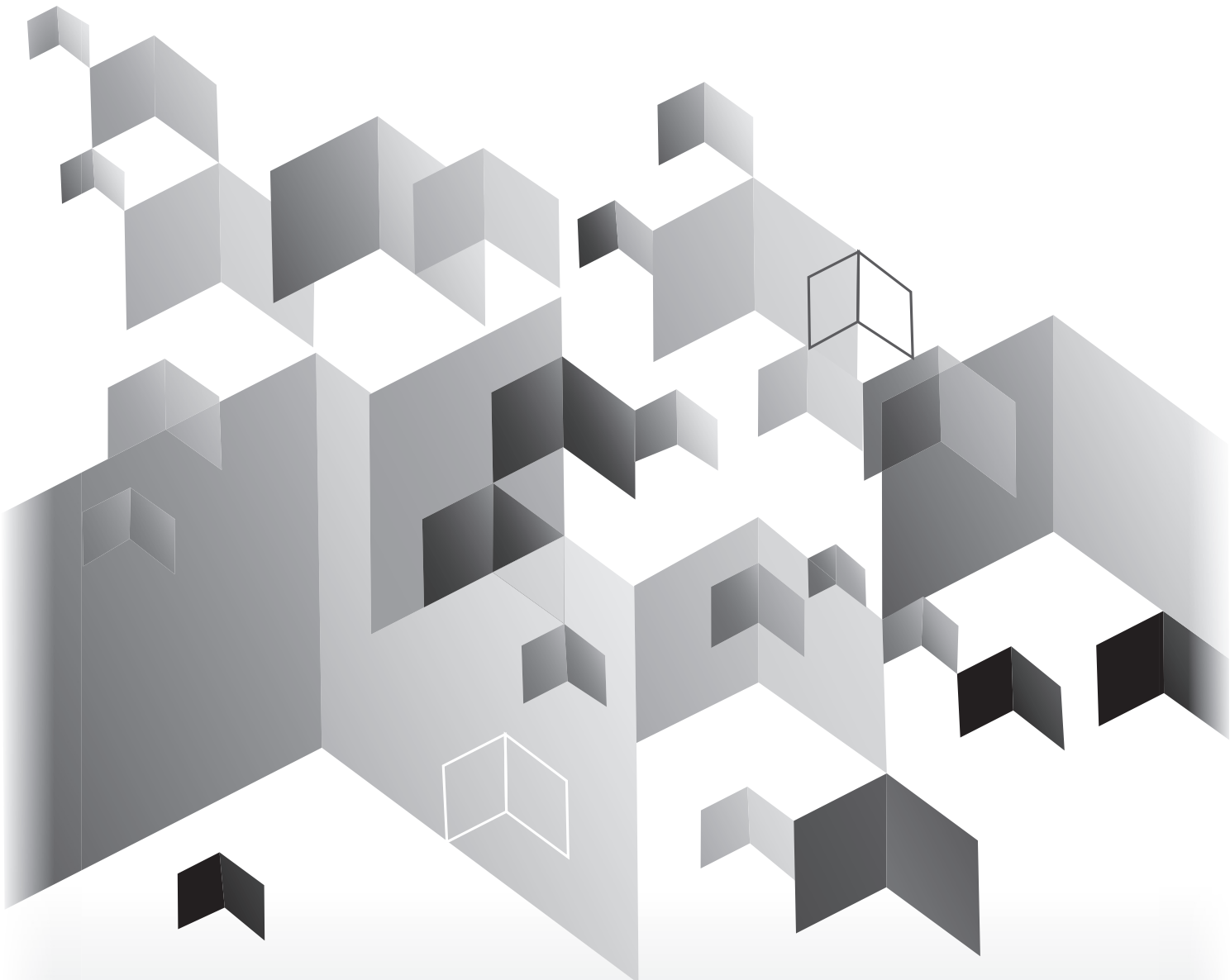
平成 30 年度「専修学校による地域産業中核的人材養成事業」

【IoTへのドリル1 】
組込みシステム開発技術



情報通信技術に対応した組込みシステム開発技術者育成のモデルカリキュラム開発と実証事業

平成 30 年度「専修学校による地域産業中核的人材養成事業」

【IoTへのドリル1】
組込みシステム開発技術

情報通信技術に対応した組込みシステム開発技術者育成のモデルカリキュラム開発と実証事業

このテキストの使い方

このテキストは、IoT を構築するための方法を習得するためのトレーニングドリルという位置づけで編集しました。全体では 2 冊構成となっています。それぞれが、無線マイコン（TWE-Lite）と WiFi マイコン（ESP8266）の 2 つのマイコンについて利用方法を解説しています。第一分冊では、マイコンの基本（DO/DI）からアナログ計測（ADC)、シリアル通信を含みます。第二分冊では、ネットワークに接続して、各種 WEB サービスを利用するためのシステム開発方法を含んでいます。他のマイコンを経験した方は、何処から読まれてもよいでしょう。また、未経験の方でも、前提知識なしに各々の第 1 回から読み進めて、解説の通りの実験を進めれば、このテキストを終える頃には、IoT デバイスに必須のマイコン知識を一通り経験できます。

マイコン毎に WEB サービスまでを通して学習したい方は、無線マイコン編の第 1 回から第 11 回の後、第二分冊の第 12～14 回でトレーニングすれば､無線マイコンで WEB を活用したシステムが開発できるでしょう。また、WiFi マイコン編の第 1 回から第 9 回の後、第二分冊の第 10 回から第 14 回でトレーニングすることで､WiFi を利用して WEB 接続する IoT システムを開発することができるでしょう｡それぞれが全 14 回の講座となっていますので、前期・後期制の学習スケジュールに乗りやすいと思います。

なお、本文中に掲載したプログラムソースコード等は、実習キット付属の CD で提供しています。

【IoT へのドリル1】

組込みシステム開発技術

目次

【無線マイコン(TWE-Lite)編】

【WiFiマイコン(ESP8266)編】

テキストの範囲

図　1

写真は、千葉県柏市・我孫子市などにまたがる手賀沼に隣接する農地の様子を８月初旬に撮影したものです。大きく広がる水田に、元気に育つ稲の様子が写っています。稲穂も太くなっていて、良い収穫が見込まれます。写真をよく見ると、右側奥手にはいくつかのハウスも写っていて、稲だけではなく様々な野菜なども生産されていることが連想されます。この写真では奥行き 3km ほどの田園風景ですが、ちょうど反対側（背中側）も同じように広い水田が広がっています。

農業分野の IoT では、このような広いフィールドや環境をコントロールするハウスなどに、様々なセンサーなどを数多く配置して、それらが取り込んでくれる環境情報を収集して、統計・解析などを行い、次の農業生産に活かす取り組みが盛んに進められています。

図　2

上図は、2016 年度に行われた、農水省のトラクター自動運転技術実証事業で公開されたロボットトラクターです。写真にはオペレータが写っていますが、よく見ると両手は膝の上に在り、何も操作していません。ではどこで操作しているのかというと、トラクターに搭載されたコンピュータシステムが、畑をどのように耕すかを判断して、数 cm の位置精度で自動走行しています。

図 3

上図は、公開されたトラクターの仕様です。キャビンの屋根上には、衛星電波を受信するGPS用アンテナが設置されていて、この中にはマイコンが内蔵されています。現在では準天頂衛星システム「みちびき」に対応したものになっており、この実証実験よりさらに精度が向上しています。また、自動ステアリング装置やコントローラにもマイコンが内蔵されて、ネット通信が行われます。

図　4

その結果、上図の様に、離れているロボットトラクター管制部にも、トラクター搭載カメラからの映像や、計器類の指針、耕うん状態などがリアルタイムで送られてきます。これらすべてが IoT 技術の成果と言えるでしょう。

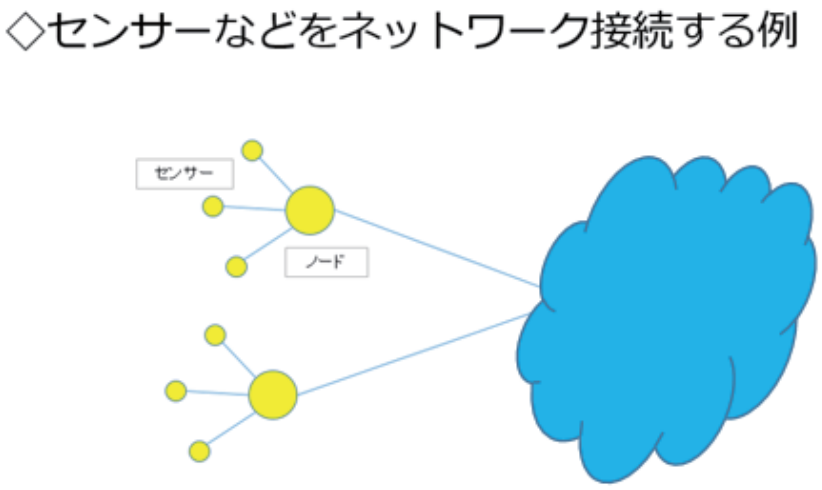


図　5

上図は、センサーなどをネットワークに接続する簡単な例示です。図 1 の様なフィールドで環境情報を取得するセンサーが繋がっているノードというものを考えて描いた図です。カエルの指先の様に見えるセンサ

一部が繋がっているのがノードです。2組しか描いていませんが、実際は無数のノードが存在します。ノードの上流にはネットワークが繋がっています。分かり易いようにとても簡単に描いたものなので、実際とは違っています。もう少し詳しく描いたものが下図です。

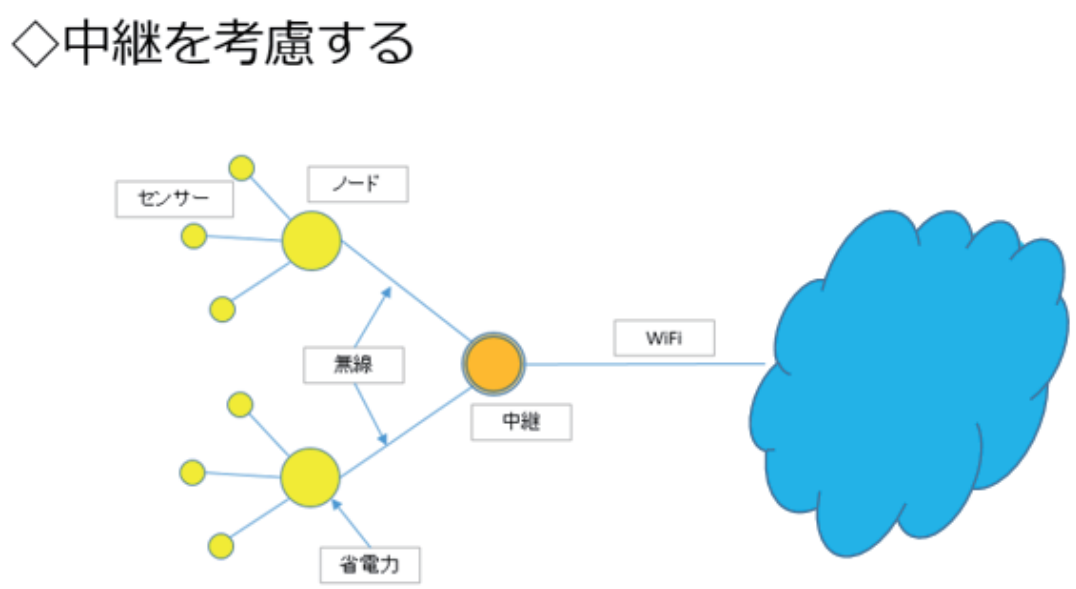


図 6

上図では、センサーが繋がっている多数のセンサーノードを、直接ネットワークに接続することは現実的に難しいので、センサーノードからの情報を整理して取り纏める【中継機能】を配置することを考えています。センサーは広いフィールドに数多く配置されますから、そのような場所での電源確保を考えると省電力（低消費電力）であることが望まれます。当然そのようなノードを数多く有線で接続することは困難ですから、無線機能が欲しくなります。センサーノードはセンサー制御機能と無線通信機能を持っていれば実現することができます。(従来は、マイコンと無線通信モジュールを組み合わせてこのようなユニットが実現されていました。)中継機能はセンサーノードからの情報を取りまとめてネットワークに流しますが、この接続もLANケーブルなどでの接続では不

便ですから、WiFi などを用いて無線で接続することで圧倒的に利便性が高くなります。現在では WiFi は近い距離での通信に限られますから、WiFi の接続先（AP：Access Point）は近くて電源も安定している場所を確保しようとすると、屋内または建物のすぐ近くになるでしょう。

◇無線の対向ノードを考慮したモデル

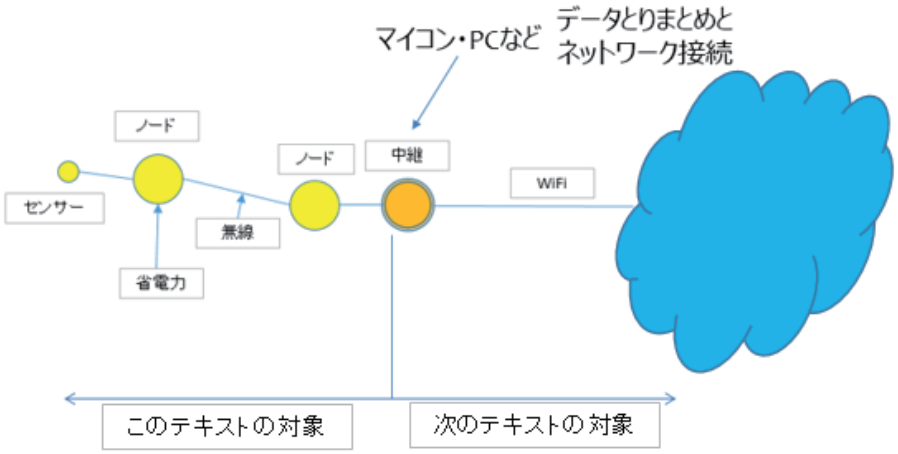


図　7

実習のできる、実際のユニットを考慮したモデルを考えたものが上図です。センサーノードはフィールドで運用されるので、電源には電池などが使えるものが良いはずです。センサーノードの無線到達距離は数十メートル～数キロメートルが望まれますが、実際に利用できるモジュールでフィールドに沢山配置する低消費電力のものでは、数十メートル～数百メートルのものが現実になります。ノードに利用できる無線モジュールが WiFi 機能を兼ね備えていると便利です。実習では、センサーノードの無線を受ける対向ノードを配置することにします。その対向ノードを、中継機能を持つ機器に直接接続（有線接続）して、センサーデータがネットワークに流れる仕組みを学びます。この場合の中継機能は PC で開発します。

このテキストでは、中継デバイスからフィールド側を対象として実習し、中継デバイスから上流側(インターネット側)は、第二分冊の対象とします。

IoTデバイス分析

講座を始める前に、IoTデバイスに必要な技術を少し調べておきましょう。そのために、IoTデバイスのモデルを考えます。

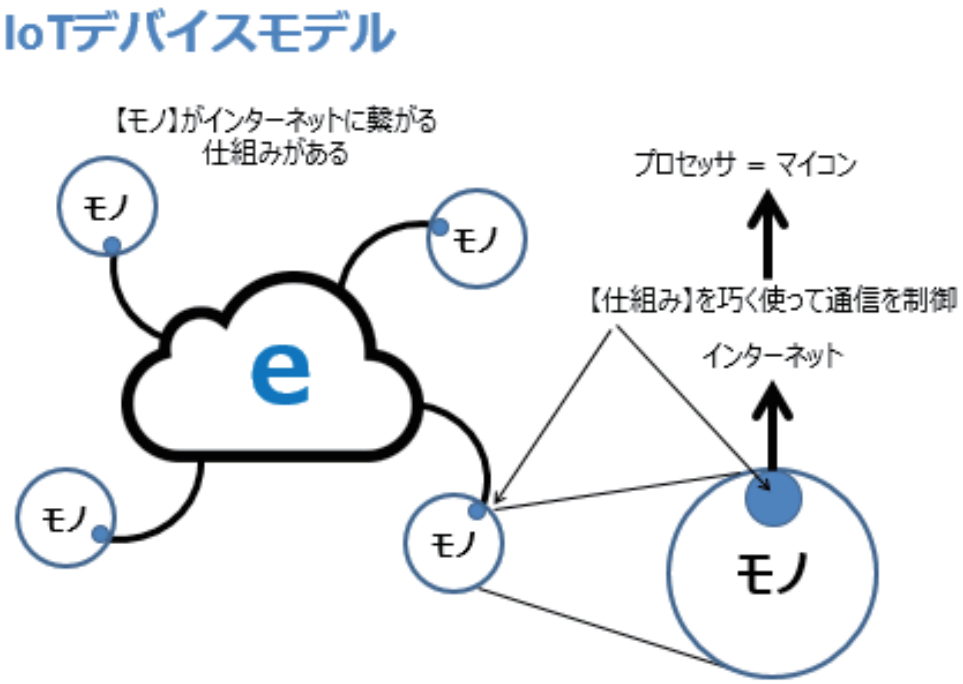


図　8

図の様にIoTデバイスには、**モノがインターネットに繋がる仕組み**があり、その仕組みを巧みに使って**通信を制御**してインターネットに繋がっています。**通信を制御できるもの＝プロセッサ**。それは**マイコン**のことです。では、IoTデバイスを分解して、内部を探ってみましょう。

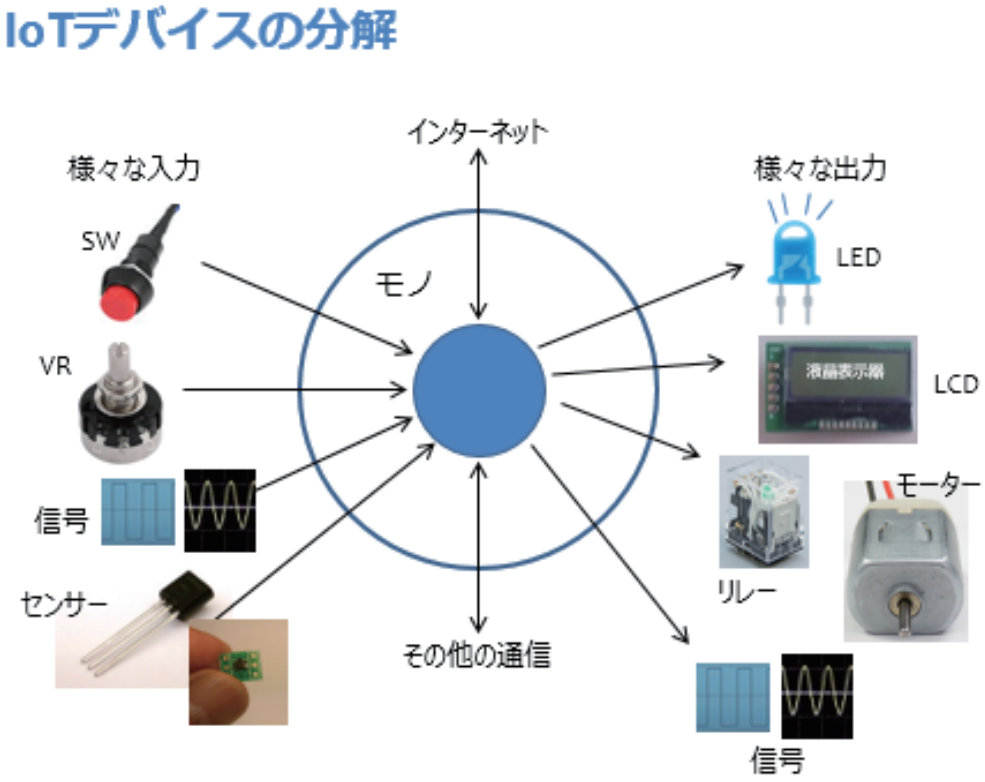


図　9

IoT デバイスには、様々な入力と様々な出力、インターネットへの接続を持っていますが、他の設備装置・機器や他の IoT デバイスにつながる通信も必要です。

入力には、SW、VR、信号、センサー、・・・デジタル、アナロ・・・etc。出力には、LED、LCD、リレー、モータ、信号、・・・デジタル、アナログ・・・etc。インターネットへの接続とその他の通信が必用です。

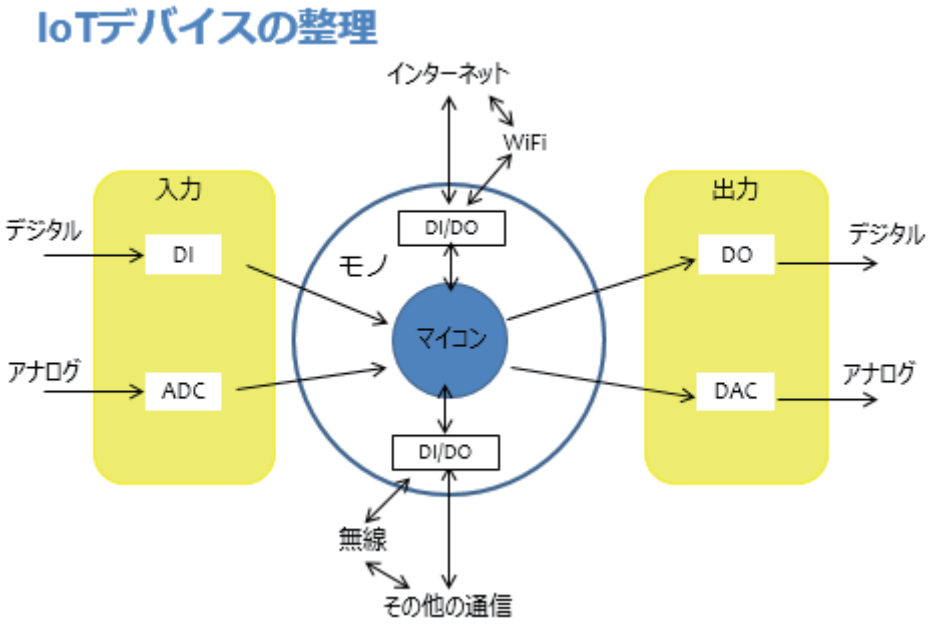


図　10

分解したIoTデバイスに必要なものを、種類ごとに整理したものが上図です。基本の入力は、デジタル（DI)ですが、アナログ入力の場合は、デジタル変換（ADC）して処理します。基本の出力は、デジタル（DO)ですが、アナログ出力したい場合は、アナログ変換（DAC)して出力します。デジタルでアナログを模擬するPWM(Pulse Width Modulation)という手法があり、マイコンの処理能力が向上したので多用されています。PWMの精度が高ければ十分アナログ出力の代わりになります。インターネット接続は、これまでケーブルによる接続が一般的でしたがこれから行う、IoTデバイス開発ではWiFiなどの無線接続によるデータ交換が必須になります。実際には、この中身もDI/DOでインターネット接続に必要なI/Fを制御しています。その他の通信も同様にDI/DOで制御されているのです。つまり、IoTデバイスは、マイコンでDI/DOをコントロールすればほとんどの機能を実装出来るのです。ADC/DACが有れば、申し分ありません。これらを踏まえて、これから無線マイコンの使い方をトレーニングしましょう！！

【無線マイコン(TWE-Lite)編】

第1回　無線マイコン

この実習では、まず末端の部分の実験を行うことで、無線マイコンの機能を確認しながら利用方法を学びます。

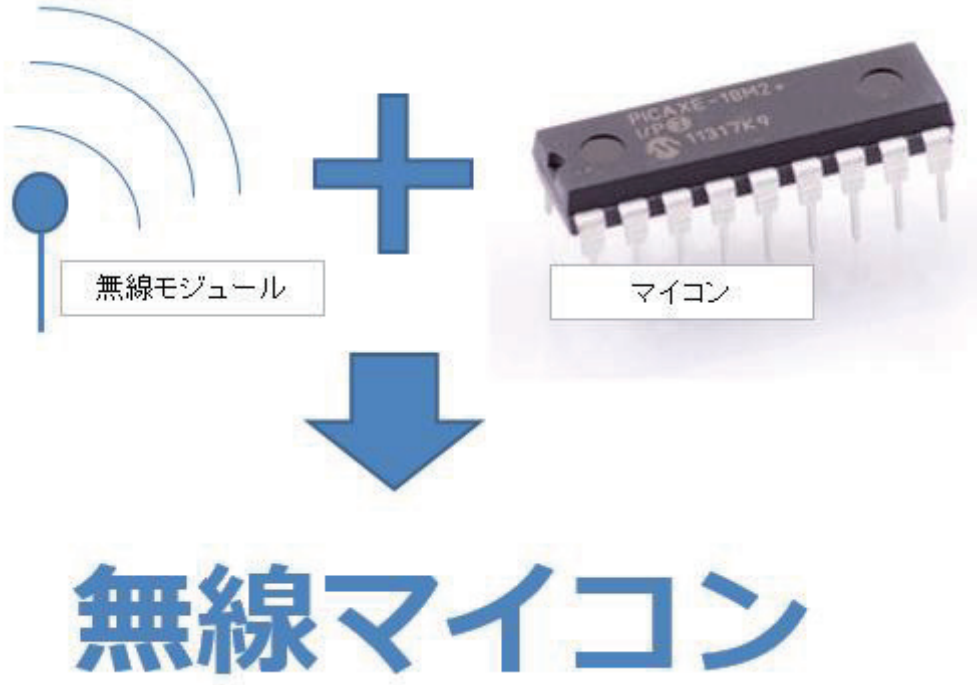


図　11

数年前までは、無線でマイコンを繋ごうとすると、上図のように、無線モジュールとマイコンを別々に用意して、それらを電気的に接続し、マイコンには、「無線モジュールを制御するプログラム」を作成して書込み、無線マイコンを実現していたのですが、数年前それらが一つになった（無線モジュールとマイコンモジュールが１パッケージになった）無線マイコンモジュールが開発されて、色々な場所で利用されています。

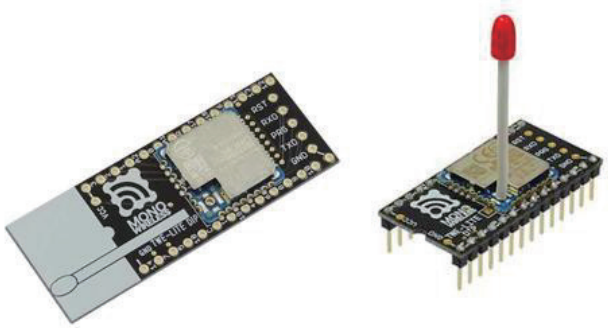


図　12

上図は、TWE-Lite（と書いてトワイライトと読む）無線マイコンモジュールです。金属のパッケージ内部に無線ユニットとマイコンユニットを内蔵しているので、このモジュール１台で無線通信のできるマイコンとして機能するように設計されています。写真左側のものは、白くプリントされた部分（アンテナの絵）の裏側にアンテナパターンが配置されているので、写真右側のようにアンテナ突起物がなくて、コンパクトになっています。

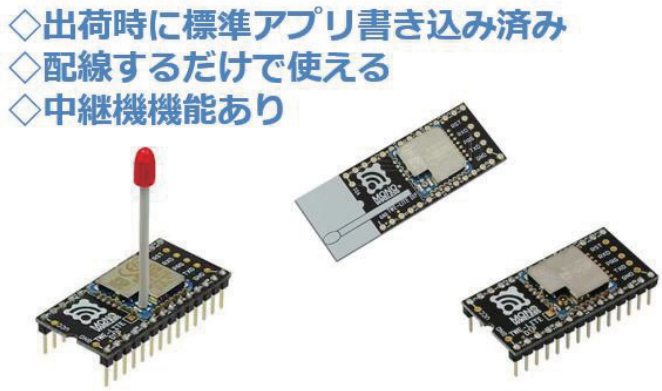


図　13

このモジュールは、出荷時に標準の機能を持つアプリケーションが書込み済みとなっています。この標準アプリケーションの機能範囲での利

用であれば、配線をするだけで無線マイコンモジュールとして使用することができます（プログラミングレス）。無線というと送信機と受信機の間での通信を行うわけですが、このモジュールは設定を行えば、送信機と受信機の間に入って中継器として働き、全体での通信距離が伸ばせるという機能があります。中継機としては、送信機と受信機の間に 2 台まで設定できる設計となっています。

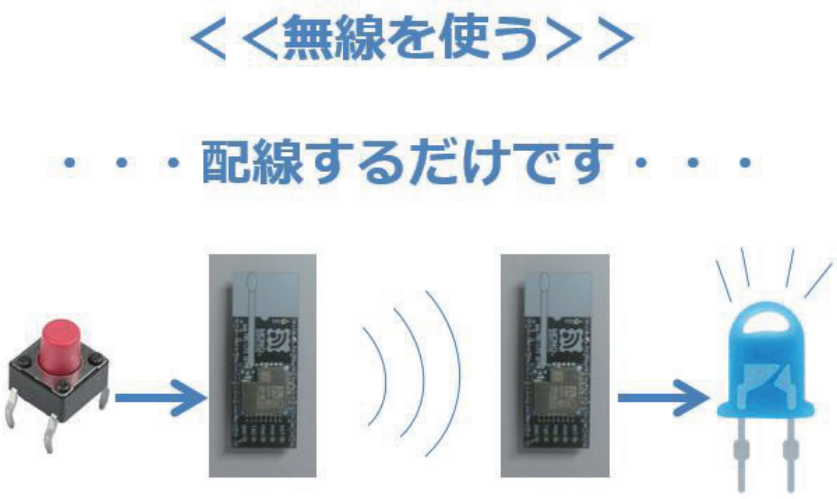


図 14

無線機能を使うとどのようなことができるのか、まず使ってみることにしましょう。上図のように、無線マイコンモジュールを 2 台使い、一方に SW、他方に LED を接続します。SW を押すとその状態が対向機のモジュールに無線で通知されて LED の点灯制御が、遠隔で行えるようになります。この機能は工場出荷時にこのマイコンに書き込まれている標準アプリケーション機能の範囲内ですので、プログラミングレスで実現できます。

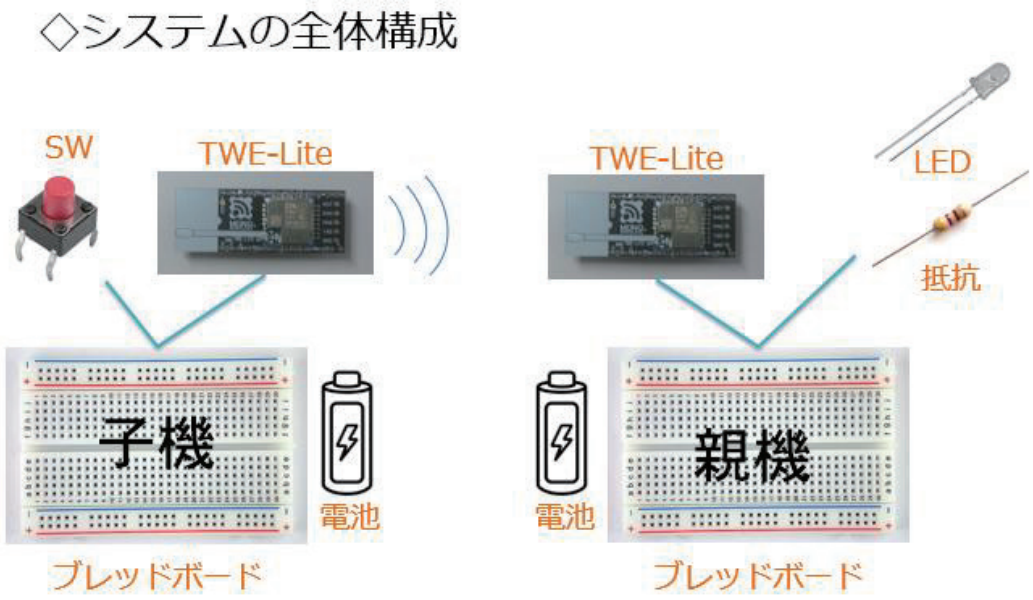


図 15

システム全体構成を図に示します。説明の都合上 SW を接続する側を子機、LED を接続する側を親機とします。必要なパーツは下記です。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. SW×１個

親機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. LED×１個
6. 抵抗器（470Ω）×1 個

無線マイコンモジュールは、親機・子機ともに全く同じものです。SW は「タクト SW」を利用します。SW 全体の大きさや押しボタン部分の色や長さの違うものがありますが、どれでも利用できます。ブレッドボードは、半田付けせずに配線ができるので大変便利です。

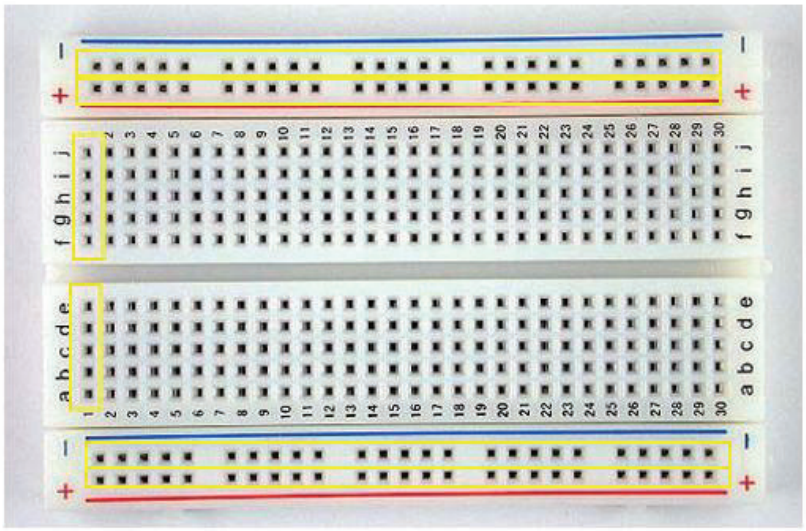


図　16(ブレッドボード内部)

◇マイナス(－)の線、プラスの線(＋)、
A～E、F～J の線はブレッドボード内部で
繋がっています。

上図のように、ブレッドボード内部で穴が接続されています。表側上

下の＋－のマークのある部分は、写真で横方向に接続されています。アルファベットが刻印されている部分は、縦方向に接続されています。中央部分の溝の上下は、繋がっていません。SW や IC などは、この溝を跨ぐように配置します。青色の線がある部分の穴には、GND（－）側を、赤色の線がある部分の穴には、電源（＋）側を接続して利用します。ブレッドボードの上下の赤・青の電源（＋）・GND（－）を両方使う場合は、上下の同じ色の線の穴をジャンパー線でつないで使用します。

電源は、1.5V 乾電池を 2 個直列に接続して、3V として使用します。この実習で使用する無線マイコンモジュールは 3V で駆動できます。

実習キットに含まれている電池ケースは 2 種類あり、単 4 乾電池を 2 本使用するもの（下図）と 3 本使用するものがあります。この実験で使用する電池ボックスは、単 4×2 本のものですので、間違えないようにして下さい。3 本のものを使ってしまうと、電圧が 4.5V になり、無線マイコンモジュールの電源電圧（2.3～3.6V）の範囲を超えてしまうので、注意してください。また、ケースには SW が付いていますので、動作確認を行うまでは、SW を OFF にしておいてください。

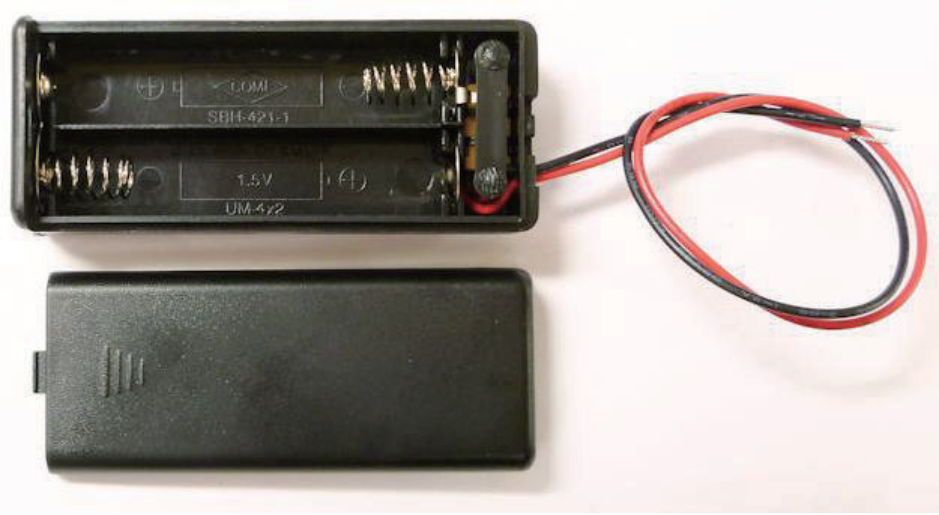


図 17

図 18

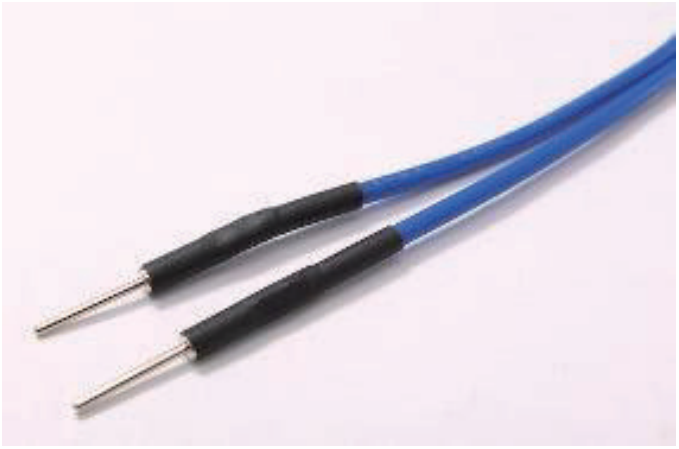
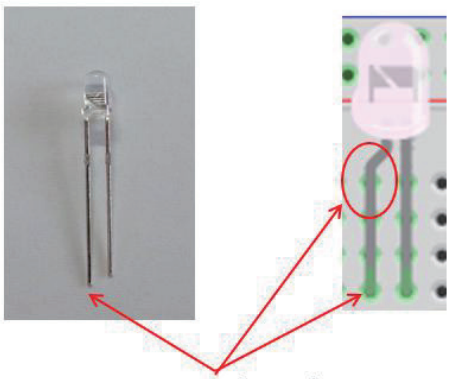


図 19

配線用ジャンパー線（上図）は、両端にピンが取り付けられています。このピンをブレッドボードの穴に差し込み配線します。

LED は、下図のように足の長さで極性を示しています。長い方の脚から電流が流れ込んで、短い方の脚から流れ出てきます。反対に接続すると電流が流れずに光りません。実体配線図では足の長さが分りにくいので、長い方の脚を曲げて表現しています。



◇図では、長いほうのピンがわかりにくいので
曲げて表現しています。気を付けて配線してください。

図 20

抵抗器は、カラーバーで抵抗値を示しています。今回使用する抵抗器は 470Ω のもので、LED に電流が流れすぎないようにするために使います。水道の水栓ハンドルのようなものです。抵抗の両端で足を直角に曲げて使用します。指で簡単に曲げられます。

◇抵抗は写真のように足を曲げて使います。
◇抵抗の値を書いたものを付けておくと、間違えにくくなります。

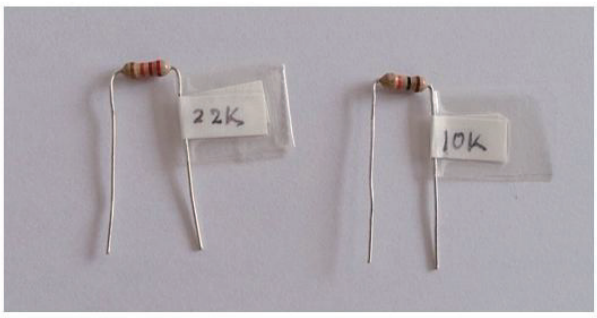


図 21

上図の様に紙片に抵抗値を書いて貼っておくと、値の違う抵抗を間違いなく利用できます。回路を作成する前に、このような準備を念入りに行うことは、後の作業の効率化や誤り排除などの面で、とても効果があ

ります。心がけてください。実習キットにはすべてのパーツがそろっていますが、個々に求めたパーツを利用する場合と同様に、極性や抵抗値などを十分に確認してから使用して下さい。

無線マイコンモジュールは、下の写真の矢印の部分が弱いので、ブレッドボードへの取付け・取り外し（特に取り外し）の際に、強い力がかからないように、差し込むときは少しずつ、取り外しにはピンセットなどを使い、少しずつ抜いて取り外してください。

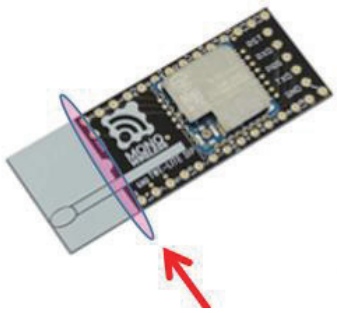


図 22

ピン配置表

機能	信号名	シルク	ピン
電源グランド	GND	GND	1
I2Cクロック	SCL	14	2
UART受信	RX	7	3
PWM出力1	PWM1	5	4
デジタル出力1	DO1	18	5
PWM出力2	PWM2	C	6
PWM出力3	PWM3	I	7
デジタル出力2	DO2	19	8
デジタル出力3	DO3	4	9
UART送信	TX	6	10
PWM出力4	PWM4	8	11
デジタル出力4	DO4	9	12
モード設定ピット1	M1	10	13
電源グランド	GND	GND	14

ピン	シルク	信号名	機能
28	VCC	VCC	電源(2.3～3.6V)
27	3	M3	モード設定ピット3
26	2	M2	モード設定ピット2
25	1	AI4	アナログ入力4
24	A2	AI3	アナログ入力3
23	0	AI2	アナログ入力2
22	A1	AI1	アナログ入力1
21	R	RST	リセット入力
20	17	BPS	UART速度設定
19	15	SDA	I2Cデータ
18	16	DI4	デジタル入力4
17	11	DI3	デジタル入力3
16	13	DI2	デジタル入力2
15	12	DI1	デジタル入力1

図 23

図は標準アプリケーションが書込まれている無線マイコンモジュールの信号配置の表です。モジュールの半円形の切り欠き部分を上に向けて、左側の上のピンから左回りに 1,2,3,4・・・と番号がついています。

「ピン」の項目がピン番号です。シルクというのは、モジュールに印刷されている文字のことです。

----------<< 各ピンに配置されている信号 >>----------

1番：電源グランドです。電池駆動の際はマイナス側に接続します。同じ機能のピンが14番にもあります。(GND)

2番：I2C クロック。アイ・ツー・シーやアイ・スクエアード・シーなどと呼ばれる、2線式シリアルインターフェイスのクロック信号です。I2Cのデータは19番ピンに配置されています。

3番：UART 受信。シリアル通信の受信信号です。送信信号は10番ピンに配置されています。

4番：PWM 出力 1。Pulse Width Modulationという、一定周期で発生するパルスの幅で、デバイスを制御する信号です。全部で4chあり、4,6,7,11番ピンに配置されています。

5番：デジタル出力 1。ON/OFFの制御を行うデジタル信号で、全部で4chあり、5,8,9,12番ピンに配置されています。

13番：モード設定ビット1。このモジュールが電源投入後、起動する際に、このモード設定ビットの信号をGND（マイナス）に接続することで立ち上がった後の動作を決める信号です。全部で 3bitあり、13,26,27番ピンに配置されています。

15番：デジタル入力 1。SWなどのデジタル入力信号を取り扱うピンです。全部で4chあり15,16,17,18番ピンに配置されています。

20番：UART 速度設定。このモジュールのシリアル通信の通信速度は内部設定で変更することができます。設定変更した場合、この信号をGND（マイナス）に接続することでその設定が有効になります。解放している場合は、通常の速度設定で動作します。

21番：Reset信号。モジュールのリセット信号です。

22番：アナログ入力1。センサーなどの出力電圧を測定するA/D変換器を内蔵しているピンです。前部で4chあり、22,23,24,25番ピンに配置されています。使用しないときは、この信号がふら付かない様に、電源の＋またはGNDに接続しておきます。

28番：電源の＋側です。この講座では電池の＋（3V）に接続します。

前掲のピン配置表は、無線マイコンモジュールに書込むアプリケーションで柔軟に変更ができるようになっていますので、書込まれたアプリケーションが異なると、ピンの機能も変わります。ですから、あくまでも【標準アプリケーションでのピン配置】と考えておいてください。

いよいよ無線マイコンモジュールの実験回路を配線します。配線と言っても線の数や使用するデバイスの数が少ないので、じっくりと確認しながら行ってもすぐに終わります。各パーツには、くれぐれも余計な力がかからないようにして､回路の配線を行ってください｡手順としては、半田付けする基盤の場合は、高さの低いものから順に基板に取り付けてゆくのが王道ですが、ブレッドボード・ジャンパ線での配線なので、作業しやすいと思った順に行えばよいでしょう。回を重ねれば、自ずと最適な作業手順が身についてくると思います。

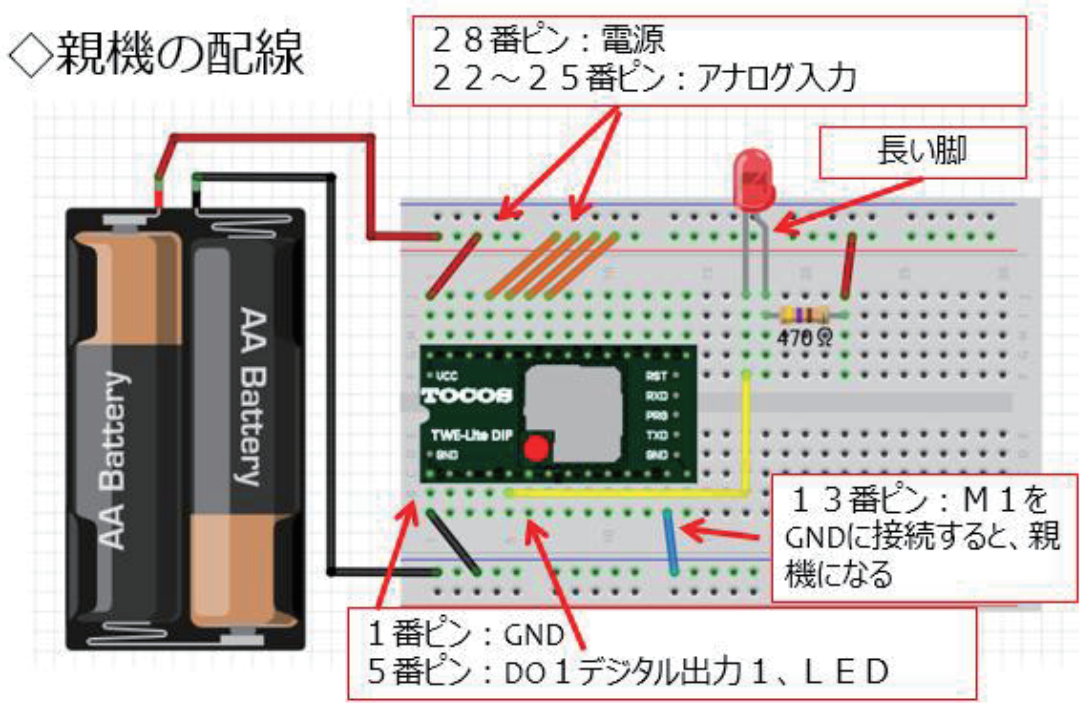


図　24

まずは、親機の配線です。ブレッドボードを小さい数字が左、アルファベットのＡが左下になるように置いて、親機を上図のように配線してください。電池ケースのSWがOFFの状態であることを確認して配線しましょう。上図ではアンテナパターンが印刷されている白い部分が省略されていますので注意してください。実際にブレッドボードに無線マイコンモジュールを取り付けると、アンテナパターンの部分が左側にはみ出します。ピン配置は同じです。使用しているジャンパー線の色は、特に指定ではありませんが、電源（＋）側は赤系統、GND（－）側には青や黒を使うことが多いので、覚えておくと別の回路を見たときに役立ちます。

1番ピンは、電源のGND（－）に、28番ピンは電源の（＋）に接続します。22～25番ピンは電源の（＋）に接続しておきます。このマイコンモジュールはアナログ信号の変化により状態が変わると、対向機に状

態が変化したことを知らせる無線通信が発生してしまうため、これを防ぐ目的で、アナログ入力のレベルを固定しておくための配線です。

13 番ピンは、GND に接続します。このようにすると、起動後のモジュールは親機として機能します。

LED は、5 番ピンからジャンパー線で LED の短い方の脚に接続します。LED は極性があります。誤って長い方の脚に接続すると意図したように動作しませんので注意して下さい。LED の長い方の脚は抵抗器を経由して電源の（＋）側に接続します。LED には、5 番ピンの信号レベルが Low になったときに、電源（＋）側から、抵抗を経由して LED の長い方の脚から電流が流れ込み（LED が光る）、短い方の脚から流れ出て、5 番ピンに吸い込まれると考えて下さい。

実際に配線した親機の様子が次の写真です。

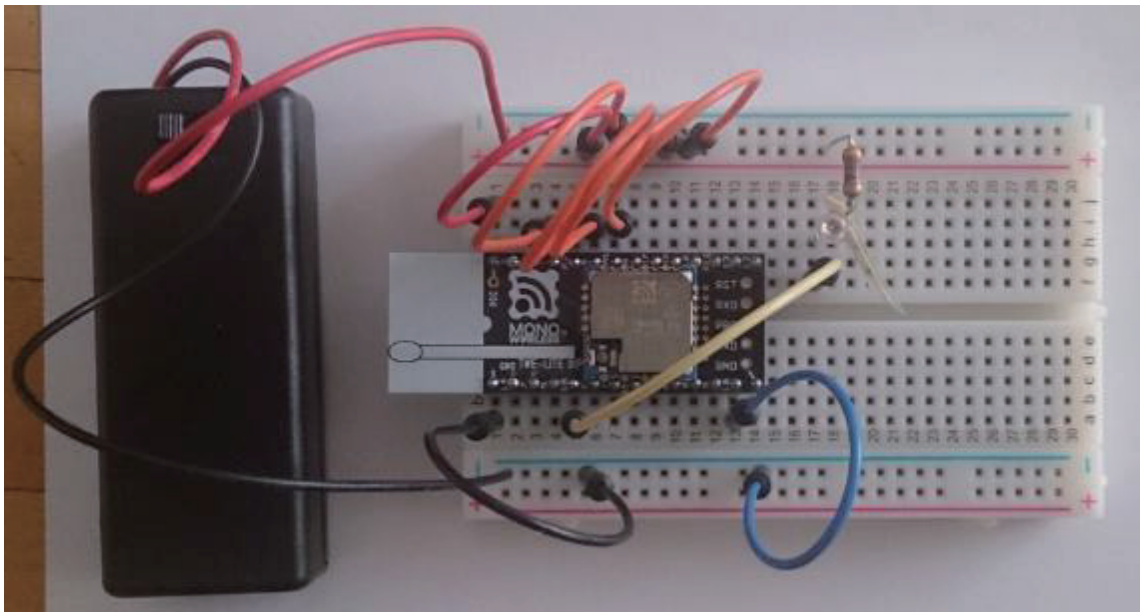


図 25

◇子機の配線

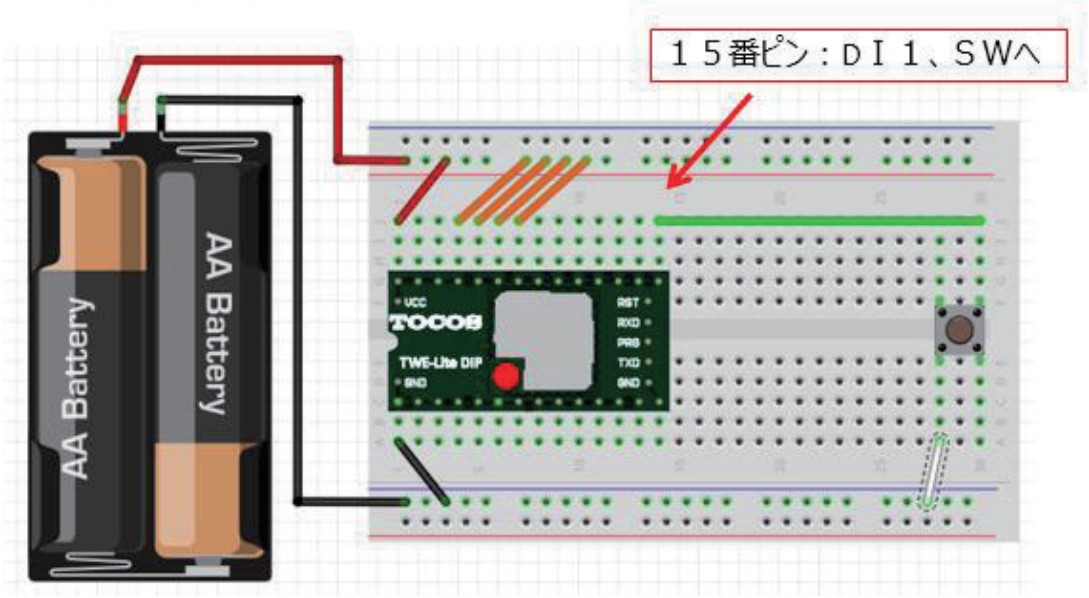


図 26

次に子機の配線を行います。子機は親機に比べて、とてもシンプルですが、侮ると配線間違いをしてしまいます。落着いて配線してください。親機と似ている（というより同じ）部分が多いですね。図の左半分は親機と全く同じです。親機の 13 番ピン（M1）の信号を GND に接続するジャンパー線がありません。繰り返しですが 13 番ピンを解放するか GND 接続するかによって、親機として振舞うか、子機として動作するかが決まります。

