$$y = \frac{+1 \cdot 0xA000}{2^{\{16-(0x44-0x40)\}}}$$

$$= \frac{40960}{2^{12}} = 10.00$$

注意：ここで分子の 0xA000 は符号なし整数として変換する必要があります。

1.26.1　C# のサンプルコードです。

３バイトから浮動小数点をつくるための例です。

```
        private double converter(byte[] buffer)
        {
        byte a = buffer[0];
        byte b = buffer[1];
        byte c = buffer[2];

        int sign = ((a & 0x80) == 0)?1:-1;
        if( sign < 0)
        {
            a = (byte)(a & ~0x80);
            }
            double s = Math.Pow(2,(16-(int)(a-0x40)));
            return sign*(double)(b*256+c)/s;
        }
```

1.26.2　Java のサンプルコードです。

３バイトから浮動小数点をつくるための例です。

プログラム言語 Java の場合には、符号なし整数がプログラム内にないため変換には注意が必要です。C#言語など符号なし整数を扱える言語は変換クラスなどで簡単に変換できます。

```
// ******************************************************
// BDKG02 は、3 バイト浮動小数点という変わった形式で値が保存されています。
// この計算によって（double）に変換しています。
// ******************************************************
private double dose_rate_converter(/*Byte*/byte[] buffer) {
  /*Byte*/int a = buffer[0] & 0xFF;
  /*Byte*/int b = buffer[1] & 0xFF;
  /*Byte*/int c = buffer[2] & 0xFF;
  int sign = ((a & 0x80) == 0) ? 1 : -1;
  if (sign < 0) {
    a = (/*Byte*/byte) (a & ~0x80);
  }
  double s = Math.pow(2, (16 - (int) (a - 0x40)));
  return sign * (double) (b * 256 + c) / s;
}
```

1.26.3　Python のサンプルコードです。

３バイトから浮動小数点をつくるための例です。

プログラム言語 Python は変数の型がないため符号なし整数がプログラム内にないため変換には注意が必要です。

```
    # 線量率は、3 バイト浮動小数点という奇妙な形式で保存されています。
    # 通常、浮動小数点は 4 バイトです。
    # この見たことがない形式の浮動小数点を、普通の数字に直すための
    # アルゴリズムがこちらになっています。
    def dose_rate_converter(self, buffer):
        a = buffer[0] & 0xFF
        b = buffer[1] & 0xFF
        c = buffer[2] & 0xFF
        sign = 1 if (a & 0x80) == 0 else -1
        if sign < 0:
            a = (a & ~0x80)
        s = math.pow(2, (16 - (int)(a - 0x40)))
        return sign * (b * 256 + c) / s
```

1.27　線量率の偏差（%）を取得する 0x1A

線量率の偏差（%）を取得する命令です。

送信: 01-1A-00-1A-00

受信: 01-1A-01-0B-26-00

線量率の偏差　　: 11 [%]

送信パケット

項目	バイト
機器アドレス	0x01
コマンド	0x1A
データ数	0x00
誤り訂正符号(Lo)	0x1A
誤り訂正符号 Hi)	0x00

受信パケット

項目	バイト
機器アドレス	0x01
コマンド	0x1A
データ数	0x01
偏差(%)	0x0B
誤り確認(Lo)	0x26
誤り確認(Hi)	0x00

意味の解説: データ数 0x01＝1 バイトのデータが入っていることを示しています。

偏差 0x0B は整数に直すと 11 なので 11% となります。

1.28　線量率の測定リセット 0x0A

線量率の平均化をリセットする命令です。

送信で６バイトの命令文を送ると、５バイトのデータを返してきます。

応答には特に何も含まれていません。

線量率の平均値をリセットすると、その場所の放射線量を０から測定開始します。測定器の電源を入れ直したような状態となります。たとえば新しい場所に測定器を持って行ったときに、これまでいた場所の平均値を引きづってほしくないような場合には、このリセット命令を送信してください。測定器がこの命令を受けると、誤差が 99%となり平均値がリセットされたことが分かります。

送信: 01-0A-01-00-0B-00

受信: 01-0A-00-0A-00

これで平均化がリセットされて、線量率の測定が０から行われます。

送信パケット

項目	バイト
機器アドレス	0x01
コマンド	0x0A
データ数	0x01
データ	0x00
誤り訂正符号(Lo)	0x0B
誤り訂正符号 Hi)	0x00

受信パケット

項目	バイト
機器アドレス	0x01
コマンド	0x0A
データ数	0x00
誤り確認(Lo)	0x0A
誤り確認(Hi)	0x00

7　その他

1.29　RS485 ケーブル

RS485 ネットワークに利用できる LAN ケーブルをご紹介します。

サンワサプライ CAT 5e

KB-C5L-CB300W

エレコム CAT 5e

LD-CT2/BU300/RS

市販の LAN ケーブル(CAT 5e)の中には、ケーブル長が 30m を超えてくると RS485 通信ができないものもあります。こちらのケーブルは 100m 程度まで確認しております。LAN ケーブルは安価なので購入して試してみるのがよいかと思います。

