2.5 シリアル通信

ロボットに搭載されたセンサの情報やロボットの内部状態を知るために，マイコンとパソコン間でデータをやり取りする通信機能を用いる。本節では，Arduino の持つ通信機能のうちシリアル通信に関する実験を行う。シリアル通信は，古くからコンピュータ間の通信に使用されてきた通信方式であり，多くのマイコンにシリアル通信機能が搭載されている。シリアル通信を用いることで，マイコンの外部に接続されたデバイスと決められたプロトコルに従って通信を行える。

実験目標：

・シリアル通信の原理を理解する。
・シリアル通信を使いこなす。
・マイコンとパソコン間でデータを送受信できる。
・センサのデータをマイコン・パソコンで処理できる。
・ディジタル I/O，AD 変換，割り込みなどを統合的に扱える。

2.5.1 シリアル通信の原理

・1 ビットディジタル通信

最もシンプルな通信の例として，ディジタル IO ポート 1 本の配線を使ってマイコン間で通信することを考えよう。これまでに，ディジタル IO ポートを使って LED の点灯・消灯やスイッチの状態の検出を行った。ディジタル IO ポートを用いると，2 つの状態（HIGH または LOW，1 または 0）を表すことができる。このことから，ディジタル IO ポート 1 本で，マイコン外部に接続されたデバイスと 1 ビットの情報のやりとりができると解釈できる。

図 2.71 のようにマイコンを 2 台用意して，各々のディジタル IO ポートを 1 本の配線で接続する。2 台のマイコンをマイコン 1 とマイコン 2 として，マイコン 1 のディジタル IO ポートの D13 を出力に設定，マイコン 2 の D9 を入力に設定する。この状態でマイコン間の 1 ビット通信を行う。通信の方向は，マイコン 1（出力）からマイコン 2（入力）の一方向である。1 ビットディジタル通信では，

・マイコン 1 は，データを送りたいときに D13 に HIGH または LOW を出力する。
・マイコン 2 は，ポーリングで 100ms 毎に D9 の状態を監視しておく。

と通信のルールを決めておく。マイコン 1 の出力ポート(D13)より送られた HIGH または LOW の電圧レベルは配線を通じてマイコン 2 の入力ポートに伝わる。マイコン 2 は，一定の時間間隔

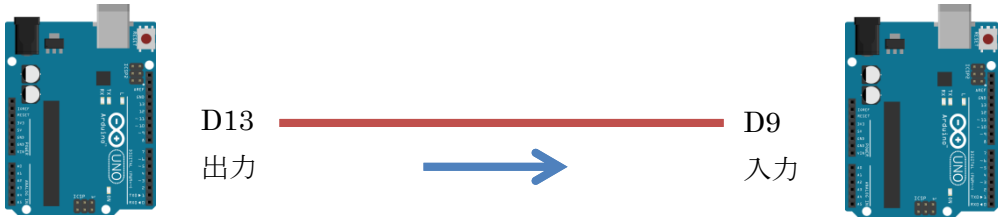


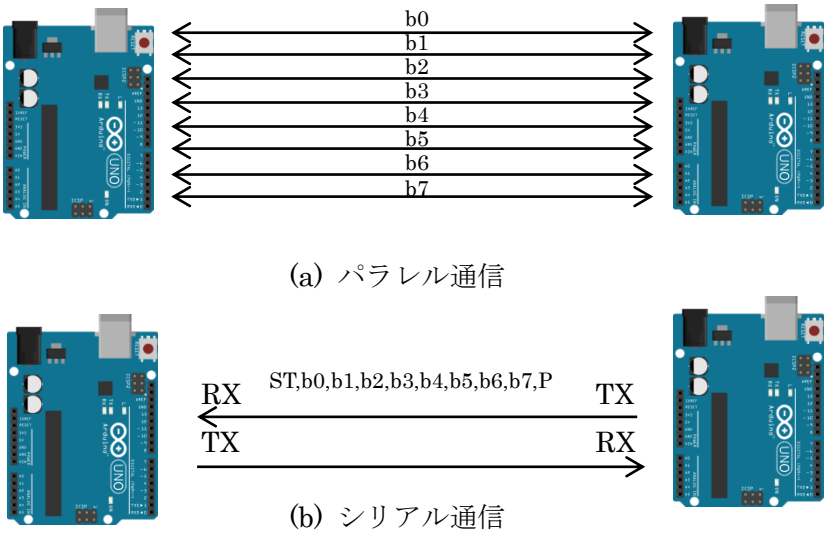
図 2.71　2 台のマイコン間の 1 ビット通信

で電圧レベル HIGH または LOW を検出する。これにより，マイコン 2 は，マイコン 1 から送られてきた情報が 1 または 0 であることを知ることができる。このように，通信の基本は情報を物理的な媒体を介して決められたプロトコルに従って相手に伝えることである。

・パラレル通信とシリアル通信

一般的なコンピュータの通信には，大きく分けると図 2.72 に示すようにパラレル通信とシリアル通信がある。パラレル通信は，図 2.72(a)に示すように複数本のディジタルポートをコンピュータ間で配線して通信を行う方式である。先の 1 ビットのディジタル配線が複数配線になった構成である。パラレル通信は，コンピュータ内のバス配線に使用されており ATA，PCI，SCSI などの規格がある。パラレル通信では，図 2.72(a)に示すようにデータ線 8 本(b0～b7)と制御線であるストロー部信号線 1 本を配線して，ストローブ信号の立ち上がりと同時に各ビットに対応する通信線からの信号の状態を受信側が読み込むことで通信が成立する。図 2.73(a)の例では，ストローブ信号の立ち上がりの時刻に受信側が b7～b0 の信号（01010101）を読み込んでいる。パラレル通信では，データを同時並列に通信できるため高速に大容量の通信ができる。その反面，通信速度を上げようとすると物理的な配線が多くなるため，配線間の干渉によるノイズの影響や基板上の配線が複雑になる問題がある。

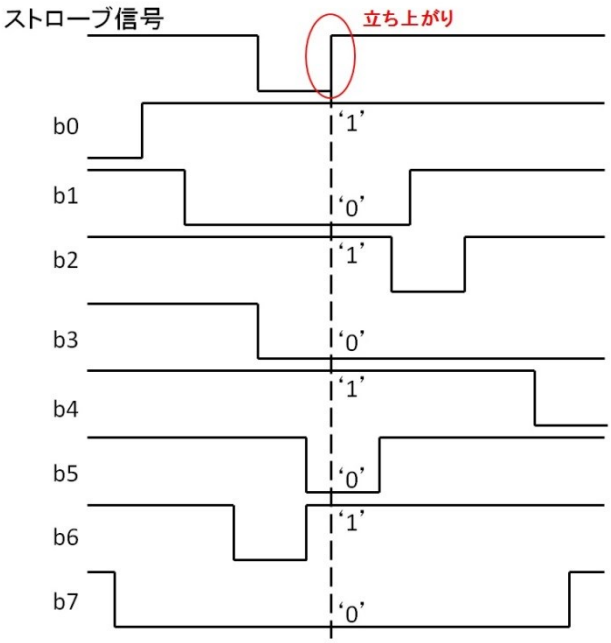
一方，シリアル通信は，図 2.72(b)に示すように 1 本または 2 本の通信線を使用して，1 ビットごとにデータを送る方式である。シリアル通信の基本は，図 2.73(b)に示すように，8 ビット(b0～b7)のデータを送るときには，スタートビットと呼ばれる通信の開始を表すビットに始まり，続けてデータビット(b0～b7)を送り，最後にストップビットと呼ばれる通信の終了を表すビットを送る。受信側では，スタートビットによりデータの開始を確認して，8 ビット分のデータを順に受け取り，ストップビットによりデータの受信を終了する。シリアル通信の規格として，パソコン間の通信の RS-232C，パソコンのハードディスクの SATA，バス接続の PCI-Express，外部デバイスとの接続の USB，LAN 接続の Ethernet などがある。シリアル通信は，パラレル通信に比べて配線数が少なくノイズの影響を受けにくいため通信の高速化が可能である。