15 番ピンは、デジタル入力 1 の信号です。ここにジャンパー線で SW を接続します。SW の反対側は、ジャンパー線で GND に接続してやると、デジタル入力 1 の信号は SW を押したときには Low レベルになります。これをマイコンのプログラムで読み込むと該当するビットが 0（ゼロ）になります。この状態を対向機に知らせて、LED を点灯したいので、【SW を押す→デジタル入力が Low→デジタル出力が 0→LED 点灯】となります。デジタル出力ピンで電気を吸い込んで LED が光るように、親機で

配線した理由が分かりますね。図のジャンパー線の色は適当なものを使用して配線して構いません。大切なことは、どんなに簡単な回路でも、念入りに確認をすることです。実際に配線した子機の様子が次の写真です。

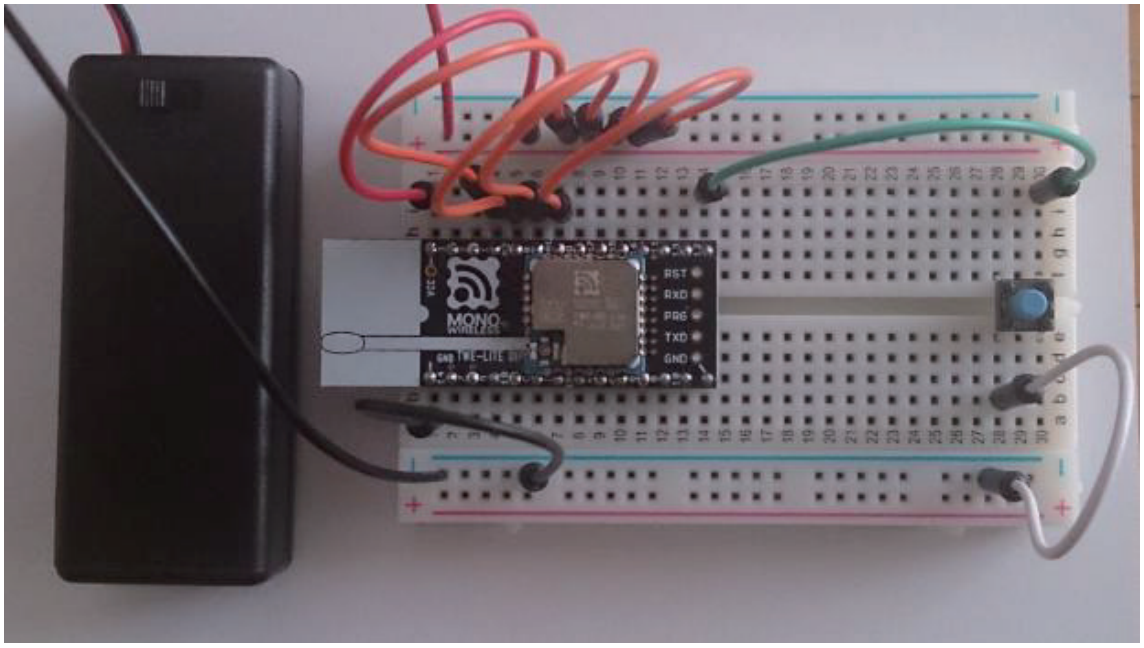


図 27

ジャンパー線を使うと、どうしても写真写りがごちゃごちゃしてしまいます。配線図を見ながら、よく確認をしてください。

配線の確認ができたら、いよいよ動作確認です。親機・子機の電池ボックスにある SW を ON 側にスライドさせてください。次に子機の基板上の SW を押してください。SW を押すと同時に親機の LED が光るでしょうか？SW から指を離すと LED は消灯するはずです。上手くいかない様でしたら、配線をチェックしましょう。配線が間違っていなくても、しっかりと差し込まれていないと接触不良で正しく動作しません。電池ボックスの中の乾電池の向きはいかがでしょうか。確認することはいっぱいありますが、一つ一つ自分で行った作業ですから、自分で確認をし

て正しい動作をするように修正をしましょう。動作の様子を写真に示します。

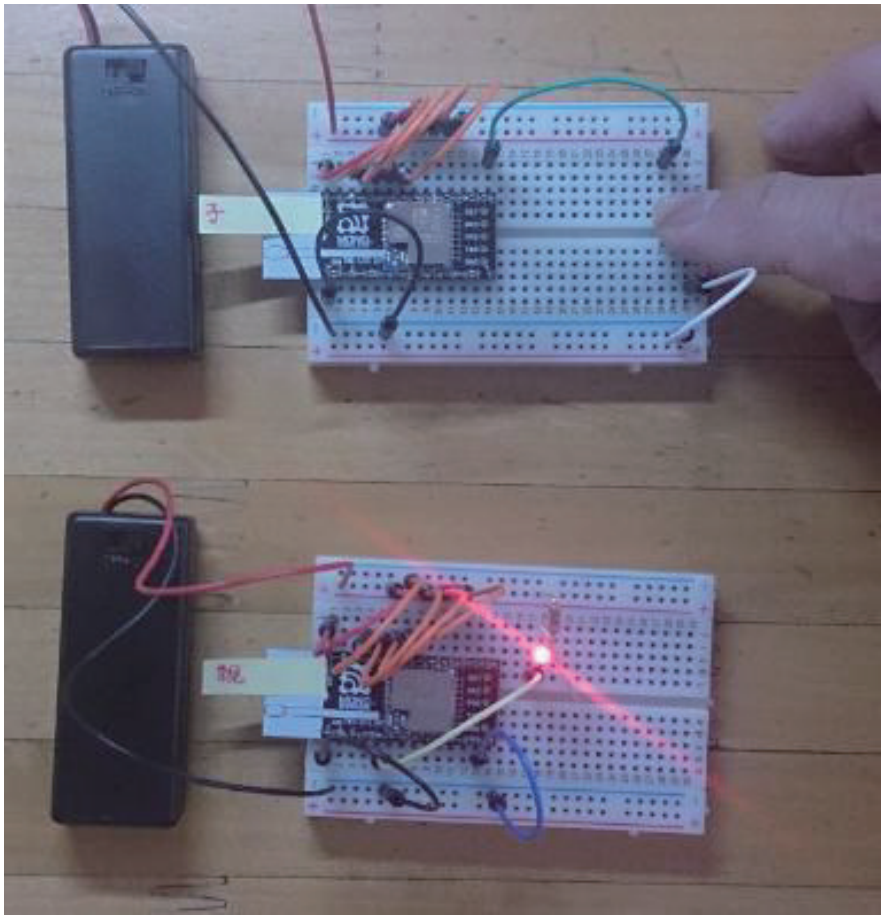


図 28

この回路が作れると、LED の代わりに例えばガレージの扉開閉や、ハウスの天井にある換気窓の開閉、ファンの起動などがリモートコントロールできるようになります。LED の ON/OFF 信号で AC（交流）を制御するときはリレーや SSR（ソリッド・ステート・リレー）などを使います。

この回で作成した回路は、第 2 回でもそのまま使用できます。可能であれば、そのまま保管してください。但し、しばらく時間が空くときは電池ケースのＳＷを OFF にして、電源の配線は抜いてください。電池を取り出しておけば完璧です。

第2回　双方向無線通信

次は双方向の無線通信で、機器の制御を行ってみましょう。第１回では、子機と親機を設定して子機側の SW の情報を無線通信で親機に送り、その内容に応じて LED を点滅させることができまました。第 2 回は、同じことを双方向で行います。

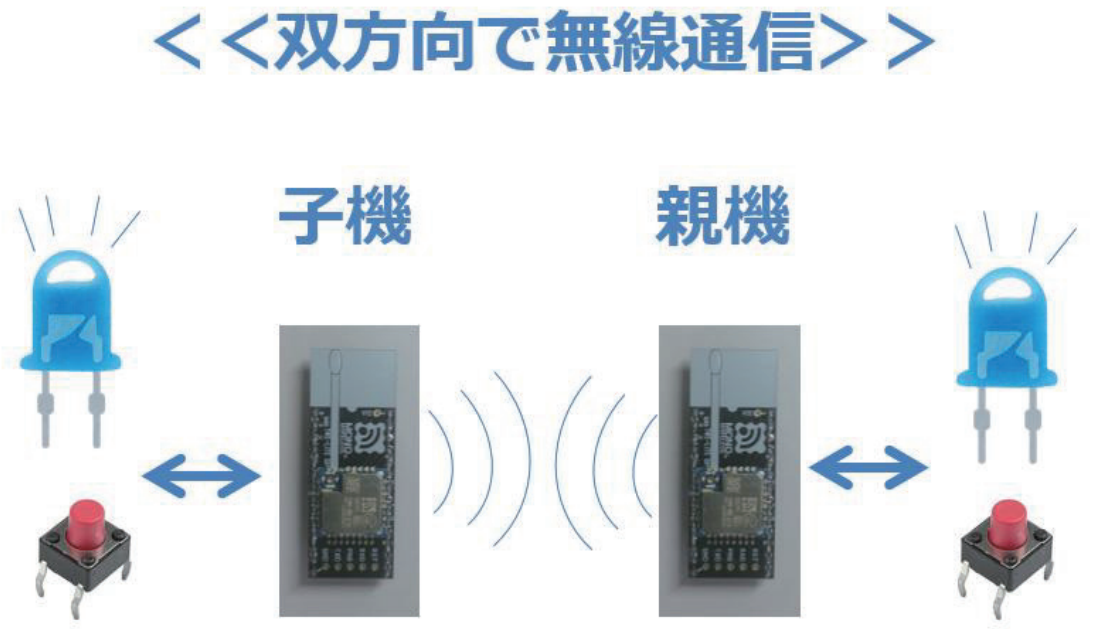


図　29

上図は、今回の実習の内容を描いたものです。前回は親機と子機で担当する機能が違っていたので、無線マイコンモジュールに取り付けたデバイスは別のものでしたが、今回は双方向で同じことを行います。子機側の SW の状態を親機側の LED に反映し、親機側の SW の状態を子機側の LED に反映します。この機能も工場出荷時にマイコンに書き込まれている標準アプリケーションを使って実現します。と、言うことは回路の配線だけで実現できるので、配線の正確性が求められます。

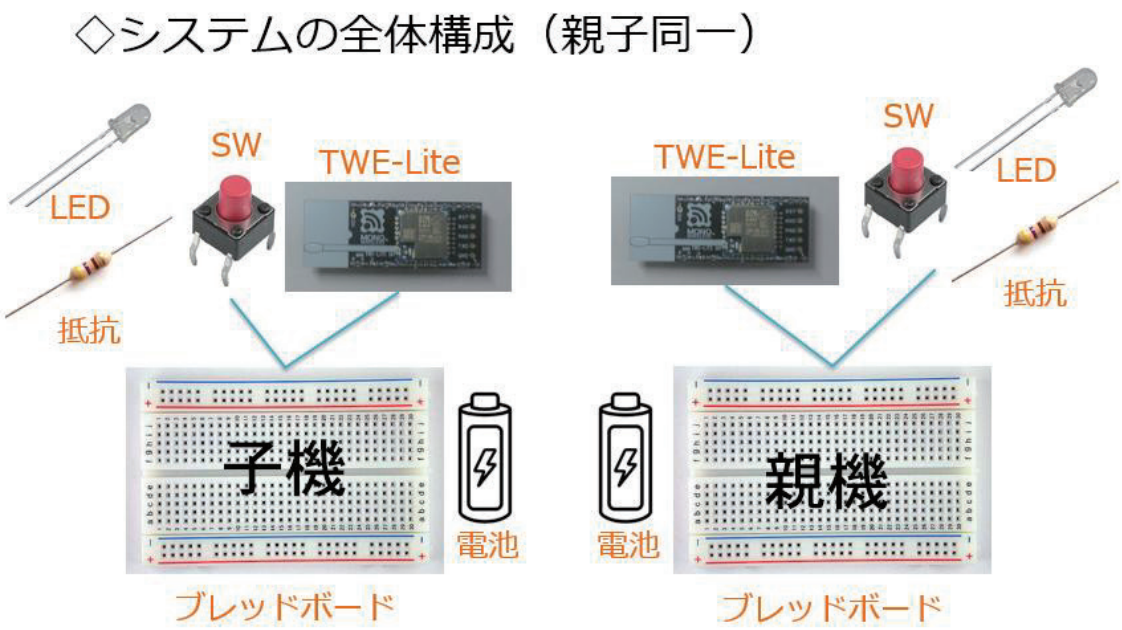


図　30

システム全体構成を上図に示します。親機・子機ともに同じパーツを使用するので、必要なパーツは各々2組です。（下記）

子機側＋親機側：

1. 無線マイコンモジュール×2台
2. ブレッドボード×2個
3. 電池（単4乾電池×2個＋SW付電池ケース×1個）×2セット
4. 配線用ジャンパー線
5. SW×2個
5. LED×2個
6. 抵抗器（470Ω）×2個（LED電流制限用）

各パーツの内容は第1回の説明を参照してください。

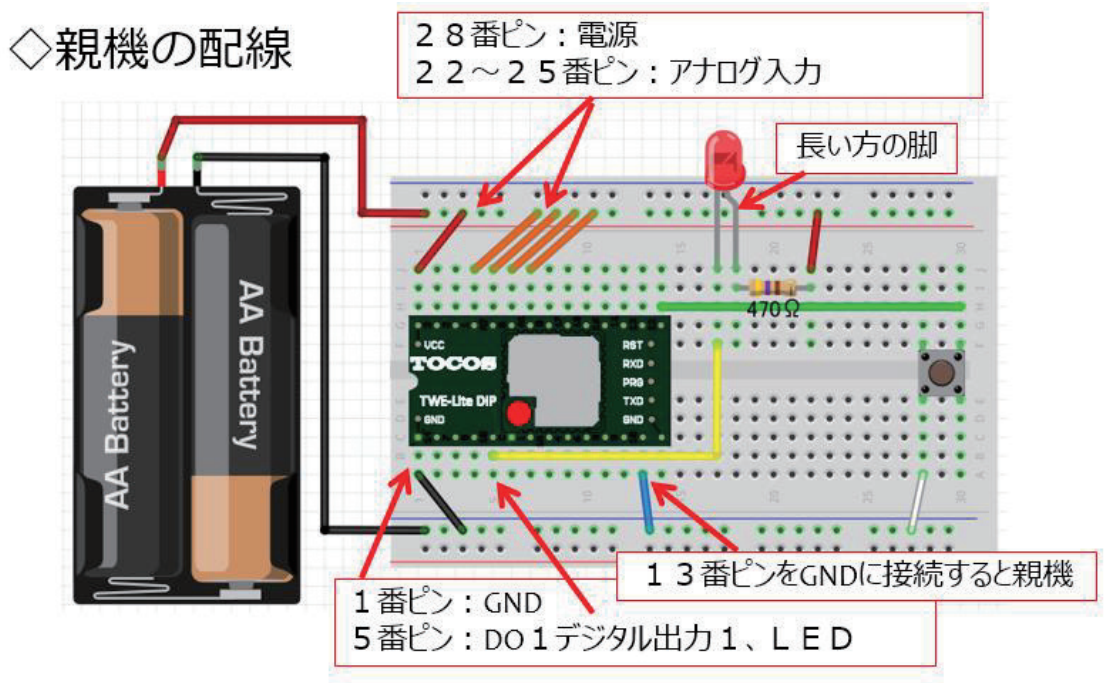


図　31

まず、上図に従って親機を配線しましょう。第１回で配線した親機と子機をミックスした回路になっています。回路がそのまま残っているのであれば、親機には子機の SW を、子機には親機の LED と抵抗器を取り付けて配線します。新たに作成する場合は、第１回目の説明を思い出して配線してください。

実際に配線した親機の様子が次の写真です。LED と抵抗器、SW が同じブレッドボード上に配線されたので少しごちゃごちゃしていますが、配線に誤りはないでしょうか。写真中央下の青いジャンパー線が親機として機能する配線です。（第１回を参照）

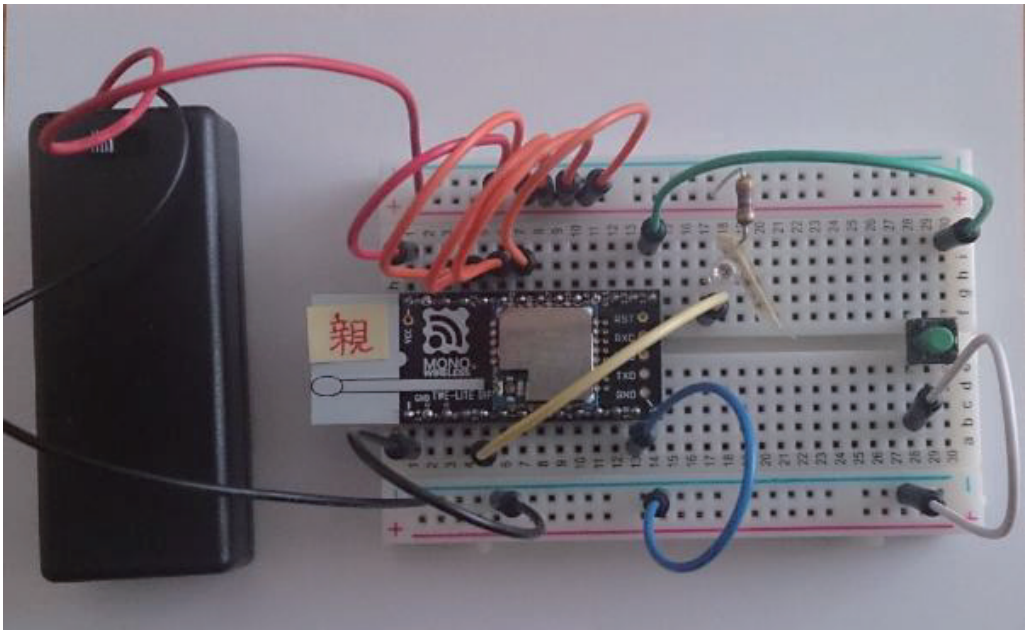


図　32

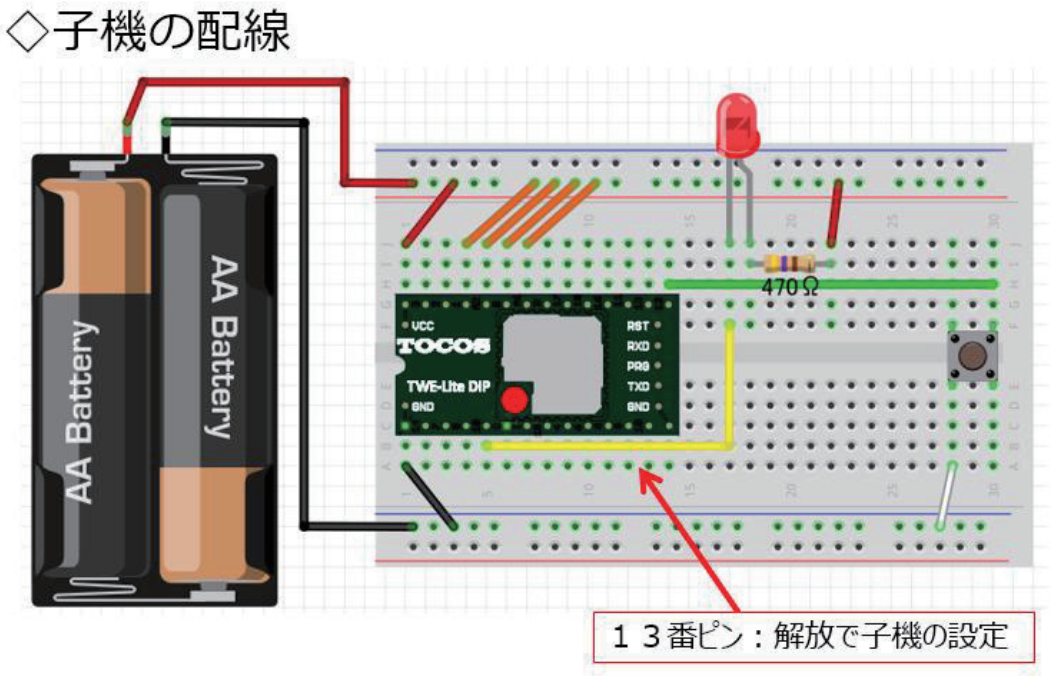


図　33

次に子機の配線を行います。上図に従って配線します。すでに気づいたと思いますが、子機側の配線は親機の回路の 13 番ピンのジャンパー線が無いものです。13 番ピンを GND に接続するとこの無線マイコンモジュールは親機として機能し、13 番ピンが解放ならば子機として機能し

ます。ですから**2組の同じ回路を作り、13番ピンのジャンパー線の有無で回路の区別をしても構いません。**実際に配線した子機の様子が次の写真です。**親機との違いはジャンパー線1本だけ**です。

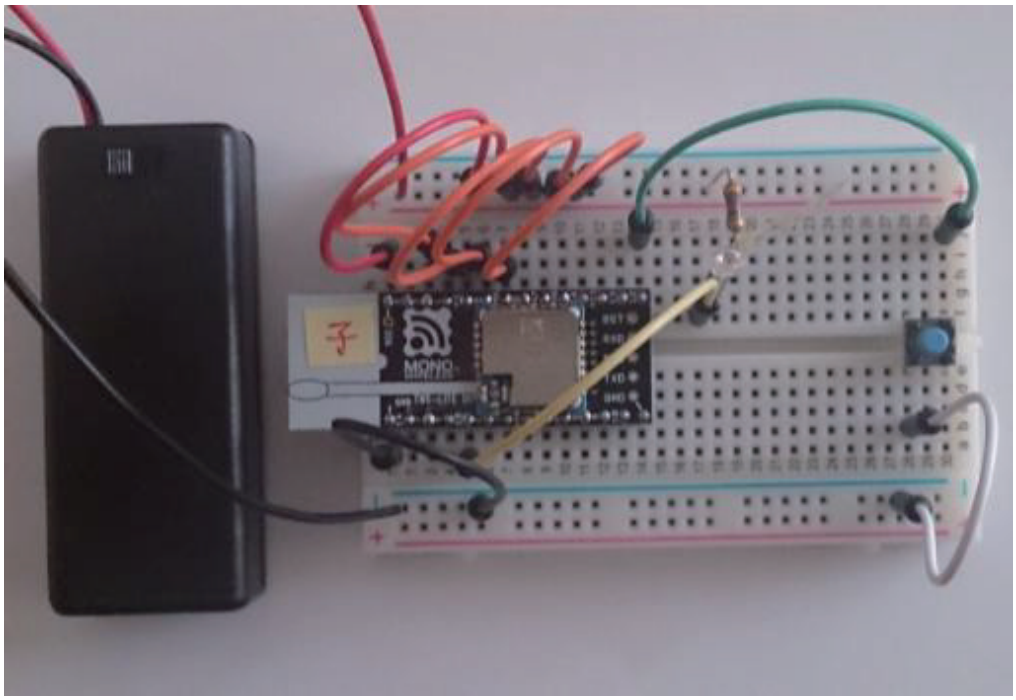


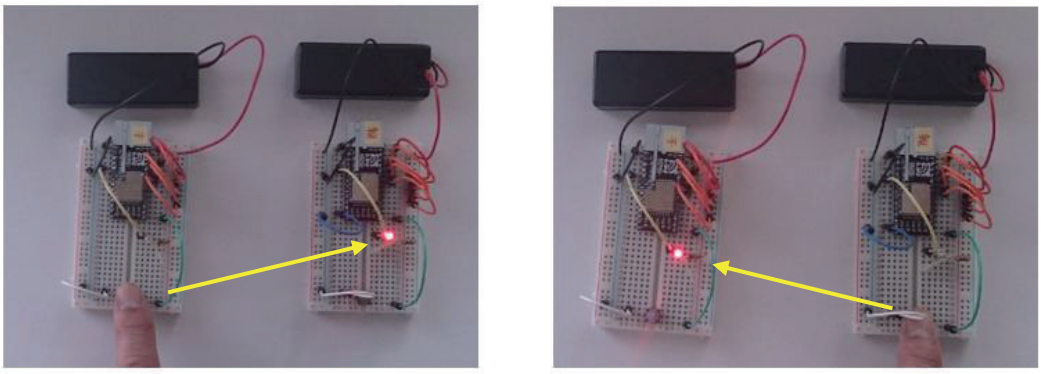
図 34

回路の作成ができたら、油断せずに良く配線を確認してください。第1回の回路に配線を付加して作成した場合は、ジャンパー線やSWの緩みも確認して下さい。電池ボックスからの配線も要点検個所です。

◇動作確認：

配線の確認が済んだら、電池ボックスのSWをONにスライドして電源を入れます。まず子機のSWを押してみましょう。親機のLEDが反応するでしょうか。次に親機のSWを押してみましょう。同様に子機のLEDの点灯が確認できるでしょう。両方のSWを細かく押したり離したりするとどうでしょうか。

◇とても簡単に双方向の無線通信ができました。
◇デジタル制御の通信がプログラミングレス！



子機から親機へ　　親機から子機へ

図　35

◇無線到達距離について

今回のような双方向の無線通信ができると、二人で親機・子機それぞれを持ち、だんだん離れながらどのくらいの距離まで、通信ができるか測ることができます。この講座で使用している無線マイコンモジュールは、アンテナの状態や周囲の電波環境が良ければ 1km ほどの到達距離があるとされています。無線通信の到達距離を伸ばすには次の点を確認しながら実験を行ってみてください。河川敷などの広い場所で他の電波の影響を受けないような環境で実験すると良いでしょう。この実験では携帯電話を機内モードにするなどとした手順を考えてみてください。

◇アンテナを親機と子機で平行になるように
◇アンテナ高さをできるだけ高くする
◇電波ノイズの少ない環境がよい
　※2.4GHz帯を使用しているので、
　　同じ周波数帯の機器との併用は要配慮
◇高さ３m以上で、電波環境の良いところでは
　およそ１ｋmの距離で通信が可能

図　36

第3回　スマートフォン連携

この講座で使用している無線マイコンモジュールには、親機としてスマートフォンに USB 接続のできる機器が開発されています。このモジュールを使用することで、無線マイコンモジュールとスマートフォンが連携するシステムを作ることができます。

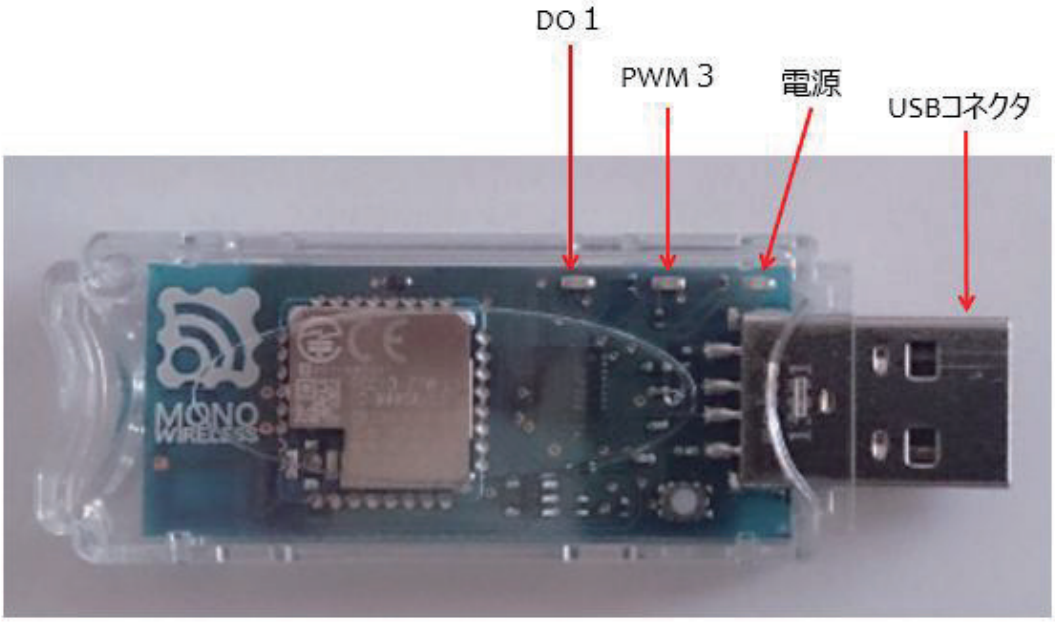


図　37

図は MoNoSTICK モジュールです。USB コネクタがあり、これでスマートフォンに接続をします。一見 USB メモリとそっくりな外観です。内部には基板があり、無線マイコンモジュールと同様に金属ケースが中央にあります。この中にマイコンユニットと無線ユニットが入っています。基板左側のメーカーロゴが印刷されているあたりの裏側にアンテナパターンが配置されていて、無線通信が行えます。USB コネクタの付け根部分の上側には、小さい LED が配置されています。それぞれ、電源、

PWM3（PWM 出力の 3 番）、DO1（デジタル出力の 1 番）の LED です。スマートフォンとの連携は下図のようになります。

図 38

左側無線マイコンモジュールの子機と USB I/F を持つモジュール（親機）との間で双方向の無線通信をおこないます。スマートフォンで子機の様子をモニターしたり、あるいはスマートフォンから子機をコントロールしたりできます。スマートフォンには、連携用のソフトウエアがありませんので、**専用のアプリをインストール**して使います。

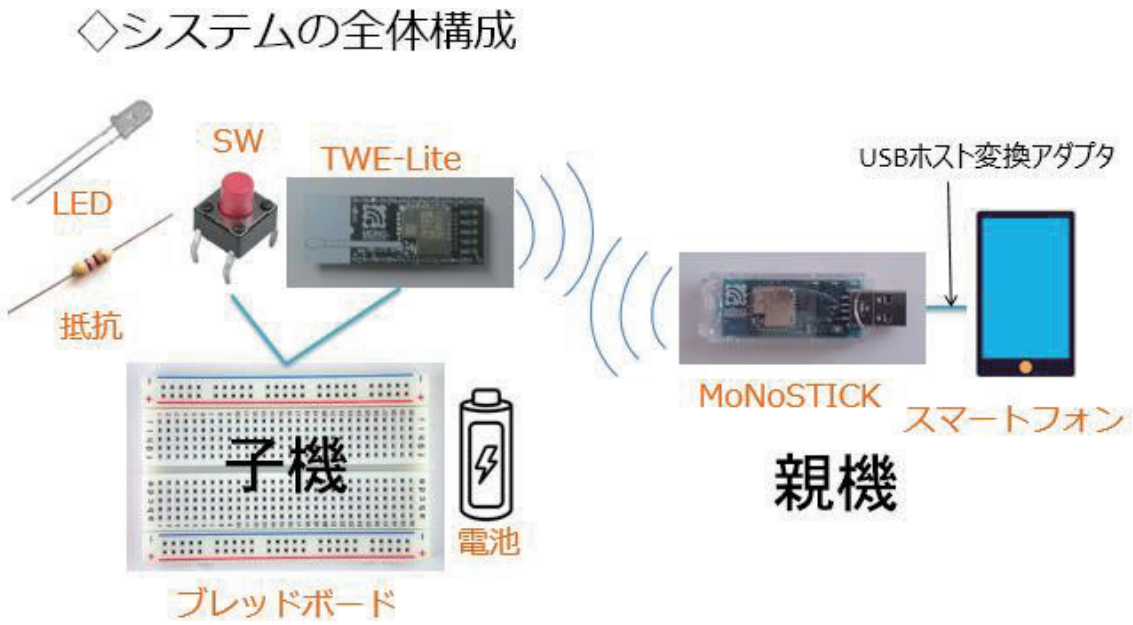


図 39

上図がシステム全体構成です。今回の実習では、第2回で使用した子機と全く同じものを使います。子機側に配置したSWの状態をモニターしたり、LEDを点滅させたりするので、子機は親機であるMoNoSTICKと双方向無線通信を行います。親機であるMoNoSTICKはわずかなLEDしか持っていませんので、モニターや操作はスマートフォンの画面を通じて行います。MoNoSTICKのUSBコネクタはそのままではスマートフォンに接続できません。次の写真に示すUSBホスト変換アダプターを使いMoNoSTICKを接続します。

図 40

USBホスト変換アダプターは、一方にmicroUSB（またはTypeC）オスコネクタがあり、こちらをスマートフォンに接続します。他方にはUSBメスコネクタがあるので、こちらにMoNoSTICKモジュールを接続します。

使用するパーツは下記です。

子機側：

1. 無線マイコンモジュール×1台
2. ブレッドボード×1個
3. 電池（単4乾電池×2個＋SW付電池ケース×1個）×1セット
4. 配線用ジャンパー線

5. SW×1 個

5. LED×1 個

6. 抵抗器（470Ω）×1 個（LED 電流制限用）

親機側：

1. MoNoSTICK×1 台

2. USB ホスト変換アダプター×1 本（コネクタ形状により選択）

3. スマホアプリ TWE Control×1 セット

◇子機の配線

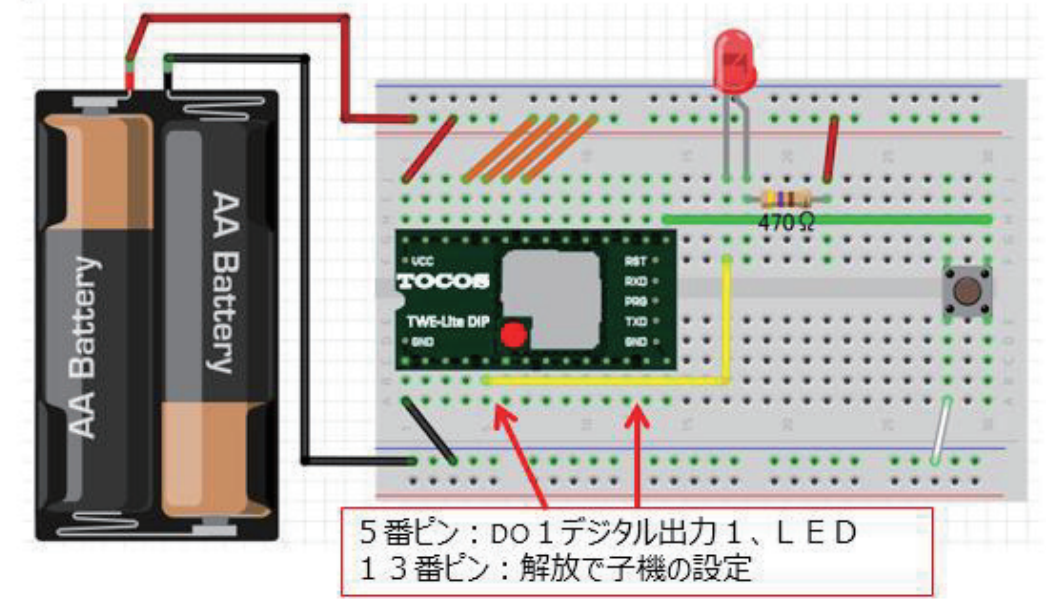


図 41

この実習で使用する子機は、第 2 回で使用した子機と同じものです。すでに作成済みの子機を利用される方は、親機と子機を間違えないようにしてください。13 番ピンが解放されているものが子機です。新たに作成される方は上図をよく見て間違いの無いように配線を行ってください。実際に作製した子機の様子が下の写真です。

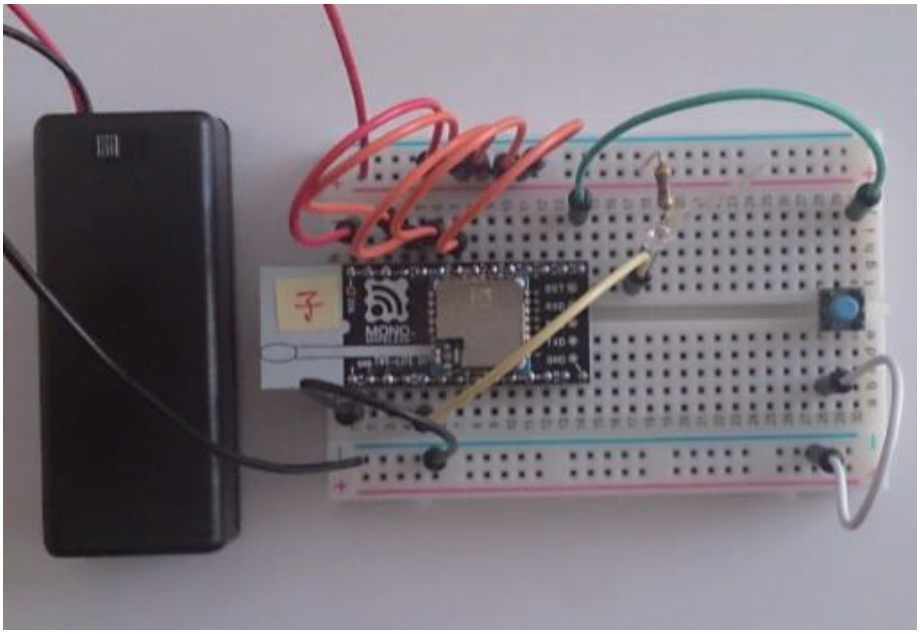


図　42

今回の実習では Android フォンを使用します。メーカーが開発したアプリで【**TWE Control**】というソフトウエアが公開されています。Google Store などで検索すると容易に見つかりますので、これをダウンロード、インストールしてください。(※この原稿執筆時にメーカーが公開しているアプリは Android 向けのみです。)

◇実験ではAndroidフォンを使用
◇アプリ【TWE Control】をインストール
※Google Storeなどで配布

図　43

次に、MoNoSTICK を準備します。次の写真のように、MoNoSTICK を USB ホスト変換アダプターで Android フォンに接続してください。写真

の左下にあるのは子機です。

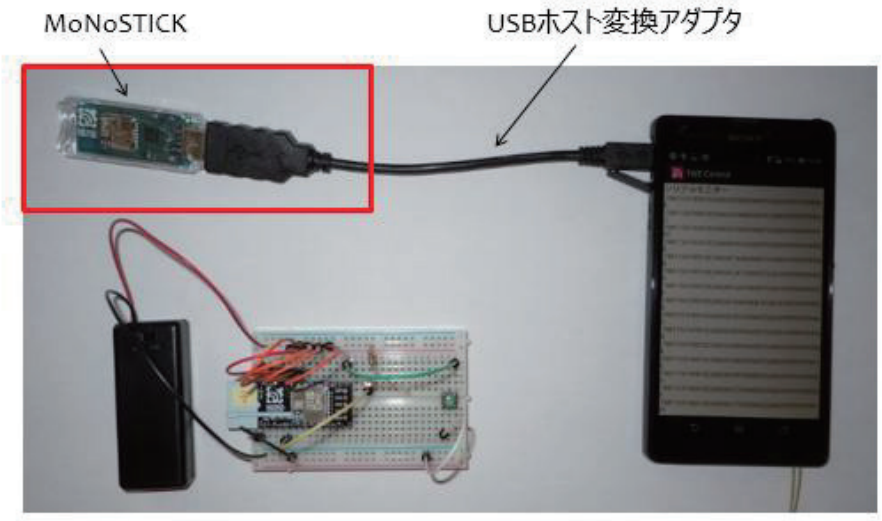


図　44

次に、子機の電池ボックスにある SW を ON にスライドします。子機の電源を入れて数秒後 Android フォンにインストールしたアプリ TWE Control を起動します。画面中央の【接続】をタップします。（下図）接続できると【未接続】から【接続中】に表示が変わります。

図　45

図　46

メニュー画面の下の部分のボタンが有効になりますので、【遠隔操作（リモートコントロール）】をタップします（上図）。すると、画面が図右のように変わります。この画面でいろいろな操作を行います。

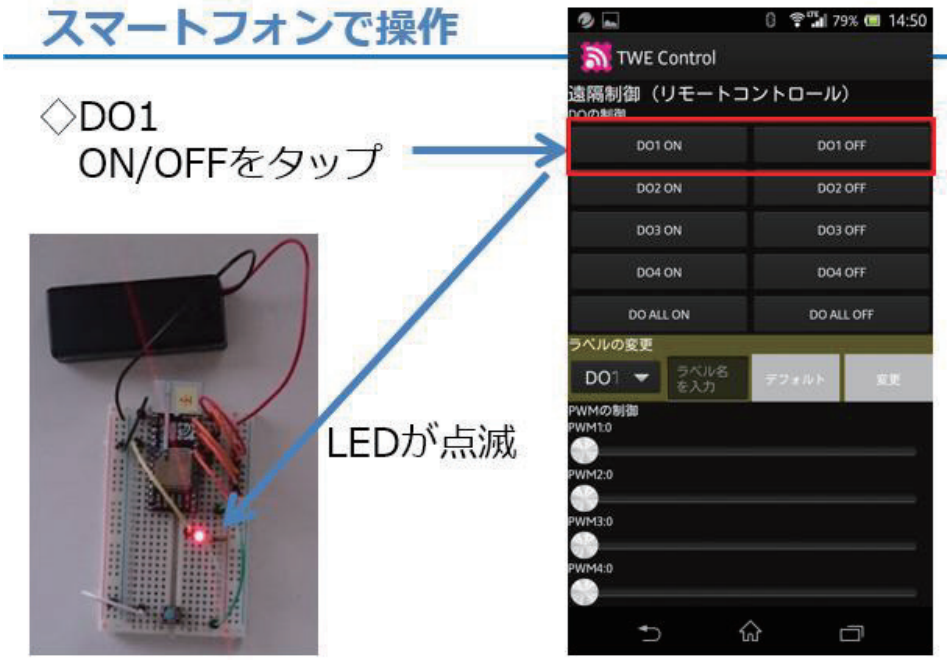


図　47

まず上図右、スマートフォン画面上にある **DO1 ON** をタップすると**子機の LED が点灯**します。次に **DO1 OFF** をタップすると **LED は消灯**します。いかがでしょうか。プログラミングレスでスマートフォンから子機を容易にコントロールできたと思います。画面のボタンを眺めると **DO2～DO4** というのがありますね。もうお分かりの通り、無線マイコンモジュールの**デジタル出力の 1～4（DO1～4）**のピンを使用すると、デジタル 4ch の無線コントロールが可能なわけです。

次にシリアルモニターという機能がありますので、それを使ってみます。

シリアルモニタの準備

図 48

上図、メニューで【シリアルモニター】をタップします。画面は右のように変わります。見ていると連続する 16 進数の文字列が順次表示されていきます。この 16 進数文字列の中に子機の情報が含まれています。

子機の SW を ON/OFF するとシリアルモニターの表示文字列に変化が起こります。

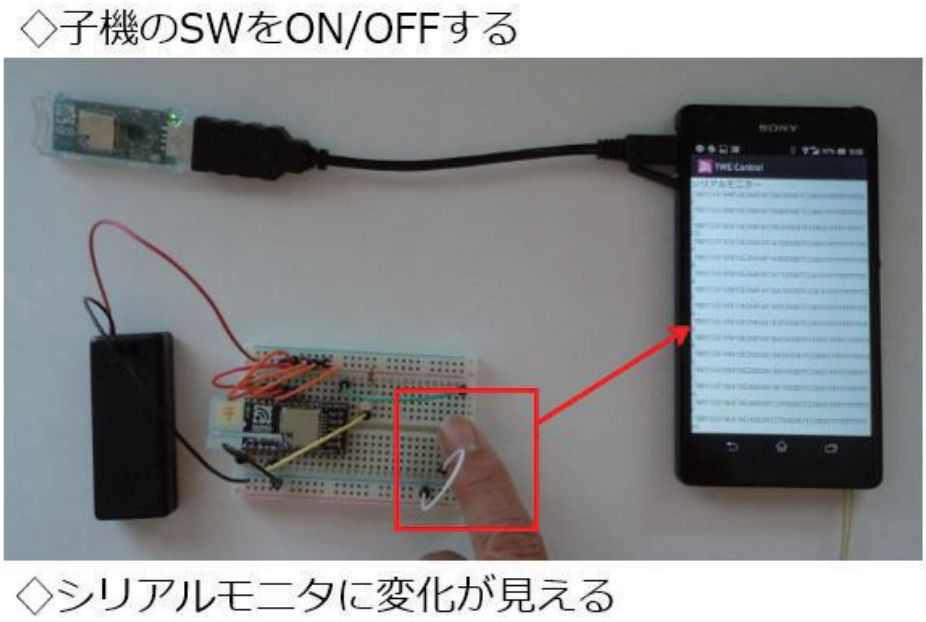


図 49

上の写真では文字列の変化が分かりませんので、その部分を拡大したのが下図です。

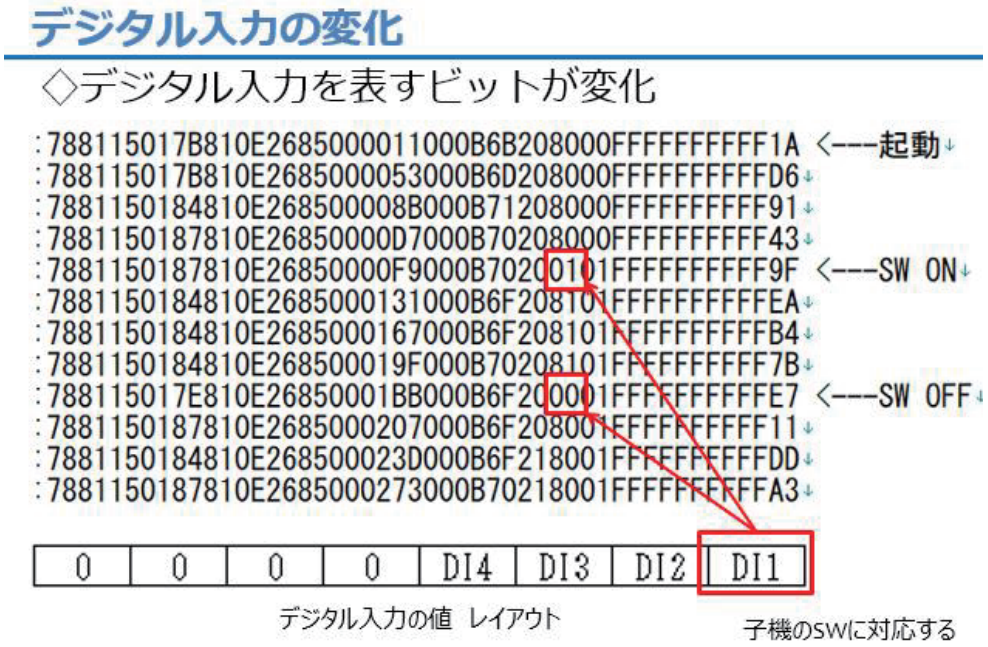


図 50

図のように【:78】で始まる文字列が出力されています。これが子機から親機に無線で送られたメッセージです。子機の SW を押した（ON した）とき、離した（OFF した）ときに変化している（赤枠）ことが分かります。この赤枠内の右側の桁は 8bit の下位 4bit であることが分かります。この下位 4bit で子機のデジタル入力 1～4 の変化を通知しています。より詳細にモニターして**子機の SW 状態に対応するコントロールを行いたいときは、この文字列を解析して対応する制御を行う**ソフトウエアを開発すればよいのです。デジタル入力の情報以外にも多くの文字が並んでいますが、これらは子機側からの情報通知に使われています。

メーカーの WEB 情報でこれらの内容は公開されていますので、参照してみると良いでしょう。

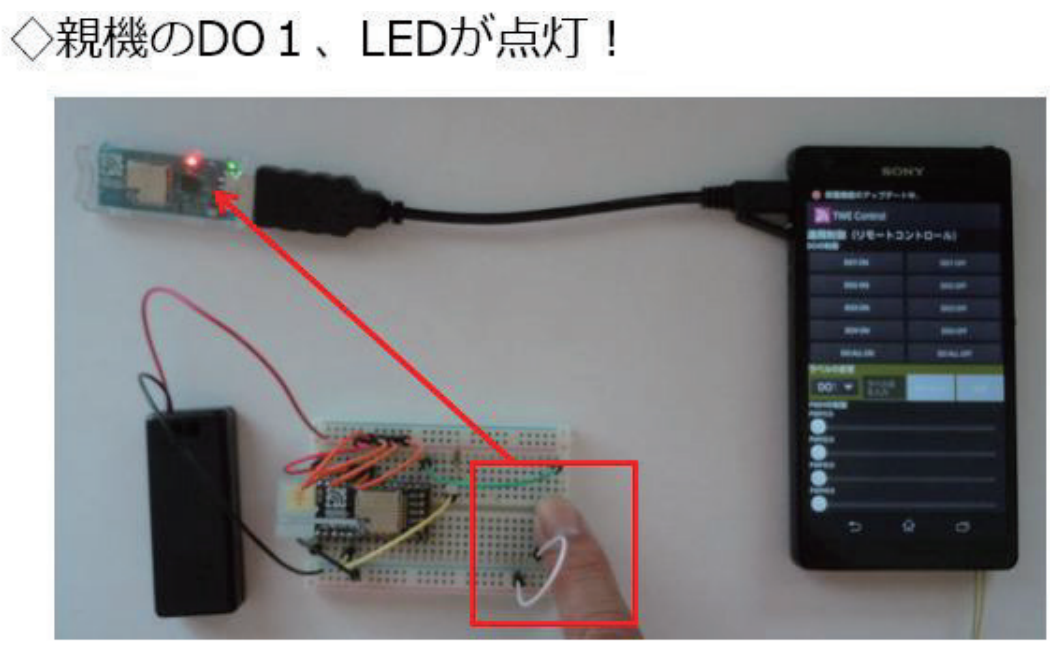


図 51

子機の SW を ON/OFF すると親機である MoNoSTICK の中にある LED が点滅していることに気付かれたでしょうか（上図）。スマートフ

ォンのボタンをタップすることにより子機の LED を制御しながら、子機の SW の状態に応じて親機 LED の制御も行われているということです。特別なことをしなくても、双方向で無線通信が常時行われていて、その機能を簡単な配線を行うだけで利用できるのは、IoT にもってこいのモジュールと言えるのではないでしょうか。