1.30 その他の通信機能のご紹介

この取扱説明書では、３つの命令のみを解説しています。

- 線量率(Sv/h)
- 線量率の偏差（%）
- 測定リセット

これ以外に以下の機能が利用できますが、仕様は英文マニュアルとなっています。

- 測定器のアドレスの変更

  測定器のアドレスは 0x01 が初期値ですが、これを変更することができます。たとえば 1 本の RS485 ケーブルに 2 台、3 台と測定器を接続する場合には、それぞれが別のアドレスをもつ必要があります。アドレスを変更することで特定の測定器のみと通信することができます。

- 通信速度の変更(初期値 9600)

  RS485 ケーブルが 100 メーター , 500 メーターと長くなる場合には、通信速度を低くすることで通信がやりやすくなります。

  BDKG-02 は、1200, 9600 の 2 つの通信速度を使うことができます。

  RS485 のケーブルの長さは、理論的には 1200m　まで延長できますが実際には 500m 程度までと考えてください。電源ケーブルは除きます。電源はこれほど長くは延長できません。

8　Raspberry Pi との接続

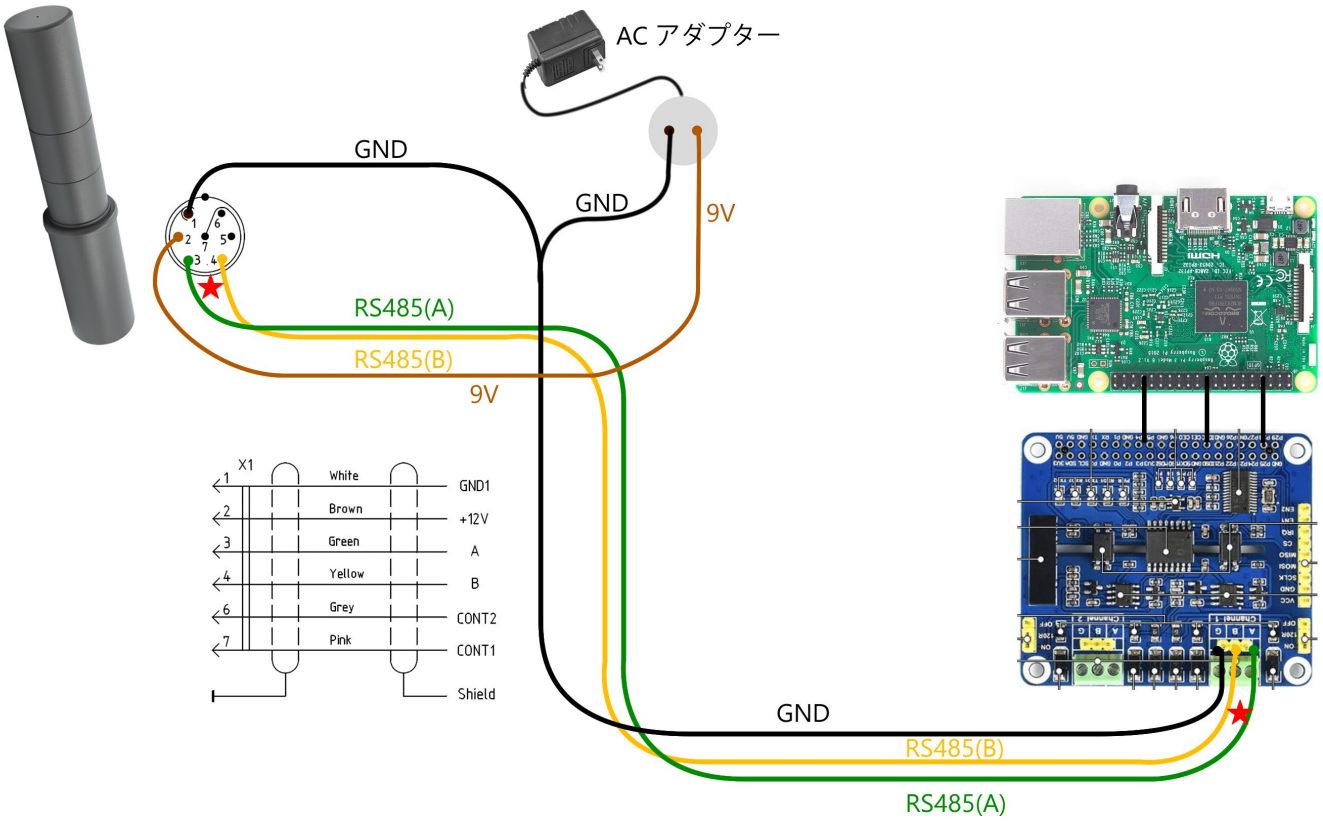
この章では、Raspberry Pi と BDKG-02 を接続する方法についてご紹介いたします。

２タイプの接続について解説しています。

SPI 接続	HAT （帽子）と呼ばれる基板を Raspberry Pi の上に載せて接続する方法です。こちらの場合には電源を AC アダプターから取ります。
USB 接続	USB から電源も取れるので便利な方法です。

放射線測定器は、高線量になると消費電流も増えるため、高線量で使う場合には SPI 接続でＡＣアダプターから測定器に電源を供給することをおすすめします。USB 接続の場合には、USB 経由での電源供給には限界があるため高線量で使用した場合には測定器の電源が落ちる場合もあります。

1.31　配線図(SPI 接続の場合)