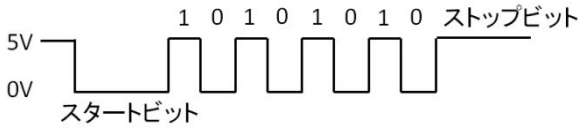
(a) パラレル通信

(b) シリアル通信

図 2.72　パラレル通信とシリアル通信



(a)パラレル通信：ストローブ信号の立ち上がりのタイミングで受信



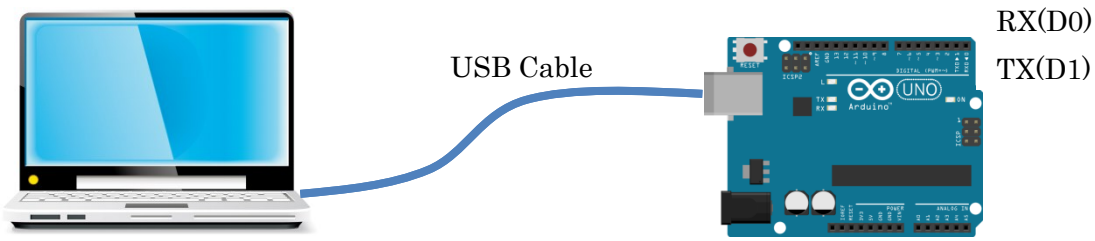
(b)シリアル通信：データが転送されたときのみ処理

図 2.73　パラレル通信とシリアル通信のデータ転送方式(‘01010101’を転送)

2.5.2 Arduino のシリアルポート

Arduino で使用できる通信方式として，シリアル通信（調歩同期式），SPI（Serial Peripheral Interface）通信，I²C (I square C)通信の 3 つの方式がある。マイコンには，各々の通信規格に対応した通信機能が内蔵されており簡単な設定で通信を行うことができる。

Arduino UNO のシリアル通信ポートは，図 2.74 に示すとおりパソコンとの USB 通信機能に割り当てられており新たな配線を追加することなく使用できる。これまでに，スケッチの書き込みやセンサの値をパソコンで確認するためにもシリアル通信を使用していた。



ノートパソコン　　Arduino　UNO

図 2.74　ArduinoUNO とパソコンのシリアル通信

Arduino UNO のシリアルポートは，図 2.75 に示すように送信ポート＜TX＞，受信ポート＜RX＞として Arduino のピンソケットに配線されている。これら 2 つのポートは，ディジタル IO ポート（D0,D1）とも共用である。そのため，シリアル通信使用時には D0 および D1 ポートが使用される。**そのため，D0 および D1 ポートをディジタル IO ポートとして使用する場合は，シリアル通信をしていないことを確認するか，もしくはシリアル通信を終了させる必要がある。**

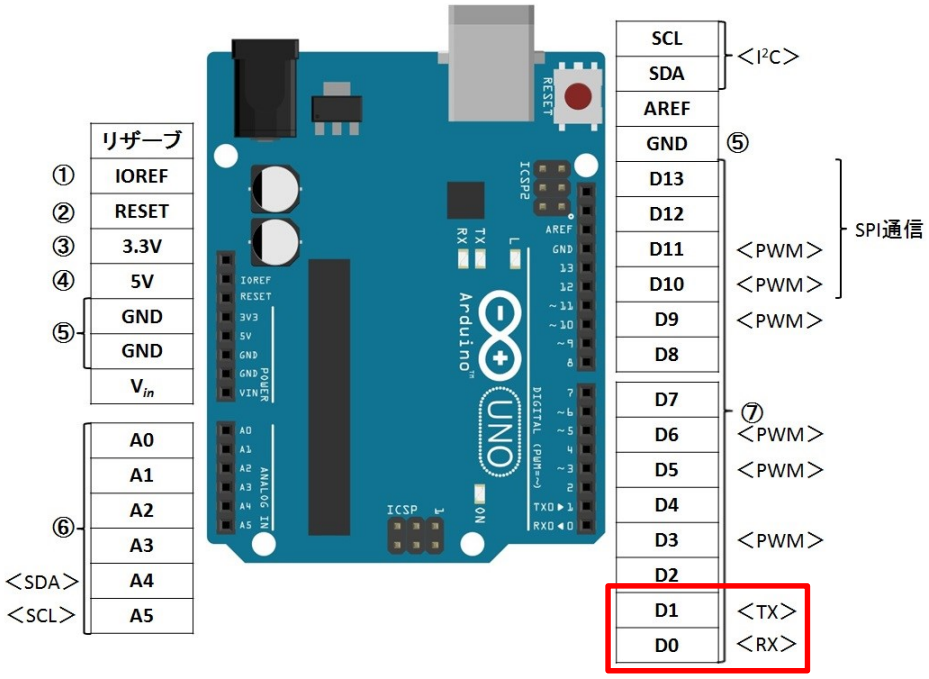


図 2.75　Arduino UNO マイコンボード（シリアル通信のピン配置）

2.5.3 シリアル通信に使用する Arduino API

Arduino UNO のシリアル通信を使用するための Arduino API 関数について説明する。

Serial.begin 関数：シリアル通信を開始するための関数である。通信に使用するデータ転送レートを指定する。シリアル通信を行うときには，**D0 および D1 ポートをディジタルポートとして設定しないことを推奨する。**また，コンピュータとのデータ転送レートには，300，1200，2400，4800，**9600**（Arduino IDE, XBee 無線通信のデフォルト設定），14400，28800，38400，57600，115200 のいずれかを指定する。ただし，マイコンの転送レートを変更した場合には，パソコン側のシリアルモニタの転送レート設定も同様に変更しないと正しくデータが受信できないので注意する。

書式：Serial.begin(speed)

引数：データの転送レートを bps（ビット/秒）で指定する。

戻り値：なし

Serial.print 関数：マイコンからデータを送信するための関数であり，ユーザが読むことの出来るデータを ASCII コードに変換してシリアルポートに出力を行う。関数を使用するには，出力する文字・文字列・数値を引数として入力する。数値を引数とした場合には，1 桁ずつ ASCII コードに変換される。また，バイト型のデータは文字として送信される。

書式：Serial.print(data);，または Serial.print(data, format);
引数 1：data　出力する値（すべての型に対応）
引数 2：format　データを変換する方法または有効桁数（浮動小数点）＜省略可＞
　format　データを変換する方法；
　・なし：ASCII コードを 10 進数で出力
　・DEC：ASCII コードを 10 進数で出力
　・HEX：ASCII コードを 16 進数で出力
　・OCT：ASCII コードを 8 進数で出力
　・BIN：ASCII コードを 2 進数で出力
　・整数値：浮動小数点の桁数
戻り値：送信したバイト数（byte）

Serial.println 関数：マイコンからデータを送信するための関数である。ユーザが読むことの出来るデータの末尾に改行に相当するキャリッジリターン（'¥r'）とラインフィード（'¥n'）を付け，ASCII コードに変換されてシリアルポートに出力する。関数を使用するには，出力する文字・文字列・数値を引数として入力する。数値を引数とした場合，1 桁ずつ ASCII コードに変換される。また，バイト型のデータは文字として送信される。
書式：Serial.println(data);，または Serial.println(data, format);
引数 1：data　すべての整数型と String 型
引数 2：format　データを変換する方法または有効桁数（浮動小数点）＜省略可＞
　format　データを変換する方法；
　・なし：ASCII コードを 10 進数で出力
　・DEC：ASCII コードを 10 進数で出力
　・HEX：ASCII コードを 16 進数で出力
　・OCT：ASCII コードを 8 進数で出力
　・BIN：ASCII コードを 2 進数で出力
　・整数値：浮動小数点の桁数
戻り値：送信したバイト数（byte）