図 52

子機のリモートコントロール、親機のリモートコントロール（LED)、子機の様子をシリアルモニターすることなどをスマホアプリで行ってみましたが、このアプリには、他にも機能がありますので、紹介しておきます。

他にも便利な機能があります

図 53

上図左はこのアプリのメニューから【遠隔監視】をタップした時の画面です。電圧グラフというボタンがあります。これは、子機のアナログ入力に与えられた電圧の変化をグラフにして表示する機能で、上図中央のように電圧変化をグラフ表示してくれる機能です。また、右側には温度グラフというボタンがありますが、これは LM61BIZ という温度センサーをアナログ入力ピンに接続したとき、この機能を使うと、子機が置かれた場所のセンサーで計測した温度変化をグラフ表示してくれます。

第4回　PC 連携

前回行ったスマートフォンとの連携を PC で行ってみましょう。PC との連携は下図のようになります。

図　54

子機としての機能は第 3 回と変わる部分はありませんので、前回と同じ回路が使えます。システム全体の構成もスマートフォンが PC に置き換わっただけです。PC には USB コネクタが備わっているので、変換ケーブルなども必要ありません。

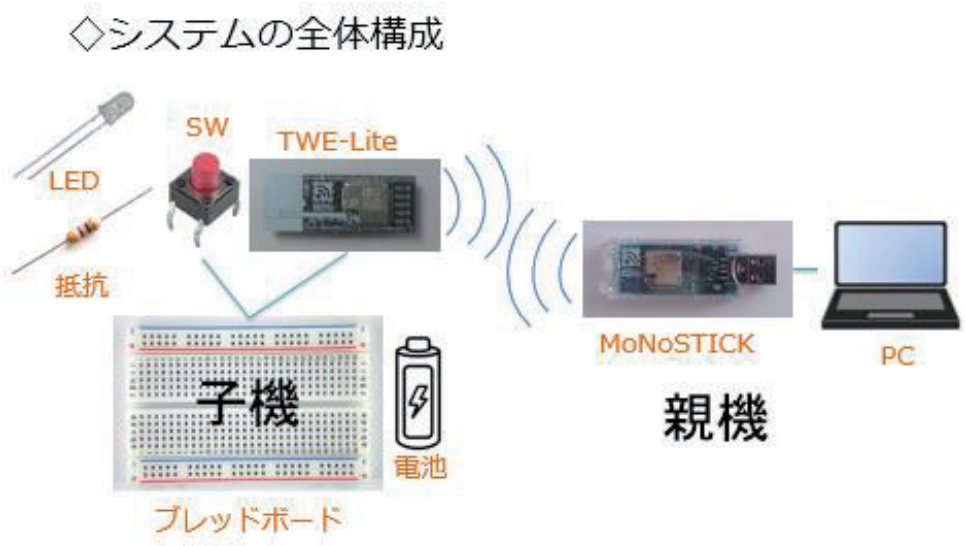


図　55

親機を通じて行う子機のモニターや操作は、PC の画面で行います。

使用するパーツなどは下記の通りです。

子機側：

1. 無線マイコンモジュール×1 台

2. ブレッドボード×1 個

3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット

4. 配線用ジャンパー線

5. SW×1 個

5. LED×1 個

6. 抵抗器（470Ω）×1 個（LED 電流制限用）

親機側：

1. PC×1 台（Windows10＋Internet 接続）

2. Windows 専用アプリ×1 セット

子機の配線は、前回（第 3 回）と同じものです。回路が保存されていればそのまま使えますが、配線やデバイスの緩みなどが無いかよく確認して下さい。新しく作られる方は第 3 回の配線図を参考にして作成してください。

◇アプリの準備

次に、メーカーWEB ページから、Windows 専用アプリをダウンロードします。

（https://mono-wireless.com/jp/products/TWE-APPS/index.html）

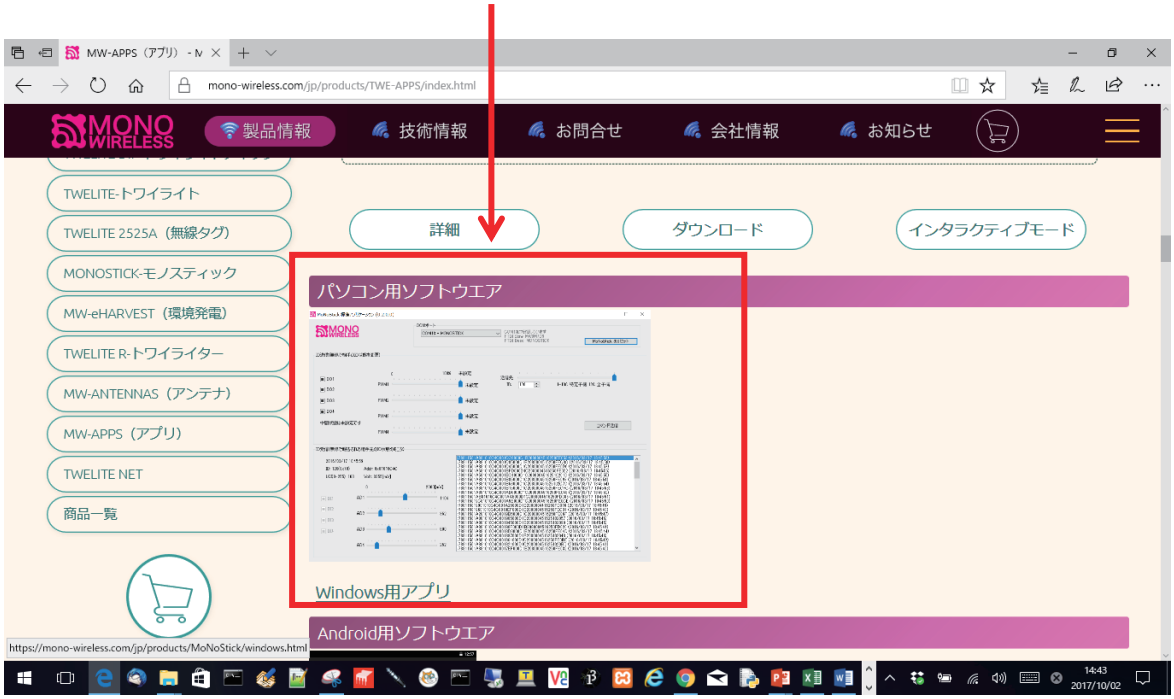


図　56

圧縮ファイルがダウンロードされますので、適当なフォルダに解凍します。その中に含まれる EXE ファイルが Windows アプリです。（下図）

図　57

インストールは必要ありません。この実行形式ファイル（exe）を開けばアプリケーションが起動するようになっています。

PC の USB コネクタに MoNoSTICK モジュールを差し込むと、**仮想 COM ポート**というシリアル通信のポートが設定されます。**デバイスドライバで確認**をすると COM ポート番号が確認できます。

◇アプリ起動　接続COMポート確認

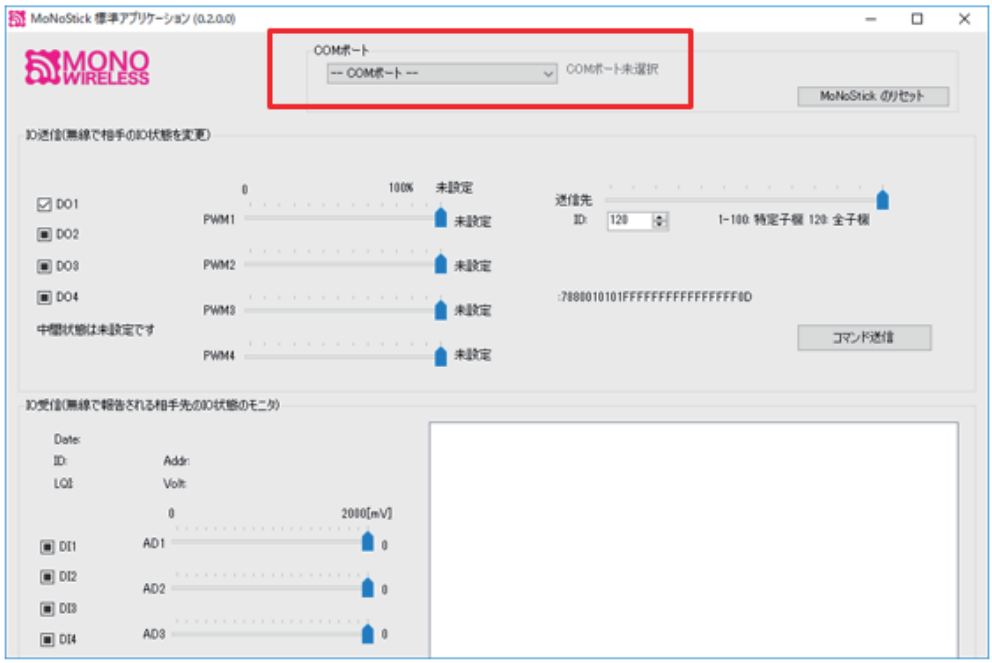


図　58

Windows アプリを起動して最初に行うことは、この COM ポートの確認と設定です。上の画面の上部にある【COM ポート】に仮想 COM ポート番号が設定されていなければ、プルダウンリストの中から、該当する COM ポート番号を選択します。

PCアプリで操作

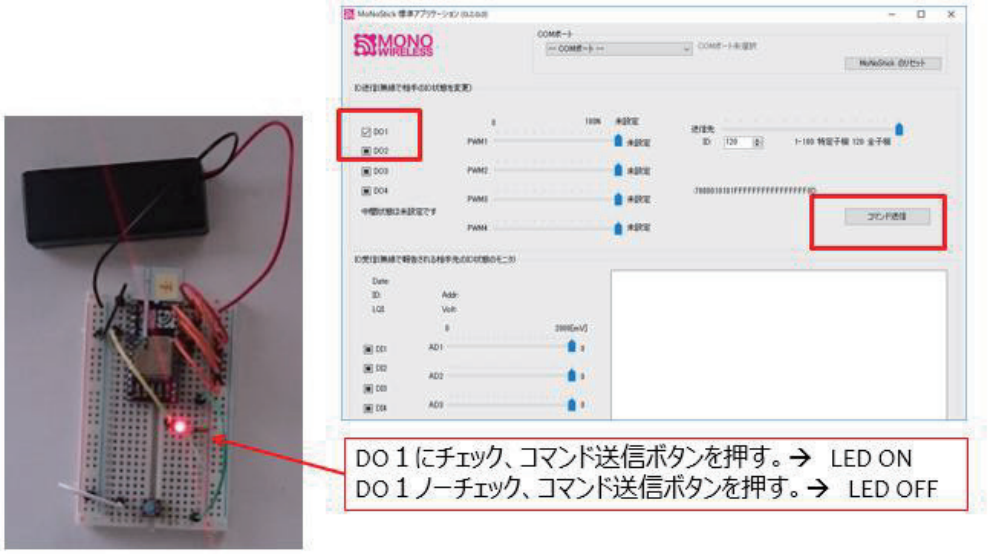


図 59

PC アプリで操作をしてみましょう。子機の電源 SW を ON にスライドしてください。そして、アプリケーションウインドウの左上にある【DO1】というチェックボックスにチェックを入れます。そして、右側中央部にある**【コマンド送信】**ボタンをクリックします。すると子機の LED が点灯します。次に DO1 のチェックをはずして**【コマンド送信】**ボタンをクリックすると、子機の LED が消灯します。この時に送信された電文（16 進数文字列）がコマンド送信ボタンの左上に表示されているので、電文中のどの部分が DO1 に対応するのかが分かります。無線マイコンモジュールの制御用通信ソフトを開発するときにはたいへん役立ちます。

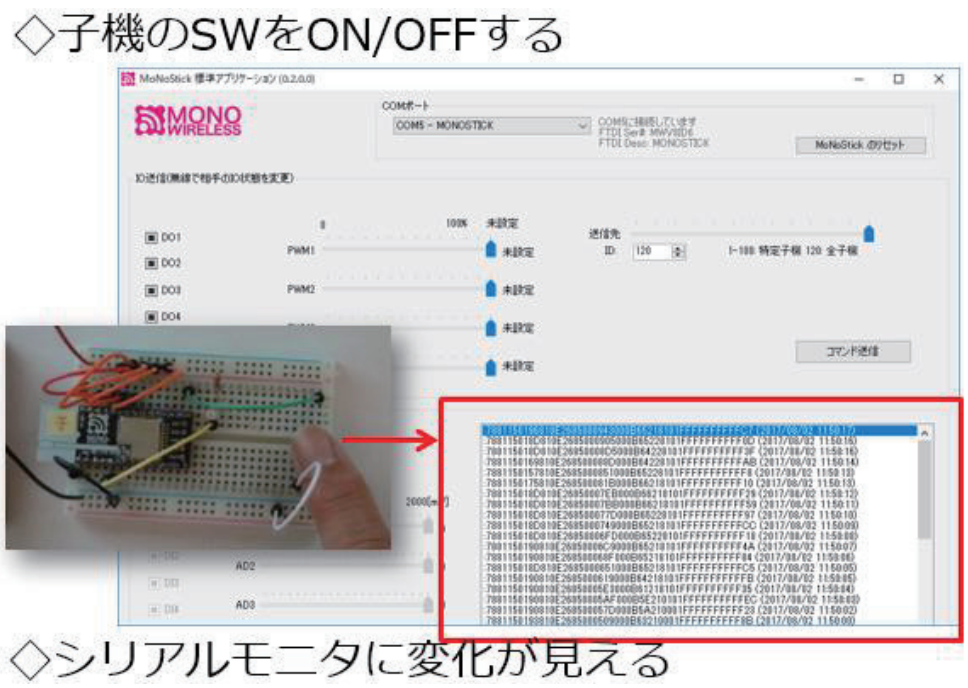


図　60

次に、子機側の SW を ON/OFF してみます。Windows アプリケーションのウインドウ右下にシリアルモニターという窓があり、そこに刻々と子機から受信した電文が表示されていきます。子機の SW の状態が変化（ON/OFF）するとこの電文に変化が現れます。この変化の様子を切り出したものが下図です。（再掲）これは第 3 回で詳しく説明したものですが、子機の SW が変化すると、その時に電文中のデジタル入力 1 のビットも対応して変化します。この変化する 16 進数の下の桁 4bit で子機のデジタル入力 1～4 を通知しています。なおこの部分は子機の SW 状態に変化があったときだけ通知されるようになっています。これは無線マイコンモジュールに書き込まれている標準アプリケーションの仕様です。

図 61

子機の SW の ON/OFF 状態でシリアルモニターの受信電文の内容が変化しますが、それと同時にウインドウ左下の DI1 というチェックボックスにも変化が現れます。子機の SW を ON すると DI1 にチェックが入り、SW を OFF するとチェックが消えることが確認できます。

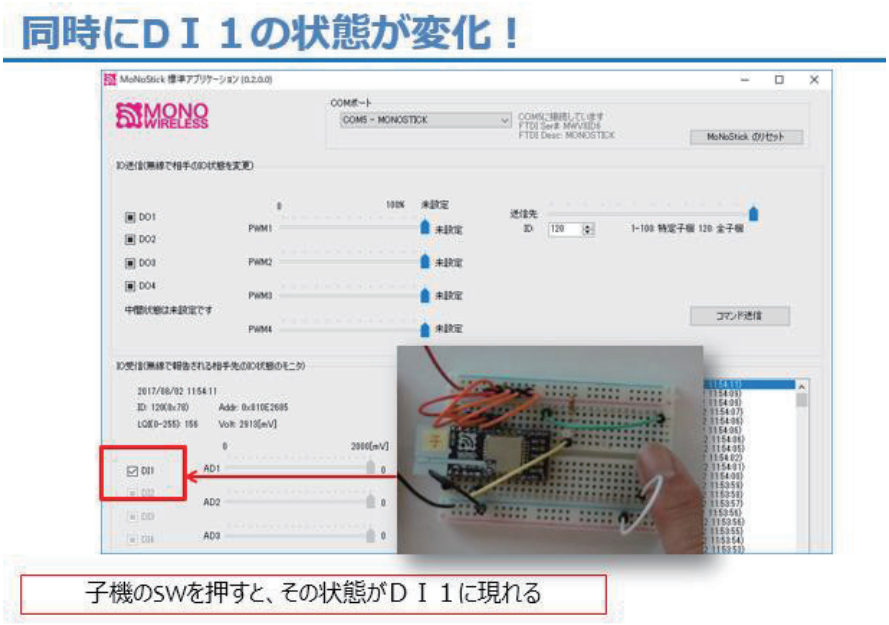


図 62

現在、アプリケーションは Windows 用のみが公開されているのですが、これらアプリは VS（Visual Studio）・VB（Visual Basic）で開発されていて、そのプロジェクトソースコードも提供されています。ここで実習に使用したアプリケーション以外にも、温度センサーの情報を表示する簡易ビュアーなどがソースコードとともに公開されています。TWE-Lite に使用するのであれば、商利用も可能ということですので、アプリケーション開発の良い教材となります。

図 63

第5回　PWM

PWM 制御という言葉をお聞きになったことがありますか。これは、Pulse Width Modulation の頭文字をとった言葉で、パルス幅変調制御というものです。下図を見てください。

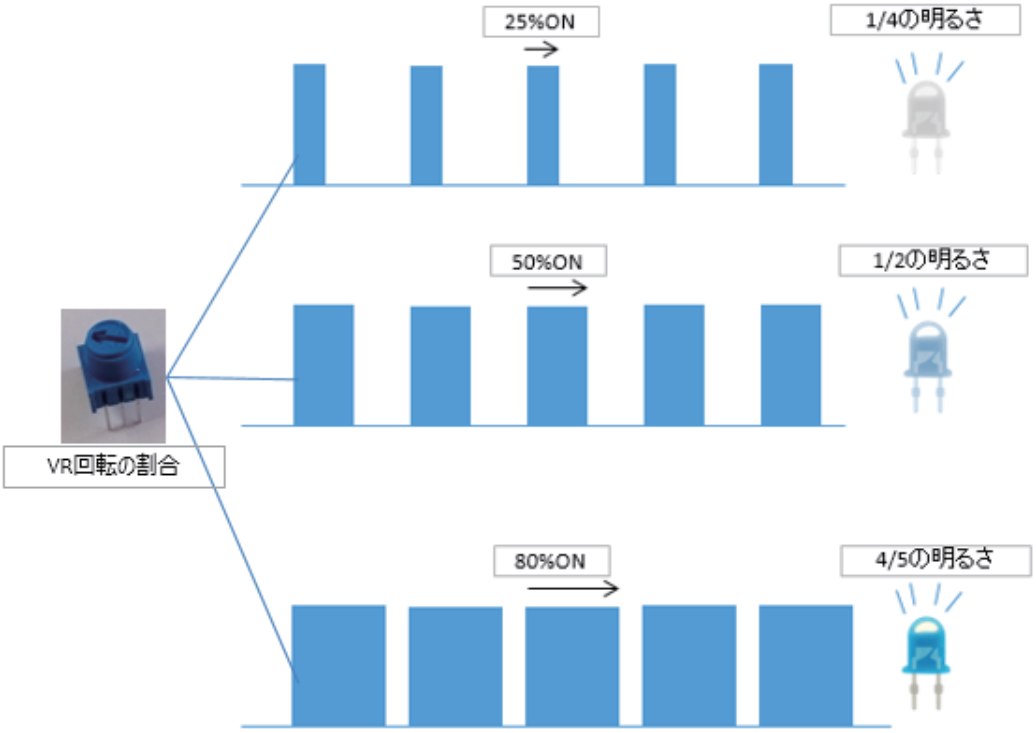


図　64

LED を点滅する信号があります。OFF では LED の明るさは 0%で、ON した時 100%の明るさだとします。この ON/OFF 信号を一定周期で繰返すパルスとして発生させます。そして、ON の割合が例えば周期の 25%の時は LED が 1/4 の明るさで光ります。次にこの ON の幅を 50%にしてあげると LED は半分の明るさで光ります・・・と云うように、ON の幅（パルス幅）をコントロールすることで、ある対象の動作を制御する仕組みが PWM 制御（パルス幅変調制御）と言われるものです。ここ

では例として LED を取り上げましたが、モータの回転数や回転角、ヒータの温度などを対象に、身近なところで利用されている制御方式です。今回は、無線マイコンモジュールの子機に VR（ボリューム：可変抵抗器）を取り付けて、VR の抵抗に対応する PWM 制御信号で、親機の LED の明るさを変化させてみましょう。今回も配線するだけでプログラミングレスです。

図 65

上図左側が子機です。VR の両端に電圧を加えて VR を回転させると、回転の割合に応じた電圧が VR から取り出せます。この可変電圧を無線マイコンモジュールのアナログ入力ピンに接続すると、電圧が変化したとき、子機は無線通信でその状態を右側の親機に伝えます。親機はこの無線で通知されたアナログ入力ピンに加えられた電圧に対応する PWM 信号を PWM 出力ピンに出力しますので、ここに LED を接続しておくと、子機の VR の回転の割合に対応して親機の LED の明るさが変化するという具合です。

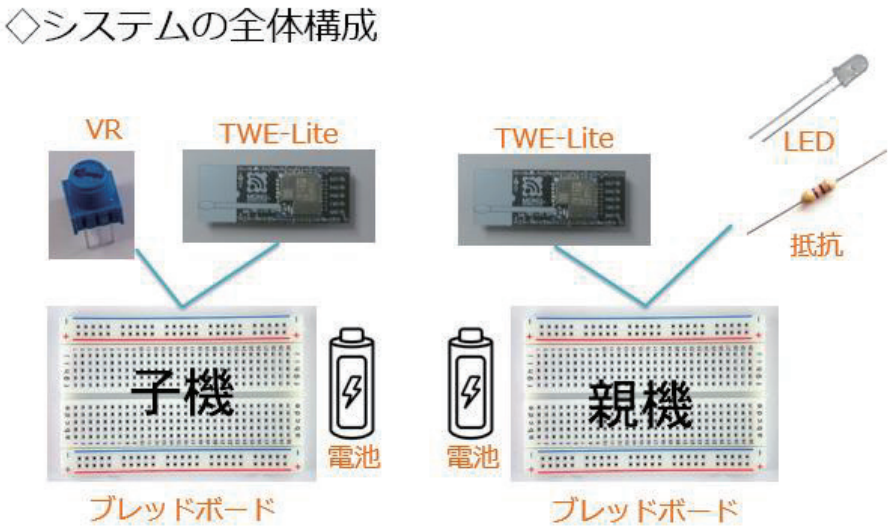


図　66

上図がシステムの全体構成です。親機には明るさを変化させる対象として LED を使います。今回初めて VR（ボリューム：可変抵抗器）を使いますが、その構造・役割を見ておきましょう（下図）。

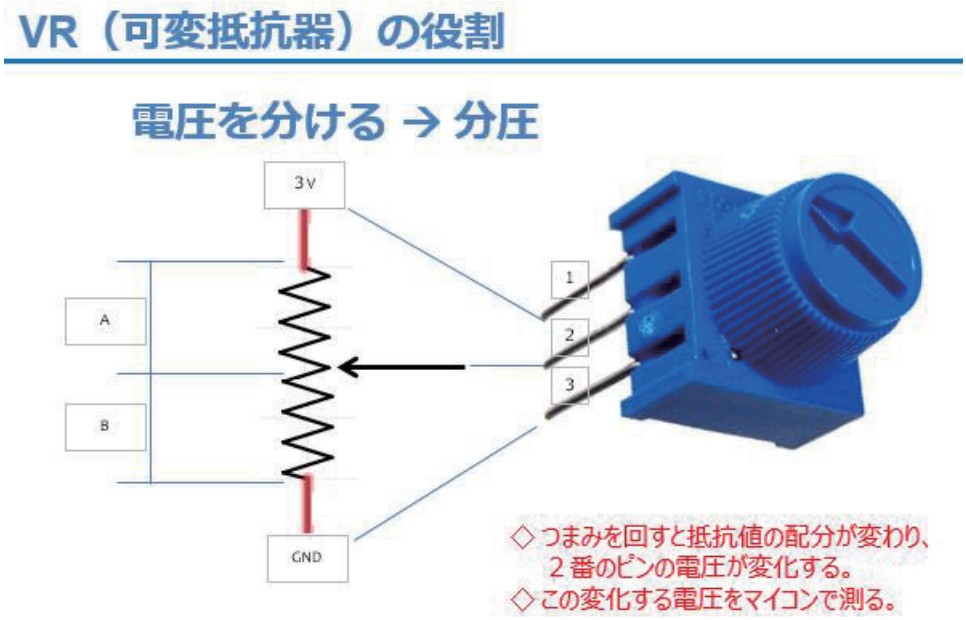


図　67

VR には 3 本のピンが出ています。このピンに図のように上から 1~3 の番号を付けます。1 番ピンに 3V、3 番ピンに GND を接続すると、2

番ピンからは、VR の回転の割合に応じて内部の抵抗が A と B の部分に分割されて、それに対応する電圧が取り出せます。VR のつまみを回すと、図の矢印の部分が上下に移動します。一番上に移動すると 2 番ピンは 3V になります。反対に一番下に移動すると 2 番ピンは 0V になります。このように内部の抵抗値の配分で両端にかかっている電圧を配分するので、このことを抵抗値による分圧と云います。この 2 番ピンをマイコンのアナログ入力ピンに接続して電圧を測ります。その値が PWM 制御の元になるパルス幅へと変換されるわけです。この電圧計測からパルス幅への変換は、この無線マイコンモジュールに書き込まれている標準アプリケーションが行ってくれますので、プログラミングは必要ありません。

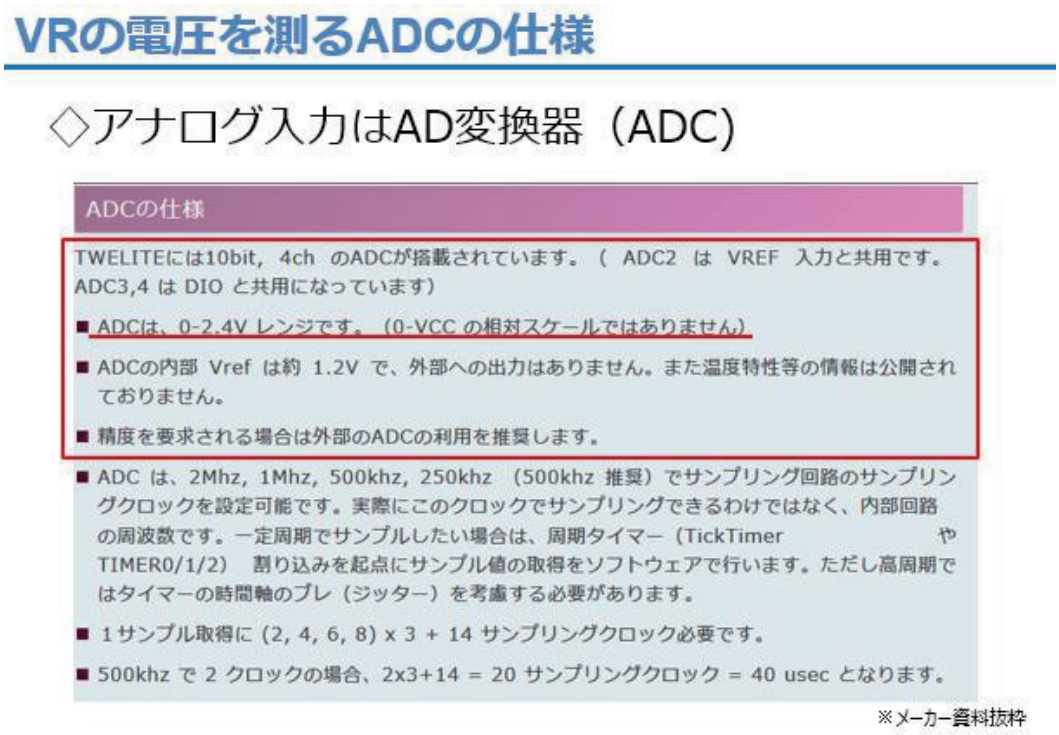


図 68

上の資料は、メーカーが公開しているアナログ入力ピンにつながっている A/D 変換器（ADC）の仕様です。これを見ると、【ADC は 0-2.4V】との記述があります。先の VR でつまみを回して矢印が一番上に移動する

と２番ピンには 3V が出力されて、アナログ入力ピンに 3V が加えられてしまいます。これは、上の仕様を超えてしまうので、少し工夫をしなくてはいけません。

VRの部分を少し工夫

◇VRに加える電圧を2.4V以下にします。
◇50kΩのVRに15kΩの抵抗を加えると、
VRの最大出力電圧は、

$$\frac{50}{50+15} \times 3 \approx 2.3 \text{〔V〕}$$

となります。
◇15kΩの抵抗をVRの電源側に接続します。

図 69

15kΩの抵抗を VR の電源（3V＋）側に接続すれば、上の計算のように VR を回し切ってしまって最大の電圧が出力されても 2.3V となって ADC の仕様の範囲内に収まりますので、VR は下図のようにします。

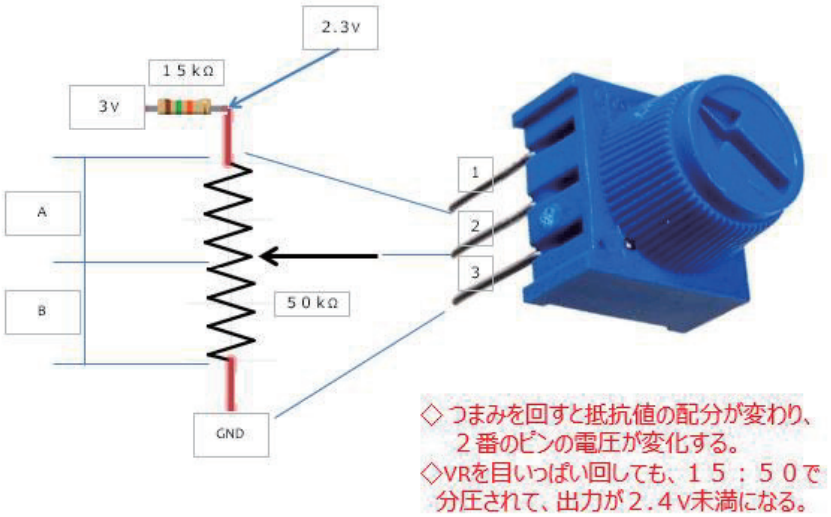


図 70

使用するパーツは次のようになります。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. VR（50kΩ）×1 個
6. 抵抗器（15kΩ）×1 個

親機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. LED×1 個
6. 抵抗器（470Ω）×1 個

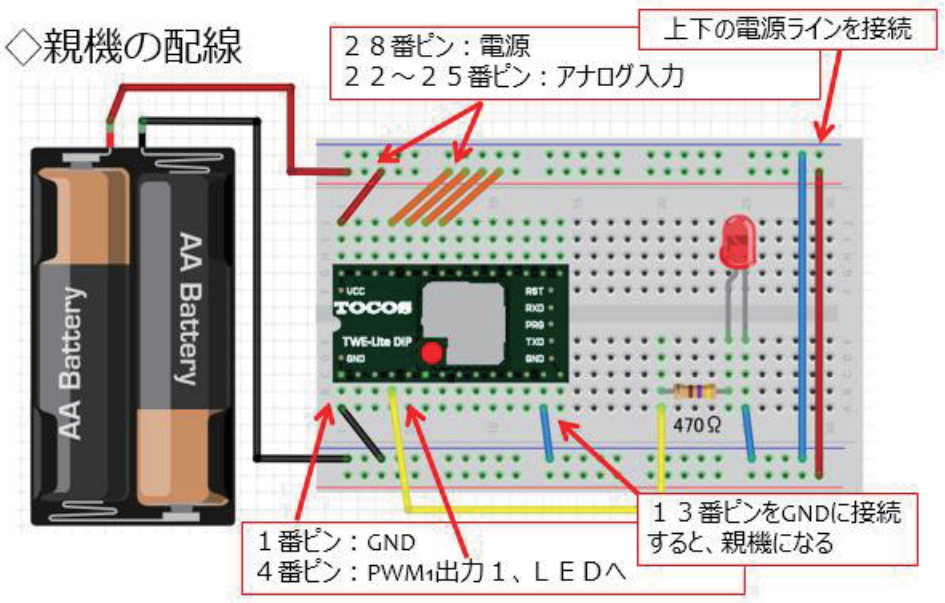


図 71

では親機の配線を上図のように行ってください。親機には LED と抵抗を使用しますが、これまでの講座で使用したピンとは違う場所に LED を接続していることに注意してください。これまではデジタル出力 1（DO1）に LED を接続しましたが、**今回は 4 番ピンの PWM1 の信号に LED をつなぎます**。LED の極性（長い方の脚の向き）にも注意をしてください。反対向きに接続すると電流が流れず LED は光りません。親機ですから、13 番ピンを GND 接続するのも忘れないようにしてください。今回からブレッドボードの右端に上下の電源ラインンを接続する赤・青のジャンパー線が加わっています。今回の実習では実際には使用されていませんが、このようにするとブレッドボードの上下どちらからでも電源（＋）と GND(－)が使えます。

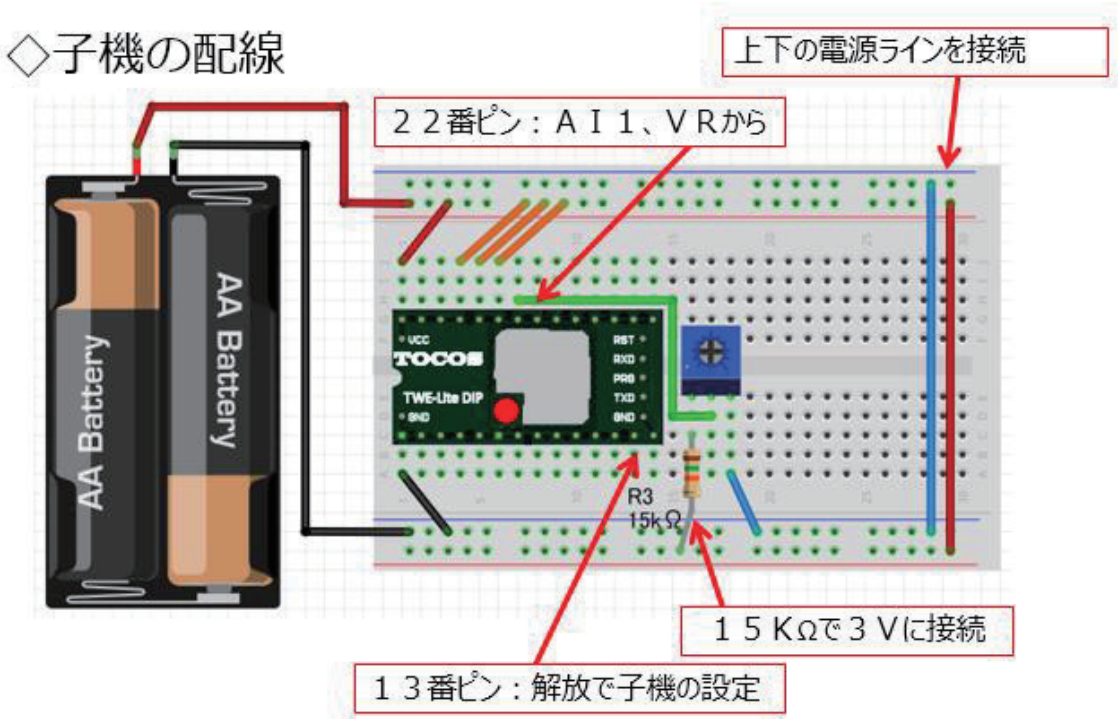


図 72

次は子機を配線しましょう（上図）。子機は既に説明した VR と抵抗器の配線がありますね。VR の 2 番ピンはマイコンの AI1 の信号である 22

番ピンに接続します。子機も右側に上下の電源ラインを接続するジャンパー線を追加してあります。

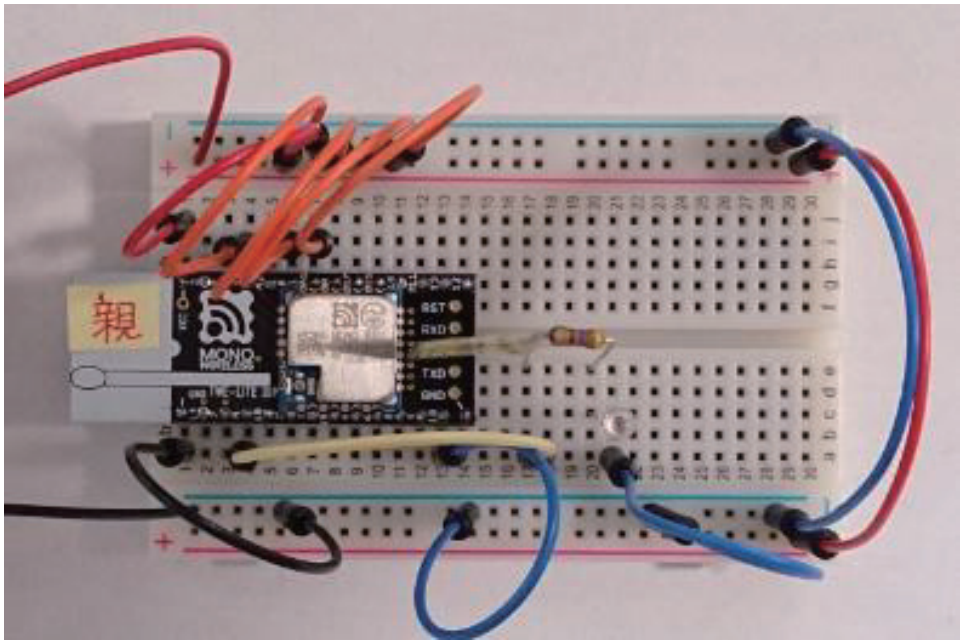


図 73

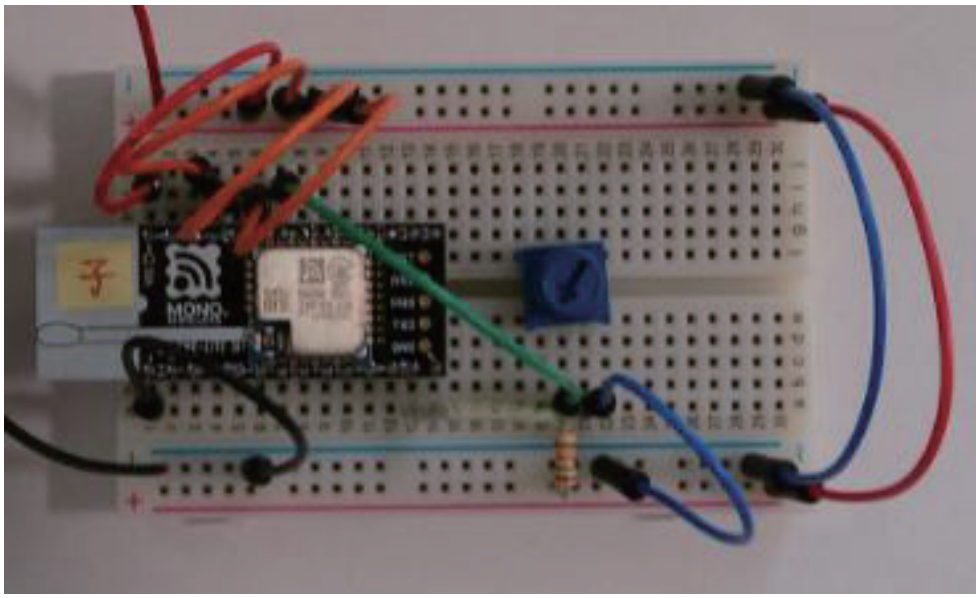


図 74

実際に配線したものが上の写真です。

もう配線にも慣れてきたと思いますが、油断大敵です。注意深く確認

をしてください。子機の AI1 ピンはこれまで電源（＋）に配線されていましたが、今回はそれが VR からの配線になっています。

◇動作確認

さて、親機・子機の電源 SW を ON にスライドしてください。そして、子機の VR のつまみを少しずつ回してみましょう。親機の LED の明るさが変化しているのが確認できると思います。

次の写真は、パルスの幅をオシロスコープで観測して、その時の LED の明るさを撮影したものです。

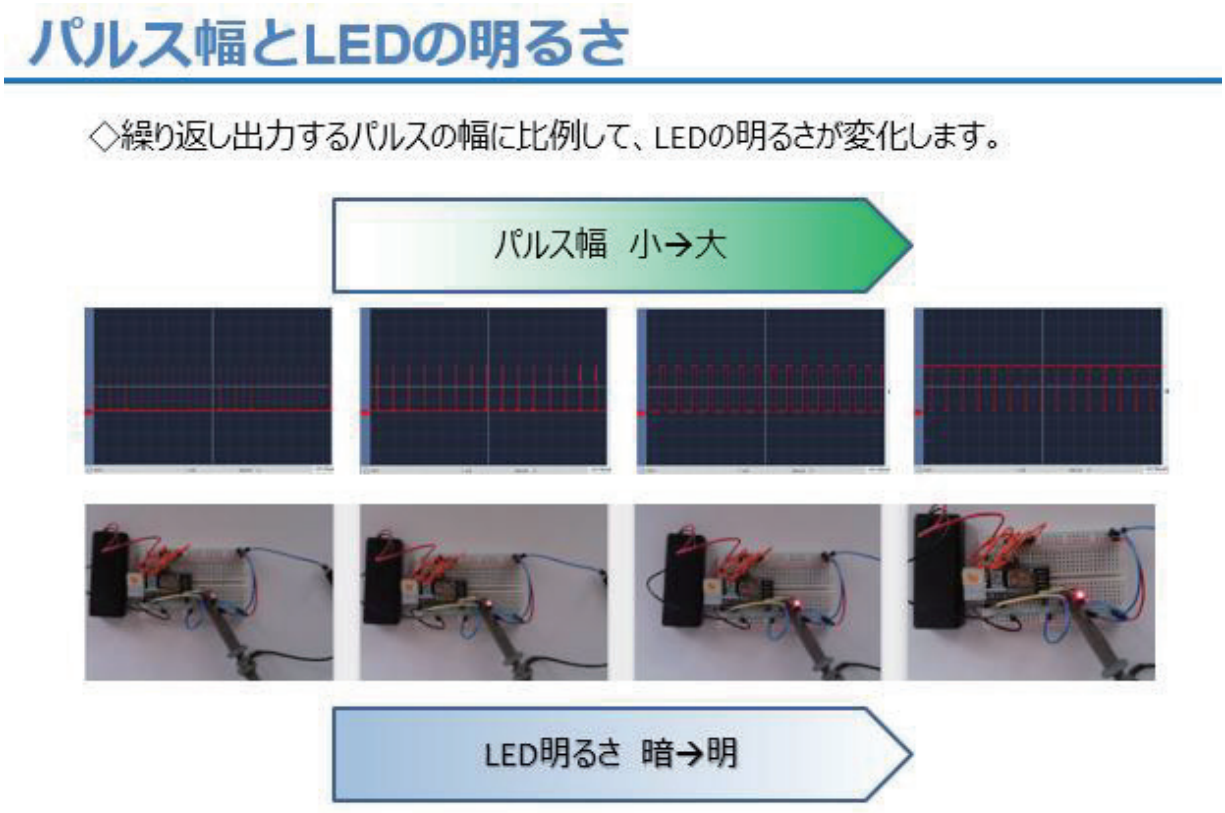


図 75

VR のつまみを回転することに対応して、パルス幅が変化していることが分かります。また、それに応じて LED の明るさも変化しています。これが PWM 制御と呼ばれている制御方式です。

第6回　双方向 PWM

第 5 回で行った PWM 制御を双方向で行います。

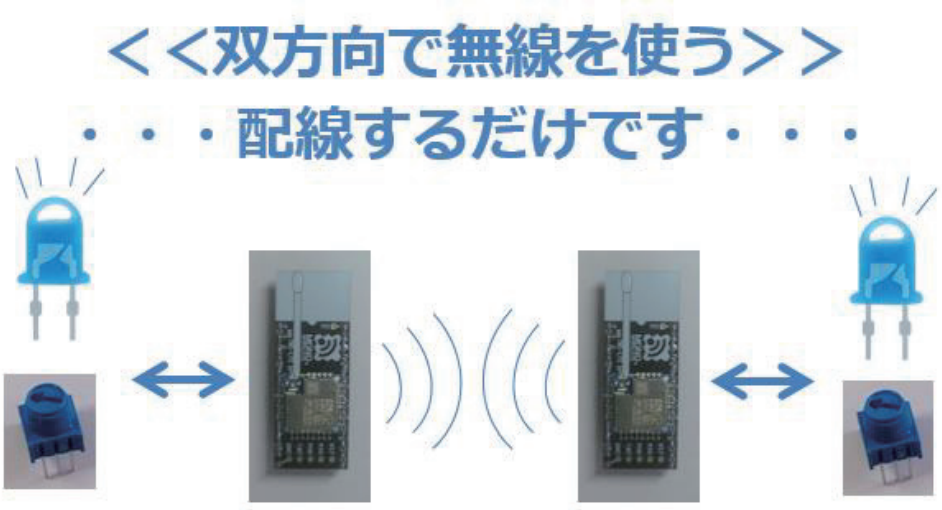


図　76

第 5 回では、VR のつまみ位置で設定する電圧を子機側マイコンモジュールで読み込んで、それを親機に無線送信して、親機側の PWM 制御のパルス幅に反映させて LED の明るさを制御しました。今回は同時に、親機でも VR で設定される電圧を読み込み、それを子機に無線送信して、子機の PWM 制御パルス幅に変えて LED の明るさ制御を行います。双方向同時 PWM 制御です。

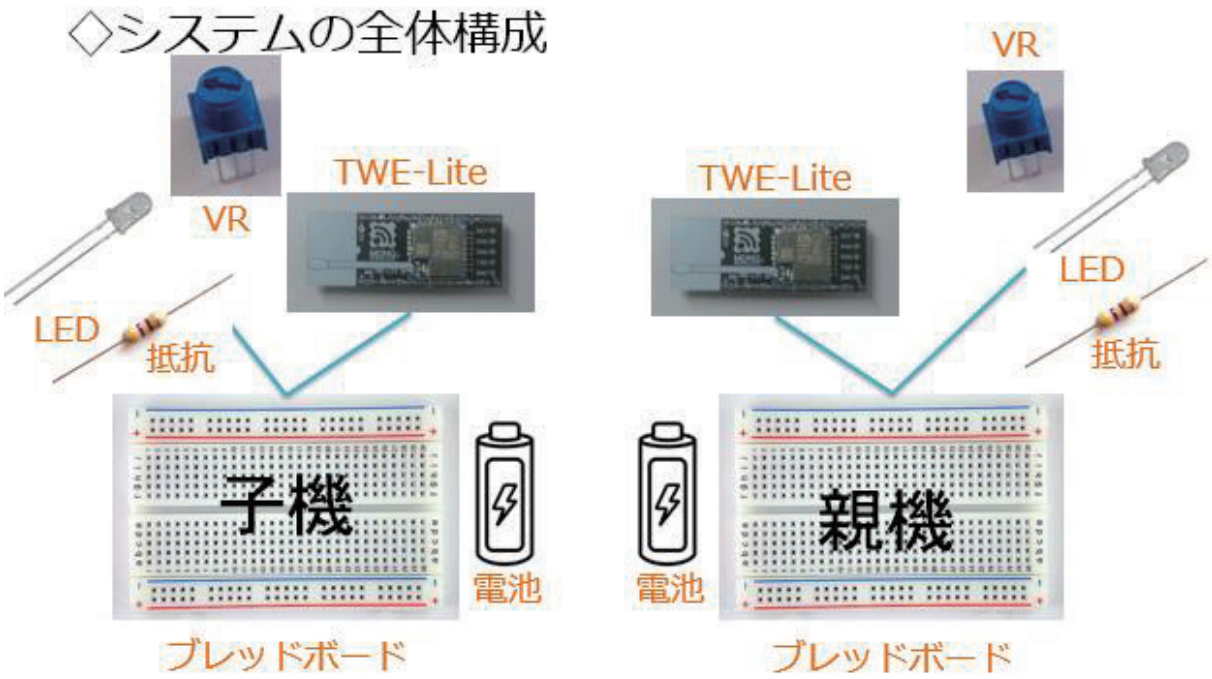


図　77

親機・子機ともに全く機器構成です。使用するパーツは各々2組です。

子機側＋親機側：

1. 無線マイコンモジュール×2台
2. ブレッドボード×2個
3. 電池（単4乾電池×2個＋SW付電池ケース×1個）×2セット
4. 配線用ジャンパー線
5. VR（50kΩ）×2個
6. LED×2個
7. 抵抗器（470Ω）×2個（LED電流制限用）
8. 抵抗器（15Ω）×2個