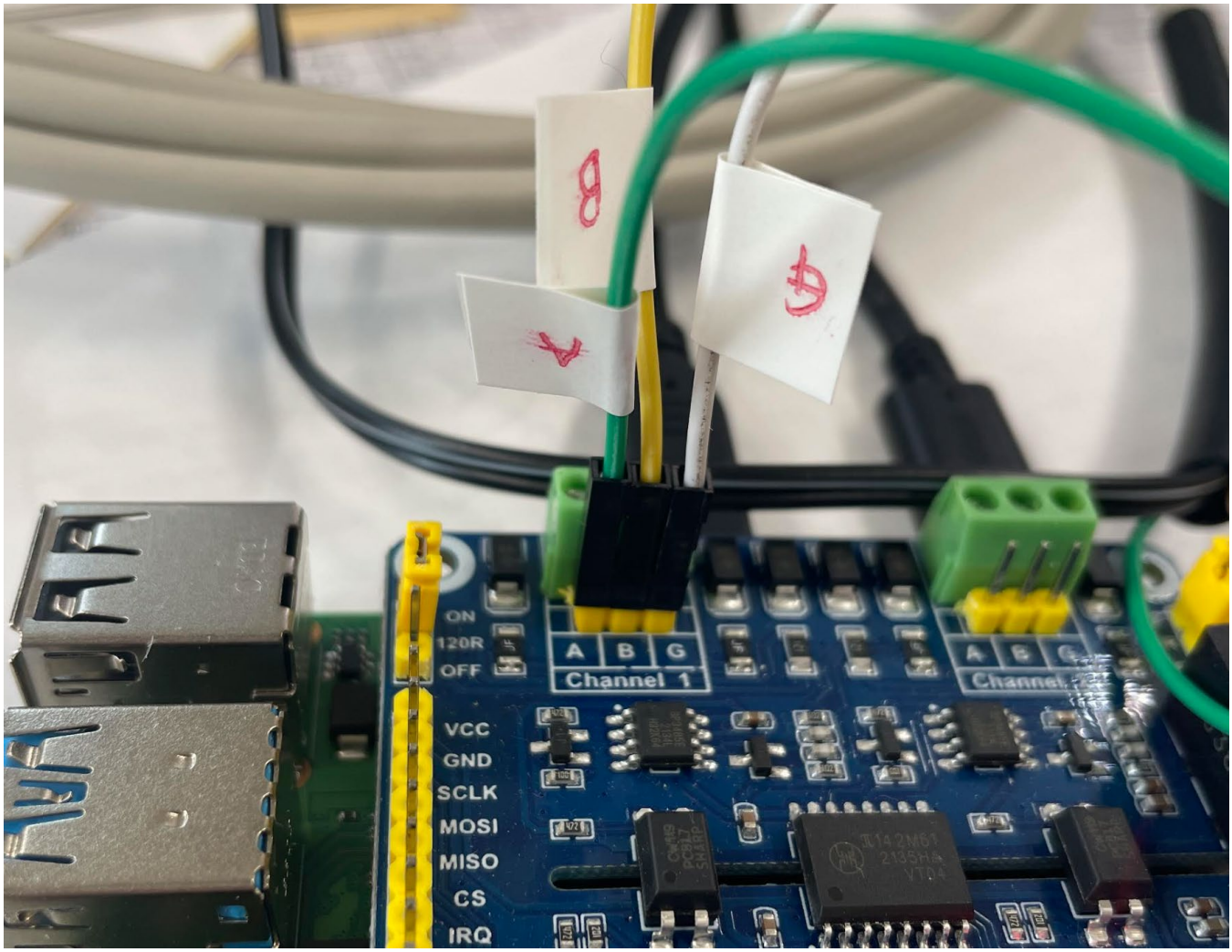
★　赤い星は、RS485(A-B)間に入った終端抵抗 120Ω です。 ケーブルが短い場合には不要ですが ケーブルが 30m 以上と長くなる場合には必ず入れてください