Serial.available 関数：マイコンが受信したデータのバイト数を返す関数である。この関数を用いるとマイコンにデータが到着したかどうかを確認できる。マイコンが受信したデータはバッファに到着順に記憶される。受信バッファの容量は 128 バイトである。
書式：val = Serial.available();
引数：なし
戻り値：バッファにあるデータのバイト数

Serial.read 関数：受信データを **1 バイトずつ**読み込むための関数である。例えば，2 バイトのデータを受信する場合には，2 回データの読み込みを行う。
書式：val = Serial.read();

引数：なし
戻り値：読み込み可能なデータの最初の 1 バイトのデータ

Serial.write 関数：シリアルポートにバイナリデータを出力するための関数である。1 バイトまたは複数バイトのデータを送信できる。
書式：Serial.write(data);，または Serial.write(buf, len);
引数 1：data　送信する値（文字列），buf　配列として定義された複数バイト
引数 2：len　配列の長さ
戻り値：送信したバイト数

Serial.end 関数：シリアル通信を終了する関数である。D0 および D1 ポートをディジタル IO ポートして使用することが出来るようになる。再度シリアル通信を有効にしたいときは，Serial.begin( )を呼び出す必要がある。
書式：Serial.end();
引数：なし
戻り値：なし

・ASCII テーブル

ASCII (アスキー: American Standard Code for Information Interchange)コードは，7 桁の 2 進数に制御コード，数字，大小のラテン文字を割り当てた文字コードである。シリアル通信に使用されるデータは ASCII コードに変換されて通信が行われる。ASCII テーブルは，表に挙げるように ASCII コードに割り当てられたコードとそれに対応する数値（16 進数，10 進数）の関係を示したものである。シリアル通信における ASCII コードの取り扱いは次のようになる。

(例 1) Serial.write(65)（または Serial.write(0x41)）：シリアルモニタに ASCII コードの‘A’が表示される。

(例 2) Serial.print(‘a’): シリアルモニタには’a’が表示される。これは, 関数内でデータ’a’が ASCII コード 0x61 に変換されて通信される。つまり，Serial.write(0x61)と同じである。

表：ASCII テーブル

	0x-0	0x-1	0x-2	0x-3	0x-4	0x-5	0x-6	0x-7	0x-8	0x-9	0x-A	0x-B	0x-C	0x-D	0x-E	0x-F
0x2-	SP 0x20 32	! 0x21 33	" 0x22 34	# 0x23 35	$ 0x24 36	% 0x25 37	& 0x26 38	' 0x27 39	(0x28 40	) 0x29 41	* 0x2A 42	+ 0x2B 43	, 0x2C 44	- 0x2D 45	. 0x2E 46	/ 0x2F 47
0x3-	0 0x30 48	1 0x31 49	2 0x32 50	3 0x33 51	4 0x34 52	5 0x35 53	6 0x36 54	7 0x37 55	8 0x38 56	9 0x39 57	: 0x3A 58	; 0x3B 59	< 0x3C 60	= 0x3D 61	> 0x3E 62	? 0x3F 63
0x4-	@ 0x40 64	A 0x41 65	B 0x42 66	C 0x43 67	D 0x44 68	E 0x45 69	F 0x46 70	G 0x47 71	H 0x48 72	I 0x49 73	J 0x4A 74	K 0x4B 75	L 0x4C 76	M 0x4D 77	N 0x4E 78	O 0x4F 79
0x5-	P 0x50 80	Q 0x51 81	R 0x52 82	S 0x53 83	T 0x54 84	U 0x55 85	V 0x56 86	W 0x57 87	X 0x58 88	Y 0x59 89	Z 0x5A 90	[0x5B 91	¥ 0x5C 92	] 0x5D 93	^ 0x5E 94	_ 0x5F 95
0x6-	` 0x60 96	a 0x61 97	b 0x62 98	c 0x63 99	d 0x64 100	e 0x65 101	f 0x66 102	g 0x67 103	h 0x68 104	i 0x69 105	j 0x6A 106	k 0x6B 107	l 0x6C 108	m 0x6D 109	n 0x6E 110	o 0x6F 111
0x7-	p 0x70 112	q 0x71 113	r 0x72 114	s 0x73 115	t 0x74 116	u 0x75 117	v 0x76 118	w 0x77 119	x 0x78 120	y 0x79 121	z 0x7A 122	{ 0x7B 123	| 0x7C 124	} 0x7D 125	~ 0x7E 126	DEL 0x7F 127

＜演習 2.5.1＞ シリアル通信における ASCII コードの取り扱い

マイコンの変数 thisByte の値を Serial.write 関数, Serial.println 関数を用いて，マイコンからパソコンへの通信を行う。

【パソコン】受信　←　送信【マイコン】

(1) 新規スケッチを開く。

(2) スケッチに名前を付けて保存する（例：Tr251_20190517 など)。

(3) 次のスケッチを書き込み実行する。

```
int thisByte = 65; // 送信するデータ
int i; // データを 1 回だけ送信するためのカウンタ
void setup() {
  Serial.begin(9600); // 転送レート 9600 に設定
  i = 0; // カウンタの初期設定
}
void loop() {
  if(i >= 1){}else{ // 1 回だけデータを送信
    Serial.write(thisByte); // ASCII コード
    Serial.println(); // 改行
    Serial.println(thisByte); // 10 進数
    Serial.println(thisByte, HEX); //16 進数
    Serial.println(thisByte, BIN); // 2 進数
    i++; // カウントアップ
  }
}
```

(4) シリアルモニタを起動して，マイコンからのデータの受信結果を確認する。

(5) マイコンの変数 thisByte の値 65 が, パソコンでは次のように受信されて表示される (図 2.76 参照)。

A ・・・・・65 (0x41)の ASCII コード。ASCII テーブルを見ると 10 進数 65 であり，2 進数では 1000001 である。Serial.write はパソコンに ASCII コードを送信している。その時のビットパターンが，図 2.73(b)の b0-b7 としてスタート，ストップビットが付加されて通信される。

65・・・・・65 の 10 進数 65。通常，Serial.print では，10 進数で変数を表示する。これは，Serial.print 関数により，'6'を ASCII コード 0x36，'5'を ASCII コード 0x35 に順に変換して送信された結果である。

41・・・・・65 の 16 進数 0x41。Serial.print 関数により，'4'を ASCII コード 0x34 に変換，'1'

を ASCII コード 0x31 に変換して送信された結果である。

1000001・・・7 ビット 2 進数。Serial.print 関数により，'1'，'0'が ASCII コード（1 → 0x31，0 → 0x30）に変換して送信された結果である。