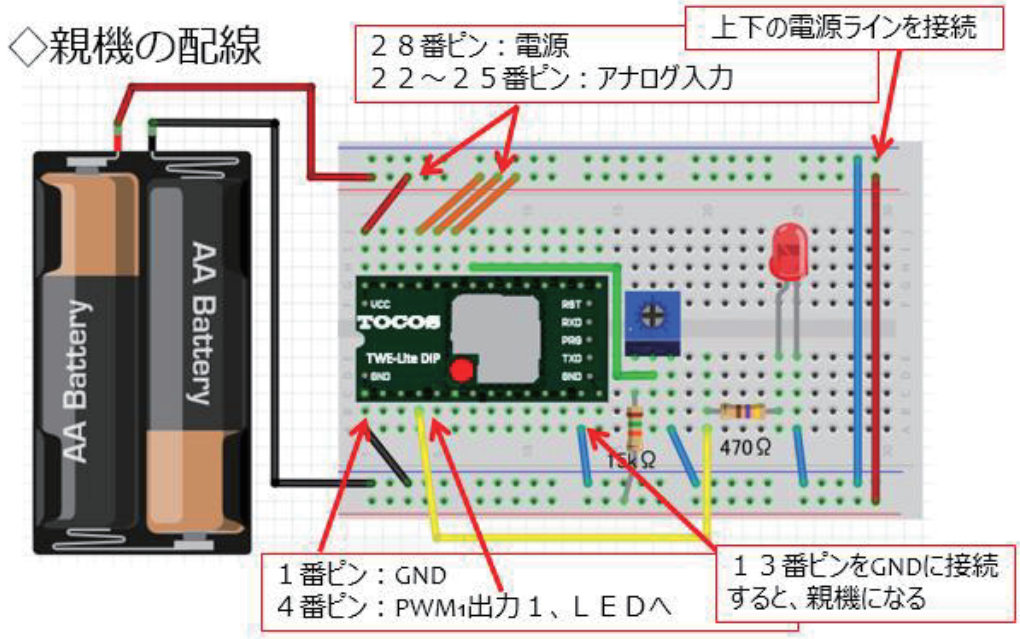


図　78

上図に従って親機を配線します。子機も同様に下図に従って配線します。

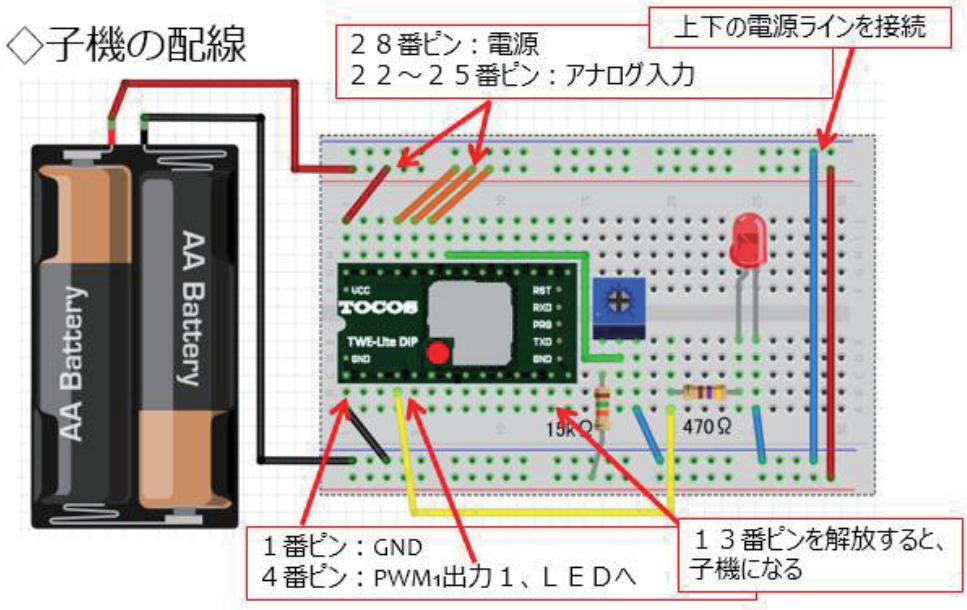


図　79

親機の 13 番ピンを GND に接続するジャンパー線を配線する以外は、親機・子機共に同じ配線ですので、確認しやすいですね。

◇実際に配線した様子　【親機】

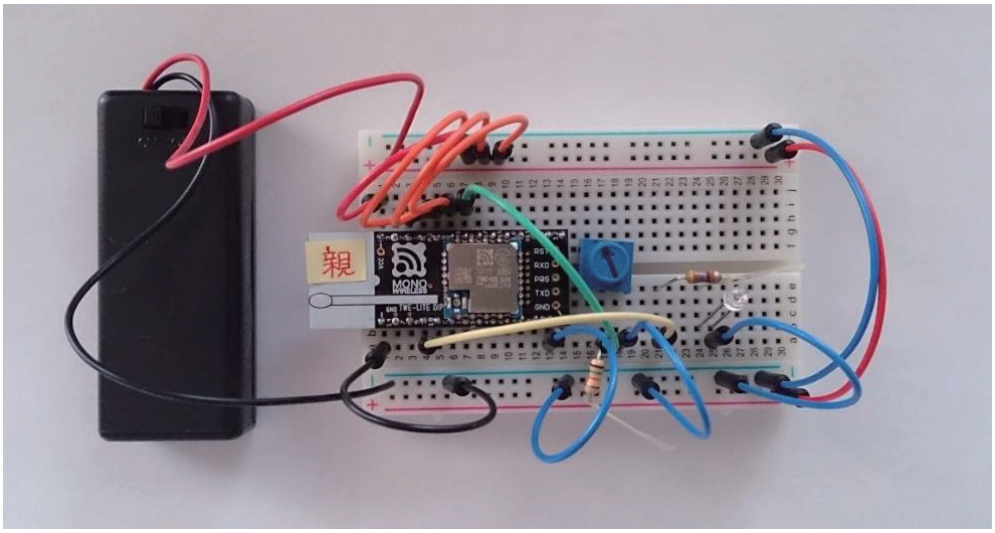


図 80

◇実際に配線した様子　【子機】

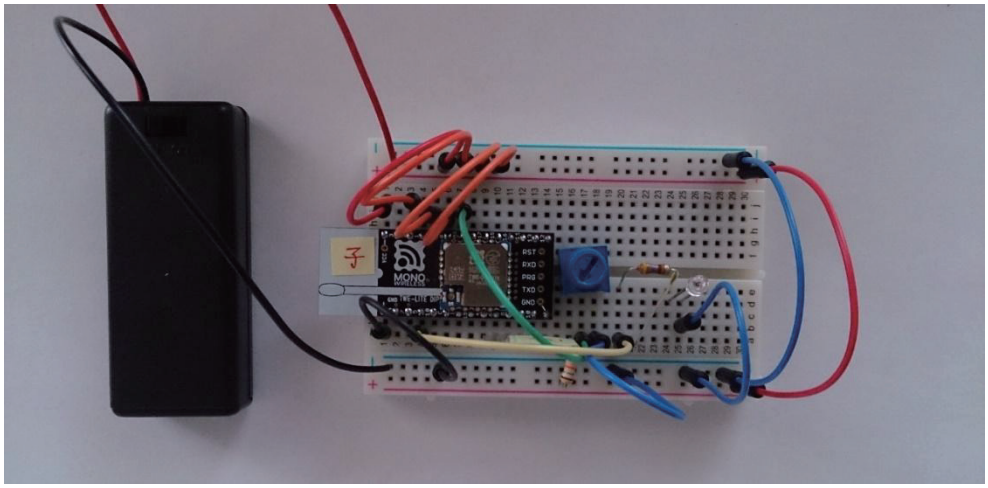


図 81

違いは、13 番ピンのモード設定信号のジャンパー線だけです。

◇動作確認

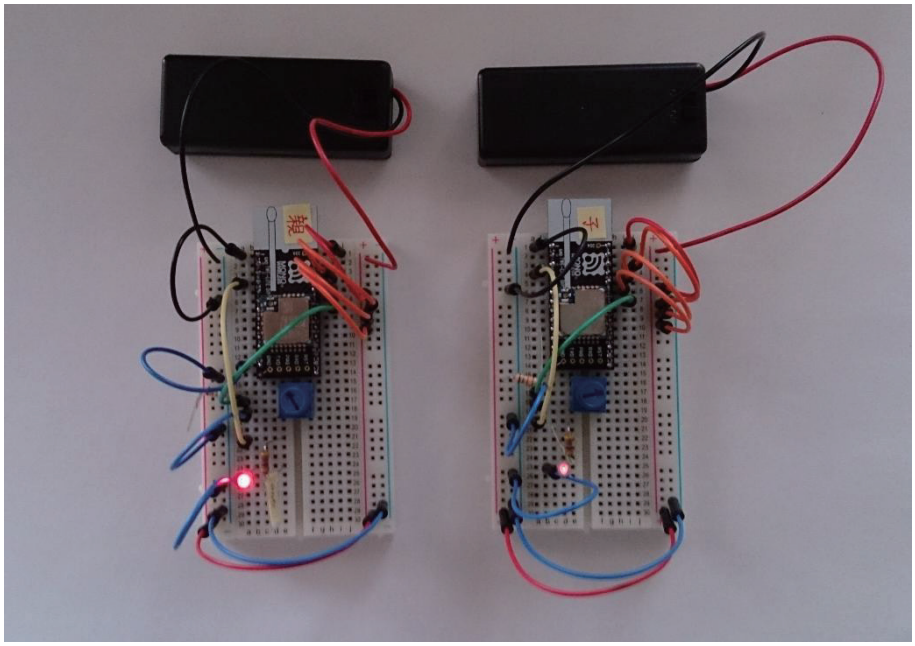


図 82

親機・子機共に電源 SW を ON にしてください。親機の VR を回転すると、その状況が子機側 LED に反映されますか。同じように子機の VR の回転位置が親機の LED の明るさとなって現れますか。

双方向 PWM 制御は、とても簡単に実現できました。

【重要】今回作成した子機は、次の回でもそのまま使用しますので、保存しておいてください。

第7回　スマートフォン連携 PWM

第 5 回、第 6 回で行った PWM 制御では、VR によって無線通信の相手側の LED の明るさをコントロールしたのですが、今回はその VR の代わりにスマートフォンを使いましょう。

図　83

上図の様に第 3 回:スマートフォン連携の考え方をそのまま使います。但し今回はスマートフォンから子機への送信のみです。スマートフォンの画面に配置されているスライダーを VR と見立てて子機の LED を PWM 制御します。

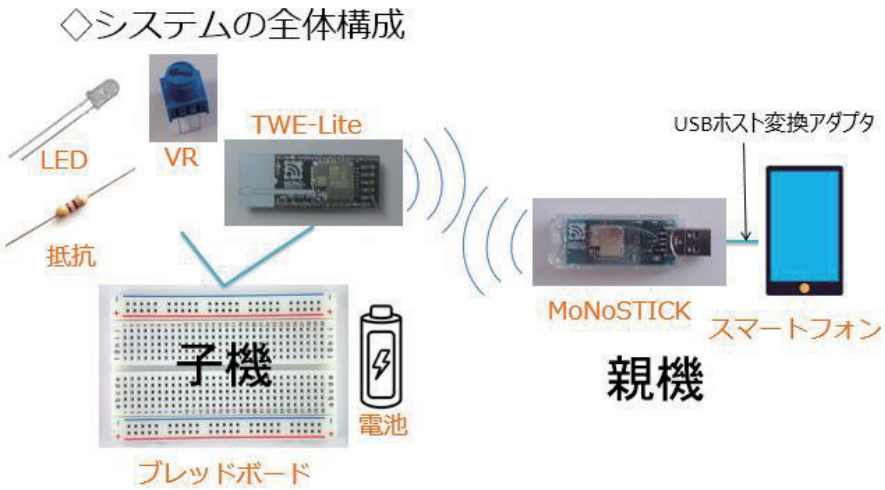


図　84

第 6 回の子機をそのまま使用しますが、子機に配置されている VR は、使いません。

使用するパーツは下記です。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. VR（50kΩ）×1 個（実験には使っていません）
6. LED×1 個
7. 抵抗器（470Ω）×1 個（LED 電流制限用）
8. 抵抗器（15kΩ）×1 個（実験には使っていません。）

親機側：

1. MoNoSTICK×1 台
2. USB ホスト変換アダプター×1 本（コネクタ形状により選択）
3. スマホアプリ TWE Control×1 セット

前回の子機をそのまま使いますので、保管してある方は配線の緩みなどが無いことを良く確認しておいてください。新たに作る方は第 6 回の配線図を参考にして配線を行ってください。第 6 回では、VR と付随する抵抗（15kΩ）は配線されていますが、それは今回未使用です。

◇MoNoSTICK モジュールの接続と子機、アプリの準備

USB ホスト変換アダプターを使い、下の写真のように、MoNoSTICK モジュールをスマートフォンに接続します。

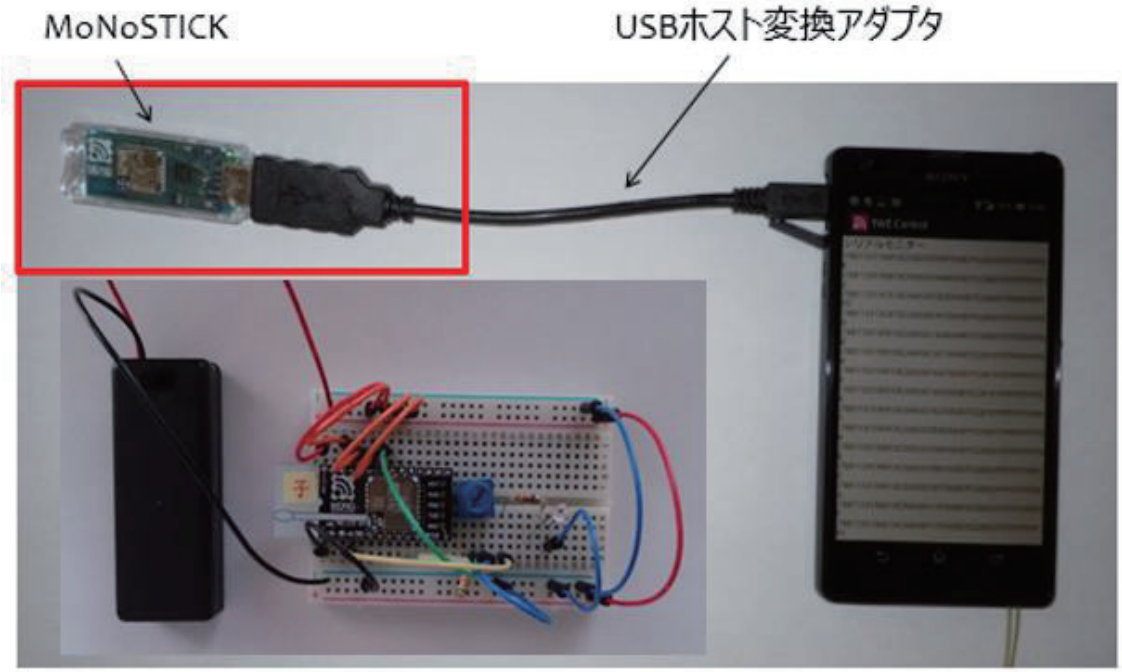


図 85

MoNoSTICK モジュールが接続されるとモジュール内の電源 LED が点灯します。次に、子機の電池ボックスにある SW を ON にスライドします。**第 3 回でインストールしたスマートフォンアプリ【TWE Control】**を今回も使用します。まだインストールされていない方は第 3 回を参照してください。

◇スマートフォンアプリの起動

子機の電源を入れて数秒後 Android フォンにインストールしたアプリ【TWE Control】を起動します。画面中央の【接続】をタップします。（下図）接続できると【未接続】から【接続中】に表示が変わります。

図　86

◇アプリ起動　接続、遠隔操作を選択

図　87

メニュー画面の下の部分のボタンが有効になりますので、【遠隔操作（リモートコントロール）】をタップします。すると、画面が上図右のように変わります。この画面で操作をおこないます。

◇動作確認

図　88

上図で示すように、スマートフォン画面の PWM1 のスライダーをスライドしてみましょう。**スライダー位置に対応して子機の LED の明るさが変化する**でしょう。この PWM 出力信号を取り出して、様々な機器を制御することが容易にできます。皆さんなら、どのような使い方を考えますか。

第8回　PC 連携 PWM

第 7 回で行ったスマートフォンに代わって、PC で PWM 制御の操作を行ってみましょう。少しの時間があれば試すことができます。

図　89

前回のスマートフォンの代わりに PC を使用します。PC へのアプリケーションは既に第 4 回でインストール済みですが、まだ準備が済んでいない方は第 4 回を参照していただき、下記メーカーWEB ページからダウンロードしてインストールしておきます。

（https://mono-wireless.com/jp/products/TWE-APPS/index.html）

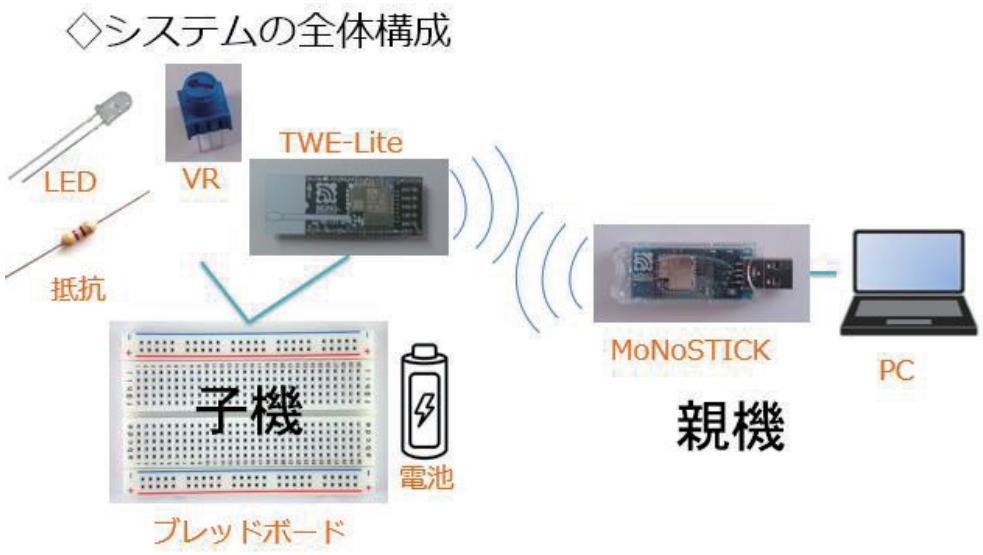


図 90

上図はシステムの全体構成です。第 7 回とほぼ同じです。MoNoSTICK モジュールは USB コネクタでそのまま PC に接続できます。使用するパーツなどは下記のとおりです。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. VR（50kΩ）×1 個
6. LED×1 個
7. 抵抗器（470Ω）×1 個（LED 電流制限用）
8. 抵抗器（15kΩ）×1 個

親機側：

1. MoNoSTICK×1 台
2. PC×1 台（Windows10）
3. Windows 専用アプリ×1 セット

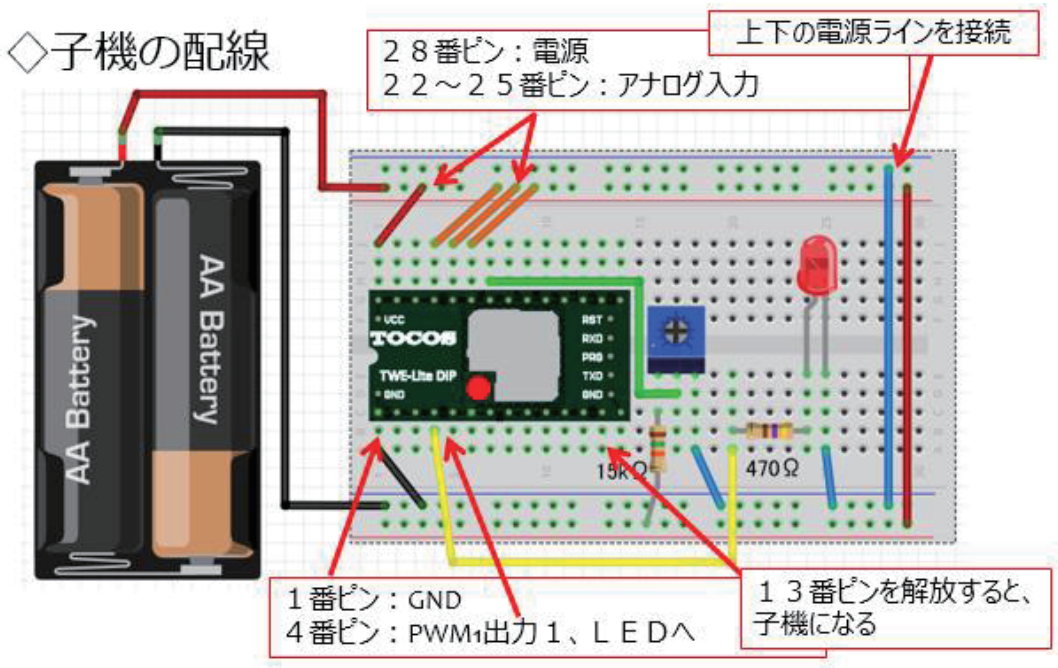


図　91

第７回の子機をそのまま使いますので、保管してある方は配線の緩みなどが無いことを良く確認しておいてください。実際の子機は下の写真のようになっています。確認したら子機の電源を入れておいてください。

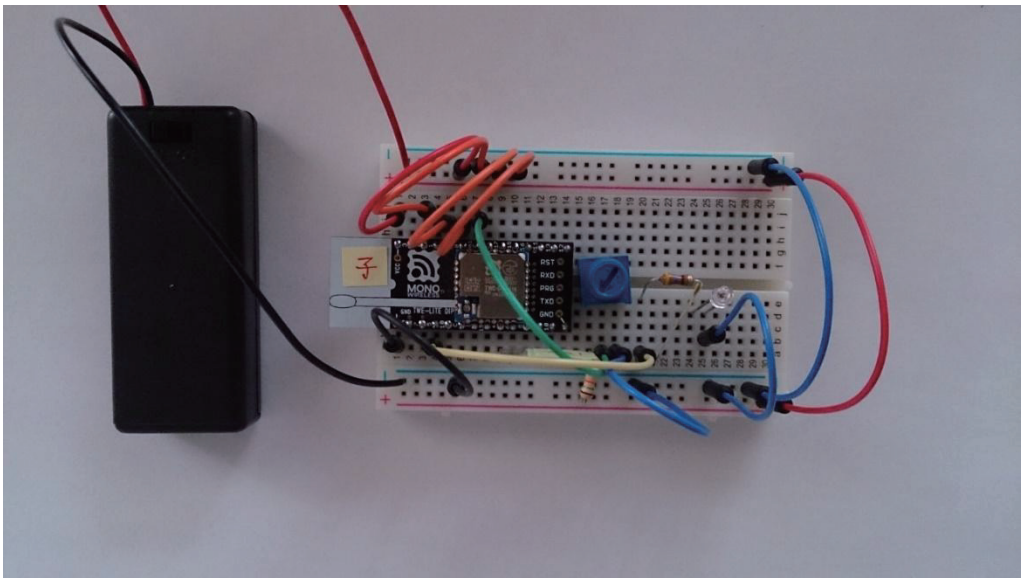


図　92

◇MoNoSTICK を PC にセット

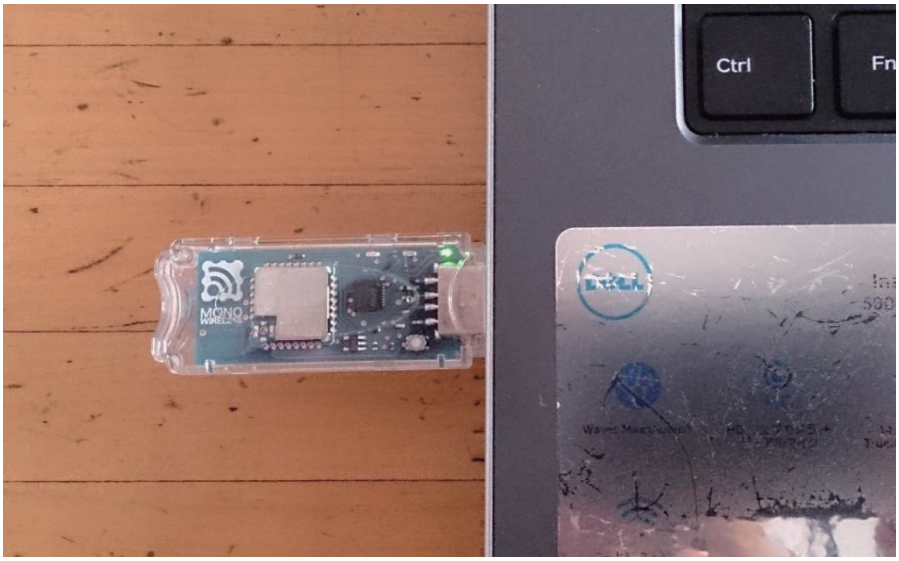


図 93

MoNoSTICK を PC の USB コネクタに差し込んで、OS に認識させます。デバイスマネージャで仮想 COM ポート番号を確認してください。第 4 回で使用したものと同じ PC 環境であれば COM ポート番号はもう分かっているはずです。

◇Windows アプリケーションの準備が整っていれば、起動します。

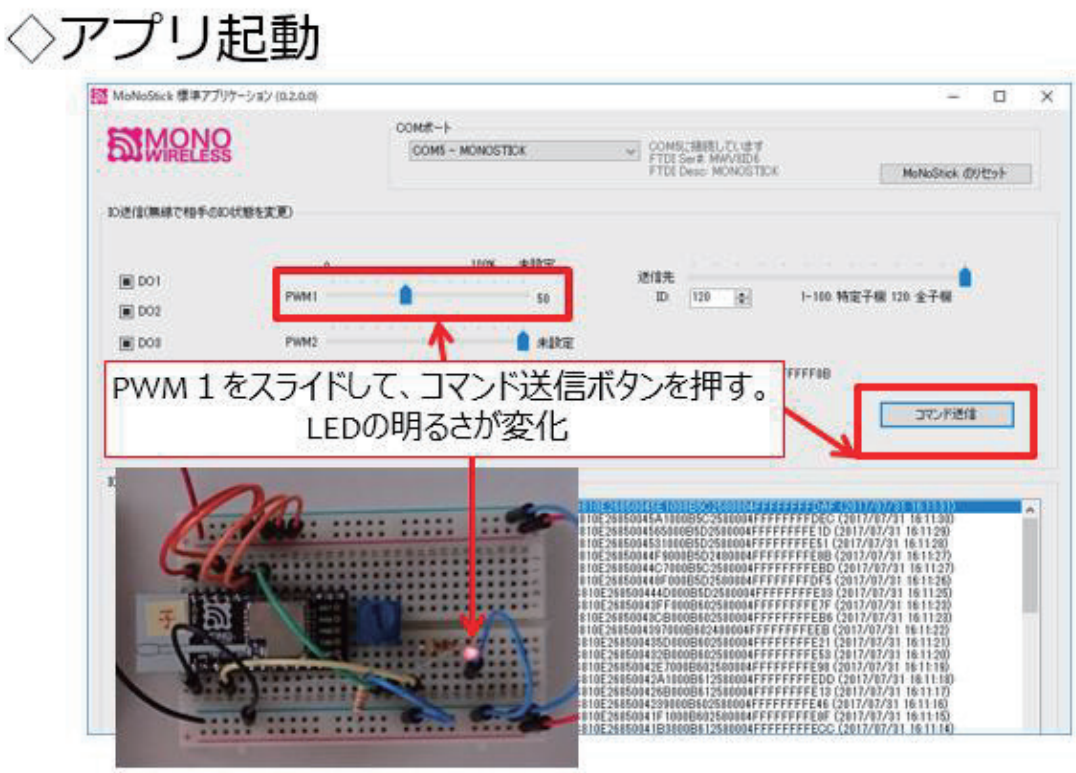


図 94

アプリケーションのウインドウが開いた際に中央上部に COM ポート番号が選択表示されていない場合は、デバイスドライバで確認した COM ポート番号をリストから選択してください。(子機の電源は、既に入っていると思いますが、ここで ON にしてもかまいません。)

◇動作確認

起動したアプリケーションの中央やや左の上段にある PWM1 と表示されている**スライダーを中央にスライド**して右側中程にある【コマンド送信】というボタンをクリックします。すると、LED が点灯します。次に、**スライダーをさらに右に移動**して再び【コマンド送信】ボタンをクリックすると、LED は先ほどより明るくなります。反対に最初の状態

より**左側にスライド**させてコマンド送信ボタンをクリックすると LED は暗くなります。Windows アプリではスライダーの位置を決めてから【コマンド送信】ボタンをクリックすることで、通信電文が送られるような仕様になっているので、スマートフォンアプリと少し使い勝手が違っていますが、同じように PWM 制御が行えます。次に、子機の VR を回転してみます。

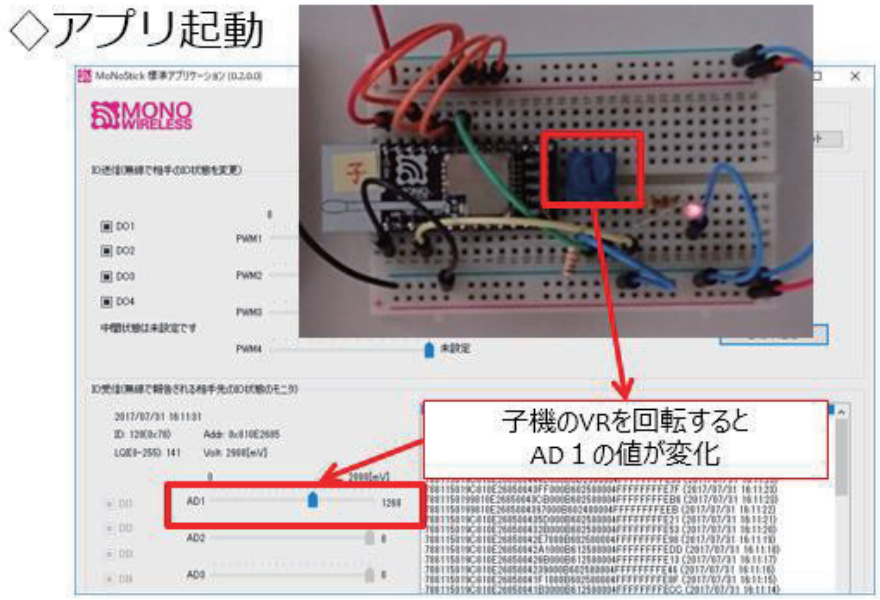


図 95

子機の VR を右に回転すると、Windows アプリの下の方にある AD1 というスライダーが右に移動し、その右にある数字が大きくなります。また、VR を左に回転するとスライダーが左の方に移動しながら、数字が小さくなっていきます。先ほど LED の明るさを PC 側からコントロールしたのに対して、子機の VR で作られる電圧によって、親機側（PC 側）のアプリ上のスライダー位置をコントロールしているということになります。このシステムでは親機と子機の間で双方向無線通信を行い、それぞれの状態を通知しあうことで、互いをコントロールしているのです。

第9回　温度センサー

今回は、温度センサーを使用します。計測値を離れたところに無線通信で送ります。主な内容は次の通りです。

図　96

温度は電圧として温度センサーから得られますが、それはそのまま PC に送信して、PC 側の処理プログラムを作り、電圧から温度に変換計算を行って温度を求めます。PC 側プログラムには **Python** という言語を使うことにします。Python は、最近ビッグデータの処理システムや AI などに使われてきています。豊富なライブラリの公開などにより、あらゆるシステム開発を少ない時間で行うことができ、コスト削減につながるので産業分野でも利用され始めています。ここで Python を経験しておくことは、皆さんのスキルアップにつながります。

◇システム概要

図 97

今回の新たなデバイスは温度センサーです。使用している無線マイコンモジュールは、温度センサーを取り扱うための機能が標準アプリケーションに含まれているので、実習にはもってこいのテーマです。**計測値から温度を計算して PC で表示**します。

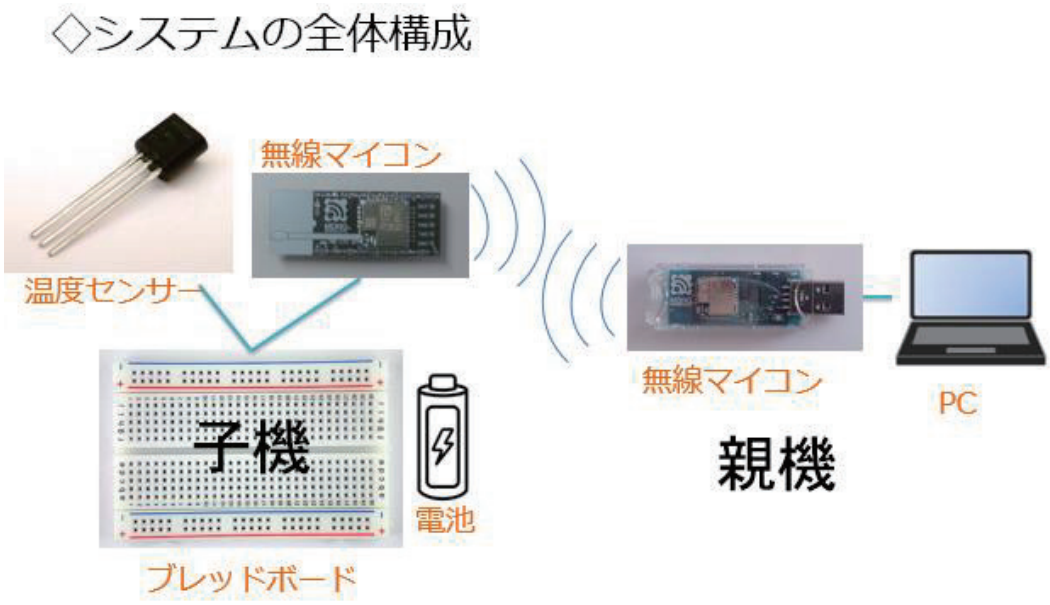


図 98

システムの全体構成は、上図の通りです。前回の実習の子機の VR の代わりに温度センサーが使われていると思えばよいでしょう。

◇温度センサーはアナログ電圧出力

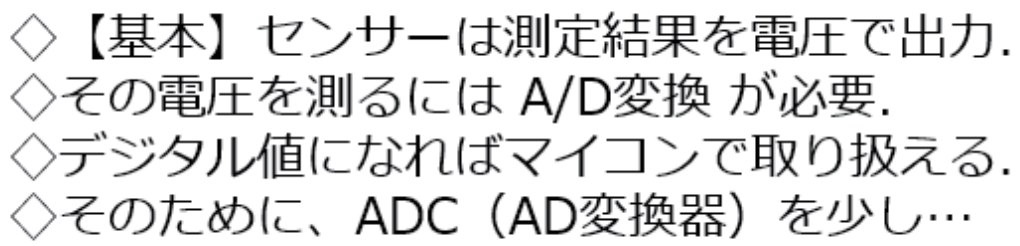


図 99

一般にセンサーは計測結果を電圧で出力します。電圧はアナログ値ですからそのままではデジタルのコンピュータでは計算などができません。そのために電圧のデジタル値を得るために A/D 変換（アナログ→デジタル変換）を行います。デジタル値になって取り込めればマイコンでも容易にいろいろな計算に使えるようになります。そのために、使用している無線マイコンモジュールの A/D 変換器について少し知っておきましょう。

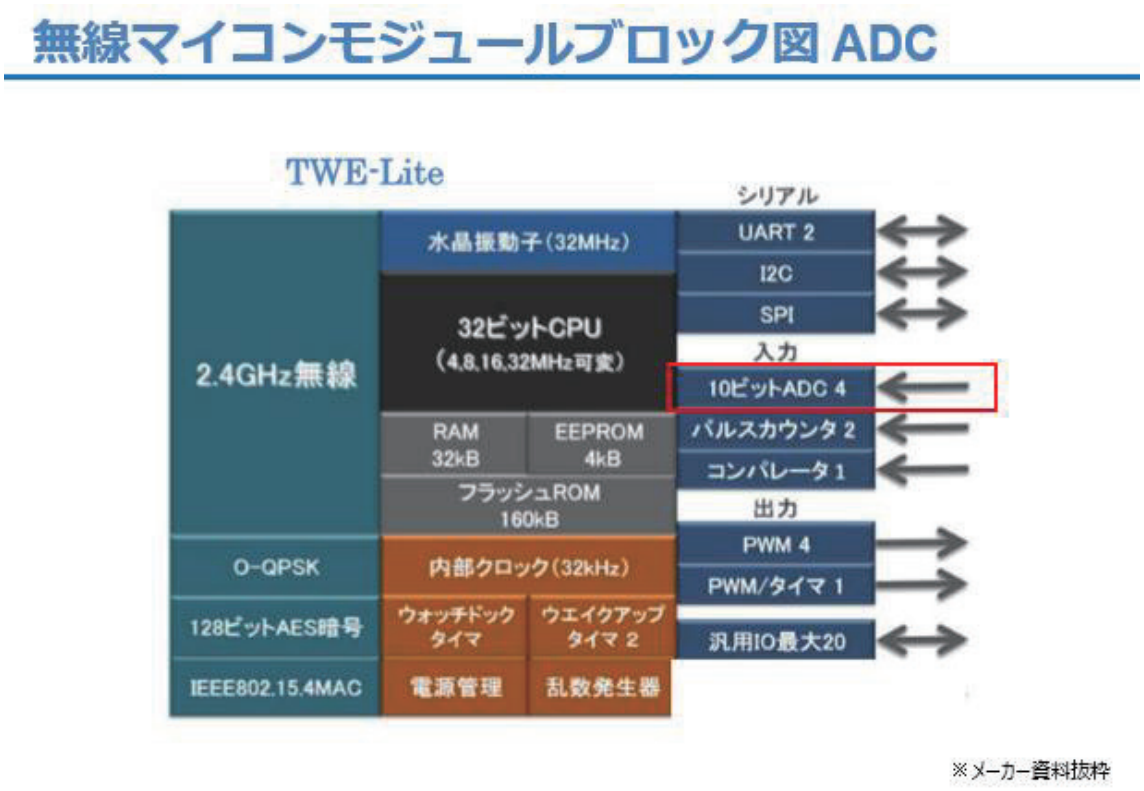
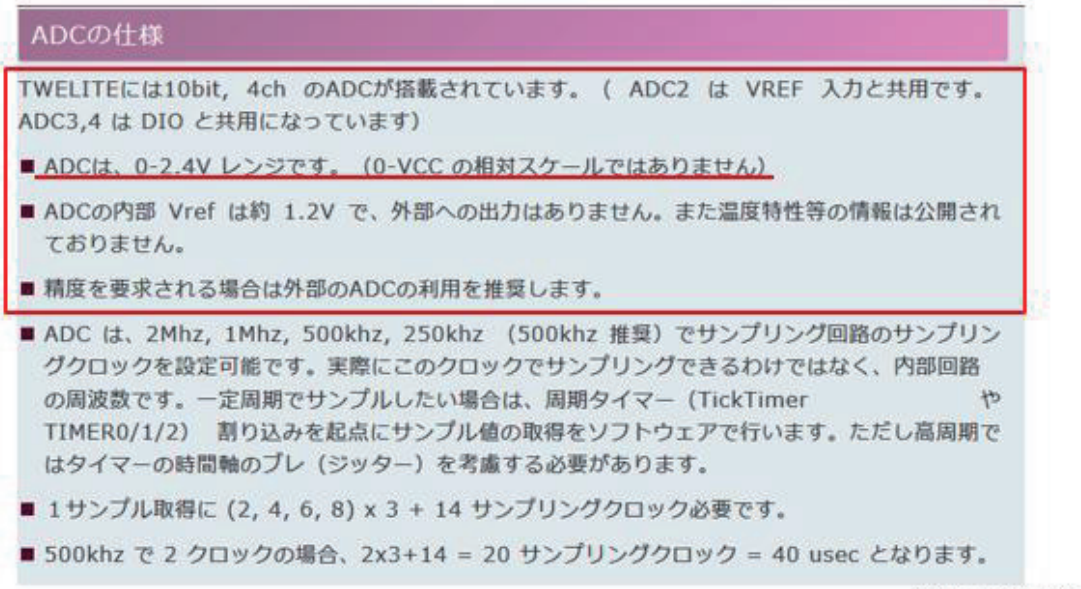


図 100

上図は、無線マイコンモジュールのブロック図です。この中に【10 ビット ADC】と書かれているのが A/D 変換器（A/D Converter）です。4 という数字は 4ch（4 系統）あるという意味です。



※メーカー資料抜粋

図 101

ADC の仕様が上の資料（再掲）に書かれています。【ADC の入力レンジは 0-2.4V】と書かれています。これは、その範囲内の電圧を測れるということです。前に示したブロック図の内容と合わせて ADC の内容を整理すると次のようになります。

◇レンジ：0～2.4V → 10bit（0～1023）
◇10bitのAD変換値を1byteにまとめている.
◇ADC補正値も付加している.
2bit × 4ch = 8bit

※標準アプリで通信を行いやすくするためか.
※2bit 補正は、AD 値を 1/4 しているため.

図 102

まず 1ch 分の A/D 変換の値は 10bit にまとめられているので、その内容は 0～1023 という数値になります。この 10bit の値が 4ch 分あるので、無線通信の効率にも配慮して、1ch あたり 8bit＋2bit の補正値として、

全体で 5byte（【8bit×4ch＝4byte】＋【2bit×4ch＝1byte】＝5byte）の ADC の値を子機から親機に通知するような設計になっています。具体的にどのように取り扱っているかについては、以下で説明します。

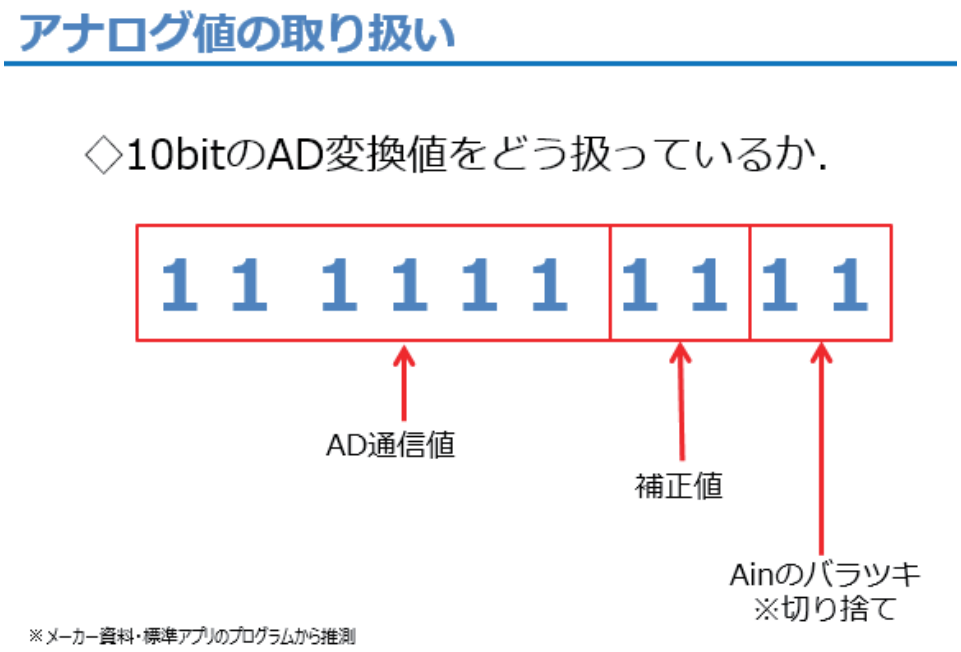


図 103

10bit の AD 変換値の下位 2bit は、バラツキがあるので切り捨てて 0 と考えます。残りの 8bit の下位 2bit を補正値とします。補正値は 4 分の 1 した値と考えられます。残りの 6bit を右に Shift して 8bit として 1ch 分のデータとします。こうしておくと将来マイコンモジュールの ADC の能力が変更になり 12bit となっても対応ができます。このデータを無線通信の電文中に配置して子機から親機に通知しています。この電文を見るとこの通信は、普通のシリアル通信と同じであることが分かります。

データ受信コマンド

◇先頭はコロン【：】で始まるテキストデータ

図　104

実際の通信電文は【**データ受信コマンド**】として上図の様に説明されています。先頭がコロン【：】で始まるテキストデータで、電文の中に含まれる情報は、上図の様に区切って解釈を行う約束です。右側に太枠で囲んだアナログ入力値と補正値があります。電文のこの位置の数値を解析すれば、子機が計測した温度センサーの出力電圧が分かり、それを元に温度の値が変換計算できるという仕組みです。

アナログ入力値

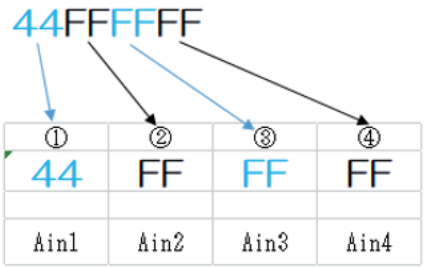


図　105

アナログ入力値の部分は上図の様に区切られていて、16 進数 2 桁ずつ

（1byte）の文字列となって 4ch 分送信されます。図で Ain1 の部分がアナログ入力 1 であり、今回温度センサーを接続する部分のデータとなります。

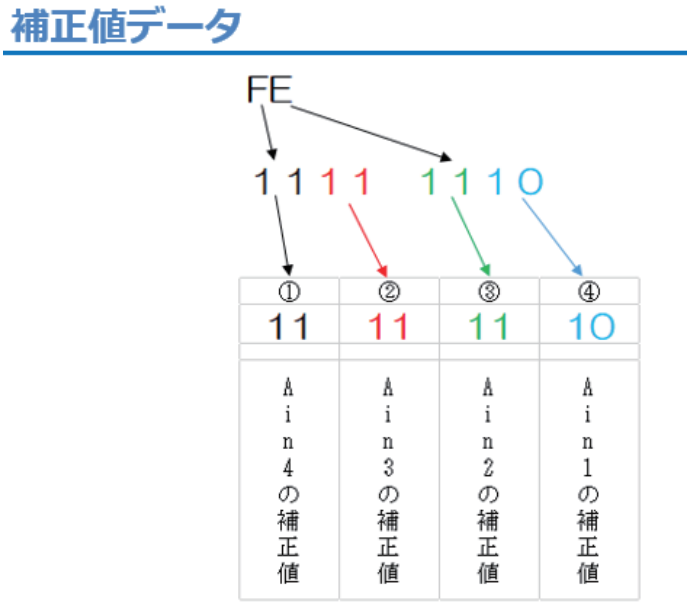


図 106

補正値データは、16 進数 2 桁で通知されているものを 2bit ずつに区切って解釈をします。各チャンネルの補正値は図の様に配置されています。ch1 の補正を行うときは、下位 2bit を使用します。

これら電文中の情報から、測定電圧を得る仕組みは、次の様に考えられます。

補正を含めた電圧計算

◇AD変換の値は標準アプリの中で処理されている.
◇計測した電圧の計算は次による.

補正ビットを含めた計算
電圧 ＝ ((AD値 × 4) ＋ ch補正値) × 4

※ この計算は、PC内プログラムで処理する.

図 107

受信電文の中に配置された AD 値と補正値を取り出して上のような計算を行えば、元の電圧を知ることができます。この電圧を基にして、次は温度を知る計算を行うのです。それには、温度センサーの特性を調べる必要があります。

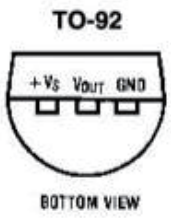
温度センサー

◇LM61CIZ　リニアな特性

◇測定範囲:－30℃～100℃
－30℃＝300mV
～
0℃＝600mV
～
100℃＝1600mV

◇温度係数:＋10mV／℃

◇動作電圧範囲:＋2. 7～＋10V



温度センサー(LM61CIZ)

温度＝（センサー出力電圧－600mV）÷１０mV

図 108

今回使用する温度センサーは、一般に広く使われているものです。LM61 シリーズとしていろいろな型番のものが作られています。その中の LM61CIZ というセンサーを使用します。次の資料は**データシート**の抜粋です。今回は LM61 シリーズの中の C グレードのセンサーを使用します。

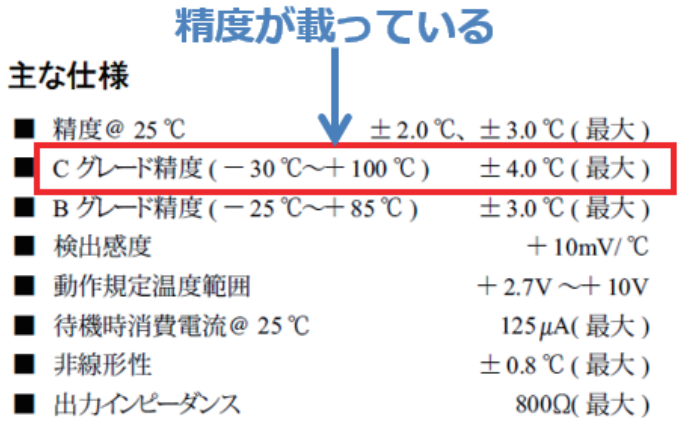
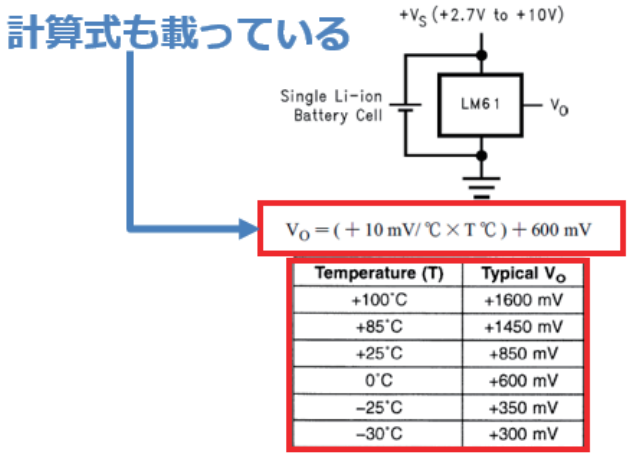


図 109

データシートには、細かな情報がたくさん書かれていて、難しく思えるところもありますが、次の資料の様に計算方法などの記載もあり、これを使って温度換算を行いますので、役立つ資料です。



Temperature (T)	Typical V_O
+100°C	+1600 mV
+85°C	+1450 mV
+25°C	+850 mV
0°C	+600 mV
−25°C	+350 mV
−30°C	+300 mV

FIGURE 1. Full-Range Centigrade Temperature Sensor (− 30 ℃〜+ 100 ℃)
Operating from a Single Li-Ion Battery Cell

図 110

センサーの温度特性グラフ

◇センサー特性をもとにA/D変換のレンジを確認。

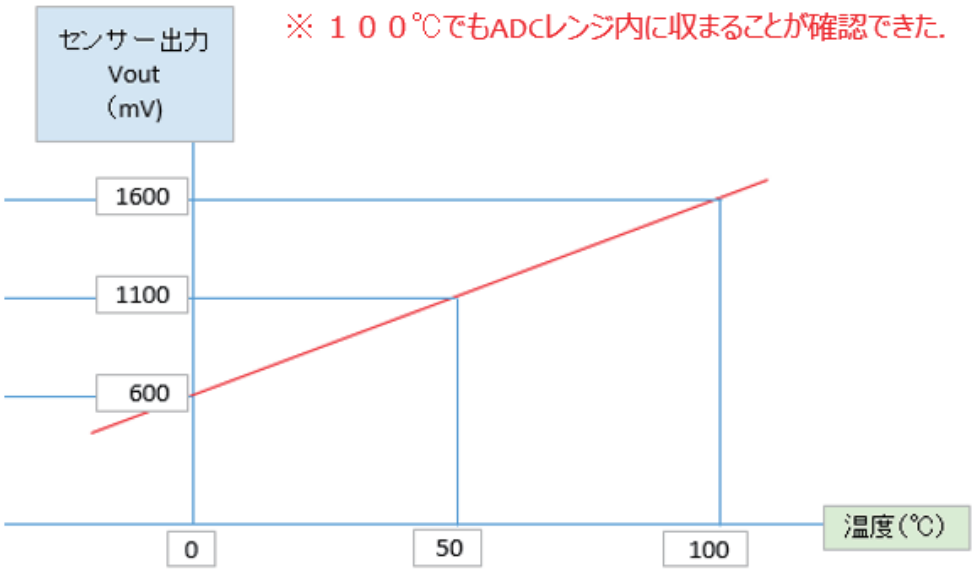


図　111

データシートに記載されているセンサーの特性をもとにして温度と出力電圧の関係をグラフにしたものが上図です。リニアな特性という表現が出ていましたが、温度と出力電圧は直線で描くことができる正比例の関係であることが理解できます。このグラフで見ると 100℃の温度でも出力電圧は 1600mV となっていて、**無線マイコンモジュールの ADC のレンジ（0-2.4V）内に収まっています**ので、安心して使えます。

この出力電圧から温度を求める計算は、つぎの様に行えばよいわけです。

◇電圧値から温度を計算

温度〔℃〕＝（電圧〔mv〕－６００〔mv〕）/１０

図　112

ピンの誤りは高熱破壊！！

ピン配置を十分確認.