Raspberry Pi に RS485 接続の SPI 接続基板を乗せて接続しています。

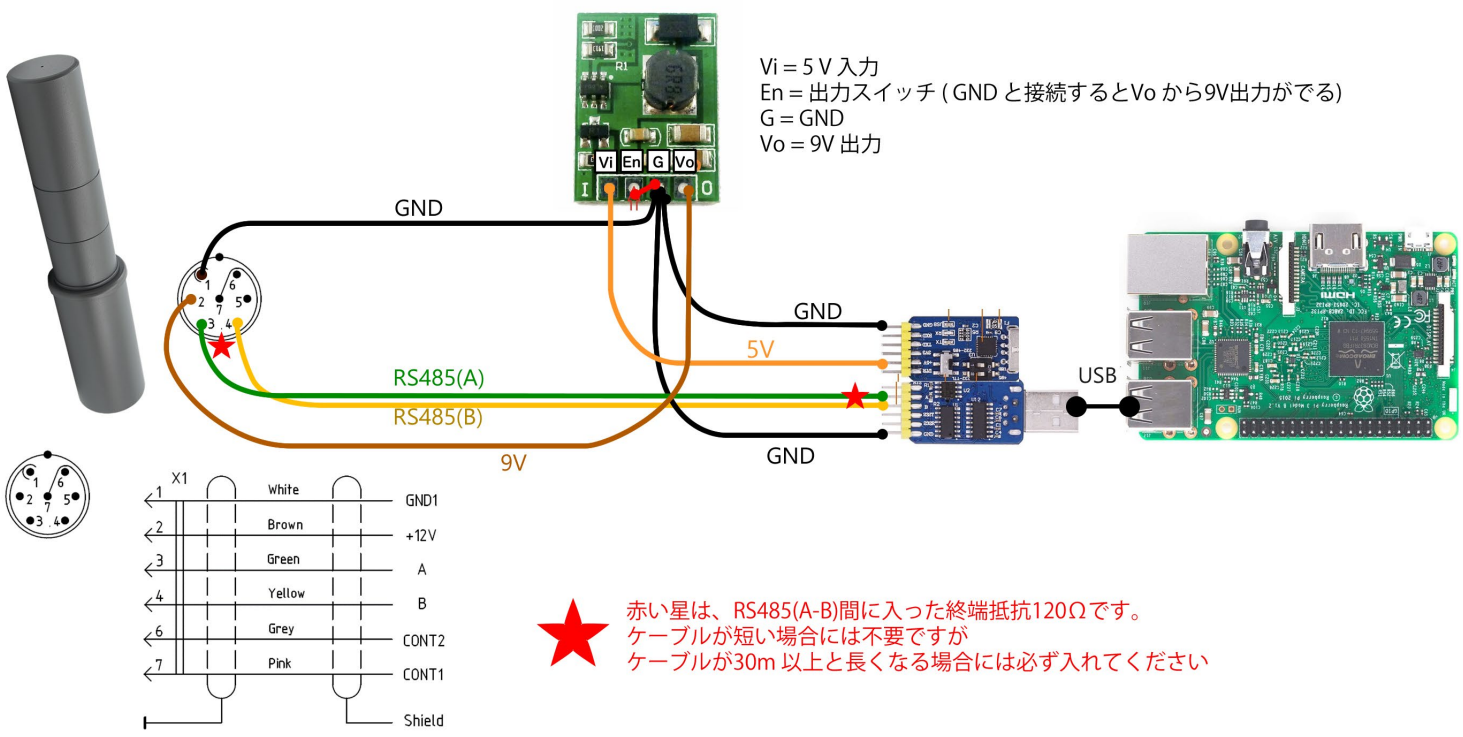
RS485 の基板は、こちらです。

- https://www.switch-science.com/products/7344
- https://www.waveshare.com/wiki/2-CH_RS485_HAT

この写真のように (A:緑) (B:黄) (GND:白) として Channel 1 に接続してください。



1.32　配線図(USB 接続の場合)



★ 赤い星は、RS485(A-B)間に入った終端抵抗 120Ω です。 ケーブルが短い場合には不要ですが ケーブルが 30m 以上と長くなる場合には必ず入れてください

USB 接続用に使用した小さい基板はこちらです。

- https://www.aliexpress.com/item/32974686257.html?spm=a2g0o.order_list.order_list_main.65.1ffc585ae6sgls
- https://www.aliexpress.com/item/32975190528.html?spm=a2g0o.order_list.order_list_main.225.1ffc585ae6sgls

1.33　テスト環境

この取扱説明書を記載するときに使用した環境です。

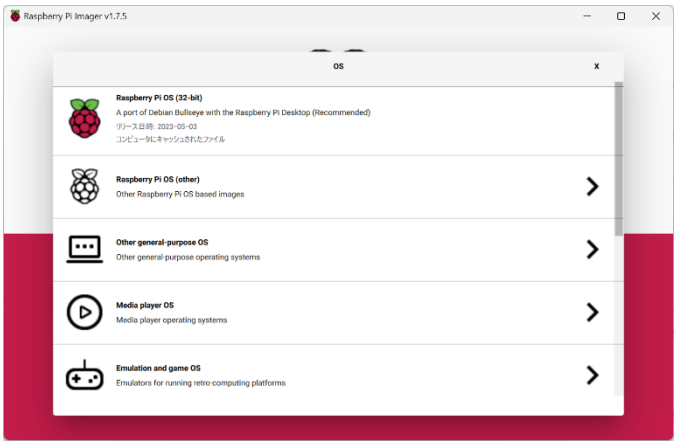
Raspberry Pi	Pi4 – Model B
プログラム言語	Python
測定器の電圧	9～12V

1.34　OS のインストール

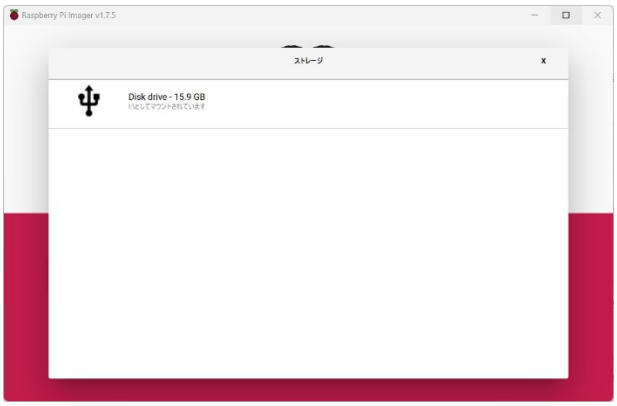
この作業には 2 時間ほどかかります。時間があるときに実行してください。スターターキットなどをお使いの場合には、この手順がすでに行われています。

1. Micro SD カードをパソコンと接続して、Raspberry Pi 用の OS ソフトウェアをインストールします。
2. https://www.raspberrypi.com/software/
   にアクセスして、imager_1.7.5.exe　をダウンロードします。
   https://downloads.raspberrypi.org/imager/imager_1.7.5.exe
3. これを実行します。