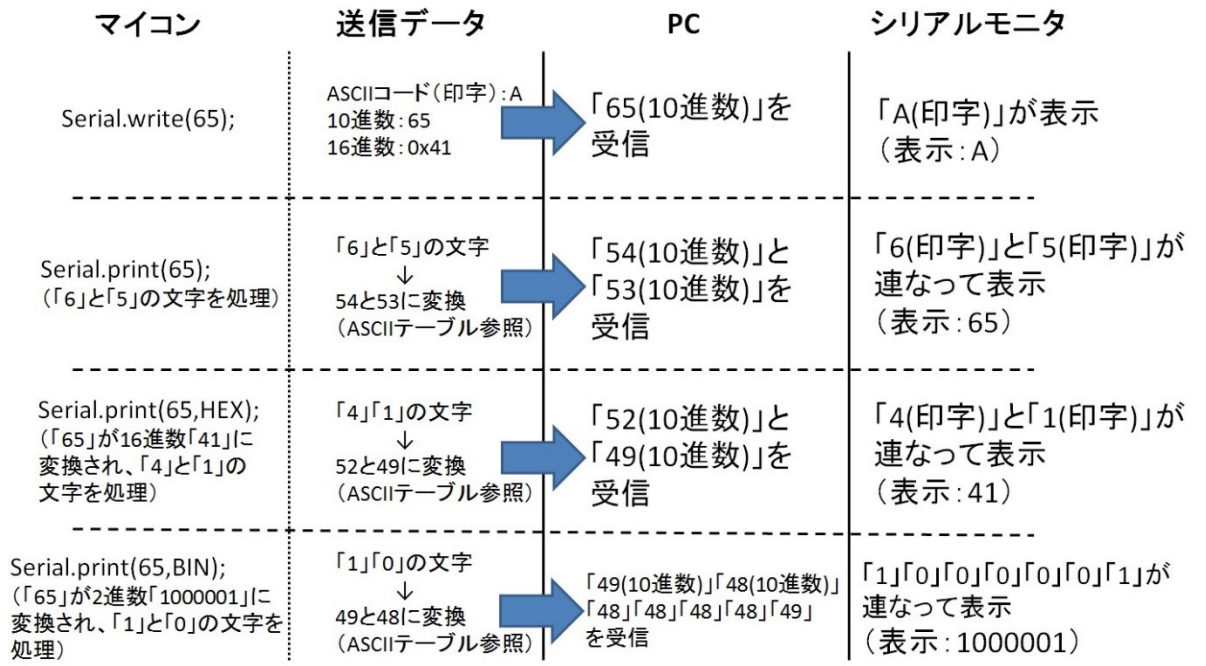


図 2.76　Serial.write と Serial.print の違い

※ シリアルモニタを開いたまま USB ケーブルを抜くとシリアルポートが認識されなくなる。USB ケーブルを抜き挿しするときには，シリアルモニタを閉じているか必ず確認する。

※ シリアルポートが認識しなくなった場合には，シリアルモニタを閉じた後に，再度，USB ケーブルを抜き挿しして，Arduino IDE のファイルメニューのツールからシリアルポートが認識されたかを確認する。

＜演習 2.5.2＞ Serial.write 関数と Serial.print の違いの確認

マイコンの変数 i の値を Serial.write 関数, Serial.println 関数を用いて，マイコンからパソコンへの通信を行う。

【パソコン】受信　←　送信【マイコン】

(1) 新規スケッチを開く。

(2) スケッチに名前を付けて保存する（例：Tr252_20190517 など）。

(3) 変数 val を用いて 32 から 127 をカウントし，その値を Serail.write および Serial.println を使用して，マイコンから PC に送信するスケッチを作成する（下記参照）。

```
int val; // カウントするための変数
void setup() {
  Serial.begin(9600); // 転送レート 9600 に設定
  val = 32; // val の初期設定
}

void loop() {
  if(val <= 127){ // 「32」から「127」までのデータを送信
    Serial.write(val); // val の値に対応した ASCII コード
    Serial.print(“,”); // ,を送信
    Serial.println(val); // val の値を文字に変換した結果＋改行
    val++; // val のカウントアップ
  }else{}
}
```

(4) スケッチを実行する。
(5) シリアルモニタを起動して，マイコンからのデータの受信結果を確認する。
(6) 図 2.76 に示されるように，i の値が 48 の場合，Serial.write では 48 に対応する「0」という ASCII コード（印字）を，Serial.print では「4」「8」の印字を，シリアルモニタで確認する。

2.5.4 シリアル通信で使用するスケッチ

シリアル通信の演習・課題・発展課題では，図 2.77 に示すように，①データの送信，②データの受信，③データの送受信，およびこれまでの実験で習得した技術を統合的に扱う。

① マイコンからデータの送信・・・Serial.print または Serial.println 関数を用いて，D1 ポートをデータ送信用ポート（TX）に設定（Serial.begin を宣言することで自動的に設定される）して，マイコンから PC にデータを送信する。その送信されたデータをシリアルモニタで確認する。

② マイコンのデータの受信・・・シリアルモニタのデータ送信機能を利用し，データ受信用ポート（RX）に設定された D0 ポートを使用して，PC からマイコンにデータを送信する。マイコンは Serial.available 関数を用いてデータが受信されたことを確認した後に，Serial.read 関数を使用してそのデータを受信する。

③ データの送受信・・・PC から送信されたデータをマイコンが受信して，受信データに対応した処理を実行し，その実行結果を PC に送信する。

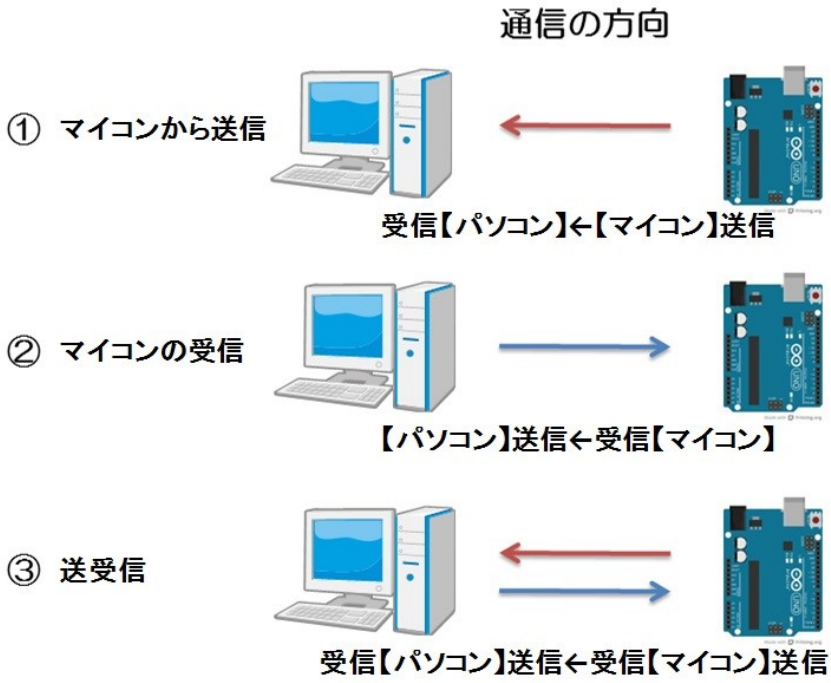


図 2.77　マイコンとパソコンの通信方向（①マイコンから送信，②マイコンの受信，③送受信）

＜演習 2.5.3＞ 演習，課題および発展課題に使用する回路の実装

図 2.78 を参照して，演習および課題に使用する LED の発光回路，スイッチの回路，照度センサの回路および温度センサの回路を実装する。（他の回路の抵抗などの素子と接触しないように注意する）