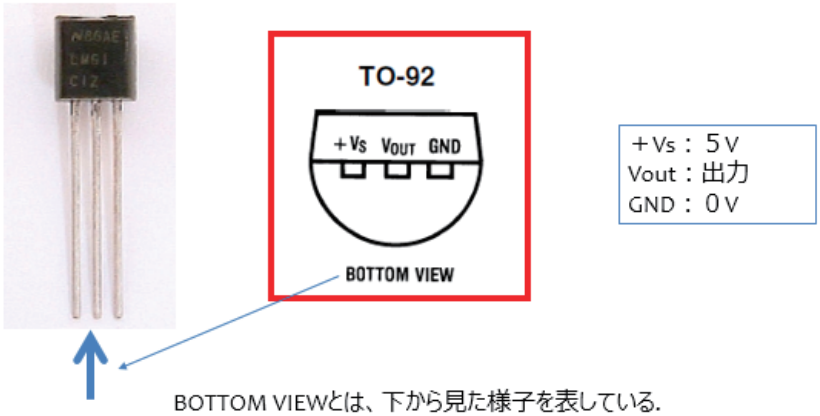


図　113

上の写真は温度センサーのピンを下にして正面（型番が印刷してある面）から見た様子です。中央に【BOTTOM VIEW】という蒲鉾を逆さまにしたような図が描いてあります。これは、センサーのピン側からみた図です。このように見たとき、「左側のピンを電源（＋）、中央のピンを出力電圧、右側のピンを GND(0V)に接続する」、ということを表しています。図の右側に＋Vs：5V と書いてありますが、今回の電源電圧は 3V です。このセンサーの動作電圧は+2.7V～+10V という記載がデータシートにあるので、この 5V は 3V に読み替えてください。

もしこの図を読み間違えて、センサーを反対向きに配線すると、電源と GND が逆になってしまいます。すると、電源を投入したとたんに火傷するほどの高熱を発してセンサーに致命的なダメージを与えてしまいます。くれぐれも向きを間違えないようにしてください。

今回の実習で必用なパーツは次の通りです。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. 温度センサー（LM61CIZ）×1 個

親機側：

1. PC×1 台（Windows10＋Internet 接続）
2. Python 開発・実行環境×1 セット
3. MoNoSTICK×１台

今回は Python という言語を使って PC の処理プログラムを作ります。この環境準備については、後ほど説明します。

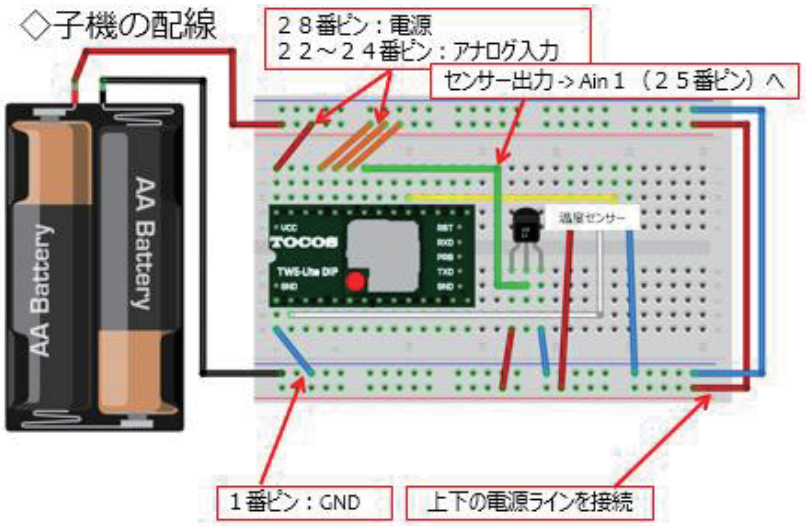


図 114

パーツの準備が出来たら上図に従い子機を配線します。温度センサーに必要なジャンパー線はわずか 3 本ですから、間違えることはないと思いますが、センサーの向きには十分注意してください。

実際に配線した様子　子機

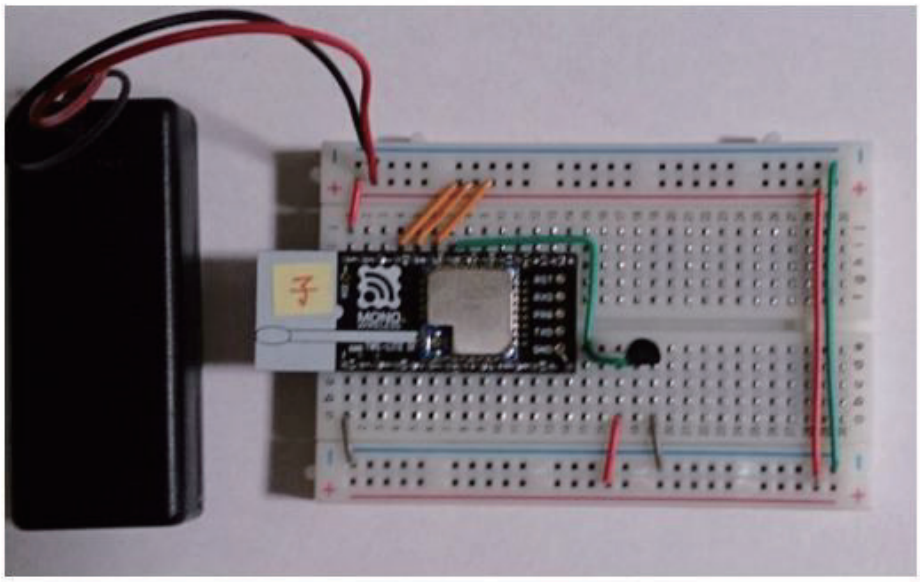


図　115

実際に配線したものは上の写真の様になりました。温度センサーの中央のピンは、ジャンパー線によって無線マイコンモジュールの AI1（25 番ピン）に接続されています。

Python Webサイト

図　116

今回、PC 側処理系では Python を使うと説明しました。皆さんは Python（パイソン）という言語を既にご存知で利用されている方もいらっしゃるのではないでしょうか。使っていなくても書店のコンピュータ関連書籍の棚に Python という見出しの雑誌や書籍を見かけていると思います。最近では AI やビッグデータの処理によく使われて、それらの特集記事が雑誌やメルマガに掲載されて、手元に届くようになってきました。

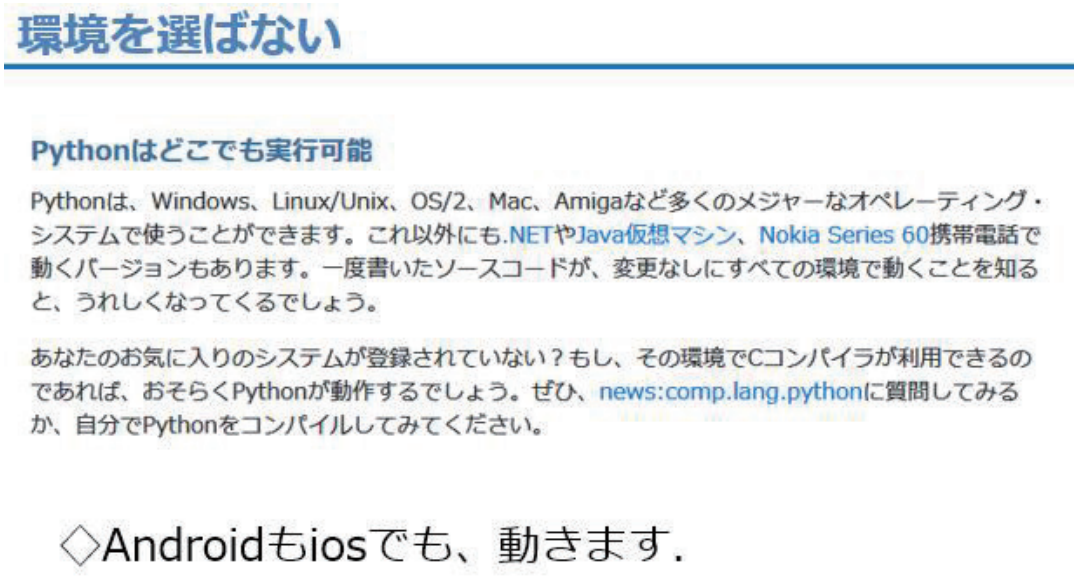


図 117

Python は、環境を選ばずどこでも実行することができ、Android や iOS でも動きます。最近では Micro Python と言って、マイコン上で稼動する環境も開発されています。技術革新は高速ですから、私が行うマイコン制御の実習も Python で行うようになるときが 1, 2 年後には到来する可能性があります。現在 Micro Python の準備を進めていますが、皆さんは PC での Python 環境の準備を行いましょう。

Python開発環境

◇この講座では・・・

1．Python2.7系を使用.
2．Pyserialを使用.

◇下記サイトから、Pythonのファイルをダウンロードします.

https://www.python.org/

図　118

この講座では、Python2.7 系を使用します。PC と無線マイコン親機間でシリアル通信を行うので、Pyserial というライブラリも使います。

まず上記の WEB サイトから Python のファイルをダウンロードします。

Python のWebサイト

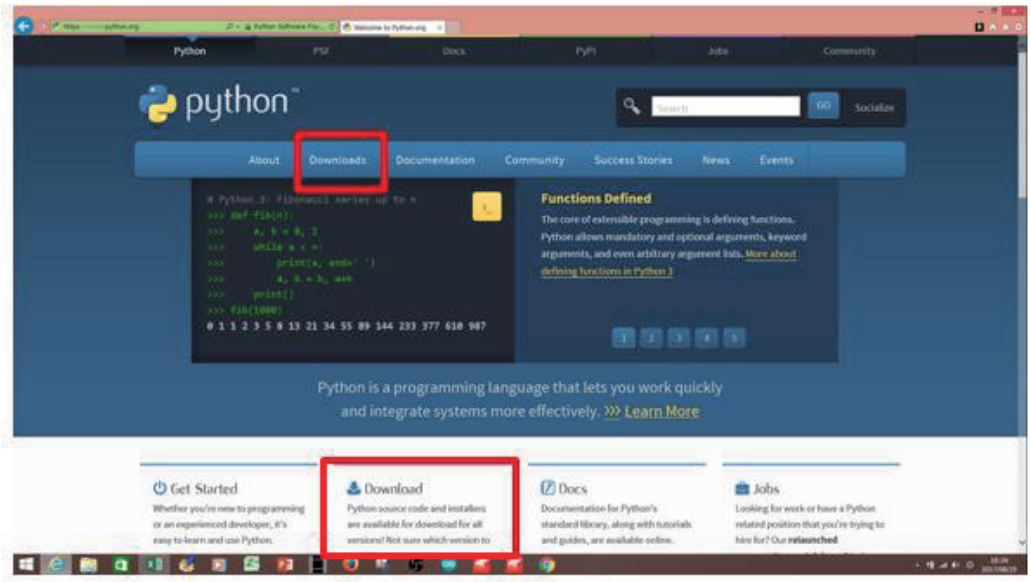


図　119

Python2.7系

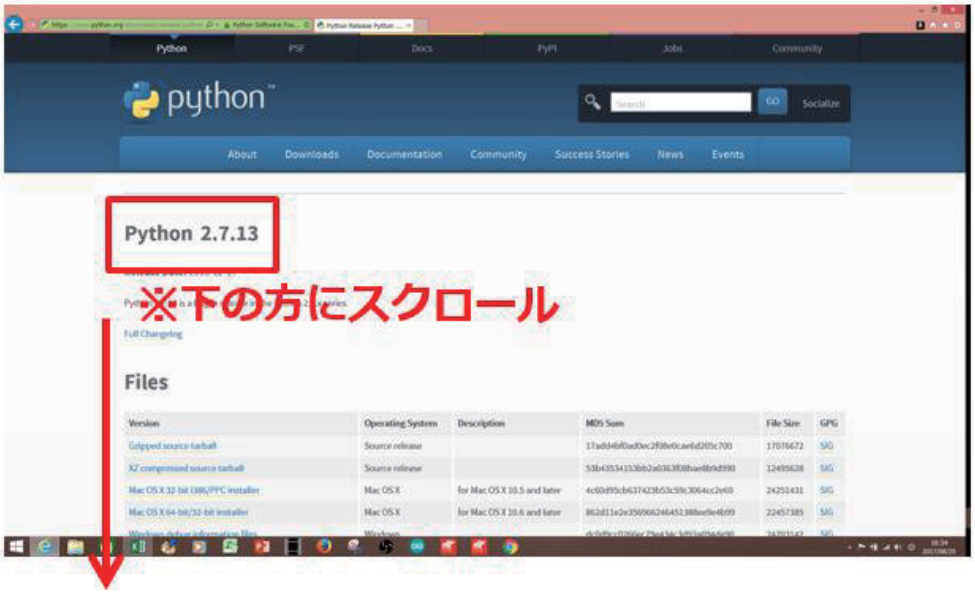


図　120

Windows x86 MSI Installer

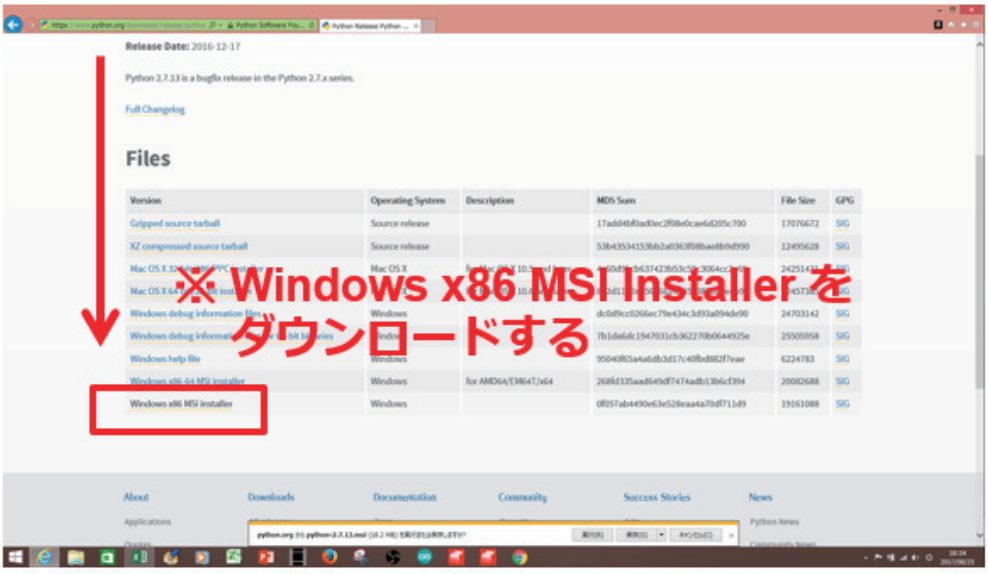


図　121

Install → パスの確認

◇ダウンロードしたファイルをInstall.
◇環境変数にPython27のパスが追加されている.

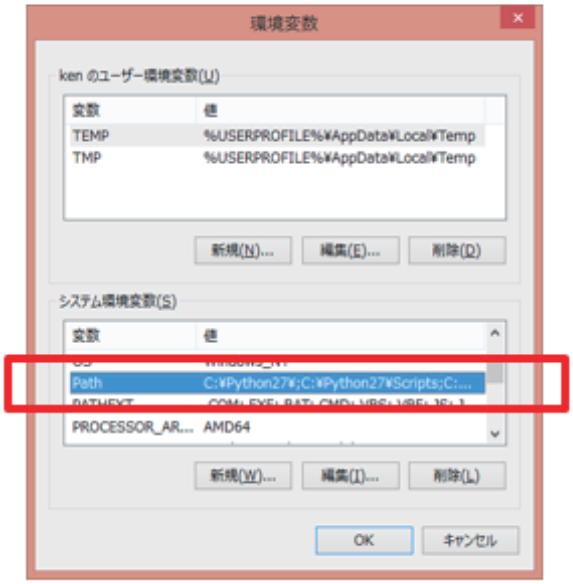


図　122

ダウンロードしたファイルをインストールすると、Windows の環境変数に Python27 の path が追加されているはずです。

起動テスト

◇コマンドプロンプトにpythonと入力し、
pythonが起動すればOK.

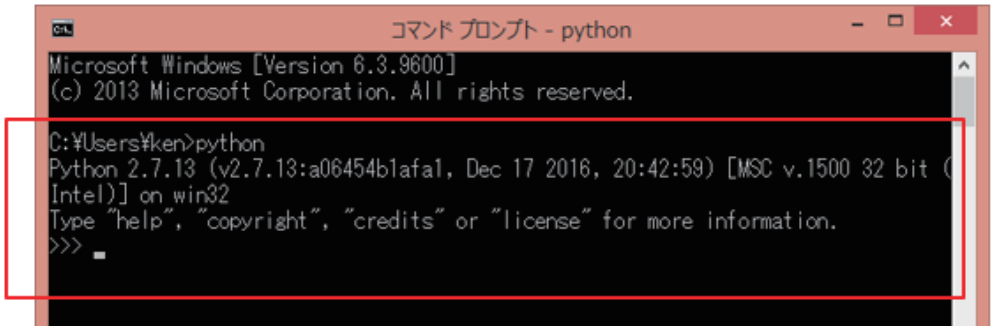


図　123

上図のように起動テストを行って下さい。コマンドプロンプトに

python

と小文字で入力して Enter キーを押下します。>>>というプロンプトが表示されれば OK です。次に **PC と親機間の通信で使用する Pyserial ライブラリをインストール**します。

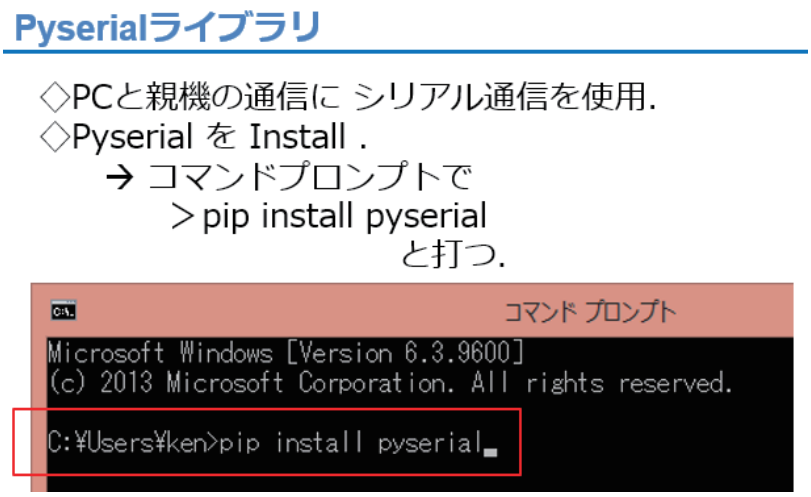


図　124

上図の様にコマンドを入力すると、下の様にインストールができます。Successfully の文字が表示されればライブライも準備完了です。

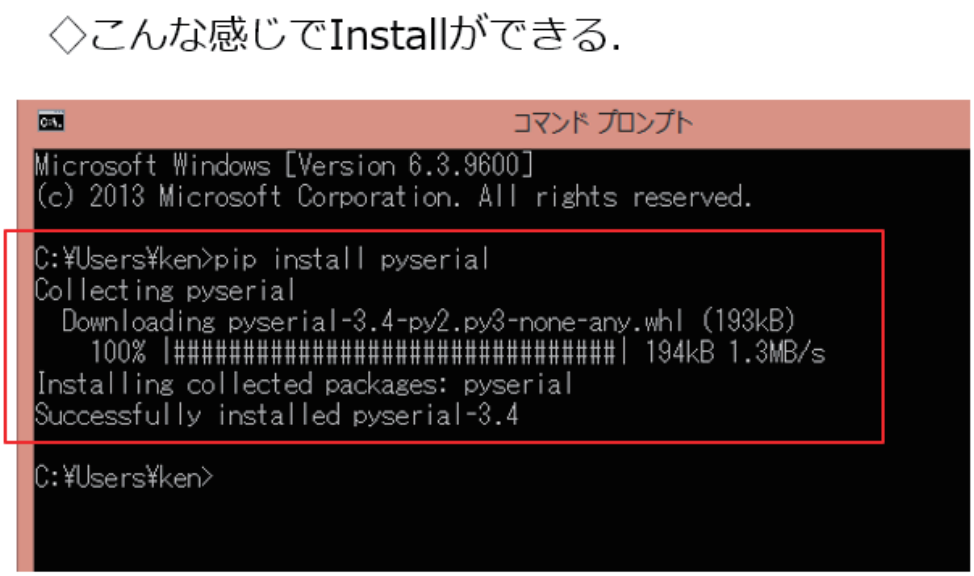


図　125

念のために、ライブラリが使えるか確認をしておきます。

Python をインストールすると IDLE というプログラムが追加されていますので、それを起動して、そのプロンプトに次の Python コマンドを入力してみます。

```
import serial
```

Enter キーを押下したとき、エラー表示も無く、プロンプト>>>が表示されれば、ライブラリはセットアップされています。

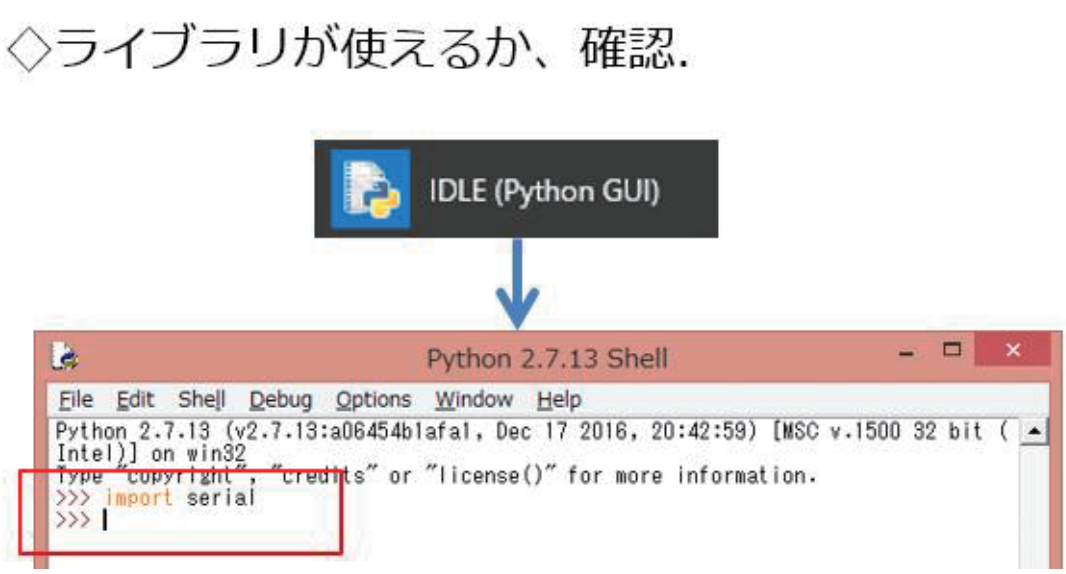


図 126

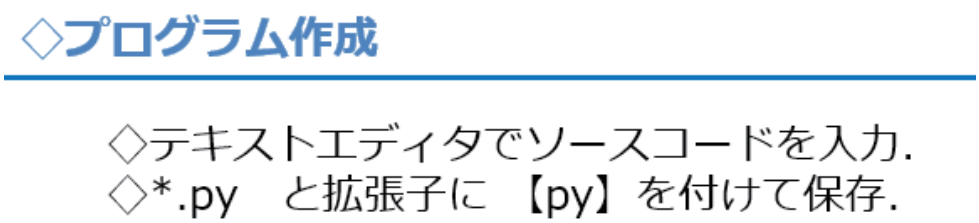


図 127

いよいよ Python のプログラム作成です。プログラムは使い慣れたテキストエディタ（私は TeraPad というフリーのテキストエディタを長年利用しています。）で、ソースコードを入力し、拡張子に【py】を付け

て保存して下さい。ソースコードは後に示しますが、この講座の Python のプログラムは大まかに次のような構造をしています。

プログラム　全体構造

図　128

最初にモジュールなどの取り込み、次に関数定義の部分、そして処理のエントリポイントとメイン処理部分となっています。＃より右側はコメントです。

では以下にソースコードを示します。

モジュールなどの取り込み

◇モジュール、パッケージなどの取り込み

```
import struct      # バイナリ<--->文字列相互変換モジュール
import binascii    # バイナリ<--->ASCII相互変換モジュール
import serial      # シリアル通信パッケージ
```

電文処理関数

◇電文（子機→親機）を解析する関数

```
def denbunKaiseki(data):    # 受信電文を解析する
    if data[0] != ":":
        return False        # 先頭が「:」でなければ、対象のデータではない
    data = data[1:]         # 先頭の「:」を取り除く

    # バイトデータに変換する
    ss = struct.Struct(">BBBBBIBHBHBBBBBBBBB")  # フォーマット文字列に従って
                                                # バイナリデータを読み書きする
    data = binascii.unhexlify(data.rstrip())    # 右端のチェックサムを取り除く
    parsed = ss.unpack(data)                    # 文字列をバイナリとして解釈
                                                # バイトデータの配列として解釈

    # デジタル入力／アナログ入力の値を計算する
    digital = [0] * 4                           # デジタル入力値4ch分
    digitalchanged = [0] * 4                    # デジタル入力変更状態4ch分
    analog = [0xffff] * 4                       # アナログ値4ch分
```

4ch分のデータ取り出し

◇デジタル・アナログ各4chあり.

```
for i in range(4):                      # 各4ch分繰り返し
    # デジタル入力
    if parsed[11] & (1 << i):           # バイトデータ配列の11番目が真か？
        digital[i] = 1                  #   真（＝１）ならデジタル入力を１にする
    else:                               # そうでなければ（偽）
        digital[i] = 0                  #   デジタル入力を０にする
    if parsed[12] & (1 << i):           # デジタル入力変更状態が真か？
        digitalchanged[i] = 1           #   真（＝１）ならば１にする
    else:                               # そうでなければ（偽）
        digitalchanged[i] = 0           #   偽（＝０）にする

    # アナログ入力
    if parsed[13 + i] == 0xff :         # アナログ入力値が0xFFならば
        analog[i] = 0xffff              #   max値とする
    else:                               # そうでなければ
        analog[i] = (parsed[13 + i] * 4 + ((parsed[17] >> (2 << i)) & 3)) * 4
                                        #   補正値も含めて元の値を復元する
```

結果をまとめた【辞書】を作る

◇項目にキーを付けた【辞書】.

```
# 結果を返すために、辞書データを作る
result = {
    "from" : parsed[0],                    # 送信元デバイスID
    "lqi" : parsed[4],                     # 受信電波品質
    "fromid" : parsed[5],                  # 相手の個体識別番号
    "to": parsed[6],                       # 宛先端末の論理デバイスID
    "timestamp": parsed[7],                # タイムスタンプ
    "isrelay": parsed[8],                  # 中継フラグ
    "baterry" : parsed[9],                 # 電源電圧
    "digital" : digital,                   # デジタル入力値
    "digitalchanged" : digitalchanged,     # デジタル入力変更状態
    "analog" : analog                      # アナログ入力値
}
return result  # 呼出元に戻る
```

処理の本体

◇main() 関数のような部分は、一番下に書く.

```
# ここから、処理開始
# COM5を開く<---自分の環境に合わせてCOMポート番号を指定する
s = serial.Serial('COM5', 115200)     # COMポート番号、通信速度

while 1:   # ずっと繰り返し
        data = s.readline()                          # シリアルポートから1行読取り
        parsed = denbunKaiseki(data)                 # 1行を分析して、項目ごとの値を求める

        t = (parsed["analog"][0] - 600.0) / 10.0     # 電圧を温度に変換する
        print t                                      # 温度を表示する

# COMを閉じる
s.close()
```

※厳密には、もっと書いた方が良いところもあるが・・・.

【注】ソースコードは、実習キット付属 CD に収録しています。ソースコードを入力したら、【py】という拡張子を付けて、適当な場所に保存してください。

【重要】上のソースコードで、‘COM5’と書いた部分は、実際に親機を接続した PC での仮想 COM ポート番号に書き換えてください。

これから動作確認を行いますが、手順を間違えると正しく動作しません。1. MoNoSTICK を PC にセットして認識させる。2. 子機の電源を ON する。3. Python のプログラムを実行する。です。

図 129

IDLE を起動し、Python のプログラムを開きます。

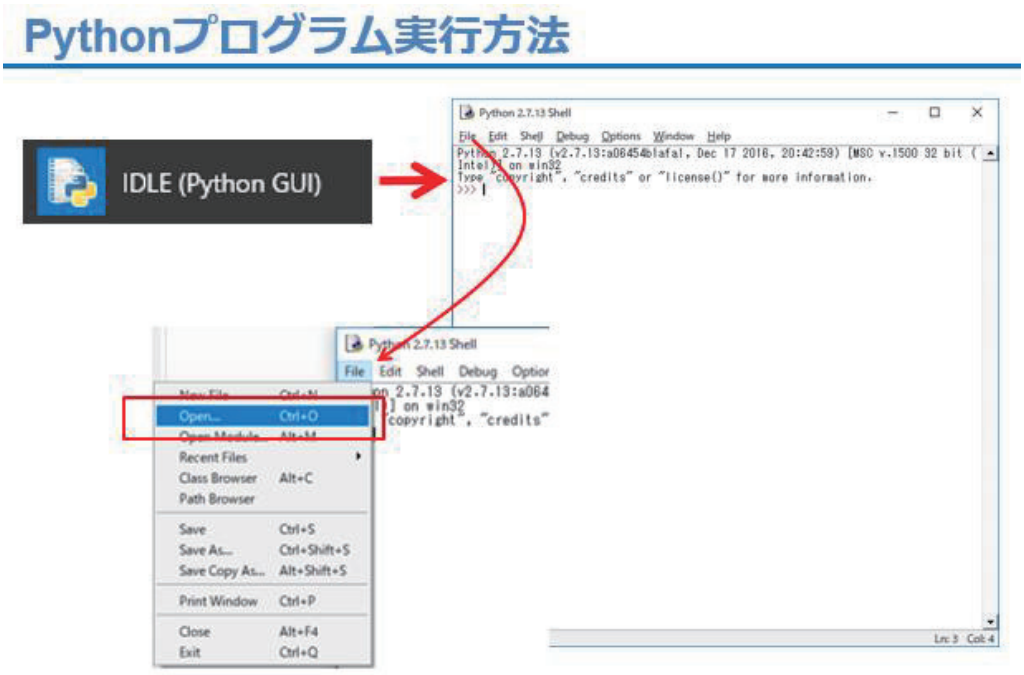


図 130

開いたソースコードファイルのメニューで Run---->Run Module と選

択すると、別のウインドウが開いて実行が始まります。

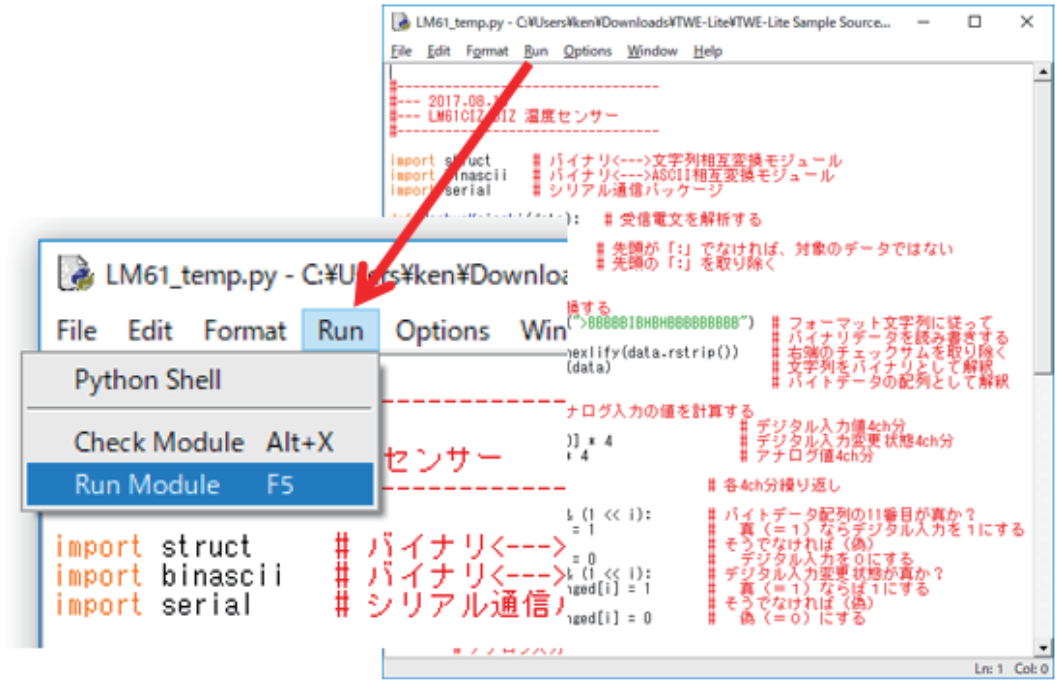


図　131

動作確認

◇温度が標示される.
◇感度や精度について考えてみる.

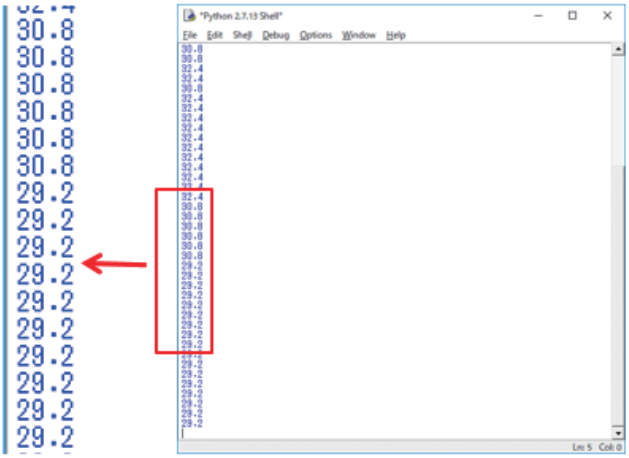


図　132

ここまでの準備が完璧なら、図のようにウインドウに温度が表示されていきます。温度センサーの頭に指を触れてみて下さい。体温が伝わり

表示温度が上昇します。指を離せば温度は低下します。

子機は電池駆動で親機とは無線通信をしていますから、測定場所を移動することが容易にできます。これが無線マイコンモジュールの便利なところです。

今回の講座では、温度表示を PC 画面に行っているので、測定場所では温度が分かりません。この点を改善するために、次回は表示器の使い方を学びます。

第10回　液晶表示器(LCD)

今回の講座では液晶表示器（LCD：Liquid Crystal Display）の使い方を学びます。無線で離れたところにメッセージを表示することができます。

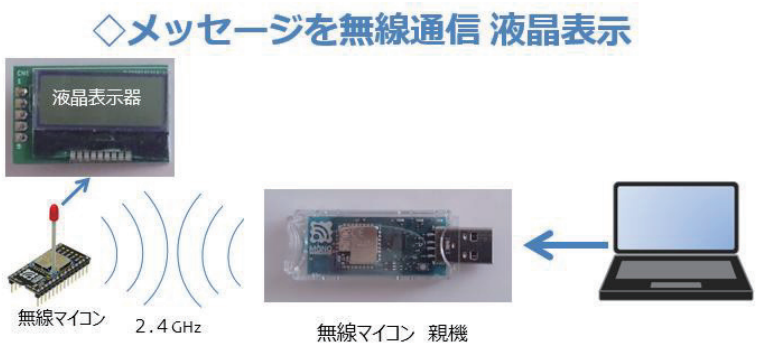


図　133

PC でメッセージを作りそれを無線通信の電文に埋め込んで、親機から子機に送ります。子機は受信した電文に含まれるメッセージを液晶表示器に表示します。第 9 回で、温度センサーの情報を子機から親機に送り、PC で処理して温度を求めて表示したのとは、情報の流れる方向が逆になります。

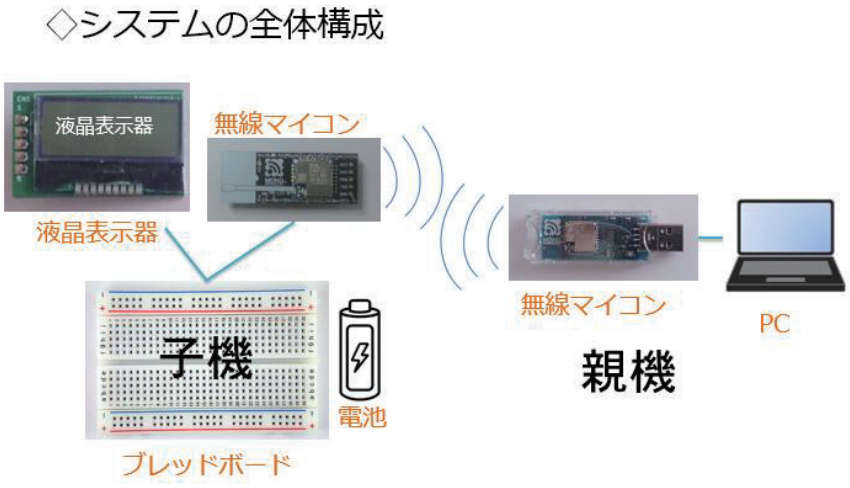


図　134

第９回の温度センサーに代わって、液晶表示器（LCD）が子機で使われます。温度センサーは子機に対する入力デバイスでしたが LCD は出力デバイスとなります。今回は PC で固定のメッセージを作りそれを子機の LCD に２行で表示してみます。PC 内部のプログラムは Python で開発します。

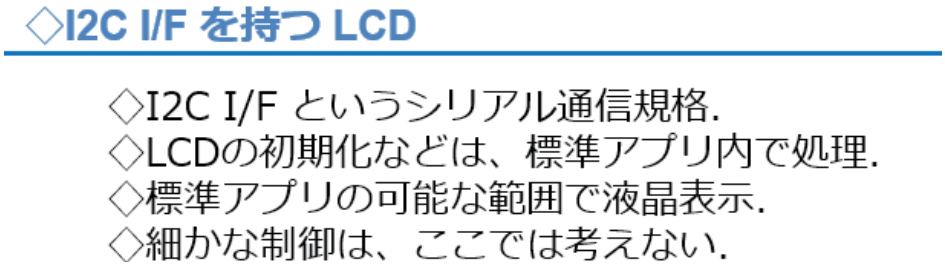


図 135

今回使用する LCD は I2C I/F という２線式のシリアル通信規格を使用するものです。I2C というのはクロックとデータを伝える電線を準備して、デバイスはその２本の信号線にぶら下がる形で種類の違うデバイスでも複数台が接続できます。デバイスごとの識別は I2C スレーブアドレスという番号がデバイスごとに決まっていて、その番号で通信相手（スレーブ）を指定して通信を行います。スレーブに対するマスターは、無線マイコンモジュールの子機が担当します。この様に書くと何やら難しそうですが、実際はそうでもありません。この講座で使用している無線マイコンモジュールに、最初から書き込まれている標準アプリケーションの機能として、サポートされている液晶表示器があります。それを使うことで、容易にシステムを開発できます。利用できる液晶表示器は型式コードが決まっています。

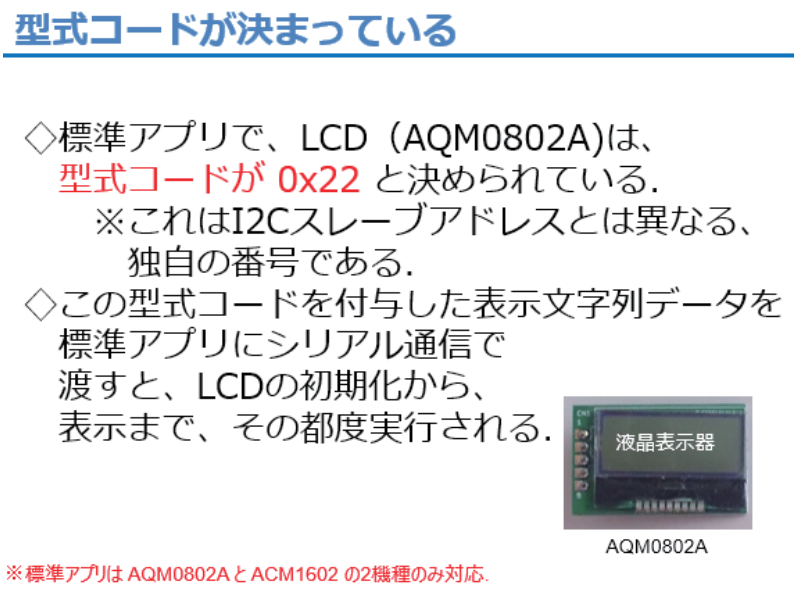


図　136

LCD は上に示す AQM0802A を使います。異なる LCD は使用できません。但し、無線マイコンモジュールのアプリケーションを作成して独自にサポートする LCD 制御プログラムを内蔵させれば、別のものでも使用できます。

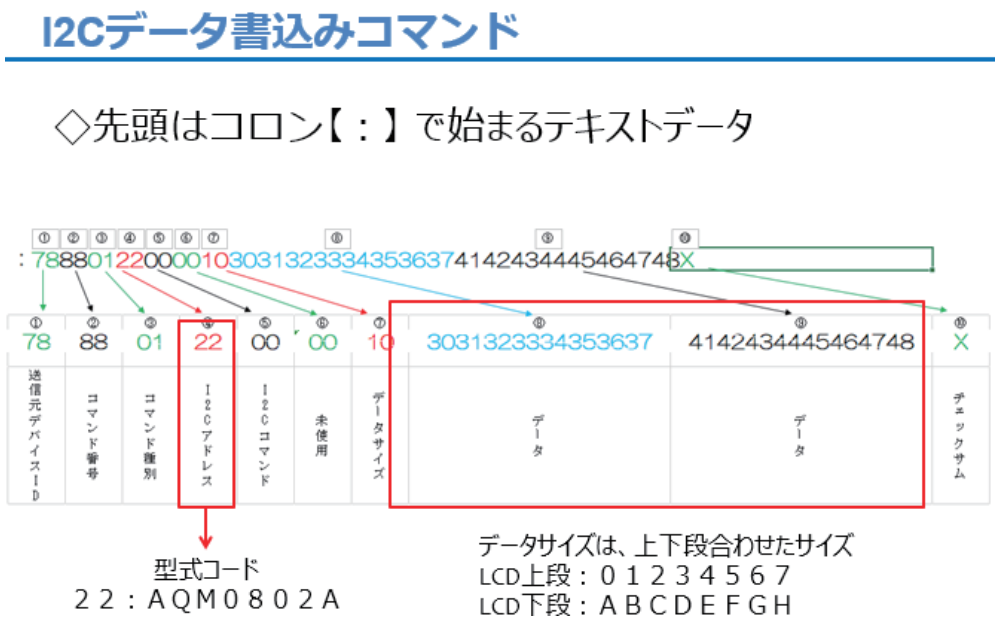


図　137

親機側から子機に対して制御を行う無線通信の電文に I2C データ書き

込みコマンドというものがあります。マイコンモジュール内蔵の標準アプリでは、この電文をサポートしています。

【注意】この電文の解説がメーカーWEBページで公開されています。また、書籍「TWE-Liteではじめるカンタン電子工作」（I/O BOOKS）でも解説されていますが、標準アプリのバージョンが更新されたためか、解説が古いものとなっていて、実際の無線マイコンモジュールの電文とは一部解説の内容が違っていますので、注意してください。

上記電文でI2Cアドレスと表記してある箇所が実は液晶表示器の内部型式コードとなっています。ここに16進数で22という型式コードを埋め込むと、無線マイコンモジュールの標準アプリはLCDとしてAQM0802Aを制御対象とした操作をするように動きます。LCDに表示するメッセージはデータと書かれた箇所に8文字×2列（LCD上段と下段）のデータとして埋め込みます。合計で16byteのデータになるので、データサイズ（データの左側）は16進で10となっています。この電文を送信すると、子機はその都度LCDをリセットして表示を消去し、必要な初期化処理を行い上下2段の文字列表示を行う動作をします。

上の電文を作り、子機に対して送信するプログラムをPythonで開発しPCで動かすことで、無線通信で子機のLCDにメッセージを表示できます。使用する液晶表示器の詳細は下図の様になっています。データの部分に埋め込む文字コードを変化させることで、自由なメッセージを表示することができます。

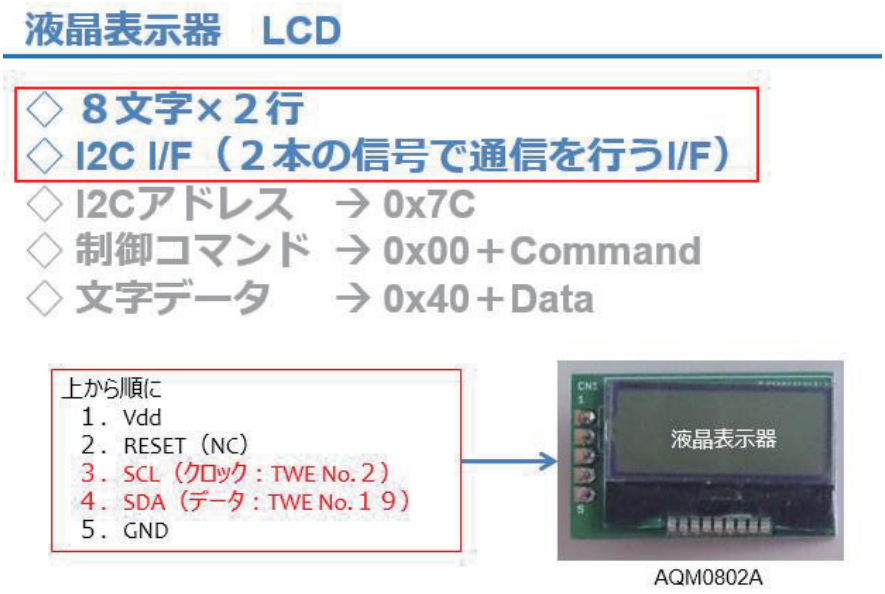


図　138

液晶表示器には、上図の様に、5 本のピンが取り付けられています。写真では上から順に番号が付いています。1 番が Vdd と記載がありますが、これは電源（＋）のことです。ここでは 3V になります。2 番は LCD の RESET 信号ですが、（NC:Not Connect）とありますので、なにもつなぎません。3 番は I2C I/F のクロック信号で無線マイコンモジュールの 2 番ピンに接続します。4 番は I2C I/F のデータで無線マイコンモジュールの 19 番ピンに接続します。5 番は GND です。

今回の実習で必用なパーツは、次の通りです。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. LCD（AQM0802A）×1 個

親機側：

1．PC×1 台（Windows10）

2．Python 開発・実行環境×1 セット

下の図に従って配線を行います。LCD とマイコンのピン接続を枠内に記載してありますので確認をしてください。

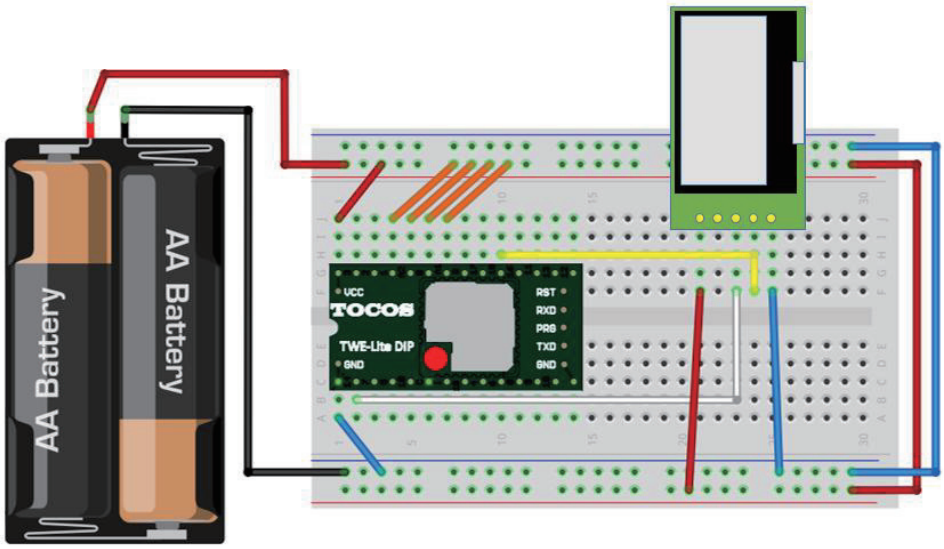


図 139

実際に配線をした子機は次の写真の様になっています。

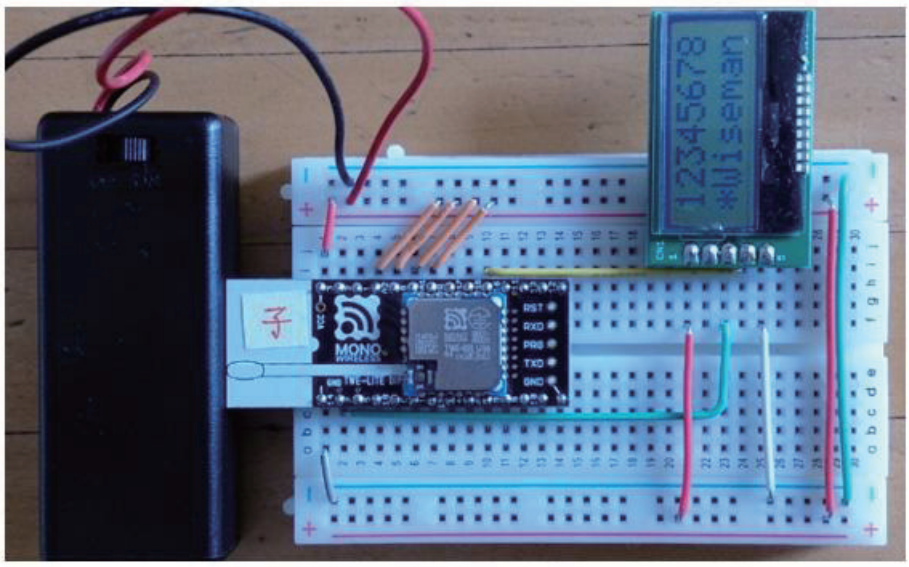


図 140

この写真は既に動かしているものですが、正しく動作すれば写真の様

に LCD の上下 2 行にメッセージが表示されます。

既に Python 開発・実行環境は整っているはずですから、次はプログラムの作成です。ソースコードを下記します。

◇モジュールなどの取り込み

```
import struct       # バイナリ<--->文字列相互変換モジュール
import binascii     # バイナリ<--->ASCII相互変換モジュール
import serial       # シリアル通信パッケージ
import time         # 日付・時刻データを取り扱う
```

◇LCD制御電文送信　序盤

```
# TWE-Liteの標準アプリ内のI2C関数をシリアル通信を経由して制御する.
def accessI2C(s, sendto = 0x78, reqno = 0x00, command = 0x01,
        i2caddress = 0x00, i2ccommand = 0x00,
        data = [], readbyte = -1):

        # データを作成する
        if readbyte == -1:   # 引数が-1のとき、I2Cに書き込むだけ
                # 送信データのリストを作る.
                sendbytes = [sendto, 0x88, reqno,  i2caddress, i2ccommand, 0x00, len(data)]

                # dataを加える
                # dataを1文字ずつリストにする
                buf = list(data)
                # buffを文字コードの数値に変換する
                buff = map(ord,buf)
                # 送信データの後ろに、表示するメッセージの文字コードのリストを追加する
                sendbytes.extend(buff)
        else:   # 引数 > 0 のとき、
                # readbyteだけ読み取る（dataは利用しない）
                # 送信データのリストを作る.
                sendbytes = [sendto, 0x88, reqno,  i2caddress, i2ccommand, 0x00, readbyte]
```

◇LCD制御電文送信　中盤　電文完成～送信

```
# 16進数文字列に変換する
bytelen = len(sendbytes)

ss = struct.Struct(str(bytelen) + "B")
outstring = binascii.hexlify(ss.pack(*sendbytes)).upper()

# TWE-Lite子機に送信する
# チェックサム省略バージョンなので、"X"を追加しておく
s.write(":" + outstring + "X" + "¥r¥n")
```

◇LCD制御電文送信　終盤　子機応答チェック

```
# 応答を待つ
# 10回繰り返す
for i in range(10):
        status = s.readline()
        if status[0:9] == ":" + outstring[0:2] + "89" + outstring[4:8]:
                # 対応する応答結果が戻った
                status = status[1:].rstrip()      # 行頭の「:」と行末の改行を取り除く
                ss = struct.Struct(">BBBBBB")     # バイトデータに変換する
                parsed = ss.unpack(binascii.unhexlify(status[0:12]))
                if status[4]:
                        # I2Cへのアクセスに成功
                        # 戻り、１バイトを返す
                        ss = struct.Struct(str(parsed[5]) + "B")
                        result = ss.unpack(binascii.unhexlify(status[12:len(status) - 2]))
                        return result
                else:
                        # 失敗のときは、偽
                        return False
                break
return False
```