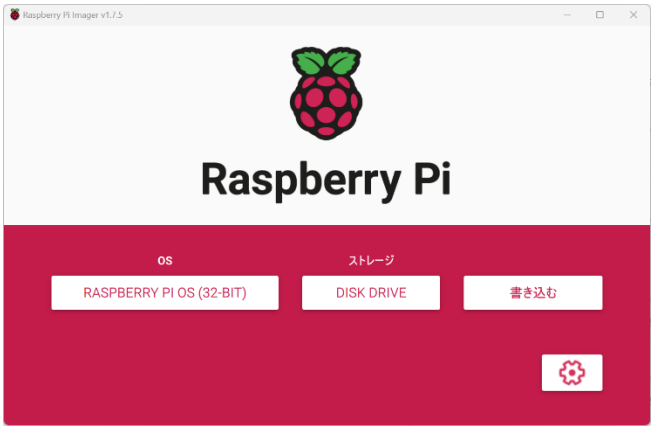
4. OS を選びます。一番上の Raspberry Pi OS (32bit) を選びます。



5. ストレージを選びます。SD カードが対象となります。



6. 書き込むボタンを押します。



ちょっと時間がかかります。

7. 書込が完了すると、書き込み内容の確認が行われます。
   書込 1 時間、確認 1 時間の合計 2 時間ぐらいかかります。

1.35 起動

Raspberry Pi にキーボード、マウス、HDMI 接続で液晶モニターを接続して起動します。詳しくは、こちらを参考にしてみてください。

https://www.indoorcorgielec.com/resources/raspberry-pi/raspberry-pi-setup/

1.36 Raspberry Pi の設定等

ここから実行できるまでのコマンドについて、順不同で記載していきます。

デバイス,OS のバージョンを知っておくことは、いろいろ役に立ちます。

```
lsb_release -a
uname -a
cat /proc/cpuinfo
```

LAN ケーブル、または WiFi と接続して Raspberry Pi 本体に割り当てられた IP アドレスも把握してください。

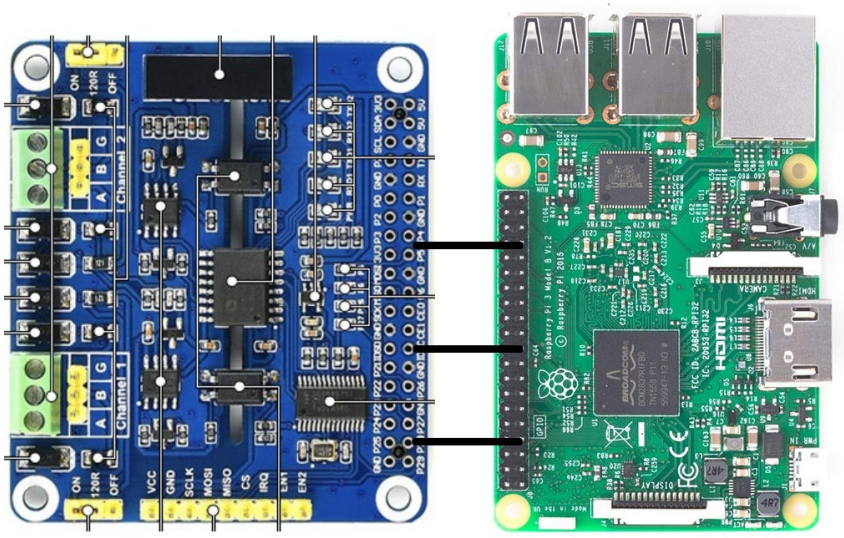
```
ip addr
```

1.37　SPI 接続の有効化

USB 接続の場合には、SPI 接続設定は不要であるためこの章を読み飛ばしてください。

SPI 接続とは、IC 間の通信を行う規格です。

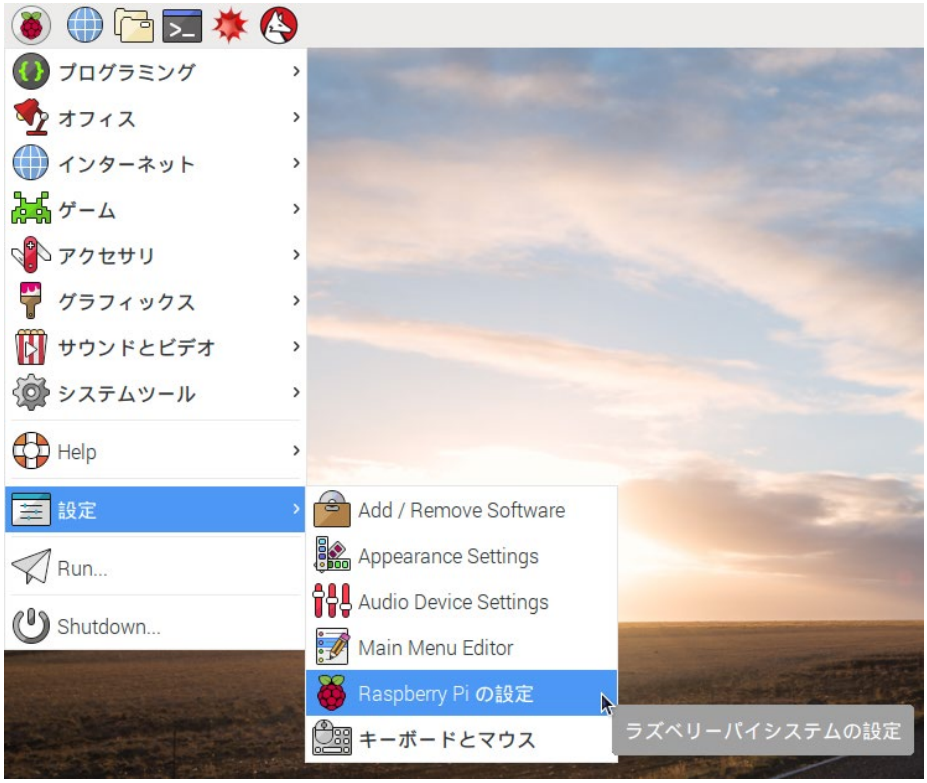
今回の場合には、RS485 通信部と Raspberry Pi の 2 つの基板の通信が SPI 接続になっています。