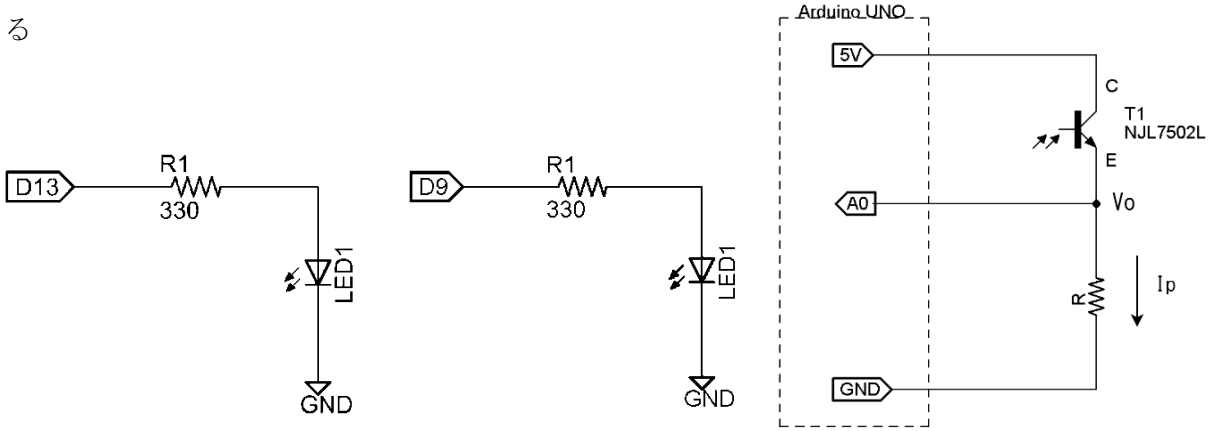
(1) LED（赤，黄）の回路を実装する（ポートには接続しない）

(2) 照度センサの回路を実装する（ポートには接続しない）

(3) 温度センサの回路を実装する（ポートには接続しない）

(4) マイコンボードの 5V とブレッドボードの赤ラインを接続するためのジャンパ線を用意する

(5) マイコンボードの GND とブレッドボードの青ラインを接続するためのジャンパ線を用意する



(a) LED（赤）発光回路　　(b) LED（黄）発光回路　　(c) 照度センサの回路

図 2.78　シリアル通信実験に使用する回路

温度センサの測定原理

温度センサ（LM61CIZ）は，PN 接合の温度による順方向電圧の変化を検出するセンサである。半導体温度センサの特性は直線性にすぐれ，温度変化に対して出力電圧の変化が 10 mV/℃と一定量で変化するため取り扱いやすい。実験に使用する温度センサの外観を図 2.79(a)に示す。温度センサにはリード線が 3 本あり，それぞれ電源端子（5V），電圧出力（Vo），グランド（GND）である。温度センサに電源供給を行うと，温度に応じた電圧が Vo 端子から出力される。温度センサ LM61CIZ では，-30℃から 100℃の範囲の温度が計測できる。

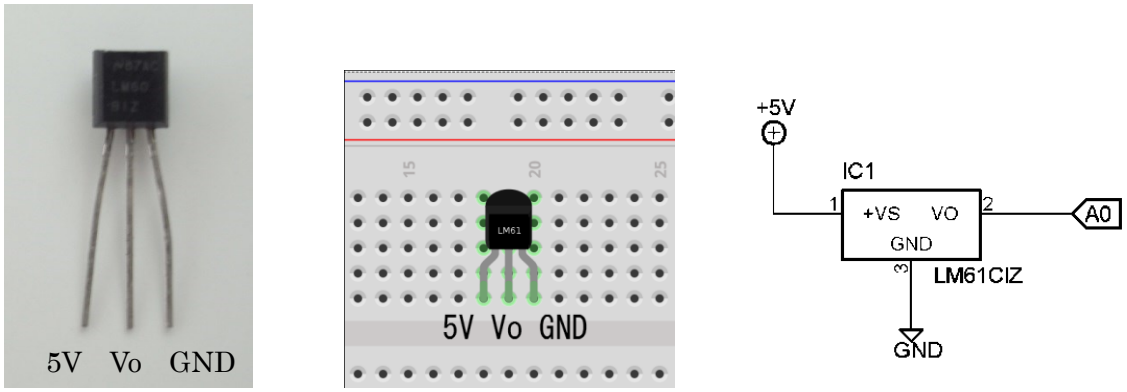
温度センサの温度変化と電圧変化の関係は，データシートより図 2.80 のグラフのように表される。温度センサの出力電圧を Vo，温度センサによって計測された温度を T [℃]とすると，

$$Vo = 10\ mV/℃ \times T℃ + 600\ mV \quad \cdots (2.9)$$

の関係がある。式(2.9)を変形すると，温度 T [℃]は，

$$T = (Vo - 600\ mV) / 10\ mV/℃ \quad \cdots (2.10)$$

により求めることができる。ここで，電圧 Vo の単位は mV である。



(a) ピン配置　　(b) ブレッドボード配置　　(c) 回路図

図 2.79　温度センサ LM61CIZ のピン配置とブレッドボード配置，回路図

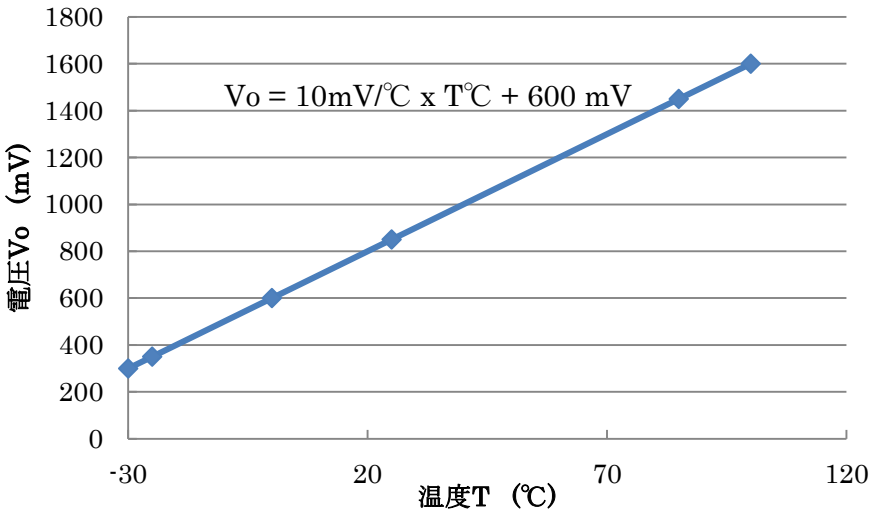


図 2.80　温度センサ（LM61CIZ）の温度と電圧の関係

＜演習 2.5.4＞ 温度に関する情報をマイコンから PC に送信：マイコンのデータ送信

演習 2.3.2 の AD 変換結果から電圧を求めるスケッチに，電圧を温度に変換する式（2.10）を追加して，温度を計算するスケッチを作成する。また，Serial.print および Serial.println を使用して，マイコンに入力された温度に関する情報（AD 変換結果，電圧，温度）を，マイコンから PC に送信するスケッチを作成する。

【パソコン】受信 ← 送信【マイコン】入力（A0）←【温度センサ】

(1) 図 2.79 を参考に，温度センサの回路をマイコンに接続する。
- 5V をブレッドボード（赤ライン）に接続する。
- GND をブレッドモード（青ライン）に接続する。
- 図 2.79 を参照して，A0 ポートと照度センサを接続する。

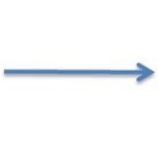
(2) 演習 2.3.2 のスケッチを開く。
(3) スケッチに名前を付けて保存する（例：Tr254_20190517 など）。
(4) 電圧を温度に変換するスケッチを追加する。
(5) シリアル通信のためのスケッチを追加する。

```
void loop(){
   ・・・
   float Temp = (vo*1000.0 – 600.0) / 10.0; //  電圧->温度
   Serial.print(sensorValue); //  センサの値
   Serial.print(“,”);
   Serial.print(vo); //  電圧
   Serial.print(“,”);
   Serial.println(Temp); //  照度
}
```

(6) USB ケーブルを接続して，マイコンにスケッチを書き込む。
(7) シリアルモニタを起動して，パソコンが受信した AD 変換結果，電圧値，温度を確認する。
(8) 温度センサを指で挟み，温度が変化することを確認する。
(9) シリアルモニタの自動スクロールのチェックを外す。
(10) USB ケーブルを抜く。

＜演習 2.5.5＞ マイコンのデータ受信による LED の点灯・消灯：マイコンのデータ受信

パソコンからのデータをマイコンが受信すると，LED（D13）の点灯・消灯が切り替わるスケッチを作成する。

② マイコンの受信   LED(D13)