◇2行のメッセージをまとめて送信

◇LCDの上段・下段のメッセージをまとめて
１回で送信

```
# AQM0802A-RN-GBWに文字列を出力する
def writeAQM0802Msg(s, msg1, msg2):
    # 初期化は、標準アプリ側で行っている

    # 出力文字列を１つにまとめる
    msg = msg1.ljust(8)+msg2.ljust(8)
    # 子機に送信
    accessI2C(s, i2caddress = 0x22, i2ccommand = 0x80, data=list(msg))
    return
```

◇処理の本体

◇main() 関数のような部分は、一番下に書く.

```
# COM5を開く
s = serial.Serial('COM5', 115200)

# データを出力する
writeAQM0802Msg(s, "12345678", "*Wiseman")

# COMを閉じる
s.close()
```

【重要】上のソースコードで、‘COM5’と書いた部分は、実際に親機を接続した PC での仮想 COM ポート番号に書き換えてください。

“12345678” ,”*Wiseman”の部分を書き換えれば、自由なメッセージを表示することができます。自由といっても、ASCII コードの範疇のみ可能で、漢字やひらがななどは表示できません。ソースコードは【py】という拡張子を付けて保存して下さい。

図 141

動作確認をしましょう。親機を PC の USB コネクタに差込み、認識させます。IDLE を起動して Python のプログラムを開きます。第 9 回で行ったように Run---->Run Module と選択すると別のウインドウが開き、Python のプログラムの実行が始まります。実行の方法に不安な方は、第 9 回を参照してください。

動作確認

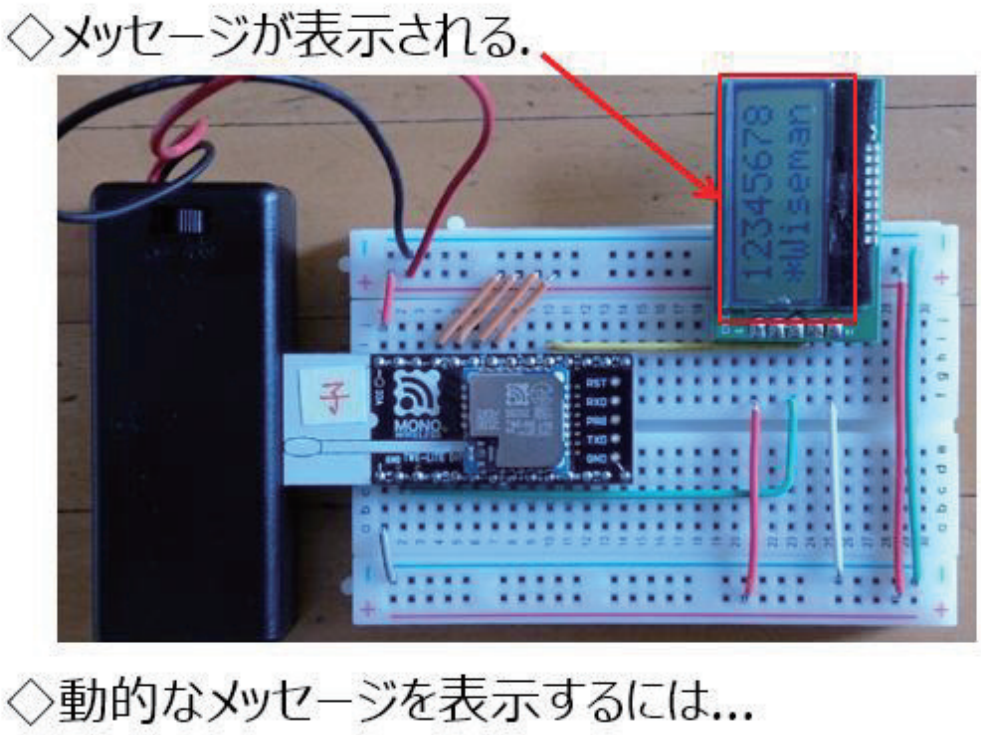


図 142

プログラムの実行はすぐに終わり、少し遅れて LCD にメッセージが表示されます。ここまでできると次に**【動的に変化するメッセージ】**を表示するにはどうすれば良いか、知りたくなります。

第11回　デジタル温度計

第 11 回の講座は、前 2 回分の講座の合体版です。温度を計測し LCD に表示するデジタル温度計です。

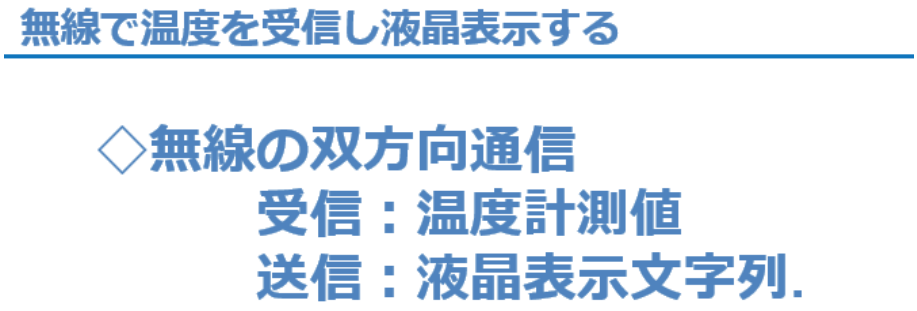


図　143

子機側には、温度センサーと LCD を使います。

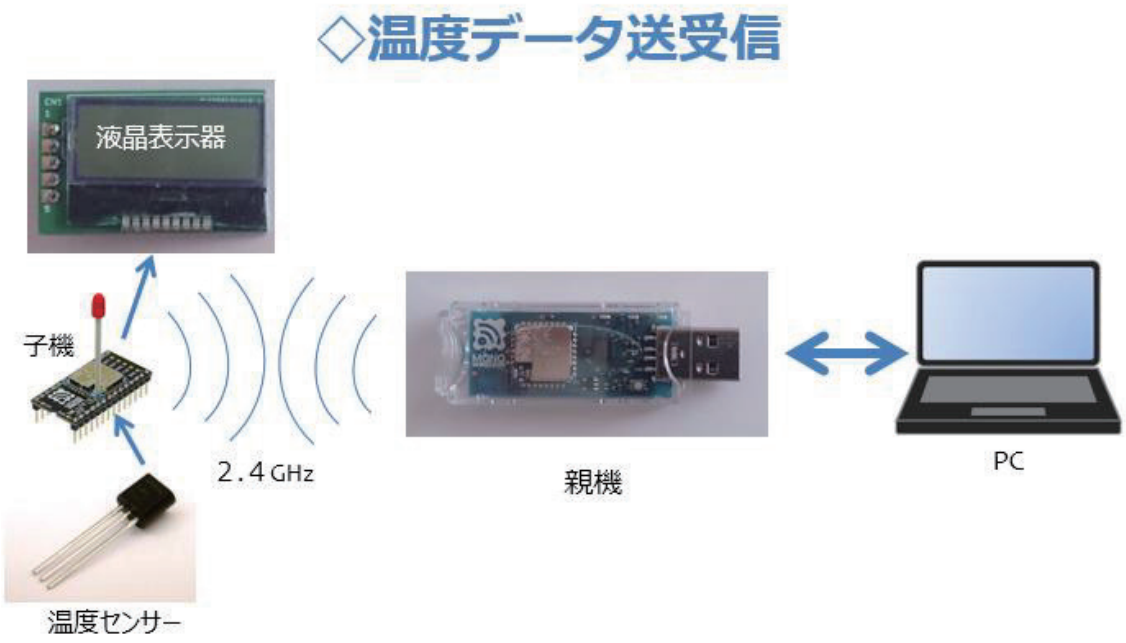


図　144

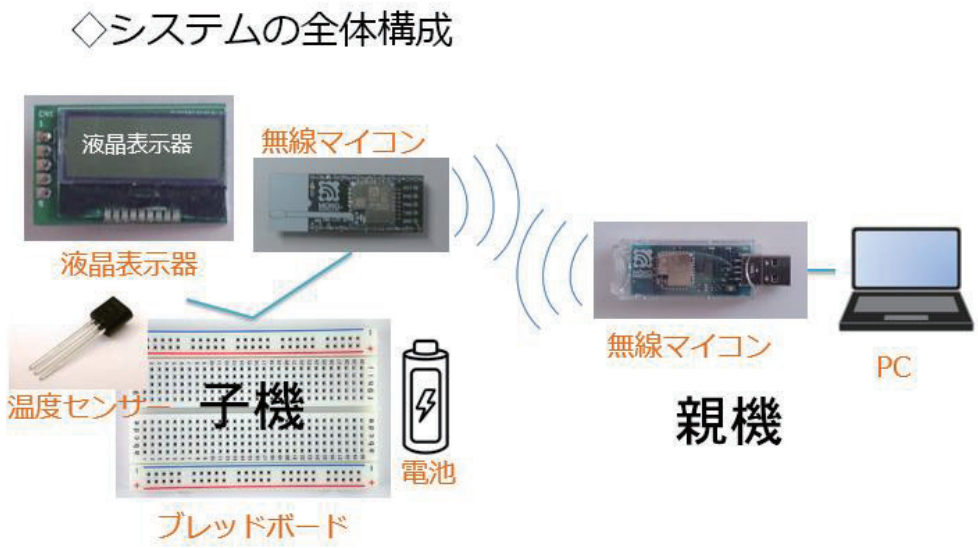


図 145

PC から見ると子機とやり取りするデータは、受信側は温度センサーの計測値、送信側は LCD に表示するメッセージです。

必用なパーツは、次の通りです。

子機側：

1. 無線マイコンモジュール×1 台
2. ブレッドボード×1 個
3. 電池（単 4 乾電池×2 個＋SW 付電池ケース×1 個）×1 セット
4. 配線用ジャンパー線
5. LCD（AQM0802A）×1 個
6. 温度センサー（LM61CIZ）×1 個

親機側：

1. PC×1 台（Windows10）
2. Python 開発・実行環境×1 セット

下図に従って配線を行います。LCD とマイコンのピン接続を枠内に記載してありますので確認をしてください。

◇子機の配線

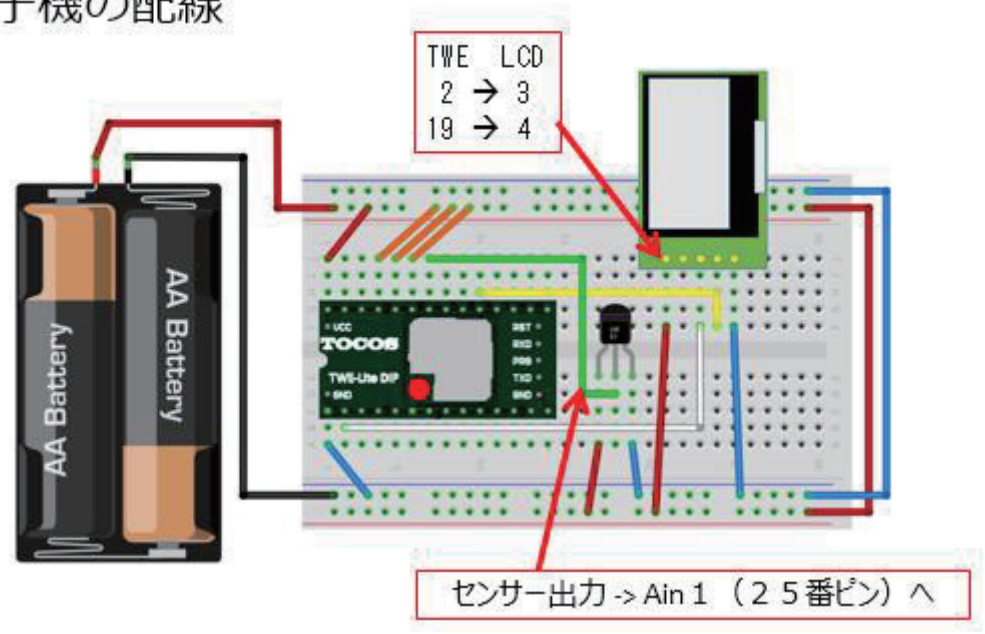


図 146

前 2 回分、第 9 回と第 10 回の子機が合体した回路になっていることが分かりますね。温度センサーと LCD の配線は、わずか 7 本ですから、簡単ですね。配線誤りがないか良く確認をしてください。

実際に配線した様子　子機

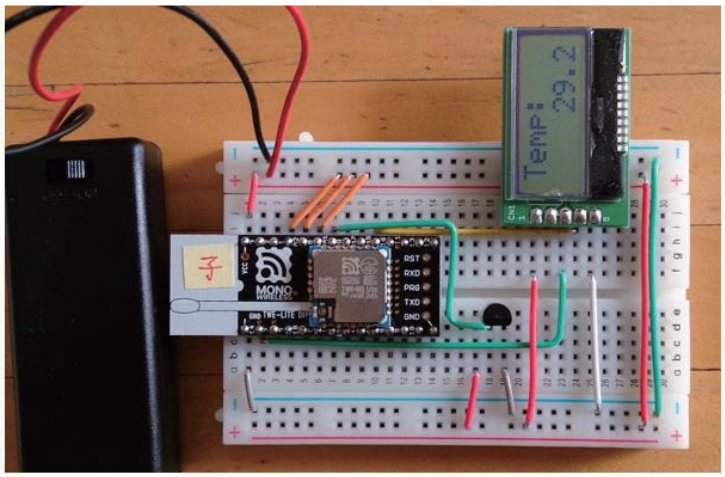


図 147

◇ソースコードを記載しますが、こちらも前 2 回分のものを合体したソースコードになっています。

◇モジュールなどの取り込み

```
import struct       # バイナリ<--->文字列相互変換モジュール
import binascii     # バイナリ<--->ASCII相互変換モジュール
import serial       # シリアル通信パッケージ
import time         # 日付・時刻データを取り扱う
```

◇LCD制御電文送信　序盤

```
# TWE-Liteの標準アプリ内のI2C関数をシリアル通信を経由して制御する.
def accessI2C(s, sendto = 0x78, reqno = 0x00, command = 0x01,
        i2caddress = 0x00, i2ccommand = 0x00,
        data = [], readbyte = -1):

        # データを作成する
        if readbyte == -1:   # 引数が-1のとき、I2Cに書き込むだけ
                # 送信データのリストを作る.
                sendbytes = [sendto, 0x88, reqno,  i2caddress, i2ccommand, 0x00, len(data)]

                # dataを加える
                # dataを1文字ずつリストにする
                buf = list(data)
                # buffを文字コードの数値に変換する
                buff = map(ord,buf)
                # 送信データの後ろに、表示するメッセージの文字コードのリストを追加する
                sendbytes.extend(buff)
        else:   # 引数 > 0 のとき、
                # readbyteだけ読み取る（dataは利用しない）
                # 送信データのリストを作る.
                sendbytes = [sendto, 0x88, reqno,  i2caddress, i2ccommand, 0x00, readbyte]
```

◇LCD制御電文送信　中盤　電文完成～送信

```
# 16進数文字列に変換する
bytelen = len(sendbytes)

ss = struct.Struct(str(bytelen) + "B")
outstring = binascii.hexlify(ss.pack(*sendbytes)).upper()

# TWE-Lite子機に送信する
# チェックサム省略バージョンなので、"X"を追加しておく
s.write(":" + outstring + "X" + "¥r¥n")
```

◇LCD制御電文送信　終盤　子機応答チェック

```
# 応答を待つ
# 10回繰り返す
for i in range(10):
        status = s.readline()
        if status[0:9] == ":" + outstring[0:2] + "89" + outstring[4:8]:
                # 対応する応答結果が戻った
                status = status[1:].rstrip()      # 行頭の「:」と行末の改行を取り除く
                ss = struct.Struct(">BBBBBB")     # バイトデータに変換する
                parsed = ss.unpack(binascii.unhexlify(status[0:12]))
                if status[4]:
                        # I2Cへのアクセスに成功
                        # 戻り、１バイトを返す
                        ss = struct.Struct(str(parsed[5]) + "B")
                        result = ss.unpack(binascii.unhexlify(status[12:len(status) - 2]))
                        return result
                else:
                        # 失敗のときは、偽
                        return False
                break
return False
```

◇受信電文解析処理

◇電文（子機→親機）を解析する関数

```
def denbunKaiseki(data):    # 受信電文を解析する
    if data[0] != ":":
        return False        # 先頭が「:」でなければ、対象のデータではない
    data = data[1:]         # 先頭の「:」を取り除く


    # バイトデータに変換する
    ss = struct.Struct(">BBBBBIBHBHBBBBBBBBB")  # フォーマット文字列に従って
                                                # バイナリデータを読み書きする
    data = binascii.unhexlify(data.rstrip())    # 右端のチェックサムを取り除く
    parsed = ss.unpack(data)                    # 文字列をバイナリとして解釈
                                                # バイトデータの配列として解釈

    # デジタル入力／アナログ入力の値を計算する
    digital = [0] * 4                           # デジタル入力値4ch分
    digitalchanged = [0] * 4                    # デジタル入力変更状態4ch分
    analog = [0xffff] * 4                       # アナログ値4ch分
```

◇受信電文解析処理　4ch分のデータ取り出し

◇デジタル・アナログ各4chあり.

```
for i in range(4):                      # 各4ch分繰り返し
    # デジタル入力
    if parsed[11] & (1 << i):           # バイトデータ配列の11番目が真か？
        digital[i] = 1                  #   真（＝１）ならデジタル入力を１にする
    else:                               # そうでなければ（偽）
        digital[i] = 0                  #   デジタル入力を０にする
    if parsed[12] & (1 << i):           # デジタル入力変更状態が真か？
        digitalchanged[i] = 1           #   真（＝１）ならば１にする
    else:                               # そうでなければ（偽）
        digitalchanged[i] = 0           #   偽（＝０）にする

    # アナログ入力
    if parsed[13 + i] == 0xff :         # アナログ入力値が0xFFならば
        analog[i] = 0xffff              #   max値とする
    else:                               # そうでなければ
        analog[i] = (parsed[13 + i] * 4 + ((parsed[17] >> (2 << i)) & 3)) * 4
                                        #   補正値も含めて元の値を復元する
```

◇受信電文解析処理　結果を【辞書】にする

◇項目にキーを付けた【辞書】.

```
# 結果を返すために、辞書データを作る
result = {
    "from" : parsed[0],                 # 送信元デバイスID
    "lqi" : parsed[4],                  # 受信電波品質
    "fromid" : parsed[5],               # 相手の個体識別番号
    "to": parsed[6],                    # 宛先端末の論理デバイスID
    "timestamp": parsed[7],             # タイムスタンプ
    "isrelay": parsed[8],               # 中継フラグ
    "baterry" : parsed[9],              # 電源電圧
    "digital" : digital,                # デジタル入力値
    "digitalchanged" : digitalchanged,  # デジタル入力変更状態
    "analog" : analog                   # アナログ入力値
}
return result  # 呼出元に戻る
```

◇2行のメッセージをまとめて送信

◇LCDの上段・下段のメッセージをまとめて１回で送信

```
# AQM0802A-RN-GBWに文字列を出力する
def writeAQM0802Msg(s, msg1, msg2):
    # 初期化は、標準アプリ側で行っている

    # 出力文字列を１つにまとめる
    msg = msg1.ljust(8)+msg2.ljust(8)
    # 子機に送信
    accessI2C(s, i2caddress = 0x22, i2ccommand = 0x80, data=list(msg))
    return
```

◇処理の本体

```
s = serial.Serial('COM5', 115200)    # COM5を開く
writeAQM0802Msg(s, "", "")           # LCDを消去する
n = 0                                # 測定回数
avr = 0                              # 平均温度

while 1:   # 繰り返し
    data = s.readline()              # 1行受信
    parsed = denbunKaiseki(data)     # 1行を分析して、項目ごとの値を求める
    t = (parsed["analog"][0] - 600.0) / 10.0   # 電圧を温度に変換する
    print t   # 温度、画面表示

    n += 1    # 加算<---平均の為
    avr += t

    if n >= 10:   # 平均計算
        avr /= n
        n = 0

        writeAQM0802Msg(s, "Temp:", "    %2.1f" % avr)   # データを出力する
        avr = 0

s.close()   # COMを閉じる
```

【重要】上のソースコードで、‘COM5’と書いた部分は、実際に親機を接続した PC での仮想 COM ポート番号に書き換えてください。

◇動作確認

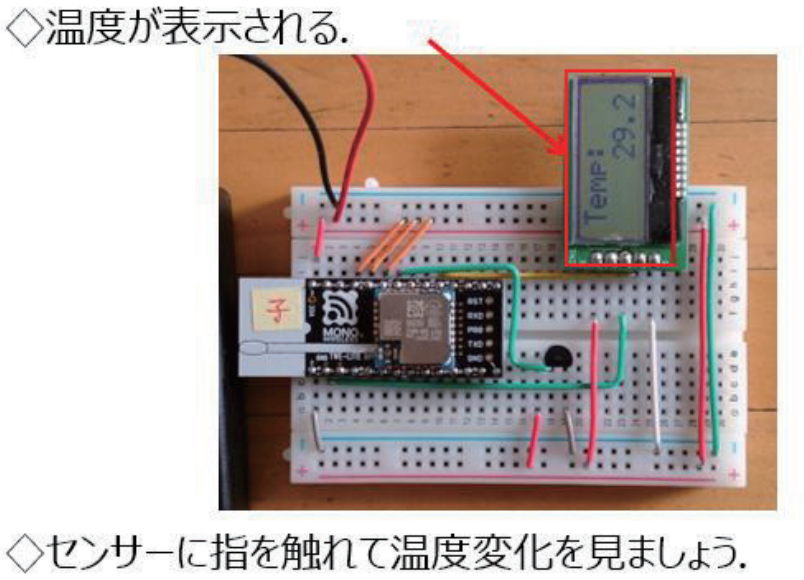


図　148

既に前 2 回の講座で、動作確認の手順は説明しましたので、省略します。必要でしたら第 9 回、第 10 回を参照してください。今回の Python のプログラムは、動作させると複数回計測した平均値を表示するようにしています。

センサーを持つノードが複数あり、色々なところの計測値をまとめて一か所の LCD で表示する。そして、その表示の内容が PC に蓄積されて後に統計､分析などに利用できるようなシステムも開発できそうですね。

【WiFiマイコン(ESP)編】

第1回　LED 点滅

これから使用するマイコンは、WiFi 機能を持つマイコンで、そのまま単独でアクセスポイントに接続できます。これからの一連の講座では、このマイコンの基本的な使い方をトレーニングします。

新しいマイコンを使い始めるときの最初のテーマは、LED 点灯です。まずはこのテーマで新しいマイコンの DO（Digital Output：デジタル出力）を使ってみるのが良いと思います。

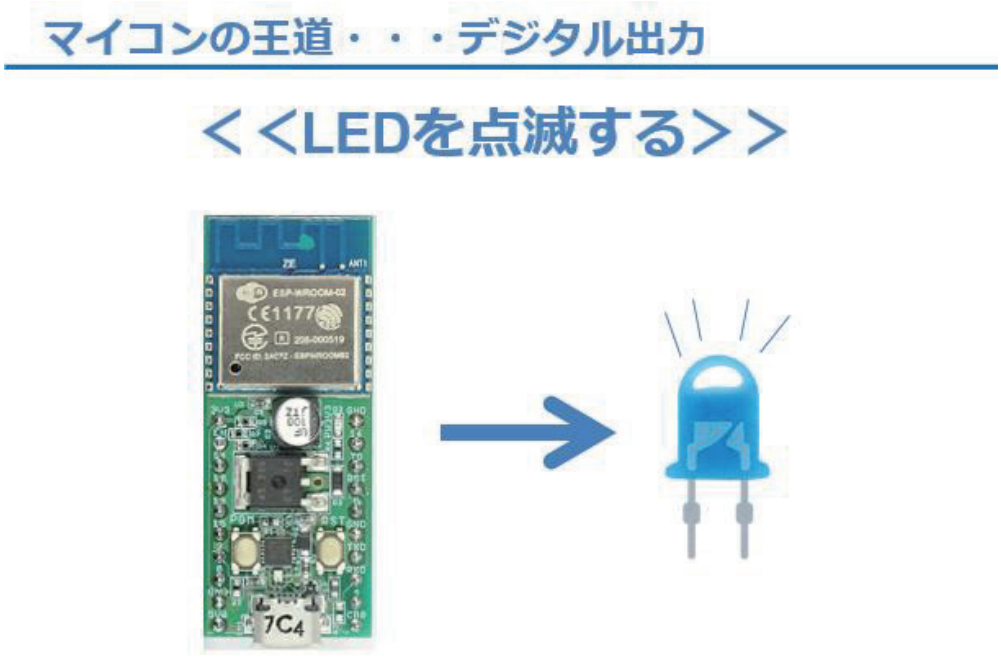


図　149

このマイコンは GPIO という汎用の入出力ポートをいくつか持っています。それを使って LED を点滅させてみましょう。

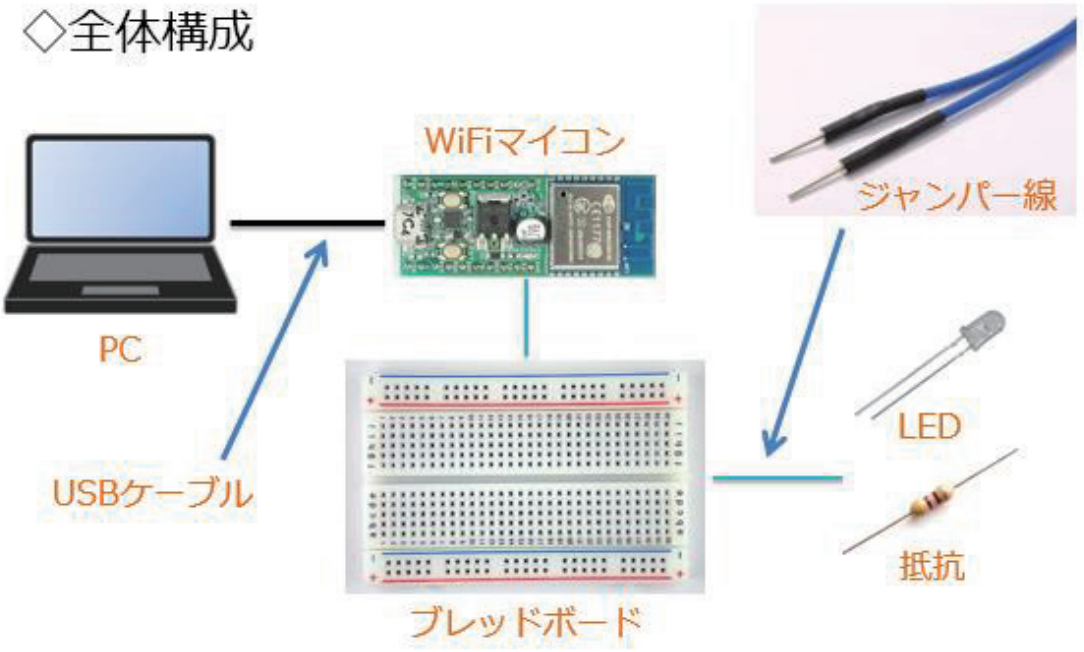
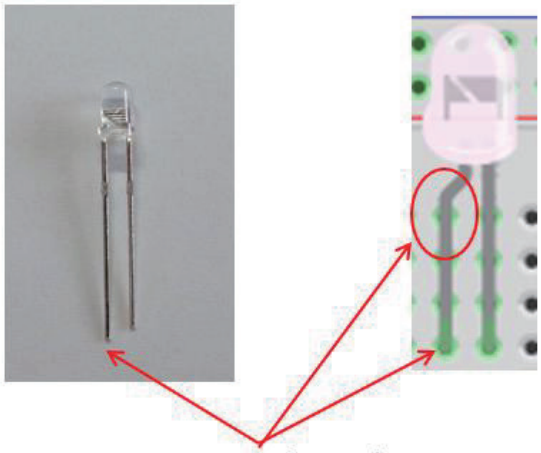


図 150

システムの全体構成を上図に示します。必要な機材・パーツは、下記です。電源は USB ケーブルで PC から供給されます。

1. WiFi マイコン（ESP8266）×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. LED×1 個
7. 抵抗器（470Ω）×1 個

LED は、下図のように足の長さで極性を示しています。長い方の脚から電流が流れ込んで、短い方の脚から流れ出てきます。反対に接続すると電流が流れずに光りません。実体配線図では足の長さが分りにくいので、長い方の脚を曲げて表現しています。



◇図では、長いほうのピンがわかりにくいので
　曲げて表現しています。気を付けて配線してください。

図　151

抵抗のカラーコード

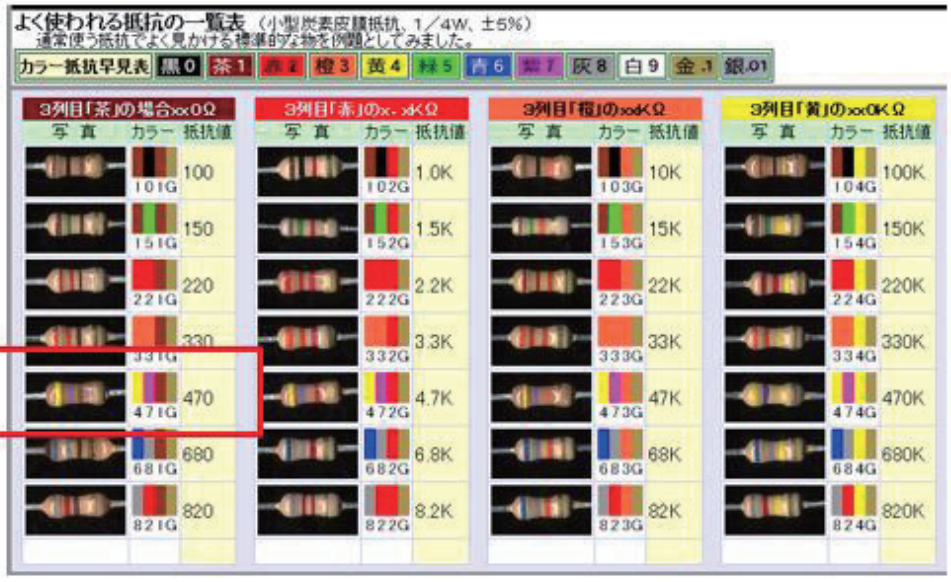


図　152

抵抗器は、カラーコードで抵抗値を示しています。今回使用する抵抗器は 470Ω のもので、LED に電流が流れすぎないようにするために使います。抵抗の両端で足を直角に曲げて使用します。

【重要】

LED に大きな電流が流れると、明るく光りますが、大きすぎる電流が流れると、寿命が短くなってしまいます。LED の最大定格電流としては 20~30mA の物が多く使われています。この最大定格電流の半分程度で、視認できる明るさが確保できますので、オームの法則に従って、E=5V,R=470Ω で計算すると、A=10.6mA となり、最大定格以下となります。この抵抗は**【電流制限抵抗】**と言って、水道でいうとバルブに相当します。バルブの絞り具合が抵抗値ということです。

◇抵抗は写真のように足を曲げて使います。
◇抵抗の値を書いたものを付けておくと、間違えにくくなります。

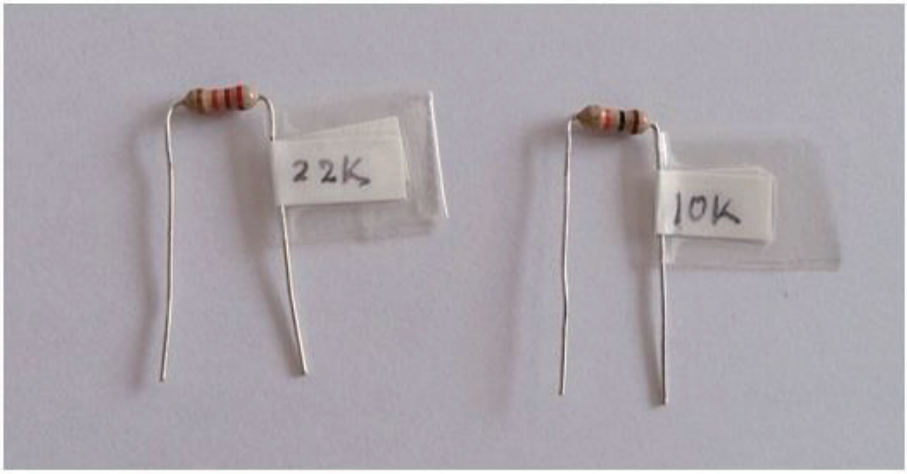


図 153

上の写真ように紙片に抵抗の値を書いたものをテープなどで貼っておくと、値の違う抵抗を間違いなく利用できると思います。回路を作成する前に、このような準備を念入りに行うことは、後の作業の効率化や誤り排除などの面で、とても効果があります。

実習キット【Appendix 参照】にはすべてのパーツがそろっています

が、個別に求めたパーツを利用する場合と同様に、極性や抵抗値などを十分に確認してから使用して下さい。

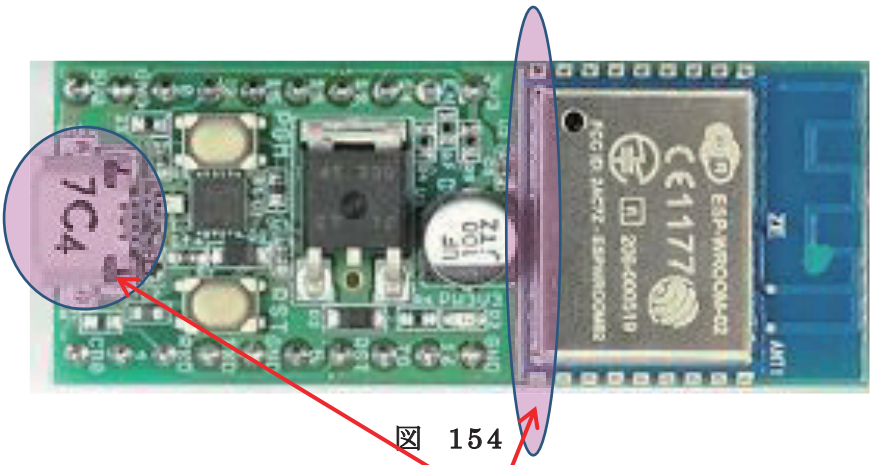
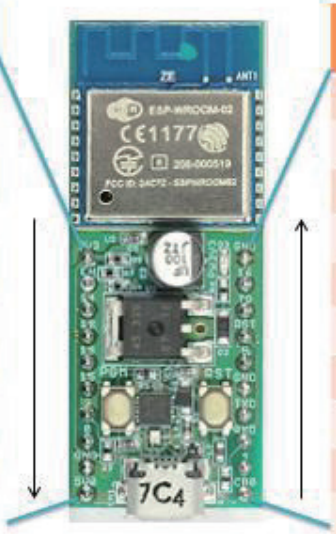


図　154

WiFi マイコンモジュールは写真の矢印で示す部分（マイコンモジュールと基板の境界部分と microUSB コネクタの部分）が弱いので、ブレッドボードへの取付け・取り外し（特に取り外し）と microUSB コネクタの挿抜時（特に抜くとき）には強い力が掛からない様に、注意して作業を行ってください。ピンセットなどの工具を準備しておくと良いでしょう。

番号	名称	機能
1	3V3	3.3V(In/Out)
2	EN	イネーブル端子
3	14	GPIO14
4	12	GPIO12
5	13	GPIO13
6	15	GPIO15
7	2	GPIO2
8	0	GPIO0
9	GND	GND
10	5V0	5V In



番号	名称	機能
20	GND	GND
19	16	GPIO16
18	TO	ADC（max1V）
17	RST	リセット
16	5	GPIO5
15	GND	GND
14	TXD	TXD
13	RXD	RXD
12	4	GPIO4
11	CB0	通信モニタ用

図　155

上図は、WiFi マイコンモジュールのピン配置を示しています。

WiFi マイコンモジュールのアンテナパターンを上に向けて、左上のピンから 1,2,3,4・・・と番号が付いています。

----------<< 各ピンに配置されている信号 >>----------

1番：3.3Vの電源で駆動するときは、このピンに直接3.3Vを接続します。USB経由または、外部5Vで駆動する場合は、このピンから3.3Vの電源を取り出すことができます。

2番：イネーブル端子で、このモジュール全体の有効・無効を設定します。通常は解放で有効です。無効（Disable）とするときは、GNDに接続します。

3～8番：汎用入出力端子です。デジタル入出力を取り扱います。

12,16,19番：同上。

9,15,20番：GND

10番：外部5Vの入力。【注意】外部への5V取り出しや3.3V入力との併用はできません。

11番：USB-シリアルI/F ICの設定により、通信状態の表示などに使用します。通常は使用しません。

13番：RXDは、シリアル通信の受信信号です。

14番：TXDは、シリアル通信の送信信号です。

17番：RSTはこのマイコンモジュールのReset信号です。

18番：TO（TOUT）は、アナログ電圧を計測出来るADC（A/D Converter）が接続されているポートです。最大1Vまで測定できます。

いよいよ無線マイコンモジュールの実験回路を配線します。**配線は必ずUSBケーブルを外した状態で行います。**配線と言っても線の数や使用するデバイスの数が少ないので、じっくりと確認しながら行ってもすぐに終わります。各パーツには、くれぐれも余計な力がかからないようにして、回路の配線を行ってください。手順としては、半田付けする基

盤の場合は、**【高さの低いものから順に基板に取り付けてゆく】**のが王道ですが、ブレッドボードとジャンパー線による配線なので、作業しやすいと思った順に行えばよいでしょう。回を重ねれば、自ずと最適な作業手順が身についてくると思います。

まず、ブレッドボードを小さい数字が左、アルファベットの A が手前になるように置いて、下図の様に配線をしてください。WiFi マイコンモジュールは、microUSB コネクタが左側に向いています。

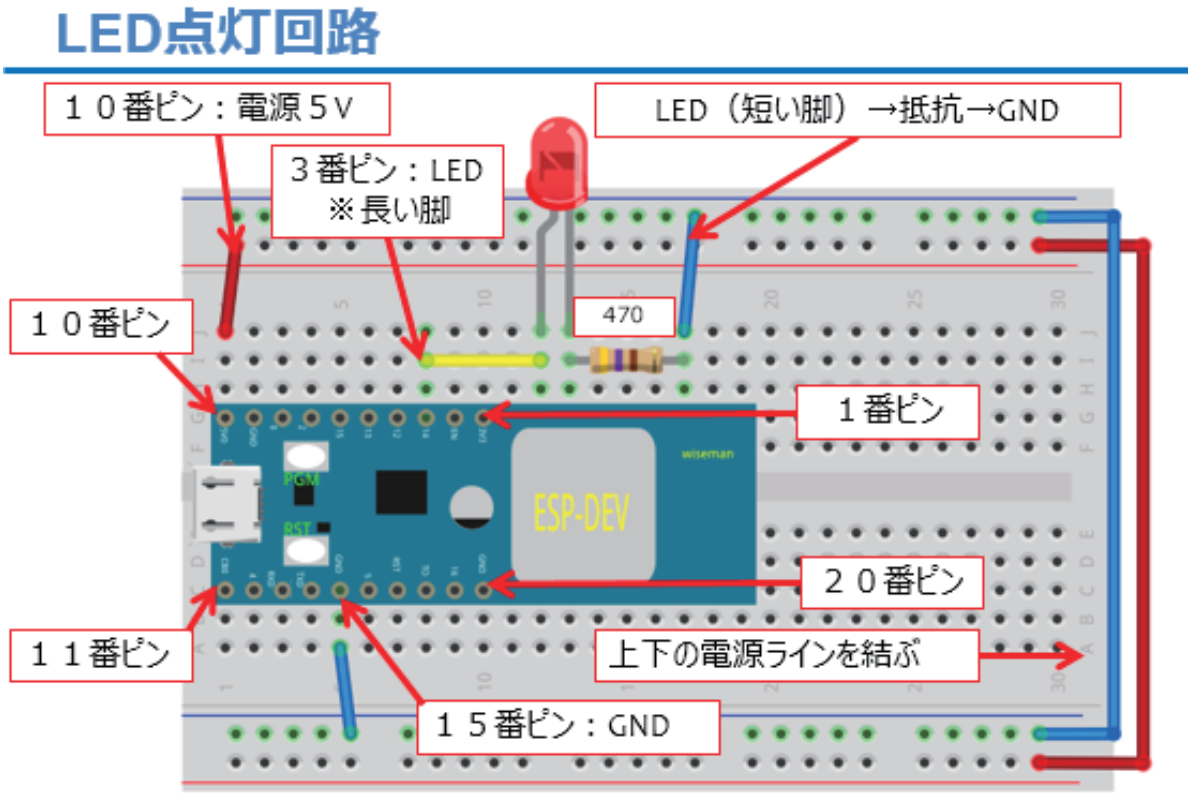


図　156

使用しているジャンパー線の色は、特に指定ではありませんが、電源（＋）側は赤系統、GND（－）側には青や黒を使うことが多いので、覚えておくと別の回路を見たときに役立ちます。WiFi マイコンモジュールの 10 番ピンから 5V を取り出し、15 番ピンは GND に接続しています。ブレッドボードの上下の電源ラインは、一番右側で上下をジャンパー線

で接続しています。3番ピンからLEDの長い方の脚に配線して、LEDの短い方の脚は470Ωの抵抗を介してGNDに接続します。このとき、抵抗の脚をそのままGNDに配線してもかまいません。そのようにして配線に使用するジャンパー線が少なくなると、基板上がすっきりして、全体を良く見る事ができるようになります。

GPIO14をHigh（=1）にすると、3番ピンから電流がLEDの長い方の脚に流れ込み、LEDを点灯させて、短い方の脚から470Ωの抵抗を経由してGNDに流れ出るという回路です。

実際に配線した様子が次の写真です。

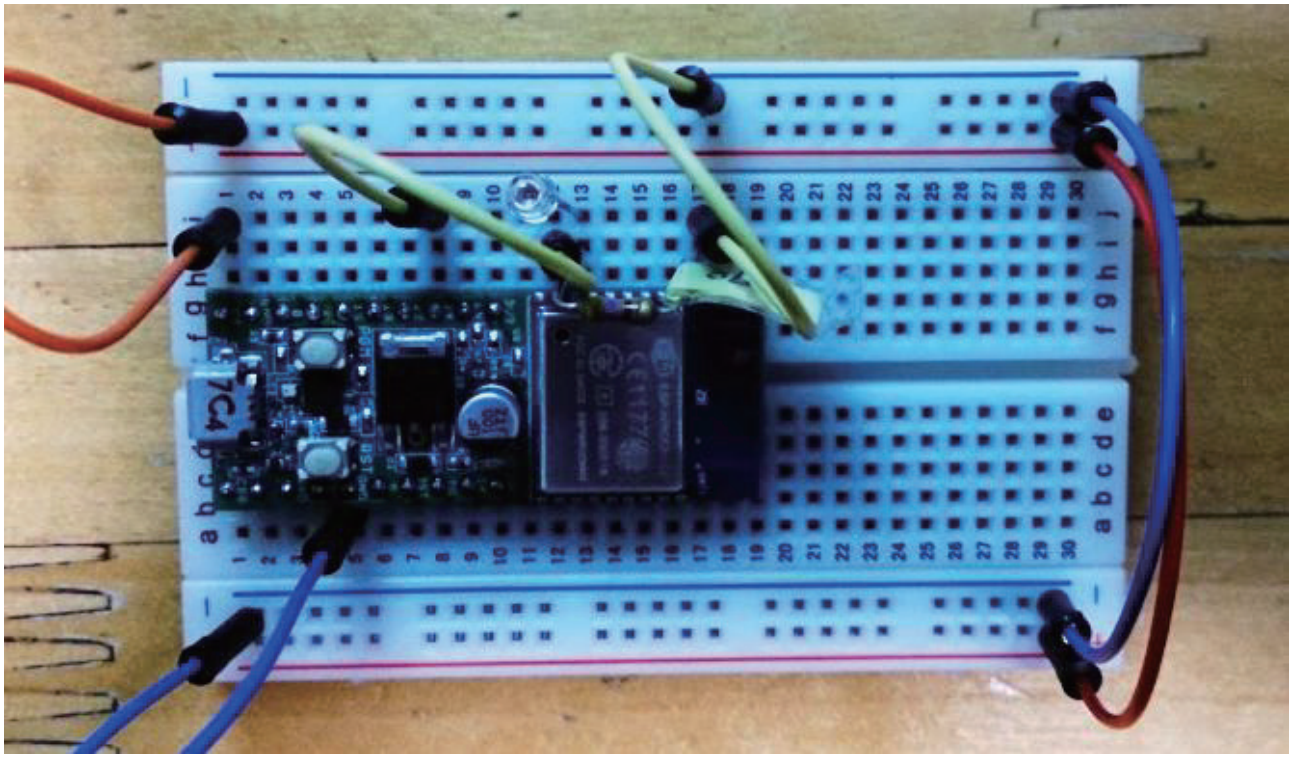


図 157

Arduino IDE

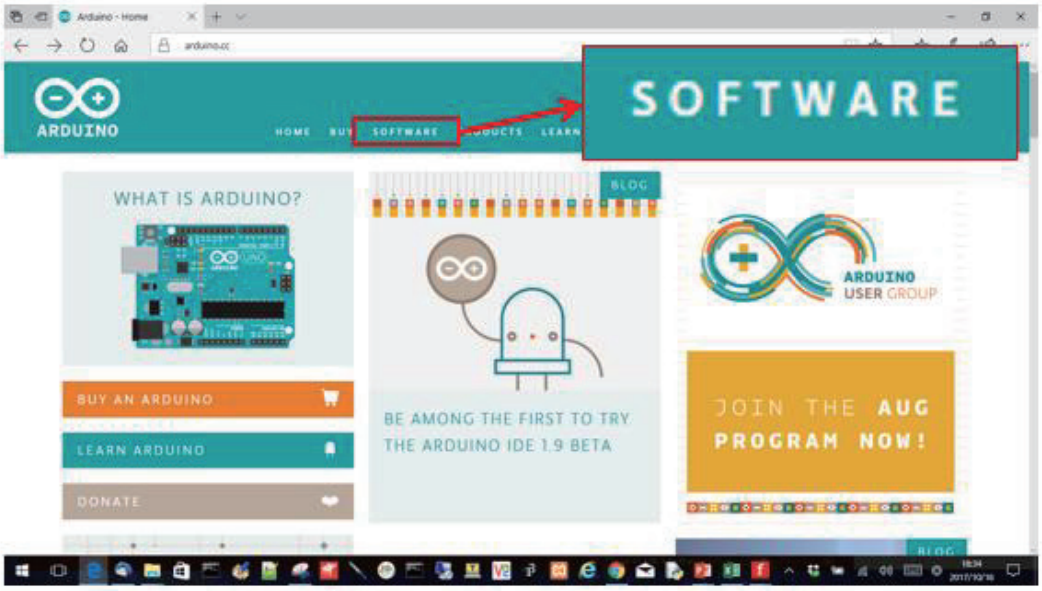


図　158

ソフトウエア開発には Arduino IDE を使用します。WEB で https://www.arduino.cc/を指定すると上のページが見つかります。ページ上部の SOFTWARE をクリックすると次のページに移動します。

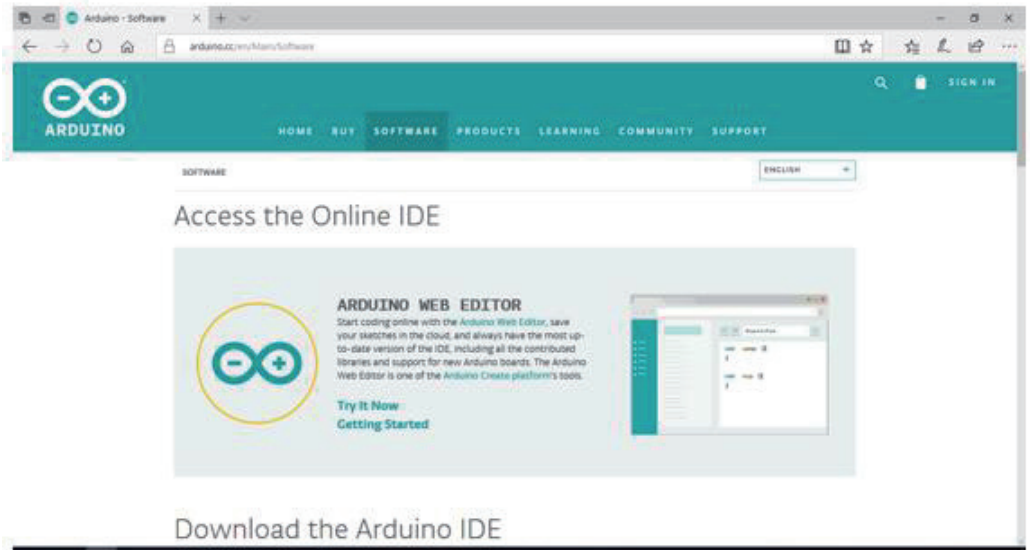


図　159

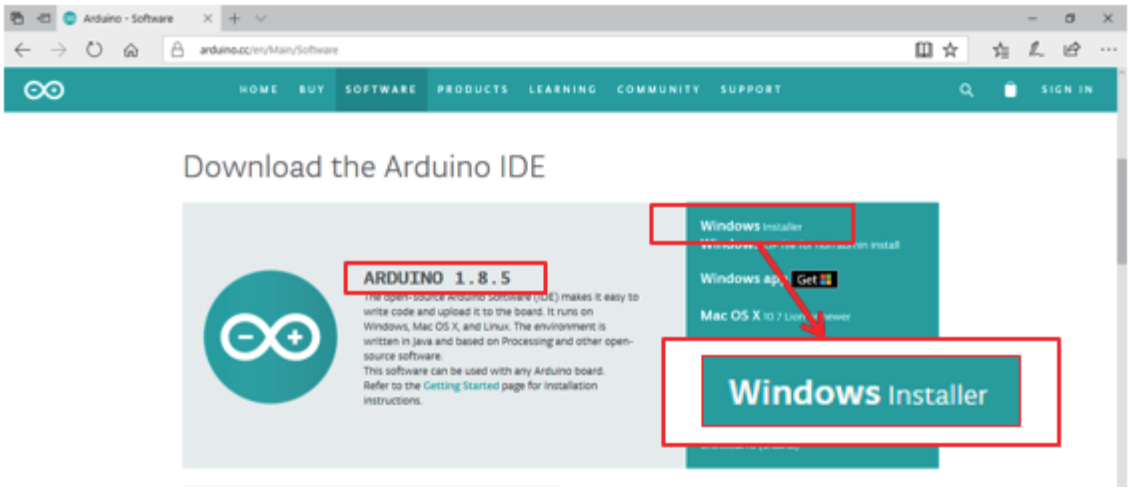


図　160

さらに、すこし下に移動すると上図のページになります。ここで、Windows Installer をクリックすると、下図のページに移動します。

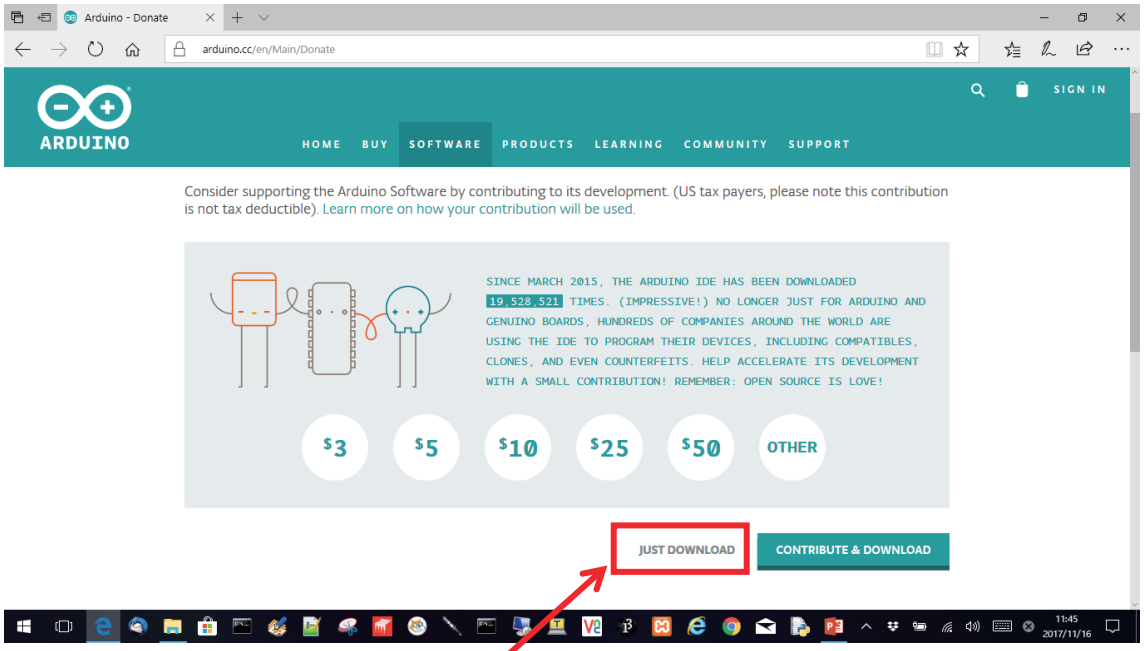


図　161

ここでは、JUST DOWNLOAD を選択します。適当なフォルダにダウンロードファイルを置き、そのファイルをダブルクリックして Install します。途中で USB-シリアルドライバなどの Install 確認メッセージが