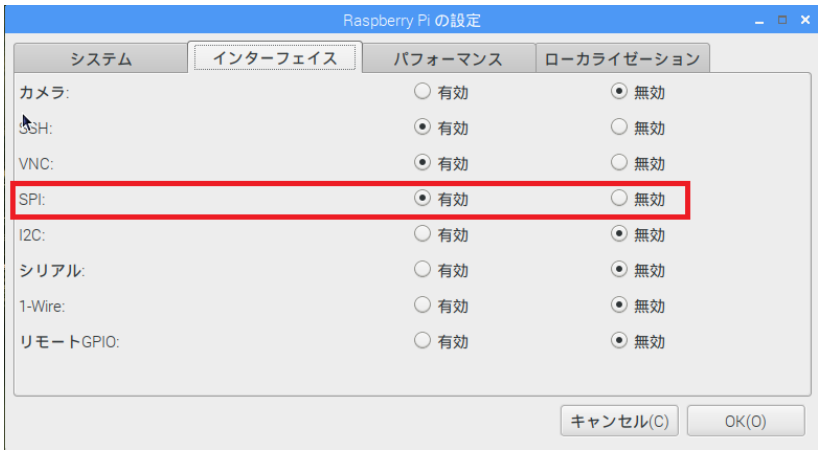
RS485通信部　　Raspberry Pi

SPI 接続を有効化する必要があります。

Raspberry Pi のデスクトップ環境から、設定-Raspberry Pi の設定を開きます。



SPI を有効にして OK をクリックします。



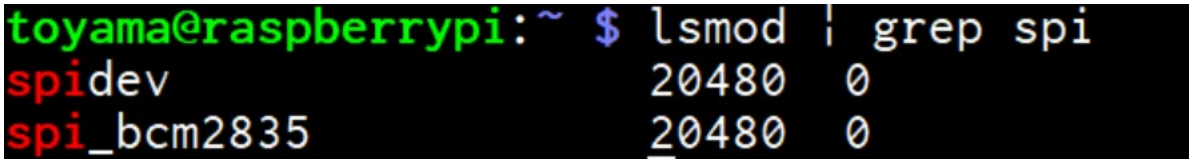
再起動すると SPI が有効となります。

設定後は、いったん電源を切ります。

```
sudo shutdown -r now
または
sudo reboot
```

起動後に、こちらのコマンドを試してみます。

```
lsmod | grep spi
```



と表示されたらＯＫです。

1.38　Emacs のインストール

Emacs は Linux 上のテキスト編集ツールです。最初は慣れないと使いづらいですが、とても役に立つのでぜひ使えるようになってください。

```
sudo apt-get install emacs -y
```

1.39　Telnet の設定

Raspberry Pi (Linux)上で Telnet サーバーを起動させることで、いつも使っている Windows PC から Raspberry Pi の操作ができるようになります。

https://spaghettiprogramming.blog.jp/archives/7193766.html

Raspberry Pi の IP アドレスは、コマンド ip addr で調べられます。

1.40　フォルダを作る

Raspberry Pi (Linux)上で、フォルダを作っておきます。

```
cd ~
mkdir ~/develop
cd develop
mkdir python
cd python
```

1.41　Tera Term (Windows 版)

Raspberry Pi (Linux)上で Telnet サーバーが稼働したら、Windows PC からログインして操作できます。

1. Windows PC に Tera Term をインストールする
2. http://ttssh2.osdn.jp/

3. TeraTeam を開いて Raspberry Pi の IP アドレスにログインします。これでパソコンから操作できるようになります。

4. もちろん Raspberry Pi のデスクトップ環境で開発してもいいと思います。

1.42　FTP サーバーの起動

FTP は、Raspberry Pi(Linux)と、手持ちの Windows PC の間でファイルを送ったり取得したするサーバーです。Windows PC で書いたプログラムを Raspberry Pi(Linux)へ送り込むには FTP サーバーを起動しておくと便利です。