【パソコン】送信　→　受信【マイコン】出力（D13）→【LED】

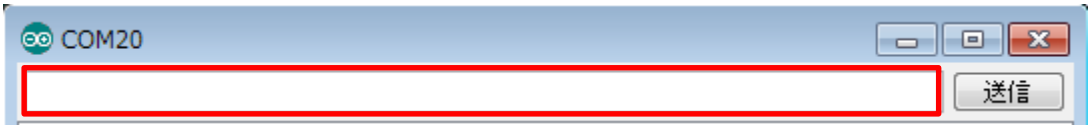
(1) LED の発光回路を実装する。
・GND とブレッドボード（青ライン）を接続する。
・図 2.78 を参考に，ディジタル出力として D13 ポートと LED を接続する。
(2) 新規スケッチを開く。
(3) データ受信のためのスケッチを記述する。

例：

```
int output = LOW; // LED への出力用
void setup(){
 pinMode(13, OUTPUT); // 13 ピンに LED を出力として設定
 Serial.begin(9600); //  シリアル通信開始：転送レート 9600
}

void loop() {
 if (Serial.available() > 0) { //  パソコンからデータを受信
     int recv = Serial.read(); //  受信データの読み込み
      output = !output; // output の反転
      // Serial.println("I received!"); //  パソコンへ応答を返す（送信）
   }
   digitalWrite(13, output); // LED の点灯・消灯
}
```

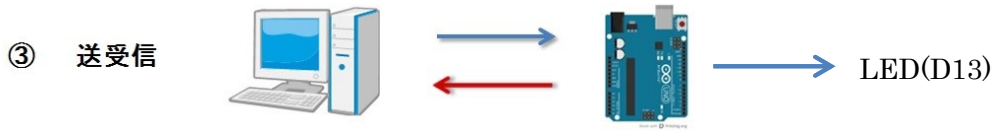
(4) 名前を付けてスケッチを保存する（例：Tr255_20190517 など）。
(5) USB ケーブルを接続して，マイコンにスケッチを書き込む。
(6) シリアルモニタを起動する。
(7) シリアルモニタに数字もしくは文字を入れて PC から送信する（マイコンは受信）。



(8) LED の発光状態が変化することを確認する。また，データの送受信時に，TX,RX の LED が点滅することも確認する。
(9) ⑦と⑧を繰り返す。このとき，パソコンから送信する文字や数字を変更すること。
(10) シリアルモニタを閉じて，USB ケーブルを抜く。

＜課題 2.5.1＞ マイコンの受信データに対応した LED の状態変化およびマイコンから LED の状態を送信：マイコンのデータ送受信

マイコンが受信したデータに基づいて，発光する LED が切り替わり，切り替わる状況を PC に送信するスケッチを作成する。‘0’を受信した場合は LED（赤）のみが点灯，‘1’を受信した場合，LED（黄）のみが点灯，それ以外の文字・数字を受信した場合，両 LED が点灯する。



シリアルモニタ【パソコン】送受信 ⇔ 送受信【マイコン】出力（D13）→【LED】

※マイコンは PC からデータを受信、PC に LED の状態を送信

(1) 図 2.78 を参考に 2 個の LED（赤，黄）をマイコンのポートに繋ぐ。
- LED（赤）は D13 ポートにディジタル出力として接続
- LED（黄）は D9 ポートにディジタル出力として接続
- 初期状態：LED（赤，黄）は消灯

(2) 演習 2.5.5 のスケッチに，受信したデータによって発光する LED が切り替わるスケッチを追記する。

(3) シリアルモニタを介してマイコンに‘0’，‘1’もしくはそれ以外の文字・数字を送信する。（マイコンは受信）

(4) LED の発光状態が変化することを確認する。

(5) LED の発光状態を PC に送信し，そのデータをシリアルモニタで確認する。
（‘1’を受信した場合の例；LED1: 0, LED2: 1）

(6) ④～⑥を繰り返す。このとき，マイコンが受信する文字・数字を変更すること。

2.5.5 マイコンからの送信データのグラフ化

表計算ソフト‘LabreOffice Calc’を用いて，マイコンから送信されたデータをグラフ化する方法を説明する。

(1) シリアルモニタを起動させ，マイコンからのデータを表示する。

(2) シリアルモニタの自動スクロールのチェックを外す（図 2.81(a)）。

(3) マウス操作により，シリアルモニタに表示されている計測値を選択する。

(4) ドラッグしたデータをマウスまたはキーボード操作（Ctrl＋c）によりコピーする（図 2.82(b)）。

(5) 表計算ソフト‘LabreOffice Calc’を立ち上げる。

(6) マウスまたはキーボード操作によりコピーしたデータを B 列に貼り付ける（図 2.83(a)）。

(7) 表の A1 と A2 のセルにそれぞれ任意の値を入力する（例：‘0’‘1’を記入する（図 2.82(b)））。

(8) セル A1 と A2 ドラッグし，A1 セルの枠の右下の小さな‘■’マウスのカーソル移動させる（図 2.82(c)）。もしくは‘■’をダブルクリックする。