表示されることがありますが、**全て Install してください**。また Java が通信するという確認のメッセージが表示されることがありますが、**全ての通信を OK（またはチェック）してください**。Install が終わるとデスクトップに下図左のような眼鏡マークのショートカットができています。それをダブルクリックすることで、下図右のような Arduino IDE が起動されます。プログラムソースコードは、ウインドウ中央の白い部分に記述します。

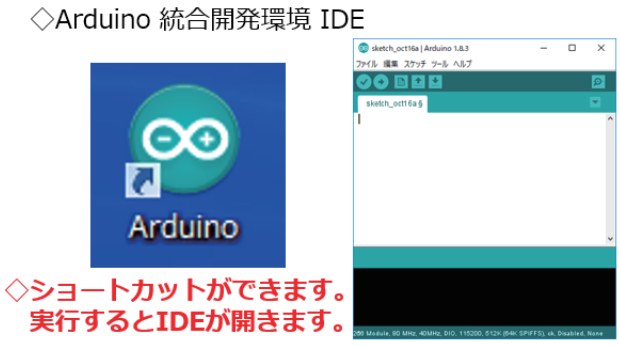


図　162

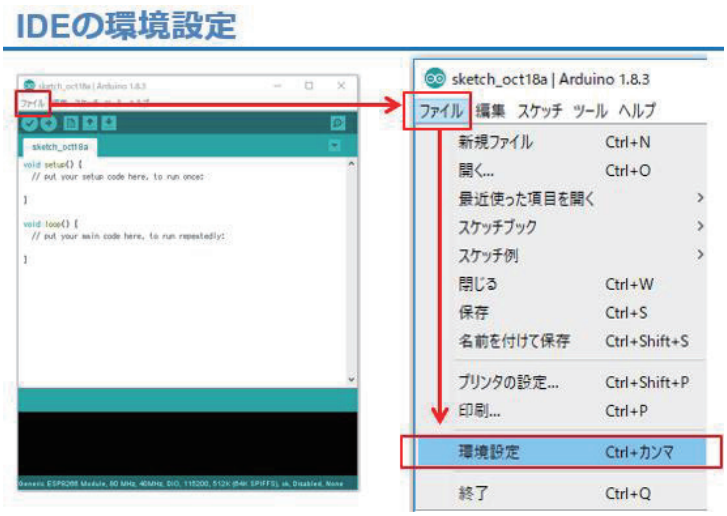


図　163

これから使用する WiFi マイコンモジュール専用のライブラリなどを準備します。上図で、**【ファイル→環境設定】** と辿ると、次のウインドウが開きます。

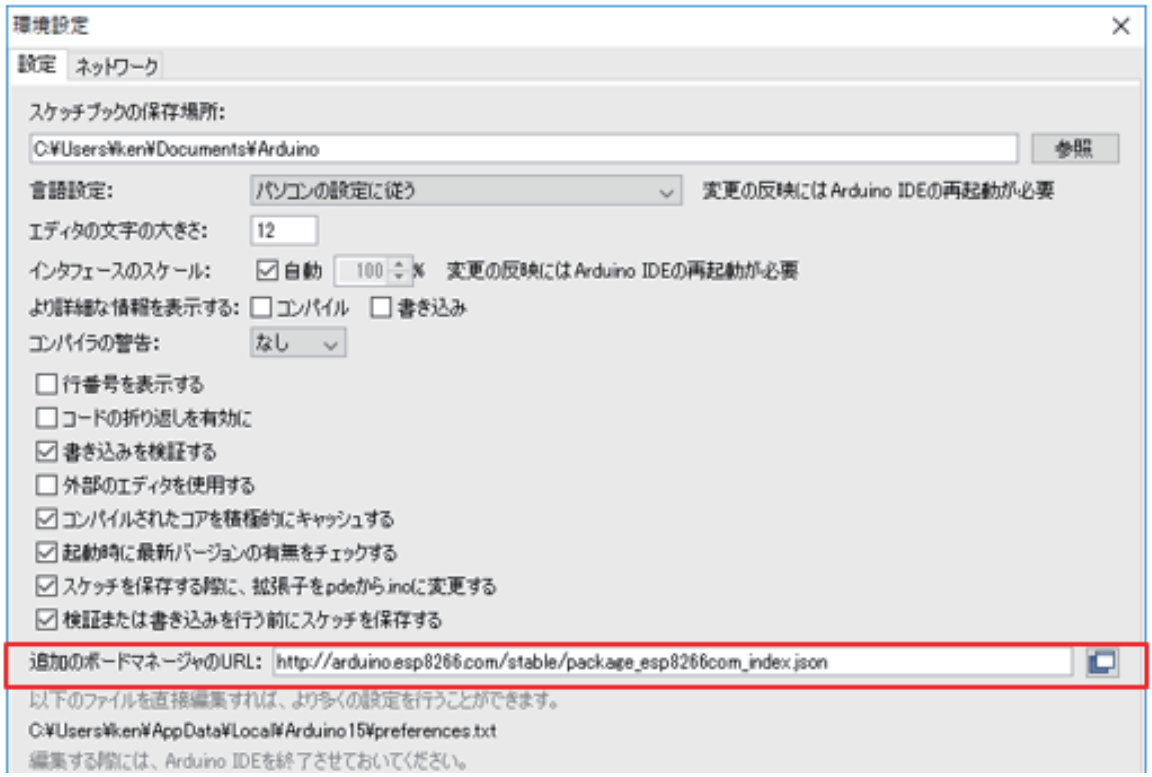


図　164

ウインドウ下部にある【追加のボードマネージャの URL】に次の URL を入力して、OK ボタンを押してください。

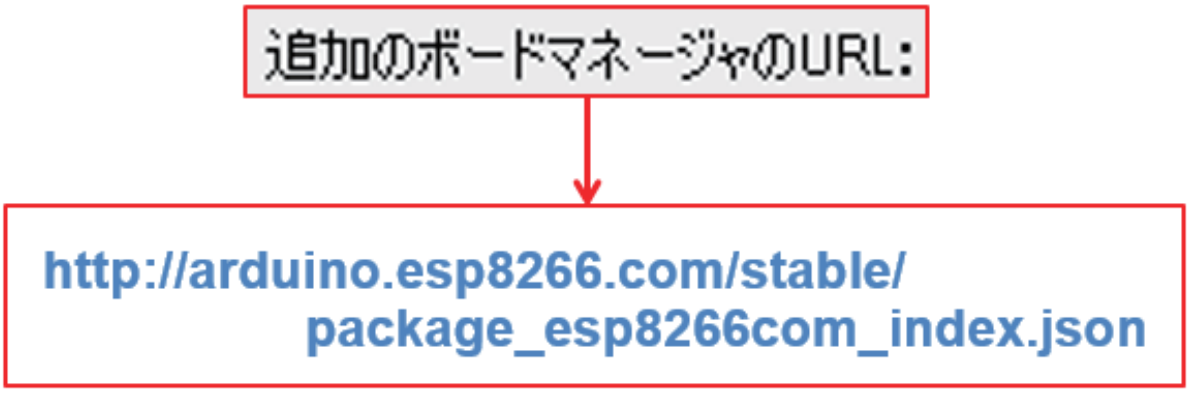


図　165

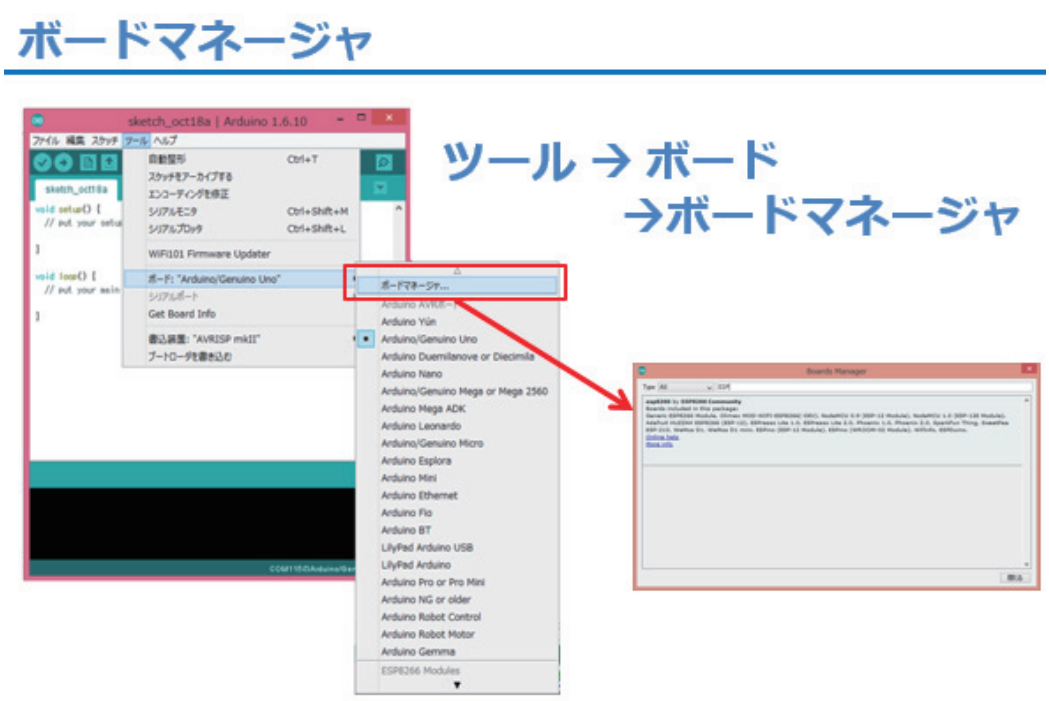


図　166

【ボード→ボードマネージャ】と辿り、ボードマネージャのウインドウを開きます（下図）。

図　167

右上にある検索フィールドに ESP と入力し、表示される esp8266 by ESP8266 Community を選択し、Install します。（下図）

図　168

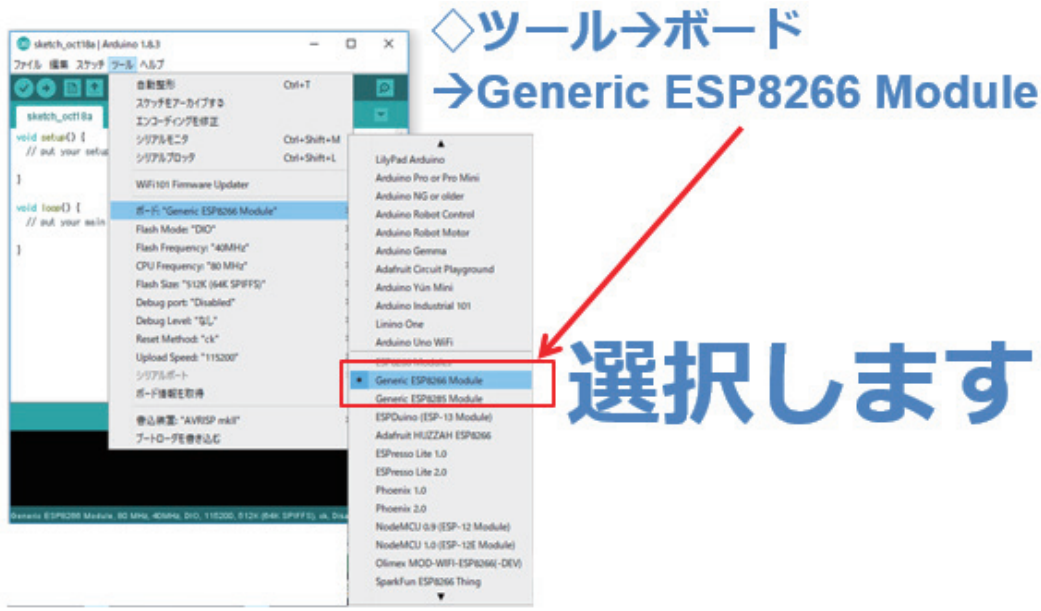


図　169

元のウインドウに戻り、【ツール→ボード】と辿り、一覧の中から、Generic ESP8266 Module を選択します。

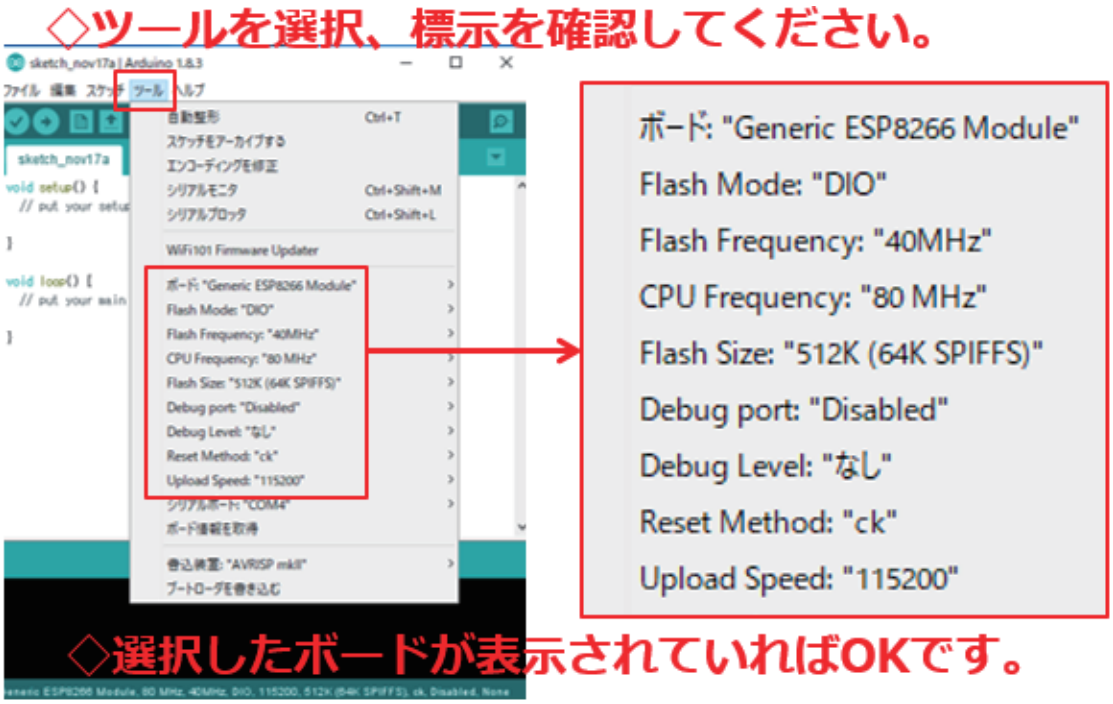


図　170

ツールで選択したボードが表示されていれば OK です。（上図）

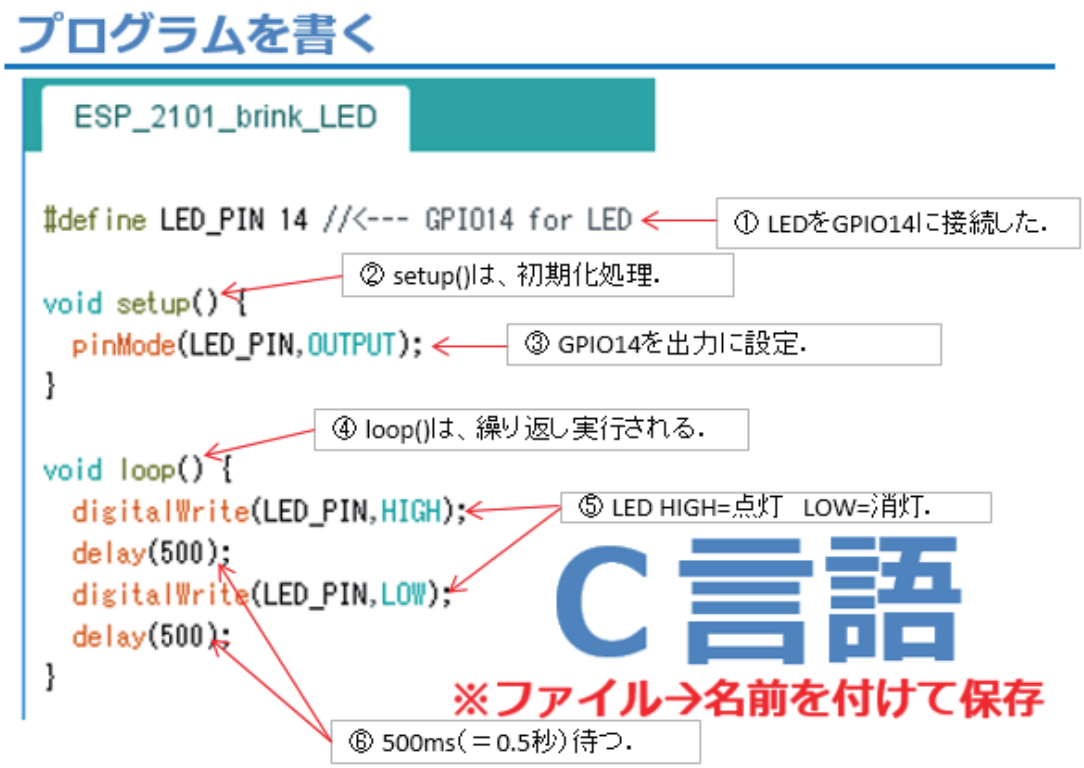


図　171

IDEウインドウ中央の背景が白い部分がソースコードエディタになっていますので、そこにプログラムを記述します。

【重要】

基本的にArduinoマイコンと同様の言語体系となっています。Arduino言語はC/C++をベースにして、C言語のすべての構造と、いくつかのC++の機能をサポートしています。

今回のLED点滅プログラムは、上図の通り10行程度ですから、容易に入力できるでしょう。もしコメントを入力する場合は、スラッシュ2つ【//】やスラッシュ＋アスタリスク【/*】とアスタリスク＋スラッシュ【*/】を利用します。詳細はCまたはC++言語の仕様を調べて下さい。

特に、今回はGPIO14にLEDを接続したので、①の部分で定義しています。また②setup()は、どのプログラムでも共通の初期化処理をまとめて記述する関数として名称が決まっています。④loop()は、繰り返し呼び出される関数で通常の処理を全てここで行います。(※割込みなどは別途記述するのですが、この講座では割込みを使用しないので、別の機会に解説をしたいと思います。) ③pinMode()は、LEDへの制御用としてマイコンの汎用入出力GPIO14を出力に設定しています。
⑤digitalWrite()は、GPIOポートに（※このマイコンでは入出力や通信の為にアクセスする部分をポートと呼びます。）HIGH(=1)かLOW(=0)を書き込みます。これにより、LEDが点灯(GPIO=1)したり消灯(GPIO=0)したりします。⑥deley()は、指定の時間、プログラムを一時停止します。単位はms（1/1000秒）です。

初期化が行われたマイコンシステムは、LEDへの出力が設定されて、loop()が繰り返し実行されることにより、0.5秒ごとにLEDが点滅を繰り返すという動作をします。

ソースコードを入力したら、名前を付けて保存してください。以後このIDEでは、ソースコードは**【スケッチ】**という名称で管理されます。

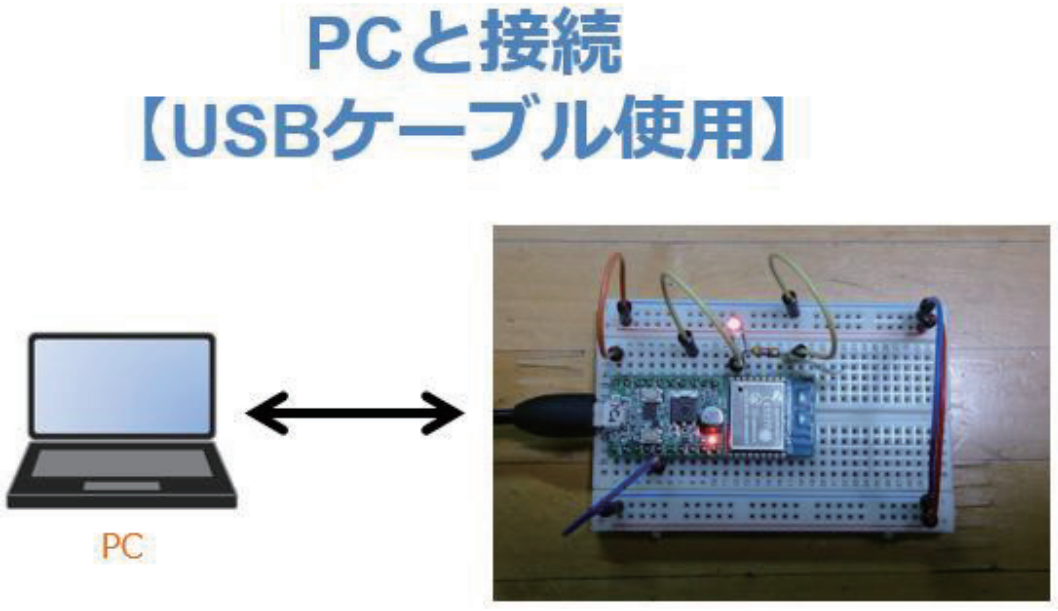


図 172

プログラムを保存したら、USBケーブルでPCとWiFiマイコンモジュールを接続します。

【注意】

上図右の写真では、LEDが2つ点灯していますが、これは分かり易いように点灯しているところを撮影したものです。実際は、WiFiマイコンモジュールの金属ケース付近にある小さな電源LEDだけが赤く点灯して電源が投入されていることを示します。

COMポート番号確認

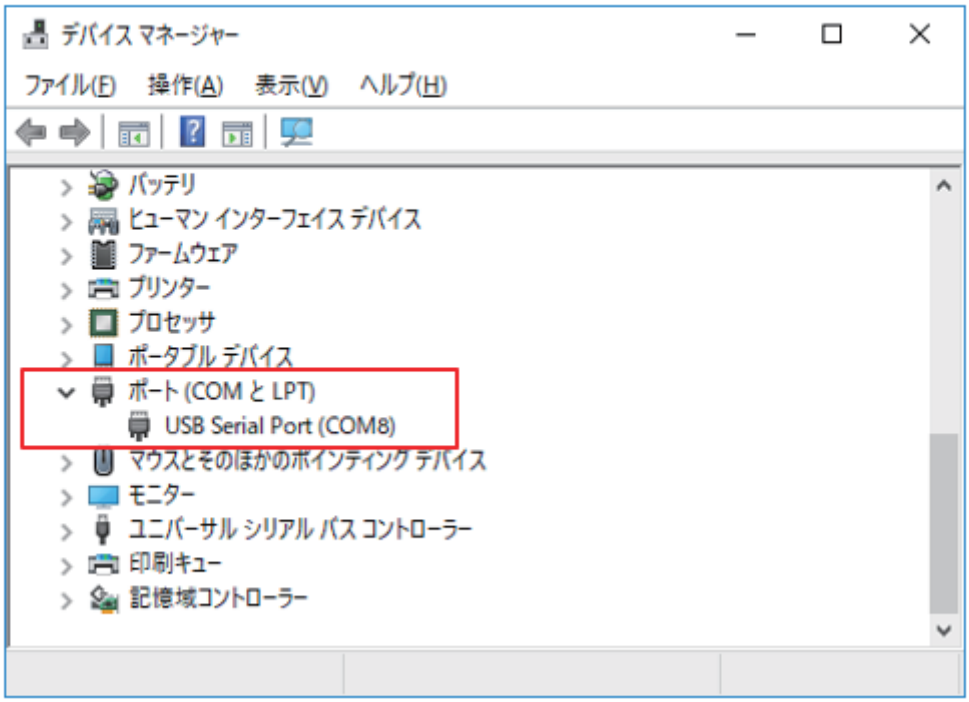


図 173

初めて、USB ケーブルで PC と WiFi マイコンモジュールを接続したとき、仮想 COM ポートドライバーのインストールが行われますので、終了するまで待ちます。終わりましたら、デバイスマネージャのウインドウで USB Serial Port の COM 番号を確認してください。ドライバーがインストールされない場合は、次頁の【注意】を参照して下さい。

【重要】

ここで確認した COM ポート番号は、他の WiFi マイコンモジュールを接続すると、異なる番号に設定されます。PC と接続するのが初めて（または、かなり時間が経過している）であれば、デバイスマネージャで番号を確認してください。同じものを繰り返し使用するときは、COM ポート番号は変わりません。

シリアルポートの設定

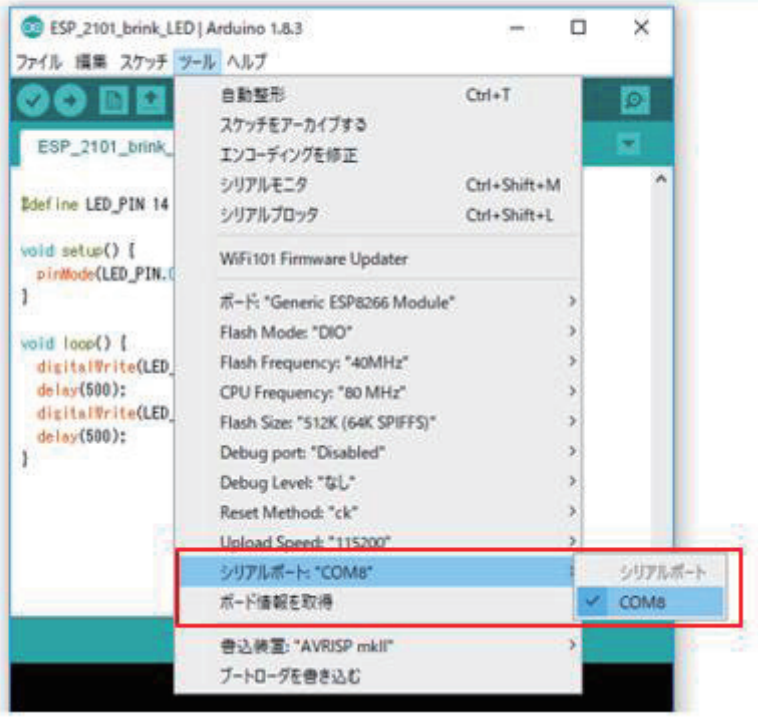


図 174

【ツール→シリアルポート】と辿り、確認した COM ポート番号を選択して、チェックを入れます。(上図)この番号は、Arduino IDE と WiFi マイコンモジュールが通信する場所(シリアルポート)です。一度設定すると次回以後も有効になっているのですが、WiFi マイコンモジュールへのプログラム書き込みが巧くいかない場合などは、確認をして再度設定することもあります。

次は WiFi マイコンモジュールへの書込み準備です。

【注意】仮想 COM ポートドライバーがインストールされていない場合は、FTDI 社(http://www.ftdichip.com/Drivers/VCP.htm)にある VCP ドライバーをインストールして下さい。

◇重要：【これからプログラムを書込むゾ！！】

書き込み可能にする

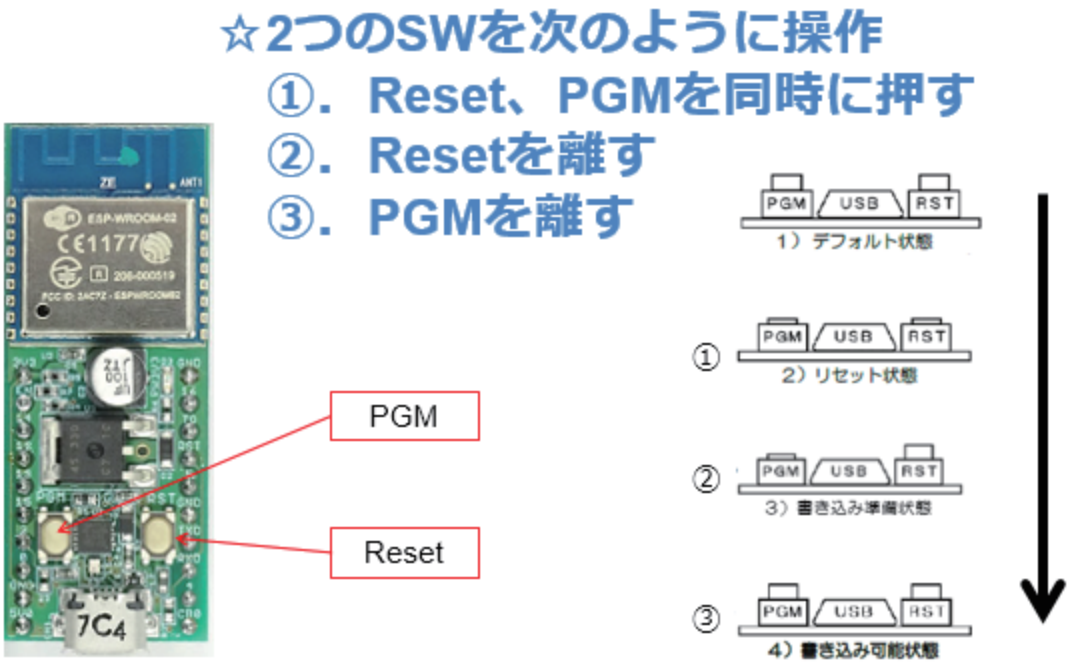


図　175

WiFiマイコンモジュールは、通常は書き込まれているプログラムを実行していて、IDEからの書込みに対して、なにも反応しませんので【これからプログラムを書込むゾ！！】と教えてやらなければいけません。その際、WiFiマイコンモジュール上の2つのSWを使います。

その手順は次のとおりです（上図）。

① まず、PGMとReset SWを同時に押します。

② 次に、Reset SWだけ離します。

③ 最後にPGM SWを離します。

これで、WiFiマイコンモジュールは、IDEからのプログラム書込みモードになります。

◇プログラムのコンパイルと書込み

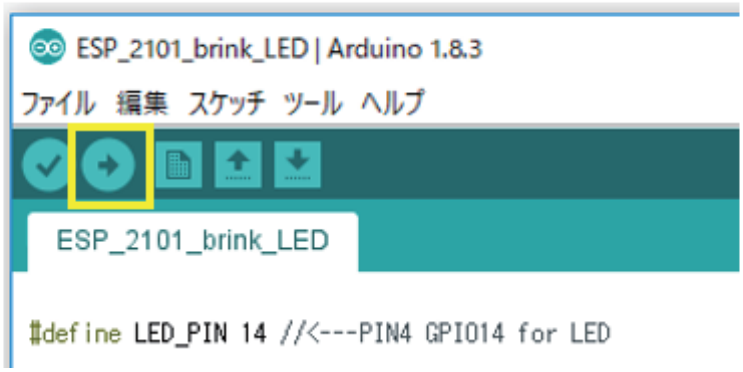


図 176

これから、プログラムのコンパイル・リンクと WiFi マイコンモジュールへの書込みを行います。IDE の上の方にある、右向き矢印ボタンをクリックしてください（上図）。

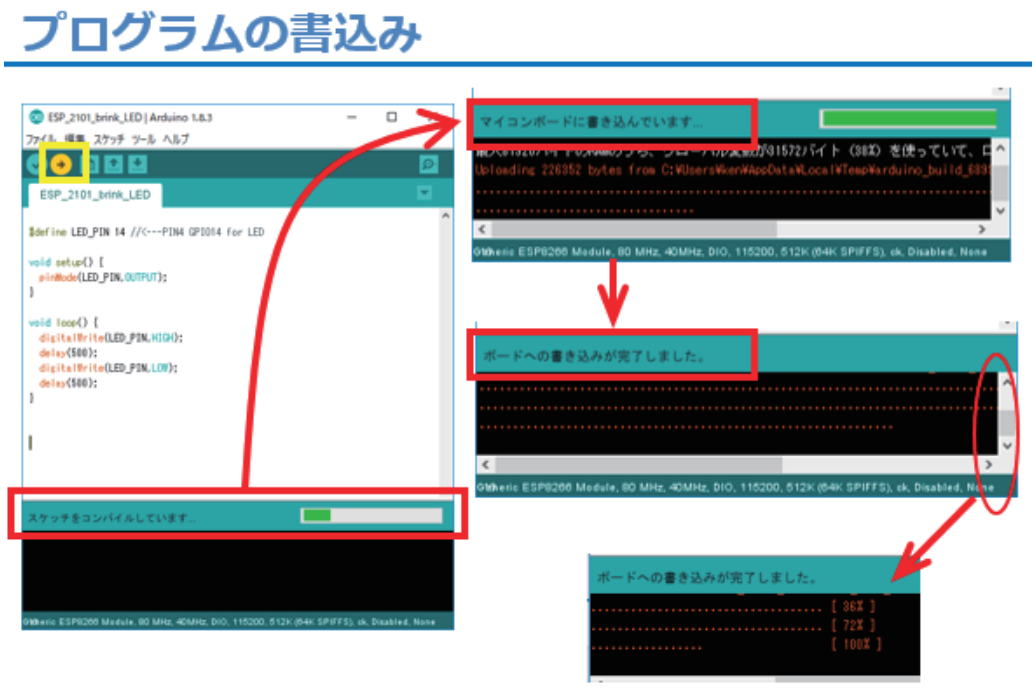


図 177

ボタンを押すと上図の様に、コンパイルから書込みまで、一連の流れで実行されて、最後に書込み完了のメッセージが表示されます。

動作の様子

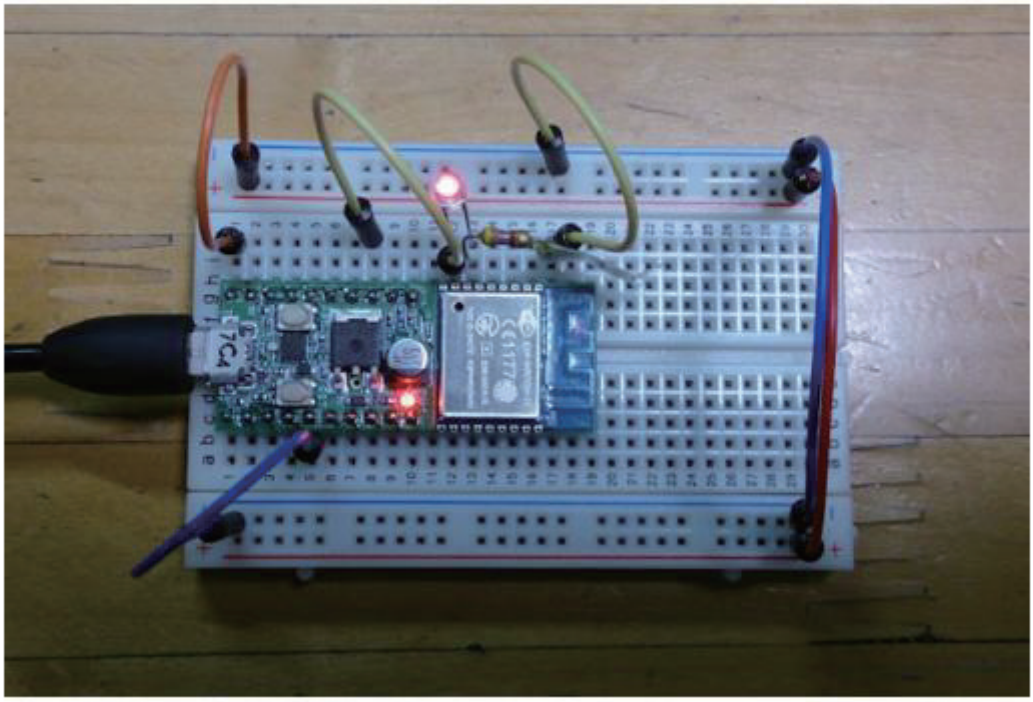


図 178

書込みが終了すると、マイコンは自動で Reset されて、プログラムの実行が開始されます。配線とソースコードの記述に間違いが無ければ、配線した LED がおよそ 0.5 秒間隔で点滅を繰り返します。

これが、第 1 回目の実験の結果です。LED は小さいので、これを点滅させることの意味を理解しにくいのですが、LED の ON/OFF 信号を取り出して外部機器・装置の ON/OFF 制御が行えます。また ON/OFF の周期を短くして制御すると PWM（Pulse Width Modulation）などに対応できます。制御対象が AC（交流）でしたら、リレーや SSR（Solid State Relay：ソリッドステートリレー）を使うことにより、ON/OFF 制御できます。

◇<< WiFi マイコン開発手順 >>

最後に、WiFi マイコンモジュールを使った開発全体の手順を振り返っておきましょう。

----------<< WiFi マイコン開発手順 >>----------

1. 回路の制作
2. IDE のダウンロード・インストール
3. ボードマネージャのダウンロード・インストール
4. WiFi マイコンモジュールのソースコード記述
5. PC と WiFi マイコンモジュールの接続（USB ケーブル）
   ※この際、USB-シリアルドライバがインストールされる
6. COM ポート番号の確認と IDE への設定
7. コンパイル・リンク・書込み
8. 動作確認

上の手順は、初めて WiFi マイコンモジュールのシステム開発を行う際の手順です。次の開発からは、2，3 の手順が省略できます。使用する WiFi マイコンモジュールが同じものであれば 6 の手順も IDE の設定を確認するだけで大丈夫です。別の WiFi マイコンモジュールを使用する場合は、同じシステムを作る場合でも COM ポート番号が変わります（前述）ので 6 番の手順を行います。また、開発対象のデバイスやシステムによっては、専用のライブラリをインストールする必要がありますので、その場合は、手順 3 と 4 の間にライブラリをライブラリマネージャでインストールします。今後の実験でも、概ね必要な手順は説明しますので、全体は上の 1～8 の手順だと理解しておいてください。

この手順は、今後も必要に応じて参照してください。

第2回　SW

第１回はLEDの点滅でした。これはLEDに対して一方的にWiFiマイコンからON/OFF信号を出力していました。第２回はSWの状態を読み込んで、その状態に対応してLEDが点灯・消灯するシステムを作りましょう。

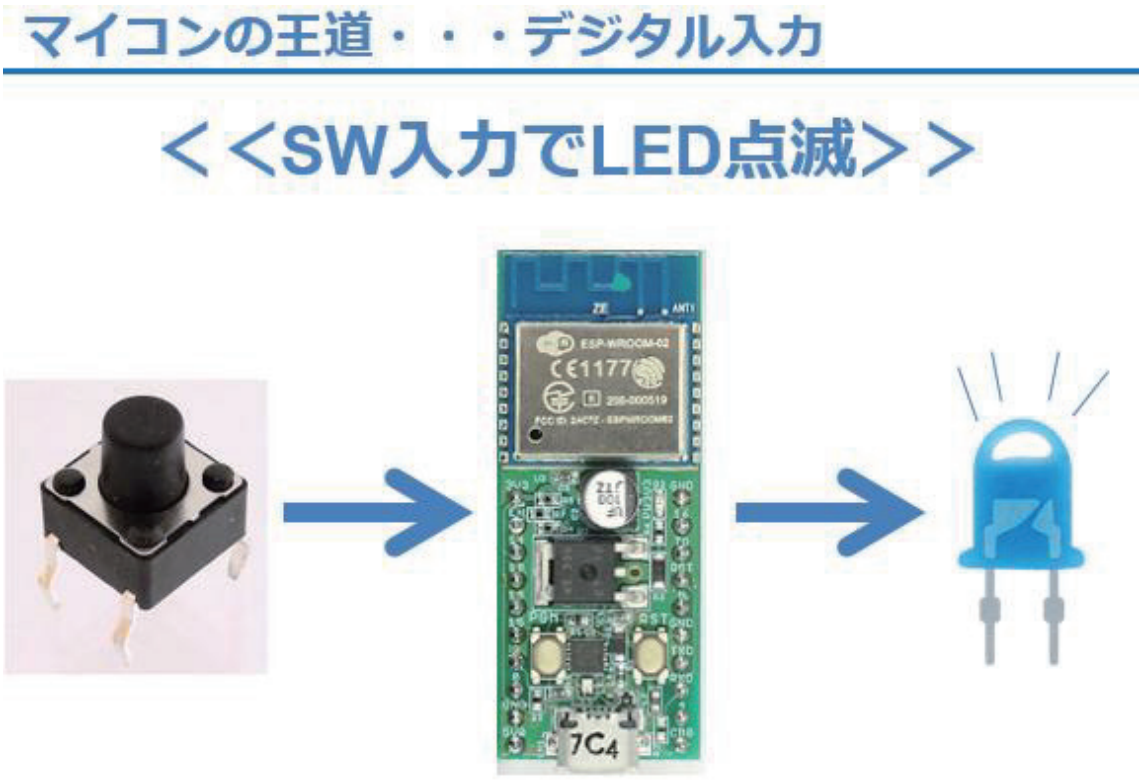


図　179

LEDのON/OFF信号がデジタル出力（DO）であったのに対して、SWからの入力はデジタル入力（DI）になります。システムが動作するとSW操作にWiFiマイコンが反応しているように見えます。

◇全体構成とパーツ

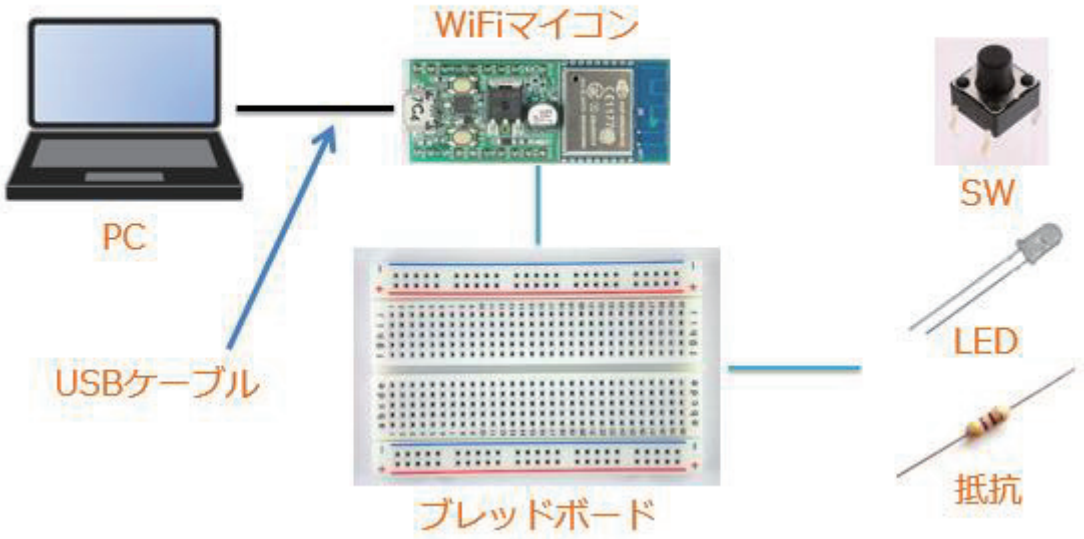


図 180

システムの全体構成を上図に示します。必要な機材・パーツは、下記です。電源は USB ケーブルで PC から供給されます。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. LED×1 個
7. 抵抗器（470Ω）×1 個
8. SW×1 個

前回のパーツに SW が加わっています。ここで、SW の構造について少し説明します。

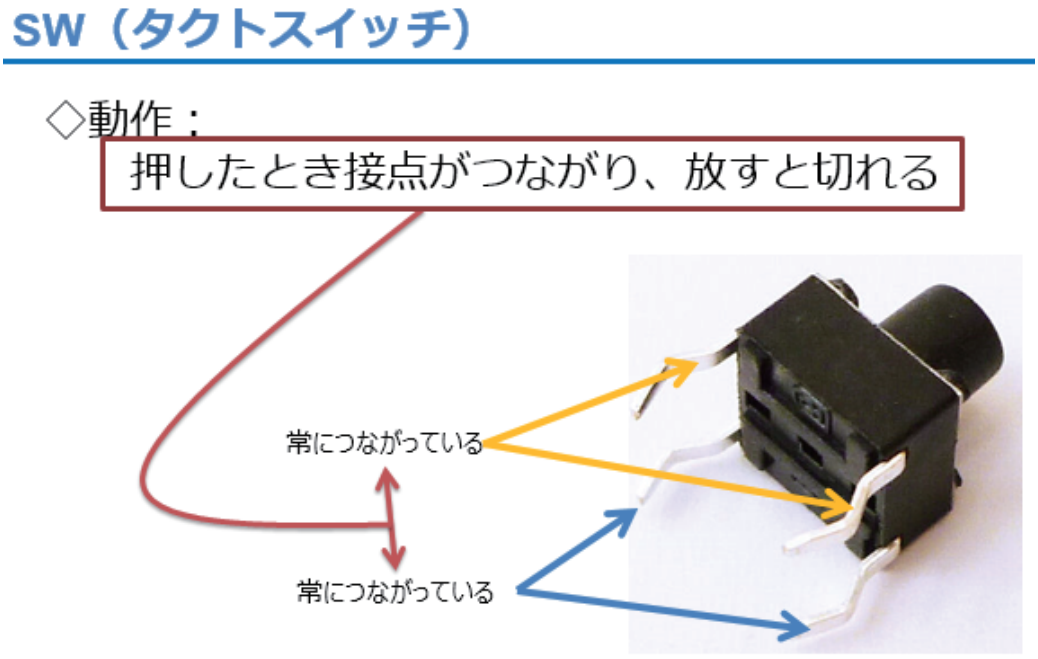


図　181

今回使用するSWは【タクトスイッチ】という名称の小さなSWです。動作は単純で、押したときに接点が繋がり、放すと切れるというものです。上図で湾曲した脚ピンがありますが、湾曲している向かい合った対になるピンが内部で接続されています。これが2対あり、ボタンを押すと、その2対が内部で繋がる仕組みです。

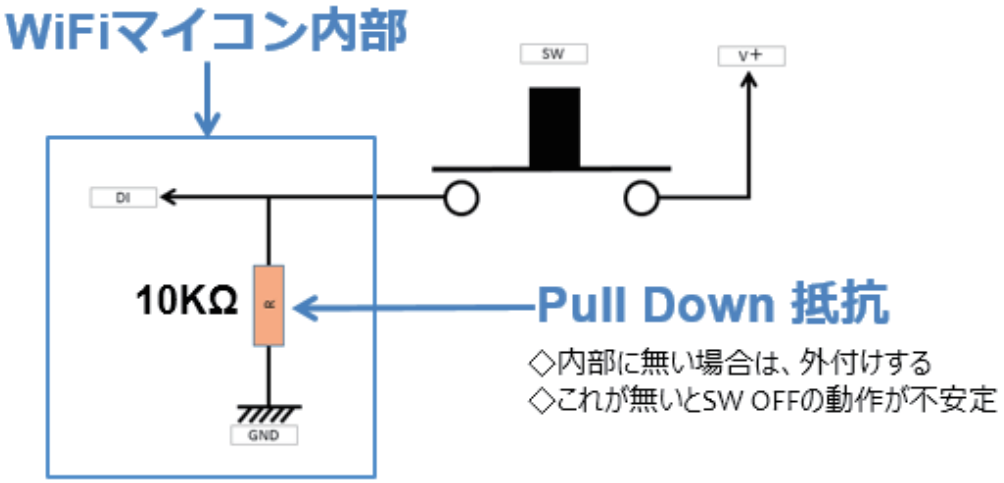


図　182

今回のSWの接続は上図のように描けます。SWを横から見た様子と考えて下さい。SWの脚ピンは湾曲している1対がSW下の○印です。片側がV+に接続されています。V+とは電源の＋側を指します。反対側はマイコン内部で10KΩの抵抗とマイコンのDI（デジタル入力の意味）に接続されています。10KΩの抵抗はGNDに接続されています。まずSWを押したときの状態をマイコン側から読み込むと、DIの部分はSWを介してV+に接続された状態となっているので、電圧レベルはV+(=HIGH)となり=1です。反対にSWを離した状態の場合は、DIの部分は電圧レベルが0V（※抵抗でGNDに接続されている）で読み込んでも=0(LOW)です。ここで10KΩの抵抗は何のためにあるかというと、なにも接続されていない解放状態のDIの信号は不安定になる可能性があるので、解放状態でも確実にLOWレベルにするために抵抗を介してGNDにつなぎます。抵抗が無いと、SWが押されたときV+とGNDが直接つながってしまい、ショート状態になってしまいます。

【重要】

この様に確実にLOWレベルにしておく為の抵抗をプルダウン（PULL DOWN)抵抗と云います。

SW・LED点灯回路

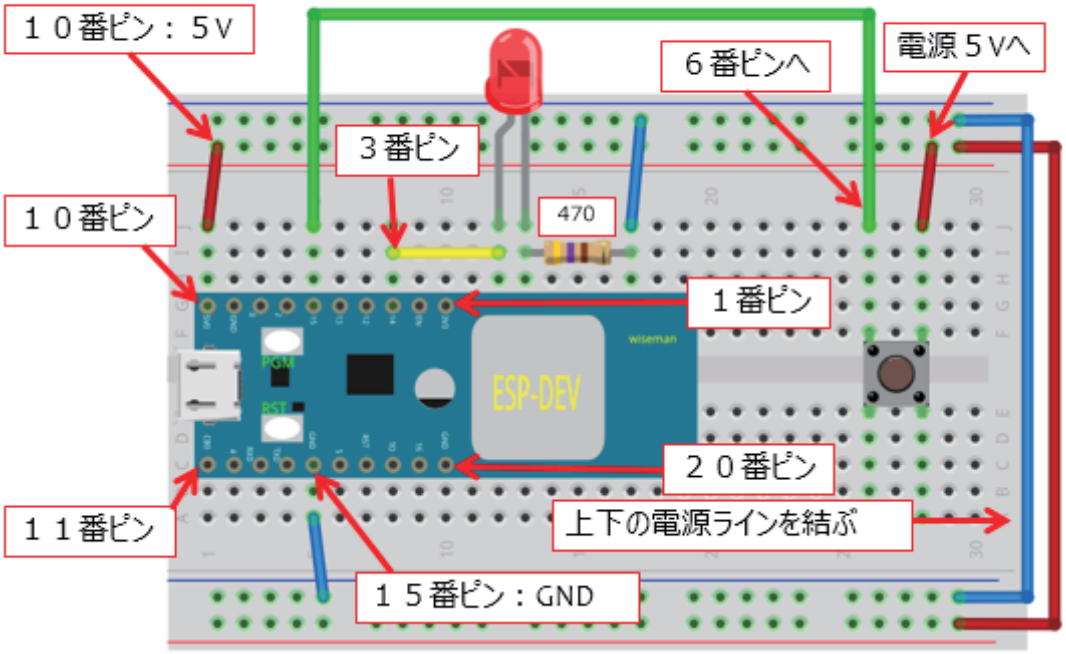


図　183

【重要】

配線を行う際は、必ずUSBケーブルを外して下さい。接続したまま配線を行ってショートが発生すると、PCのUSBポートを壊してしまうことがあります。以後の講座も同様です。