手順はこちらを見てください。

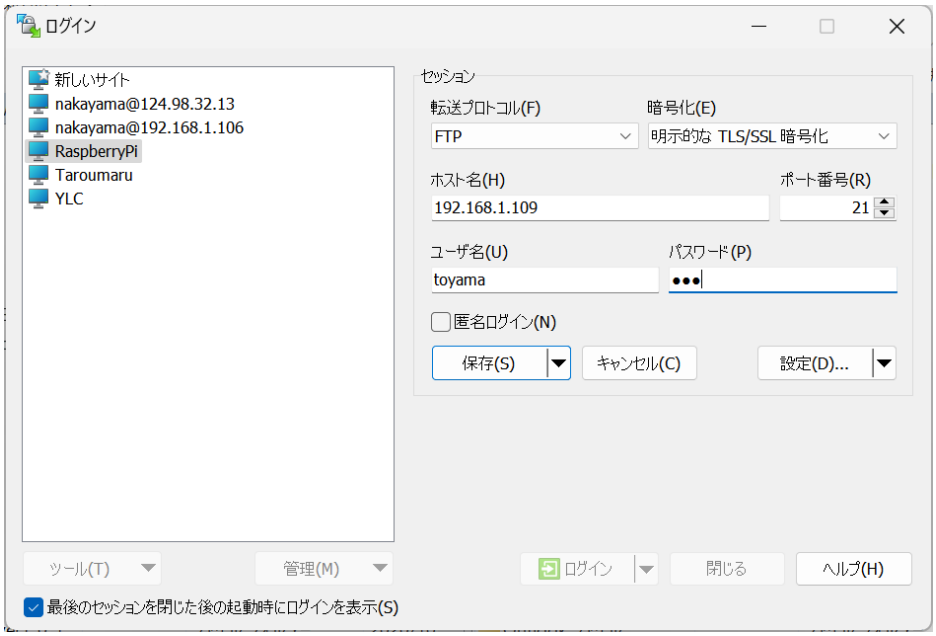
https://minicarbattle.com/raspi-remote/04-ftp/

1.43　WinSCP のインストール

Windows PC から Raspberry Pi(Linux)へ FTP 接続するには、FTP クライアントが必要です。

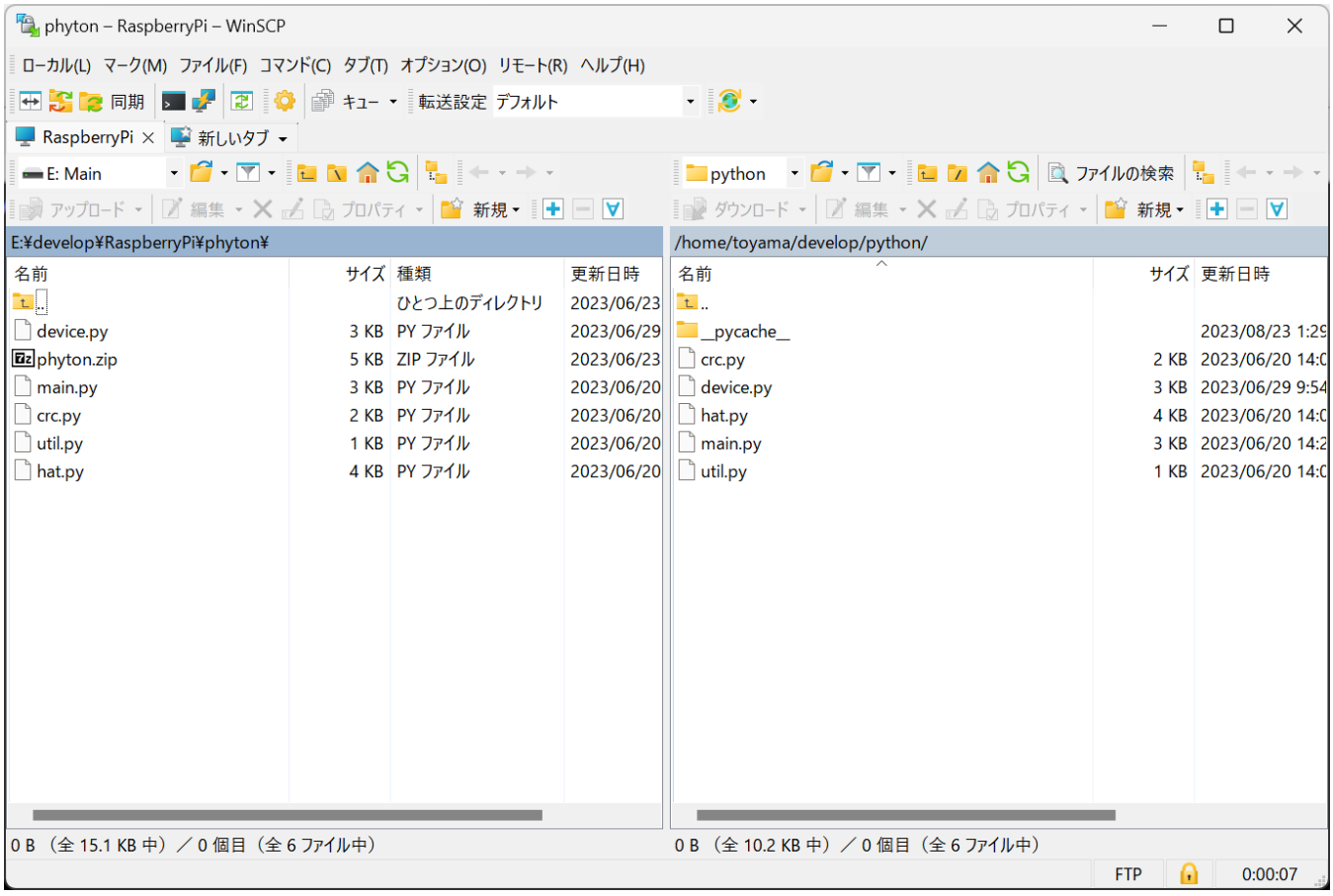
そのうちの一つ Winscp をご紹介します。

https://winscp.net/



私の場合には、Windows マシン上に動く Visual Code でソースコードを書いて、FTP で RaspBerry Pi マシンに送り込んで、実行する、という手順を繰り返しました。