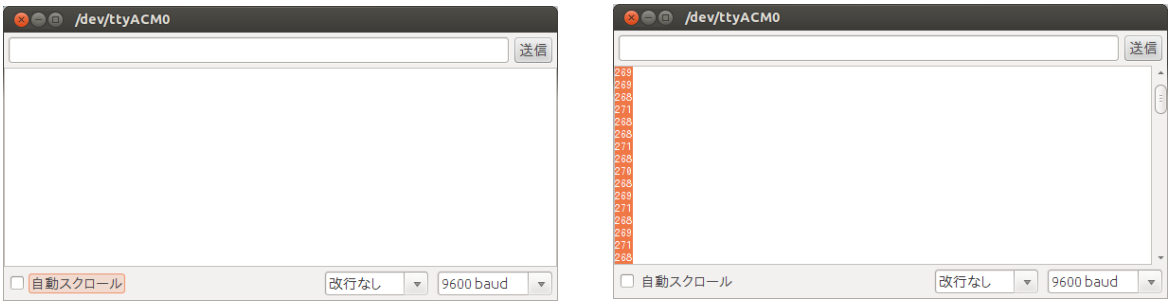
(9) 表の最上部にあるセルの列記号を意味する‘A’‘B’を選択する。

※AB の列がすべて選択されていることを確認する（図 2.83(a)）。

(10) A 列，B 列に記入されている数値群をグラフ化する**（グラフの種類は散布図を選択）**。

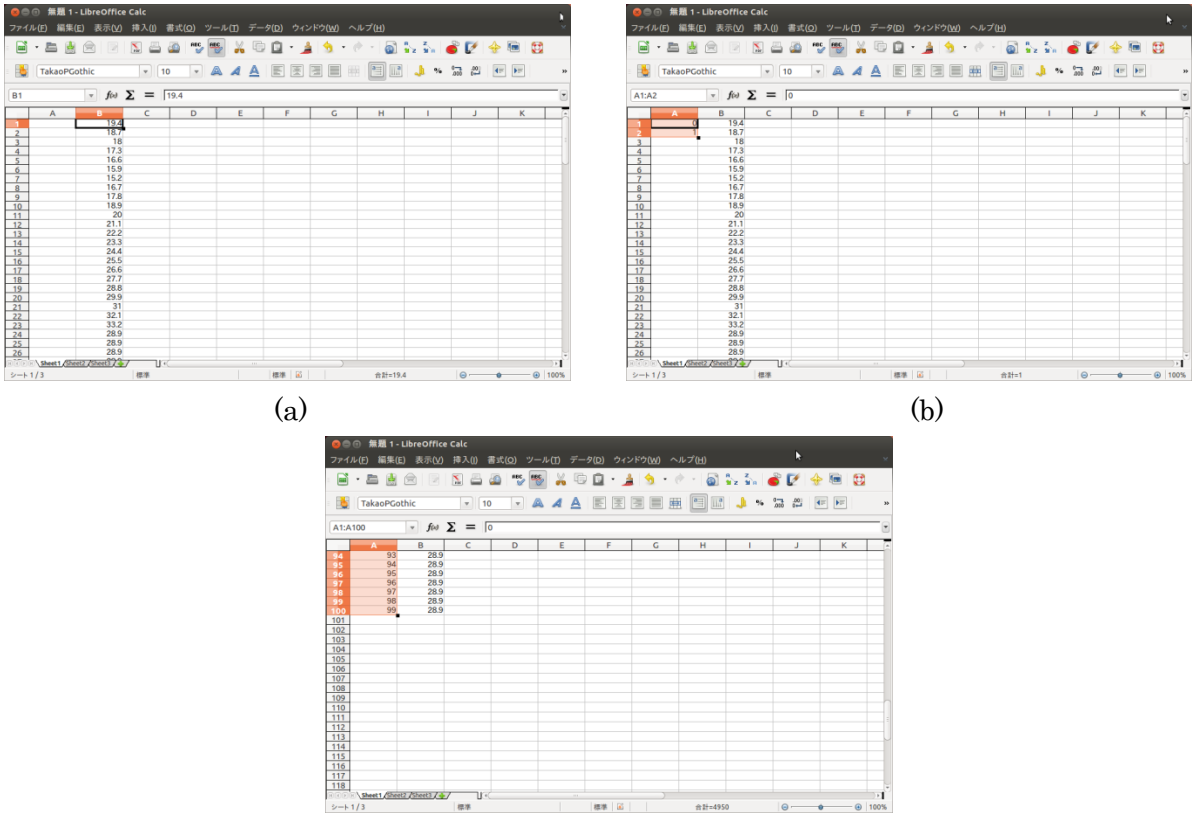
(11) A の列を横軸（時間：サンプル番号　[単位なし]など）に設定する。

(12) B の列を縦軸（物理量：温度　[℃]など）に設定する（図 2.83(b)参照）。



(a) 自動スクロールチェックを外す　　(b) データのコピー

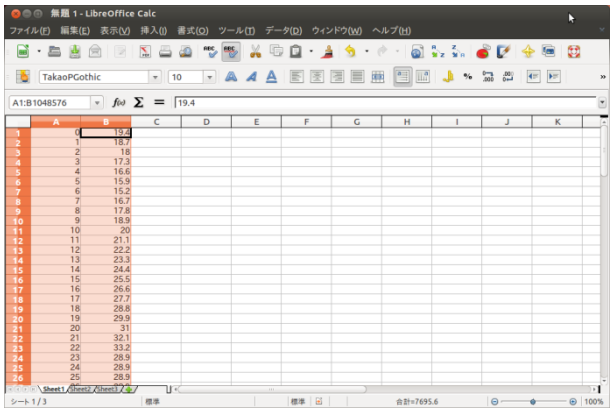
図 2.81　シリアルモニタ上での送信データの取得



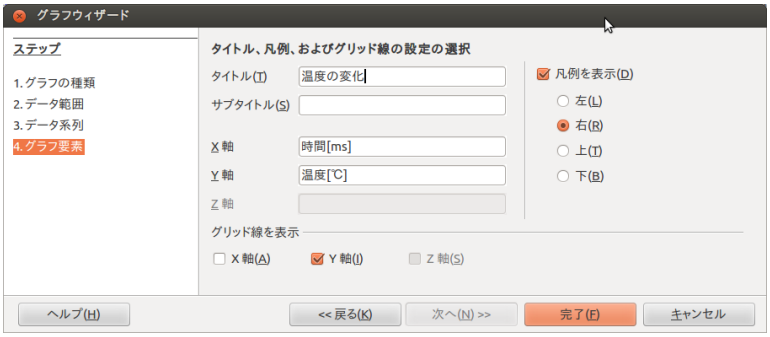
(a)　　(b)

(c)

図 2.82　データの貼り付けとグラフ化の前処理



(a)



(b)

図 2.83　グラフ化と軸のラベル設定

＜演習 2.5.6＞ 温度センサを用いたマイコンの温度情報の PC への送信およびそのグラフ化

温度センサからの電圧値をアナログ入力ポート（A0 ポート）を介して，AD 変換結果をマイコンに入力し，AD 変換結果から温度を算出し，シリアル通信で PC に温度を送信するスケッチを作成する。また，送信されたデータを表計算ソフトを介してグラフ化する。

グラフ化【パソコン】受信　←　送信【マイコン】入力（A0）←【温度センサ】

(1) 図 2.79 を参照に温度センサの回路をマイコンに接続する

・5V をブレッドボード（赤ライン）に接続する

・GND をブレッドモード（青ライン）に接続する

・A0 ポートと温度センサを接続する

(2) 演習 2.5.4 のスケッチを開く

(3) 名前を付けてスケッチを保存する（Tr256_20190517 など）

(4) 温度のみを送信するようにスケッチを修正する

－Serial.begin(9600); および Serial.println(Temp); 以外のシリアル通信用関数をコメントアウトする

(5) 上書き保存する
(6) USB ケーブルを接続する（接続した時刻から温度計測が開始される）
(7) マイコンにスケッチを書き込む
(8) シリアルモニタを起動させ，温度センサを指で挟む（10 秒程度）
(9) 約 10 秒間のデータをグラフ化する（**マイコンからの送信データのグラフ化**を参照(96～97 頁)）
(10) A の列を横軸（時間：サンプル番号　[単位なし]），　B の列を縦軸（物理量：温度　[℃]）に設定する
(11) 温度の変化が安定していないことを確認する（グラフが曲線を描けていない）

＜演習 2.5.7＞　millis 関数によるデータ解析区間の設定，温度センサの情報の PC への送信および温度情報のグラフ化：マイコンの制約付きデータ送信

温度センサを 10 秒間摘まむことで変化する温度を PC に送信して，グラフ化する。このとき，100ms 間隔で，温度センサの回路から入力されたデータを温度に変換した結果を PC に送信する。ここで，温度の計測は 10 秒として，その後の温度の情報は PC に送信しないものとする。

グラフ化【パソコン】受信　←　送信【マイコン】入力（A0）←【温度センサ】

(1) 図 2.78 を参照に温度センサの回路をマイコンに接続する
・5V をブレッドボード（赤ライン）に接続する
・GND をブレッドモード（青ライン）に接続する
・A0 ポートと温度センサを接続する
(2) 新規スケッチを開く
(3) 名前を付けてスケッチを保存する（Tr257_20190517 など）
(4) 100ms 間隔で温度を計測し，10 秒間の温度変化（100ms 間隔のデータ）を PC に送信するスケッチを作成する（下記サンプルスケッチ参照）

```
unsigned long timePrev = 0;
int count = 0;
void SendData(){
   int sensorValue = analogRead(A0);
   float vo = sensorValue * (5.0 / 1024.0);
   float Temp = (vo*1000.0 - 600.0)/10.0;
   Serial.println(Temp);
   count++;
}
```

```
void setup() {
  Serial.begin(9600);
  Serial.println("start!");
  timePrev = millis();
}

void loop() {
  if(count < 100){
    unsigned long timeNow = millis();
    if(timeNow - timePrev >= 100){
      SendData();
      timePrev = timeNow;
    }else{}
  }else{}
}
```

(5) 上書き保存する。
(6) USB ケーブルを接続する（**接続した時刻から温度計測が開始される）。**
(7) マイコンにスケッチを書き込む。
(8) シリアルモニタを起動させ，温度センサを指で挟む（10 秒程度）。
(9) 10 秒間のデータをグラフ化する（**マイコンからの送信データのグラフ化**を参照(94－96 頁)）。
(10) A の列を横軸（**時間　[ms]**），　B の列を縦軸（物理量：温度　[℃]）に設定する。
(11) 温度の変化が安定していないことを確認する（グラフが曲線を描けていない）。

＜課題 2.5.2＞ 温度センサを用いた解析済み温度情報のマイコンの PC への送信およびグラフ化

温度センサを 10 秒間摘まむことで変化する温度を，以下の条件を満たすように解析して，PC に送信するスケッチを作成する。また，その解析結果をグラフ化する。

グラフ化【パソコン】受信　←　送信【マイコン】入力（A0）←【温度センサ】

・温度センサの回路から入力されたデータの平均温度を算出する。
・平均温度を 100ms 間隔で PC に送信する。
（100ms 間の温度の平均を繰り返し PC に送信する）
・温度の計測は 10 秒とし，その後の温度の情報は PC に送信しないものとする。

※※注意：100ms 間隔で計測するのではない！100ms 間隔で温度の平均を送信する！

(1) 図 2.79 を参照して，温度センサの回路をマイコンに接続する。
・5V をブレッドボード（赤ライン）に接続する。
・GND をブレッドモード（青ライン）に接続する。
・A0 ポートと温度センサを接続する。
(2) 上記の条件を満たすスケッチを作成する。
(3) 10 秒間の解析済み温度情報を PC に送信する。
(4) グラフ化する（縦軸，横軸に適切なラベルを付ける）。
ヒント：millis 関数を使用して 100ms を計測する

＜課題 2.5.3＞ 照度センサによるマイコンから PC への解析済み照度情報の送信，グラフ化および解析結果（照度）に対応した LED の発光

約 5 秒間照度センサをそのままの状態にしておき，その後に約 10 秒間照度センサに当たる光をゆっくり手で遮っていく。このときの変化する照度を，以下の条件を満たすように解析し，PC に送信するスケッチを作成する。また，その解析結果をグラフ化する。

グラフ化【パソコン】受信 ← 送信【マイコン】入力（A0）←【照度センサ】
出力（D9）→【LED】

・照度に対応させて LED の発行状態を制御する（工夫点をレポートに記述）
・照度センサの回路から入力されたデータの平均照度（区間平均）を算出する
・平均照度を 500ms 間隔で PC に送信する
（500ms 間の照度の平均を繰り返し PC に送信する。照度センサの値の計測間隔は 100ms である。）
・照度の計測は 15 秒とし，その後の照度の情報は PC に送信しないものとする

(1) 図 2.78 を参照に照度センサおよび LED の回路をマイコンに接続する
・5V をブレッドボード（赤ライン）に接続する
・GND をブレッドモード（青ライン）に接続する
・A0 ポートと照度センサを接続する
・D9 ポートに LED を接続する（ディジタル PWM 出力）
・LED は平均照度に対応させて明るさを調整する（最小～最大：0～255 と設定しても良い）
(2) 上記の条件を満たすスケッチを作成する
(3) 15 秒間の解析済み照度情報を PC に送信する
(4) グラフ化する（縦軸，横軸に適切なラベルを付ける）

＜発展課題 2.5.1＞ マイコンのデータの送受信の確認

PC 上で起動しているシリアルモニタを使用して，マイコンに誕生日を送信する。マイコンでは，受信した誕生日に誕生月を加算して，その結果を PC に送信する。その結果をシリアルモニタで確認する。

＜発展課題 2.5.2＞ 各種センサのキャリブレーションの自動化

照度センサと温度センサのキャリブレーションをそれぞれ自動的に実施するスケッチを創生せよ。また，キャリブレーション後に，照度および温度をそれぞれ計測し，その結果をグラフ化せよ。このとき，下記の条件を満たすこと。

・キャリブレーションはマイコンが立ち上がったとき一度だけ実行されるものとする
・キャリブレーションでは，各種センサの最大値・最小値を計測し，計測された最大値・最小値に基づいて正規化する（最小値：0，最大値：255）関数を作成する
・計測時間は 15 秒とする
・温度と照度は，200ms 間隔で区間平均を算出し，グラフ化する

ヒント：キャリブレーションについて調査する

※実験内容：

(1) キャリブレーション後に, 各センサの値をグラフ化する (200ms 間隔で区間平均を算出)

【レポート 2.5 （2019 年 5 月 24 日(金) 12：50 締切)】

※スケッチには詳細なコメントを記述すること。コメントが無いものは採点対象外とする。
※自身で作成・変更したスケッチ部分を示すこと。サンプル全てをそのまま貼付けしない。

レポート 2.5.1 基礎実験第 5 回の概要

基礎実験第 5 回目の実験の目的，実施した実験の概要，および理解した事柄を 100～200 字程度で説明せよ。

レポート 2.5.2 復習と確認

レポート概要：ブレッドボード上に回路を実装する際の注意点を説明せよ。また，C 言語によるプログラムとスケッチの違いについて説明せよ。さらに，シリアルモニタの使用に関する注意点を述べよ。

レポート 2.5.3 回路実装

レポート概要：演習 2.5.3 において実装したブレッドボード配線図を報告せよ。

レポート 2.5.4 マイコンの受信データに対応した LED の状態変化およびマイコンから LED の状

態を送信

レポート概要：課題 2.5.1 において実装したスケッチを報告せよ。また，スケッチ作成において工夫した点を記せ。さらに，マイコンが受信した文字を数字に変換して処理する方法を提案せよ。

レポート 2.5.5　温度センサを用いたマイコンの温度情報の PC への送信およびそのグラフ化

レポート概要：演習 2.5.6 において作成したグラフを報告せよ。

レポート 2.5.6　millis 関数によるデータ解析区間の設定，温度センサの情報の PC への送信および温度情報のグラフ化

レポート概要：演習 2.5.7 において作成したグラフを報告せよ。

レポート 2.5.7　温度センサを用いた解析済み温度情報のマイコンの PC への送信およびグラフ化

レポート概要：課題 2.5.2 で実装したスケッチおよび作成したグラフを報告せよ。また，工夫した点を記せ。演習 2.5.6 で作成されたグラフとの違いを説明せよ。

レポート 2.5.8　照度センサによるマイコンから PC への解析済み温度情報の送信，グラフ化および解析結果（照度）に対応した LED の発光

レポート概要：課題 2.5.3 で実装したスケッチおよび作成したグラフを報告せよ。また，工夫した点を記せ。さらに，データ解析（区間平均の算出）の必要性について考察せよ。

レポート 2.5.9　発展課題 2.5.1：マイコンのデータの送受信の確認

レポート概要：発展課題 2.5.1 において実装したスケッチを報告せよ。また，作成したスケッチの工夫した点を記せ。

レポート 2.5.10　発展課題 2.5.2：創生課題

レポート概要：発展課題 2.5.2 において実装したスケッチを報告せよ。また，作成したスケッチの工夫した点を記せ。

・使用部品一覧

使用部品一覧

部品	個数	備考
Arduino Uno	1	
USBケーブル　Aオス-Bオス　1.5m A-B	1	
ブレッドボード　EIC-801	1	
ブレッドボード・ジャンパーワイヤ　EIC-J-L	1	
3mm 赤色LED OSDR3133A	1	
3mm 黄色LED OSYL3133A	1	
1/4W 330Ω	各2	LED
1/4W 10kΩ	1	照度センサ
照度センサ	1	
温度センサ	1	

参考図書

[1] Massimo Banzi，船田 巧：Arduino をはじめよう －第 2 版－，株式会社オライリー・ジャパン (2014).

[2] 河連 庸子，山崎 文徳，神原 健：Arduino　スーパーナビゲーション，株式会社リックテレコム (2012).

[3] 神崎 康宏：Arduino で計る，測る，量る，CQ 出版株式会社 (2013).

[4] 田中 博，芹井 滋喜：PIC16 トレーナによるマイコンプログラミング実習，学校法人 東京電機大学(2013).