このソフトウェアで python のサンプルプログラムを Raspberry Pi 側のフォルダ /home/toyama/develop/python に送り込みます。



1.44　SCI 接続の起動設定

USB 接続の場合には、SPI 接続設定は不要であるためこの章を読み飛ばしてください。

これは、RaspberryPi の上に載せた RS485 接続ポートを認識させる作業です。

sudo emacs /boot/config.txt
のファイルの一番最後の行に、この 1 行を追加します。
dtoverlay=sc16is752-spi1,int_pin=24
ここで再起動が必要です。
sudo reboot
再起動後に、
ls /dev/gpioc* /dev/ttySC* を実行してみてください。
/dev/gpiochip0　/dev/ttySC0　/dev/ttySC1　の 3 つのファイルが見えたら、OK です。

これについては、ここに書いてあります。

- https://www.waveshare.com/wiki/2-CH_RS485_HAT

設定後に、デバイスファイルができているか確認してみてください。

```
toyama@raspberrypi:~ $ ls /dev/gpioc* /dev/ttySC*
/dev/gpiochip0  /dev/gpiochip1  /dev/ttySC0  /dev/ttySC1
```

lsmod | grep spi
を実行してみます。

このようになっていれば、うまく認識されています。

```
toyama@raspberrypi:~ $ lsmod | grep spi
spidev                  20480  0
regmap_spi              16384  1 sc16is7xx
spi_bcm2835aux          16384  0
spi_bcm2835             20480  0
```

1.45　その他

便利な圧縮コマンドも入れておく。

```
sudo apt-get install p7zip-full
```

1.46　python の開発環境

python はすでにインストールされているため、python からシリアル接続するライブラリを入れておきます。

```
python --version
で　Python がインストールされていることを確認
２つのライブラリを入れておく。
sudo pip install pyserial
sudo pip install rpi.gpio
```

1.47　Python サンプルプログラムの実行

Raspberry Pi を使って実行した例です。

サンプルプログラムは、こちらです。

https://taroumaru.jp/app/webroot/download/ftp/phyton-RaspberryPi-Sample-Bdkg02-20231006.zip

FTP 等を使って ~/develop/phyton に転送して実行してください。

```
toyama@raspberrypi:~/develop/python $ cd ~/develop/python/
toyama@raspberrypi:~/develop/python $ python3 main.py
Write : 01 03 00 03 00
Read : 01 03 04 47 8f 3e 00 1b 01
Write : 01 1a 00 1a 00
Read : 01 1a 01 24 3f 00
----------------------
線量率 : 0.07 [μ Sv/h] 偏差 36 [%]
```

1.48　SPI 接続 RS485-HAT 基板の説明書

USB 接続の場合には、SPI 接続設定は不要であるためこの章を読み飛ばしてください。

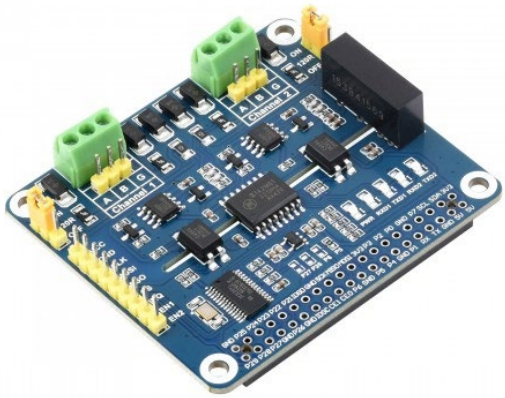
今回使った基板の取扱説明書がここです。

- https://www.waveshare.com/wiki/2-CH_RS485_HAT
- ２つの RS485 接続ポートがあるが、今回は 1 番ポートを使いました。

説明書には、こちらのような記載があります。

EN1	P27	Channel 1 transceiver enable: high level transmit enable, low level receive enable
EN2	P22	Channel 2 transceiver enable: high level transmit enable, low level receive enable

GPIO のピン 27 を High にすると、送信できる、GPIO のピン 27 を Low にすると受信できると、書いてありますのでこれに従い、送信する前、受信する前には、このピンを High, Low と制御しています。



9　パソコン(USB)との接続

RS485-USB 変換器を使った場合の接続例です。

ケーブルの色	番号	配線
白	1	GND
茶	2	9V (電圧)
緑	3	A
黄	4	B
被覆線	5	被覆線

RS485-USB 変換器はこちらを利用しています。

デバイスのドライバーはこちらです。

https://taroumaru.jp/download/software/RS485USB_port_WindowsDriver.zip

1.49 C# サンプルアプリの起動例

パソコンに USB で放射線測定器を接続した場合の起動例です。

サンプルコードのダウンロードはこちらから
https://taroumaru.jp/app/webroot/download/ftp/BDKG02_RS485_CS.zip

こちらは実行例です。