まず、上図に従って配線しましょう。前回のLED点滅の回路にSWが追加になっています。SWは湾曲した脚で、ブレッドボード中央の溝を跨ぐように配置してください。図でSW右上のピンは赤いジャンパー線で5Vに配線します。SW左上のピンは緑色のジャンパー線で6番ピンのGPIO15に配線します。ジャンパー線は2本増えただけですが、油断せず確実に配線してください。新しい配線をすると、既存の配線が外れたり緩んだりすることもありますので、確認を行ってください。

実際に配線した様子

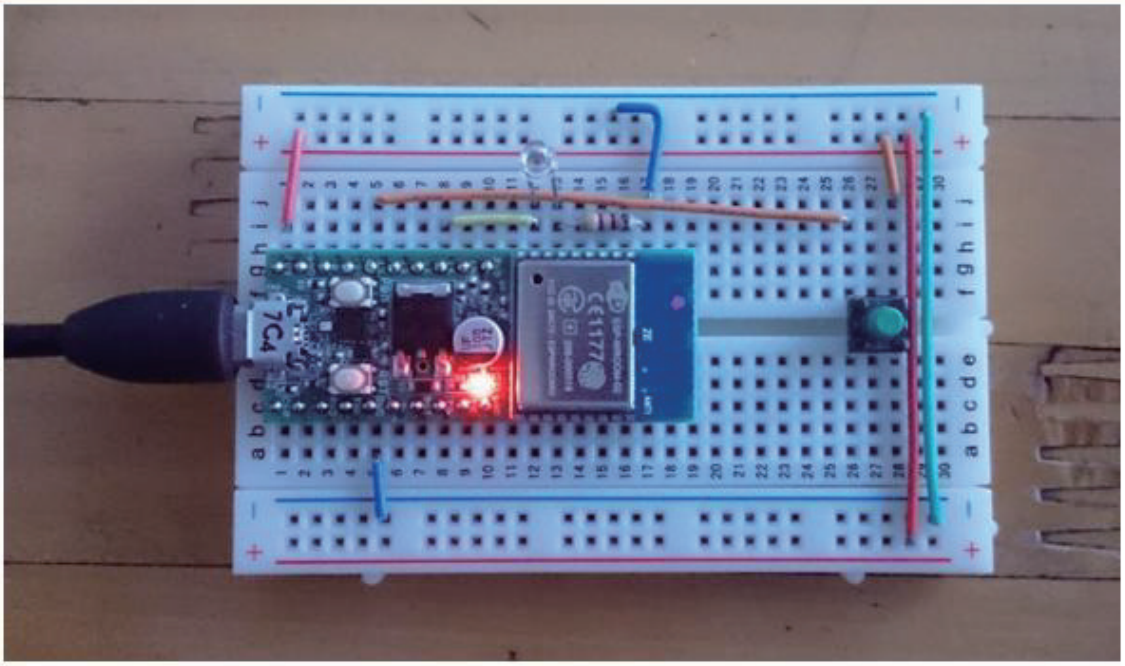


図 184

実際に配線した様子が上の写真です。

【重要】

この回路は、直後の講座でも使いますので、今回の動作確認が終わっても保存しておいて下さい。

回路ができたら、次はプログラムの作成です。

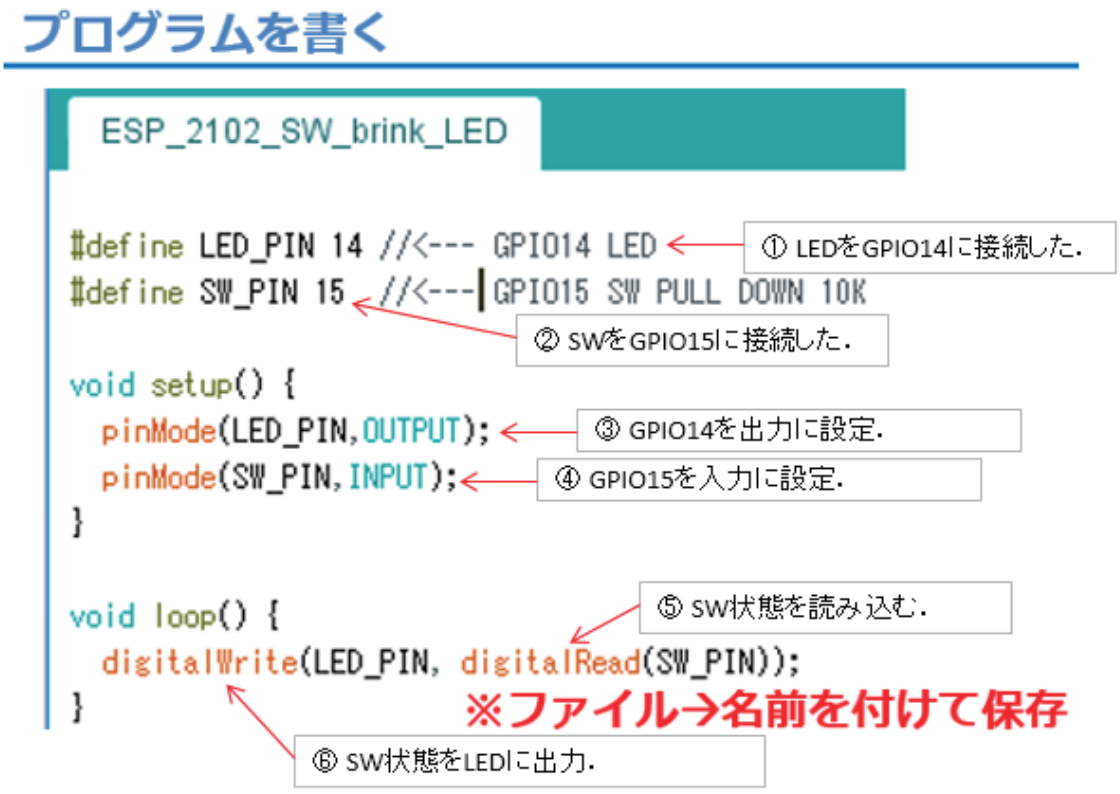


図　185

IDE のファイルメニューで新規を選択して、新しいスケッチ（Arduino IDE ではプログラムをスケッチと呼びます。）を開きます。ウインドウの中央、白い部分に上図のソースコードを入力してください。

①LED を GPIO14 に接続したので 14 を定義しています。②SW は GPIO15 に接続しました。③GPIO14 を出力に設定。④GPIO15 を入力に設定。⑤SW 状態を読み込みます。状態は degitalRead()の戻り値として返されますが、次の⑥digitalWrite ()で LED の接続されている GPIO14 にそのまま出力します。

プログラムができたら、名前を付けて保存しておきましょう。

次は、マイコンと PC を USB ケーブルで接続します。

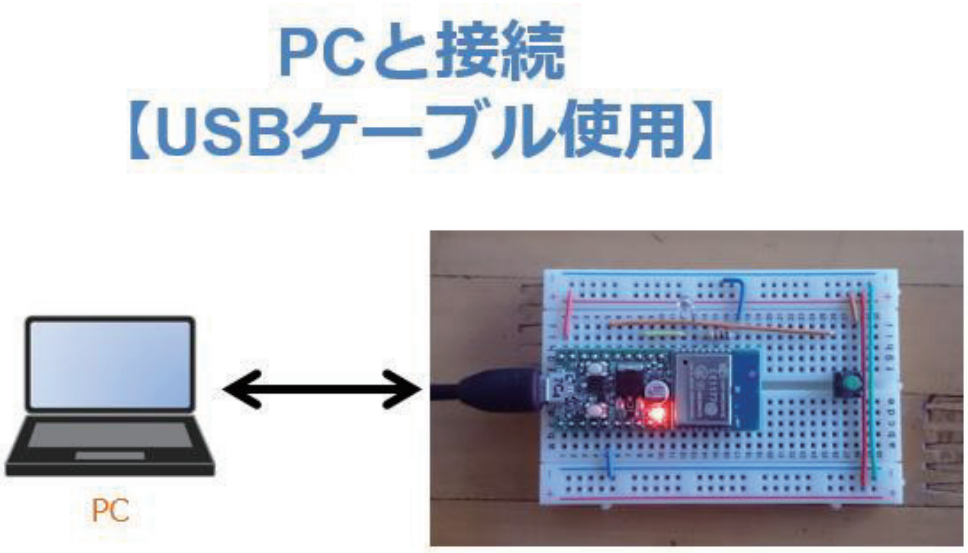


図 186

すでに、第１回目で USB-シリアルドライバがインストールされていて、COM ポート番号も設定済みだと思いますが、ここでは念のためにデバイスマネージャで確認して、IDE の COM ポート番号を確認・設定しておきます。

第１回**【これからプログラムを書込むゾ！！】**以後を参照して、プログラムのコンパイル・リンクを行い、マイコンへの書込みを行いましょう。

書込みが終了するとメッセージが表示されて、マイコンに Reset が掛かりプログラムは開始されています。出来上がったシステムの動作確認をしましょう。

動作の様子

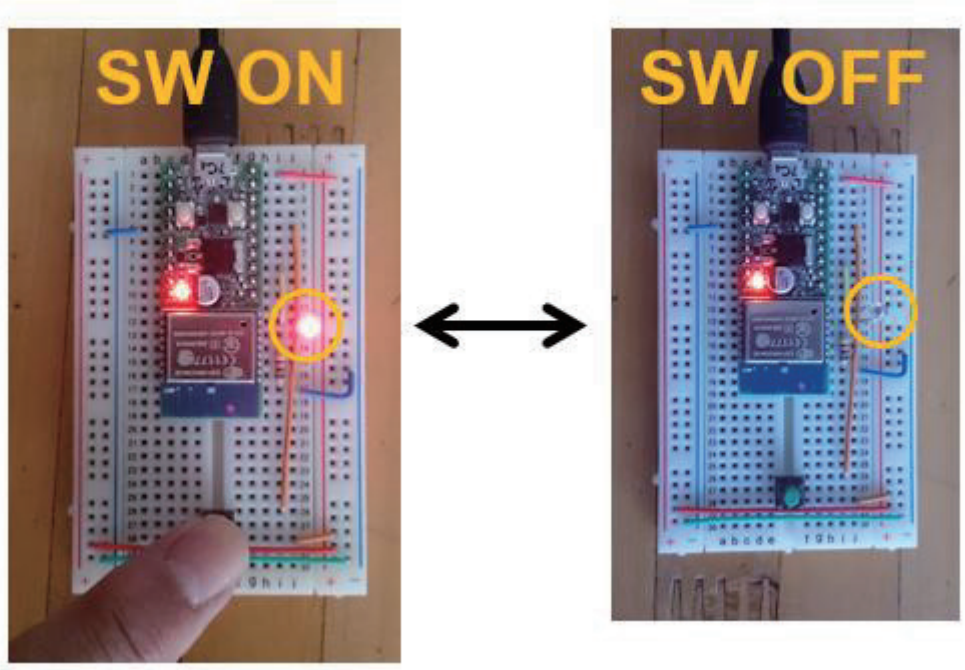


図 187

まずは、SW を押してみます。同時に LED が点灯します。（写真左）SW を解放すると LED は消灯します（写真右）。いかがでしょうか、SW 状態に反応する（または SW 状態を反映する）システムができたでしょうか。これで、この WiFi マイコンモジュールのデジタル入力・出力が使えるようになりました。今回の講座の中で解説した SW の入力を読むポートに接続されている PULL DOWN 抵抗を覚えておいてください。

第3回　シリアル通信【送信】

今回は、開発のために使用しているPCに対してシリアル通信を行い、WiFiマイコンモジュールからメッセージを送信してみましょう。

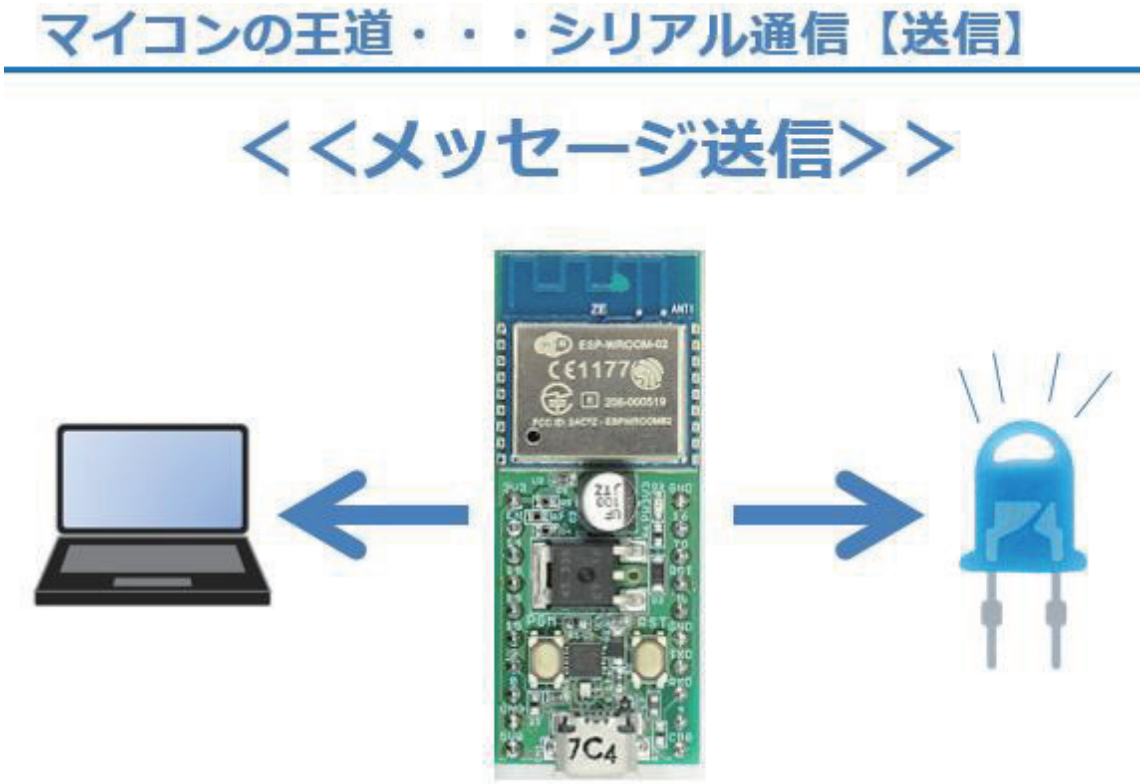


図　188

USBケーブルで接続されているPCは、プログラムをWiFiマイコンモジュールのフラッシュメモリに書き込むために、通信を行っていますが、その通信ポートを使って、マイコン側で作ったメッセージを送信すると同時にLEDの点灯を行ってみます。

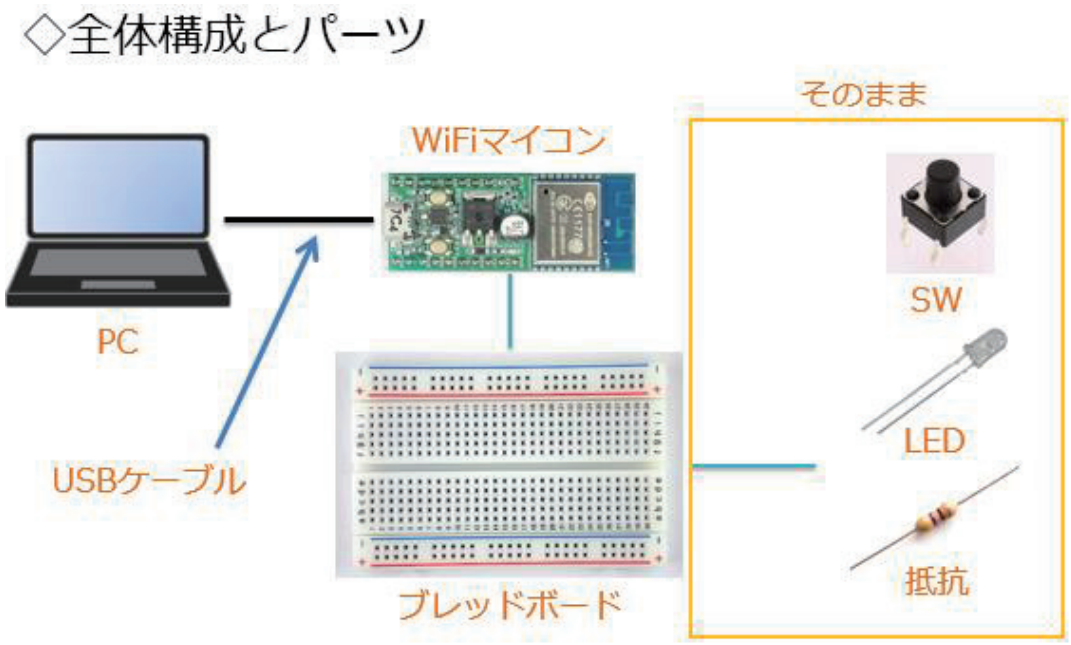


図 189

上図にシステムの全体構成を示します。前回のものと同じ内容です。

必要な機材・パーツは、下記です。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. LED×1 個
7. 抵抗器（470Ω）×1 個
8. SW×1 個

回路を下図に示します。こちらも前回と同じです。

SW・LED点灯回路

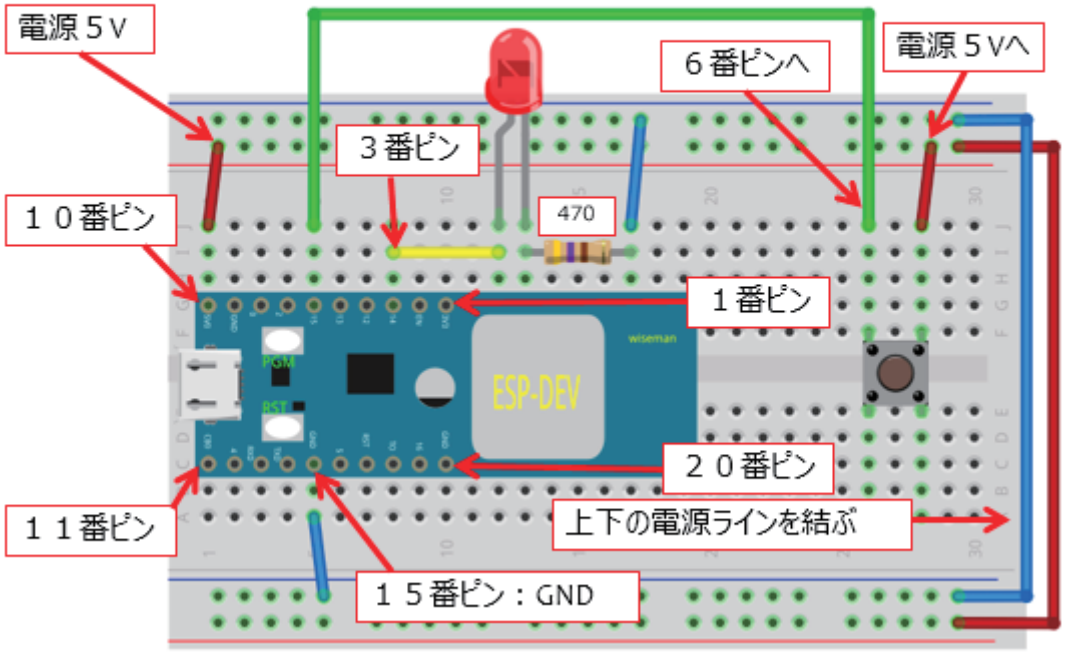


図　190

実際に配線した様子

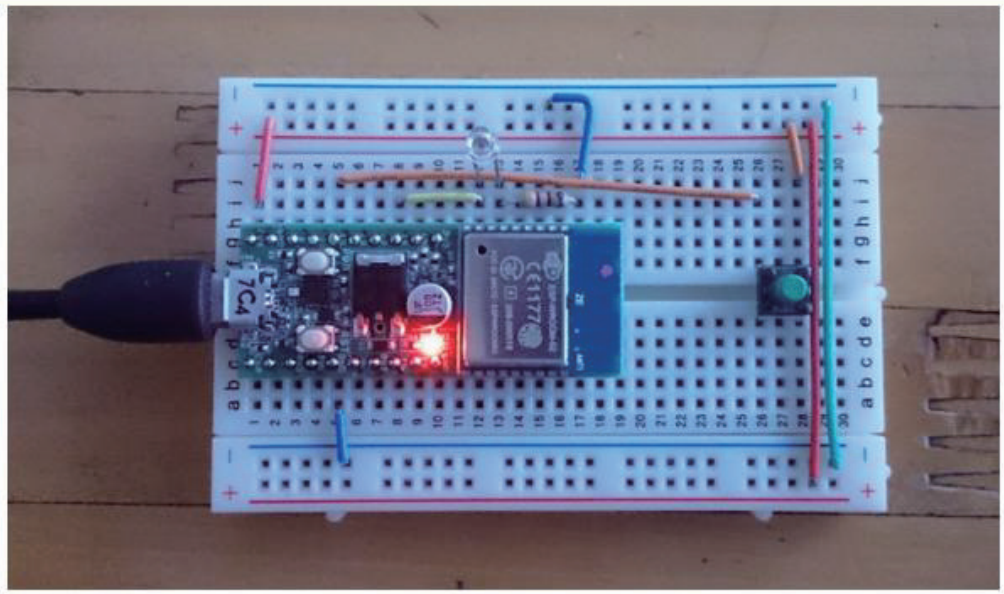


図　191

実際に配線した様子を上図に示します。前回の講座後に、改めて回路を作られる方は、第 2 回 SW を参照して作成して下さい。

【重要】
この回路は、直後の講座でも使いますので、今回の動作確認が終わっても保存しておいて下さい。

回路ができたら、次はプログラムの作成です。プログラムの作成に必要な IDE の準備についての詳細は、第 1 回 LED 点滅を参照して下さい。

◇Arduino 統合開発環境 IDE

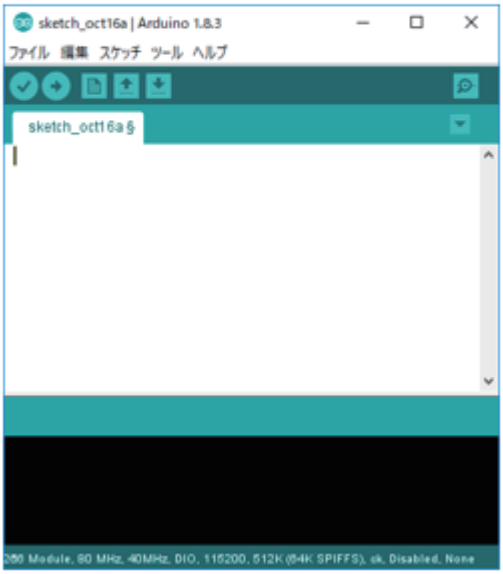


図 192

上図デスクトップ上の IDE ショートカットにより、IDE を起動します。

図　193

ツールメニューのプルダウンで、Generic ESP8266 Module が選択されていることを確認してください。異なるボードの場合は、上図に従い、ボードを選択してください。

IDE の中央にソースコードを入力します（下図）。

図 194

IDE に上のソースコードを入力してください。新たな部分は次の 2 点です。①シリアルポートの初期化です、シリアル通信は通信速度を指定できます。ここでは 9600bps（bps：bit/sec）の速度で初期化しています。②シリアル通信で文字列を送信します。()内部のパラメータで【""】ダブルクォーテーションで挟んだ文字列を、PC に向けて改行付きで送信します。Serial.println()の関数名の最後の 2 文字に【ln】が付いているのが、改行コード付きの送信関数名です。

loop()内では、固定のメッセージを PC に送信して、LED を 1 秒点灯し、1 秒消灯しています。この結果、1 秒ごとに LED 点滅を行いながら、固定メッセージをシリアル送信する動作を繰り返します。

ソースコードを入力したら、名前を付けて保存しておきましょう。

PC と WiFi マイコンを USB ケーブルで接続し、第 1 回【**これからプログラムを書込むゾ！！**】以後を参照して、プログラムのコンパイル・リンクを行い、マイコンへの書込みを行います。

書込みが終了するとメッセージが表示されて、マイコンに Reset が掛かりプログラムは開始されています。出来上がったシステムの動作確認をしましょう。

動作確認

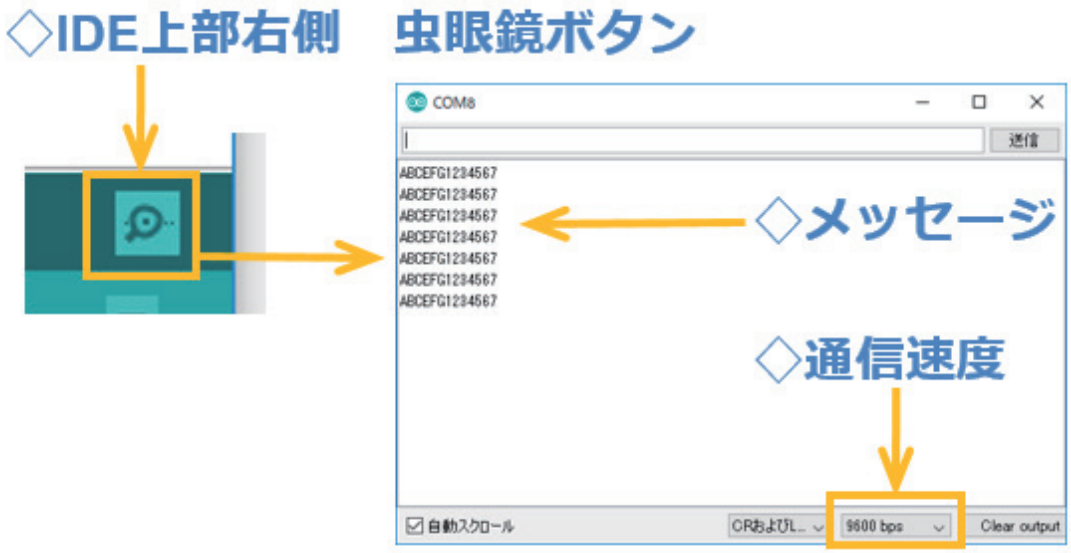


図　195

IDE ウインドウの右上にある虫眼鏡マークのボタンをクリックすると、シリアルモニターのウインドウが開きます。シリアルモニターウインドウの下部にある通信速度のプルダウンで、setup()関数で設定した 9600bps を選択します。通信速度が合っていないと、何も表示されなかったり文字化けを起こしたりしてしまいますので、忘れない様にしてください。

通信速度を合わせれば、WiFi マイコンが送信しているメッセージがウインドウに約 1 秒間隔で表示され続けます。

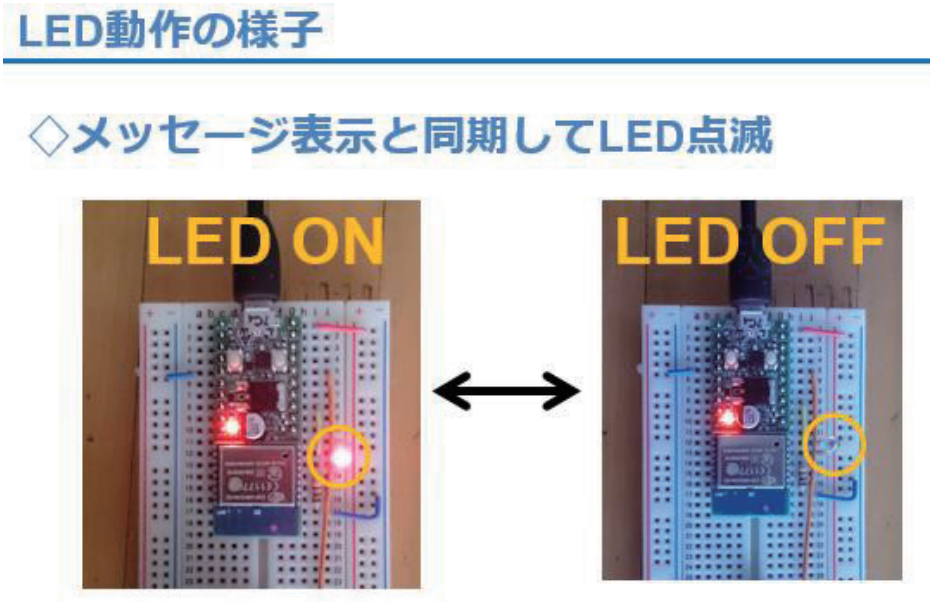


図 196

この時 WiFi マイコンの回路では LED が点滅をしていますが、シリアルモニターのメッセージと合わせて見ていると、メッセージ受信に同期して LED の点滅動作が確認できます。

【重要】

ここでは、固定のメッセージを送信していますが、このメッセージ文字列の内容をダイナミックに変化するデータなどで編集すれば、マイコンが計測した情報を監視する用途などに、シリアル通信が使えるようになります。今回は送信でしたが、次の講座では受信を行ってみましょう。

第4回　シリアル通信【受信】

シリアル通信の受信が行えるようになると、受信したメッセージ（コマンド等とも言いますが）の内容に対応する動作や処理を行わせることができて、リモコン操作ができるような仕組みが構築できます。

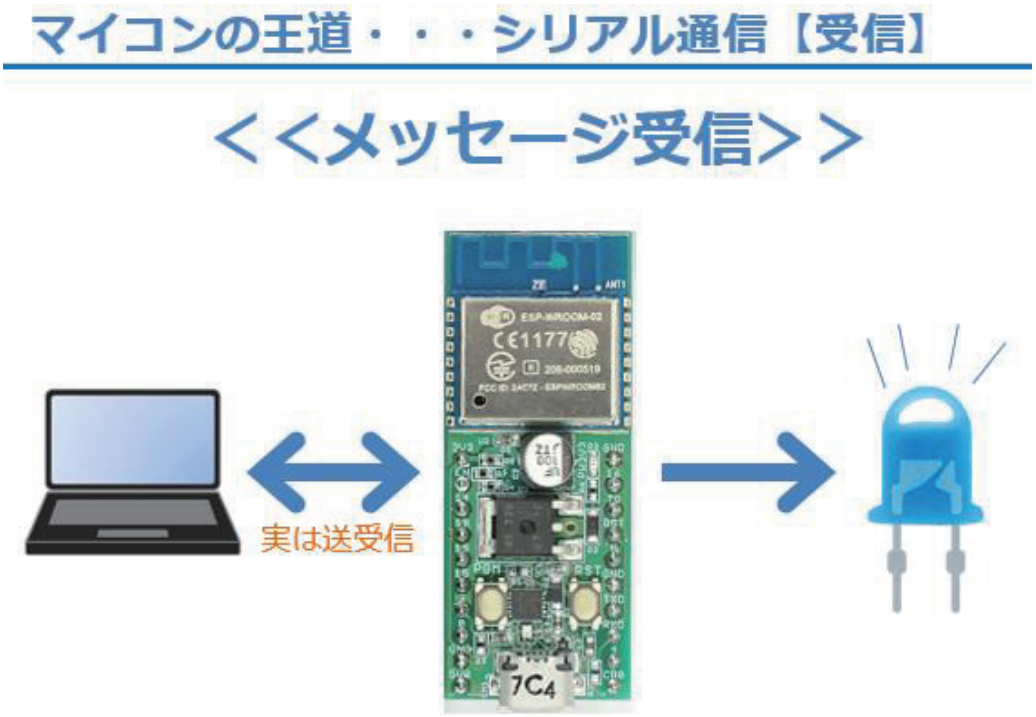


図　197

上図は、前回と同じように見えますが、PCとWiFiマイコンが双方向の矢印で結ばれています。今回のテーマは、シリアル通信の【受信】ですが、せっかく送信もできるようになっているので、欲張って双方向の通信をしてみましょう。PCから送信するメッセージの電文内容によって、WiFiマイコンの外部に配線したLEDを制御してみます。またその際のWiFiマイコンの振舞いをPCに送信します。

◇全体構成とパーツ

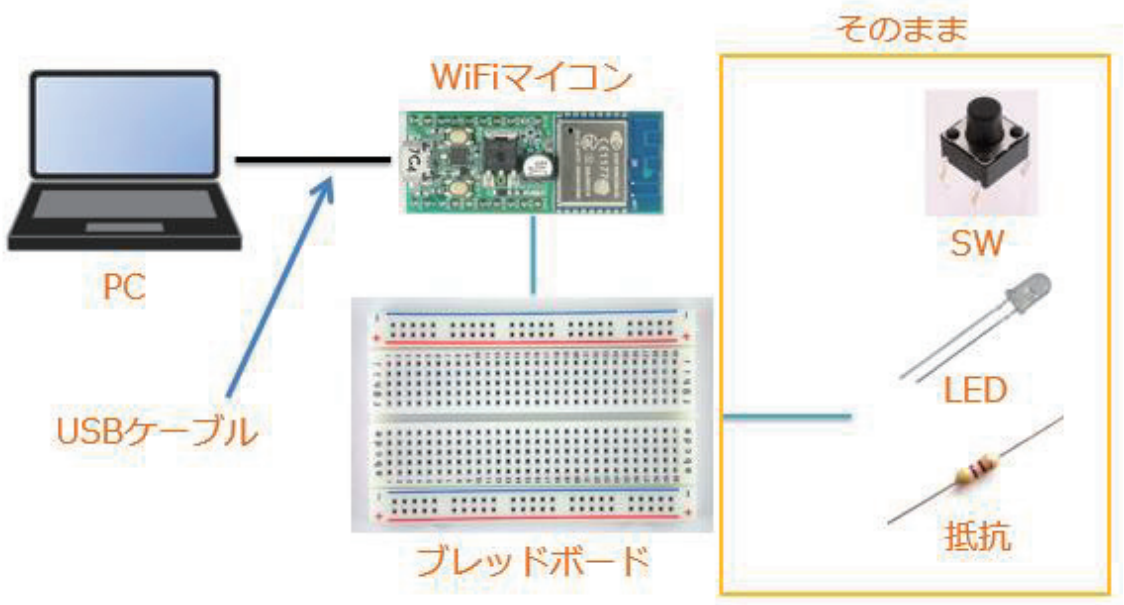


図 198

システムの全体構成は、上図の通り、第 2 回と同じです。

必要な機材・パーツは、下記です。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. LED×1 個
7. 抵抗器（470Ω）×1 個
8. SW×1 個

回路を下図に示します。こちらも前回と同じです。

SW・LED点灯回路

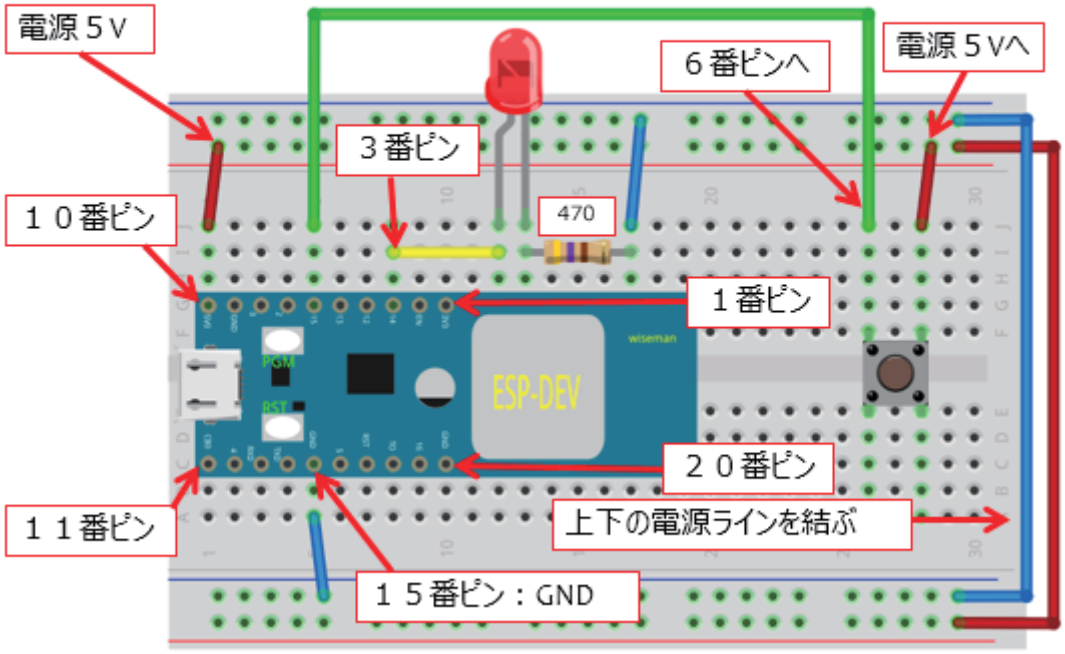


図　199

実際に配線した様子

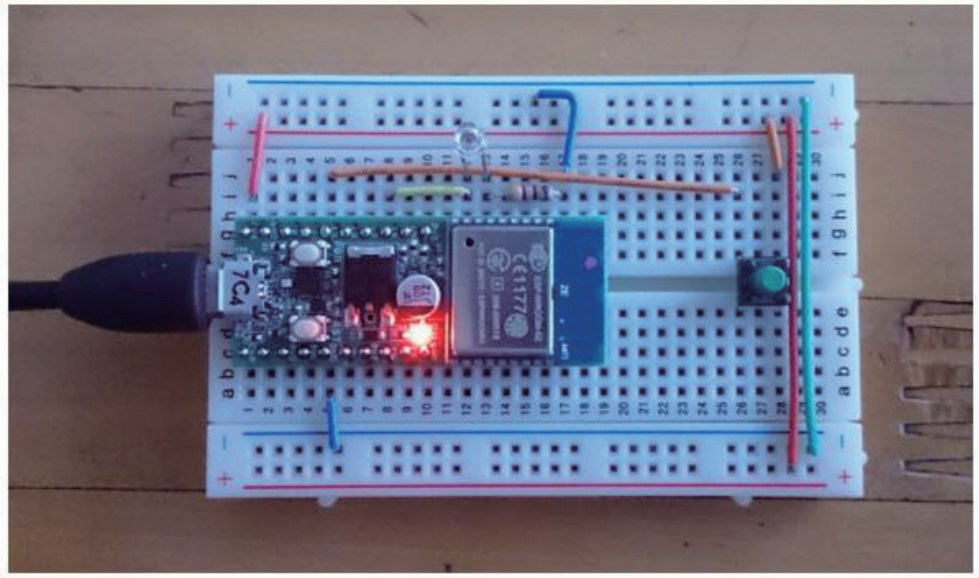


図　200

実際に配線した様子を上図に示します。前回の講座後改めて回路を作られる方は、第 2 回 SW を参照して作成して下さい。

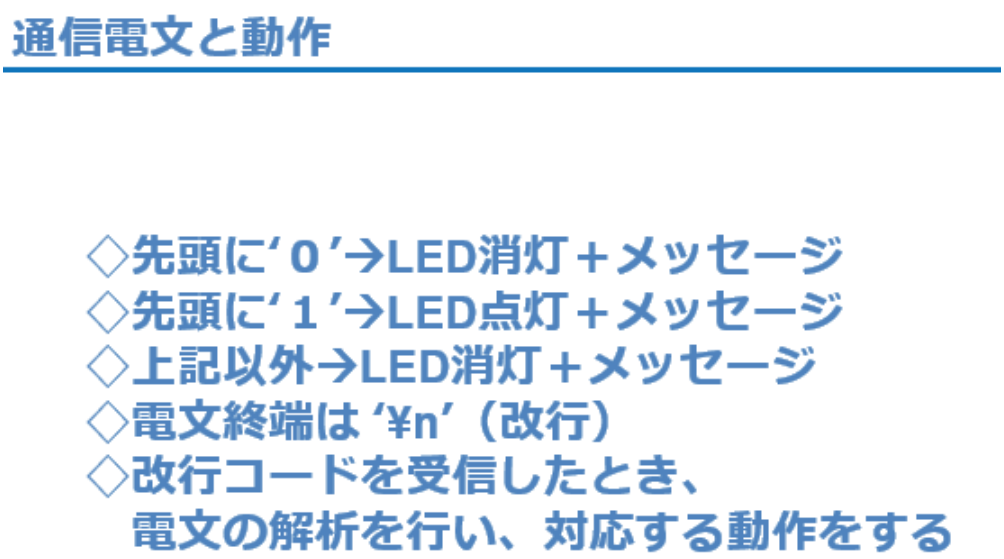


図 201

ここで、シリアル通信で使用する通信電文の設計をしておきましょう。上図を見てください。通信に使用する電文は、文字（ASCII コード）により構成します。電文の先頭 1 文字目が数字の’0’か’1’またはそれ以外の 3 パターンで処理内容が変わります。電文の終端は’¥n’（改行）とし、’¥n’を受信したとき、電文の解析・対応処理を行います。

電文が設計出来たら、プログラムの作成です。必要な IDE の準備については第 1 回 LED 点滅を参照してください。

プログラム開発のための IDE の準備とボードの選択は、ここまでの講座で既に済んでいます。（上図）IDE の【ファイル→新規ファイル】 と辿り、新しいスケッチにソースコードを記述しましょう。以下にソースコードを示します。

プログラム

◇冒頭＋初期化

ESP_2104_Serial_Rx

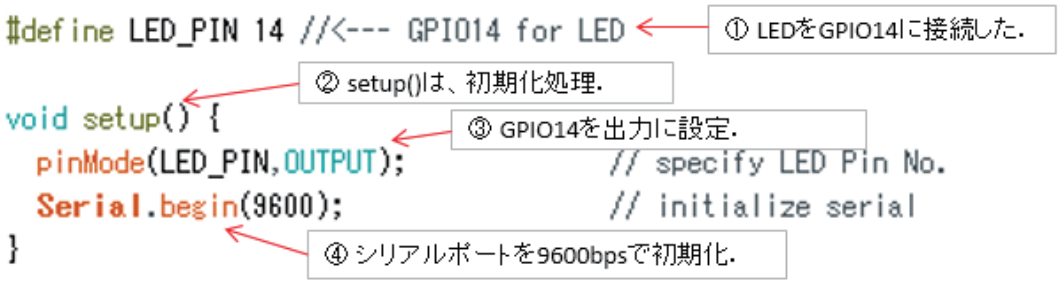


図　202

まず、冒頭の部分からです。

①LED 用の GPIO 番号定義

②初期化の専用関数は setup()という決まった名前

③LED 用に GPIO14 を出力に設定

④シリアルポートを 9600bps で初期化

次はメイン処理部分です。

プログラム　◇メイン処理部の全体

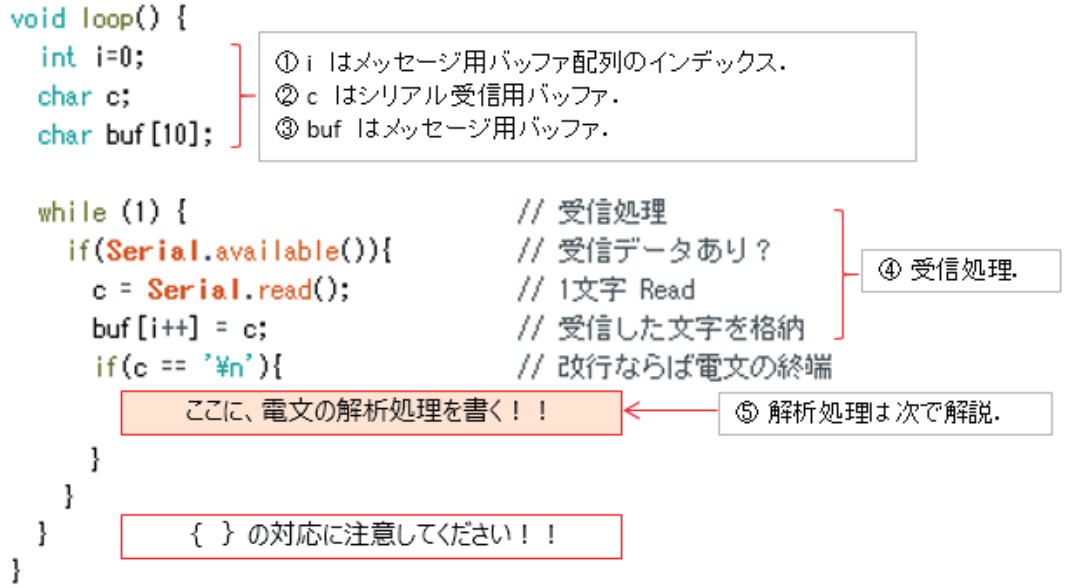


図　203

loop()関数は繰り返し実行される関数で、名称固定です。

①シリアル通信で受信される文字を③の配列に格納するインデックス

②シリアル通信で受信する 1 文字用のバッファ．10 文字分で十分

③受信した文字データを格納して電文として編成するバッファ

④受信処理部.

Serial.available()で受信しているデータがあるかを調べる

Serial.read()で 1byte を読み込む

受信した 1byte の文字を配列に格納．インデックス更新

受信したデータが'¥n'（改行）かどうか判断

⑤電文解析：次で説明

※この部分は、{}（括弧）が多いので対応に注意をしてください。

プログラム　◇電文の解析処理

```
switch(buf[0]){                  // 受信コマンド解析
  case '0':                      // ='0' LED 消灯
    digitalWrite(LED_PIN,LOW);
    Serial.print("LED OFF¥n");
    break;
  case '1':                      // ='1' LED 点灯
    digitalWrite(LED_PIN,HIGH);
    Serial.print("LED ON¥n");
    break;
 default:                        // '0','1'以外
    digitalWrite(LED_PIN,LOW);
    Serial.print("*** Unknown Command!!¥n");
    break;
}
```

① LED消灯＋メッセージ送信.

② LED点灯＋メッセージ送信.

③ LED消灯＋メッセージ送信.

図　204

上の部分は電文解析と対応処理の本体です。電文配列 buf[] 内の先頭の文字を判断して対応する処理を行います。

①先頭が'0'の場合：LED を消灯し【LED OFF】を送信

②先頭が'1'の場合：LED を点灯し【LED ON】を送信

③先頭が'0'の場合：LED を消灯し【*** Unknown Command!!】を送信

ソースコードを入力したら、名前を付けて保存しておきましょう。

ここから、WiFiマイコンへの書込みまで一気に進めましょう。

以下の手順でプログラムのコンパイルからWiFiマイコンへの書き込みを行います。

①PCとWiFiマイコンの接続：USBケーブルで接続

②シリアルポート（COM番号）の確認：デバイスマネージャで確認

③シリアルポート（COM番号）の設定：IDEで設定

④WiFiマイコンへの書込み準備：WiFiマイコンのSW操作

⑤スケッチ（プログラム）のコンパイルから書込み：IDEで操作

書込みが終了すると書込み完了のメッセージがIDEに表示される

※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。

書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作確認

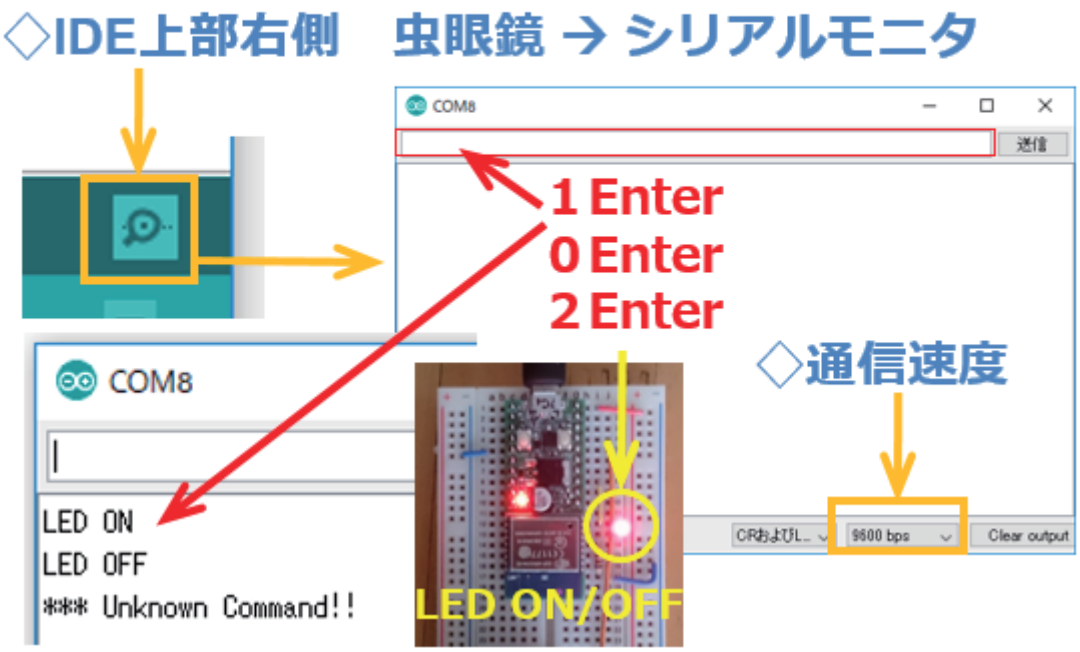


図　205

動作確認を行います。IDE 上部右側にある虫眼鏡マークのボタンをクリックしてシリアルモニターを起動しましょう（上図右）。

シリアルモニターの下部にある通信速度のセレクタで今回の通信速度 9600bps を選択します。メッセージは何も表示されません。

PC から WiFi マイコンに電文を送信してみましょう。シリアルモニター上部の COM ポート番号が表示されている直下に送信データを入力する部分があります。そこにカーソルを入れて、数字（半角）の'1'を入力し Enter キーを押下するか、右側にある送信ボタンをクリックしてください。この操作で 1 文字(='1')だけの電文が'¥n'(改行)付きで WiFi マイコンに送信されます。その時、LED が点灯してシリアルモニターには、LED ON のメッセージが表示されます。

次に送信データに'0'を入力して Enter キーを押下するか、送信ボタンをクリックして下さい。すると、LED が消灯して LED OFF のメッセージがシリアルモニターに表示されます。

今度は'2'を入力して Enter キーを押下するか、送信ボタンをクリックしてください。すると LED は消灯のままで、*** Unknown Command!! と表示されます。

このシステムでは、'0', '1'以外の文字を送信すると全て Unknown になります。送信した側では、何を送ったかが分からないので、送信された文字を PC に送り返して、再送信を促すようにしてもよいですね。実際のマイコン・装置・設備・システムの間のシリアル通信では、そのように、現在の状況をお互いに送信し合いながら、連携をするような仕組みがプログラムされています。

いかがでしょうか、マイコンを利用したシステムを PC からリモートコントロールして、状況を返信させることができるようになりました。マイコンに複数のセンサーが接続されていて、その中から特定のセンサーの今の計測値を送信させるような使い方ができますね。PC 側では、受信したデータを CSV ファイルなどに記録したり、あるいはグラフ化したりして利用できます。どのような応用ができるか、考えてみてください。

第5回　VR(電圧測定)

なぜボリュームをVRと書くのでしょうか。ボリュームの正式名称は、可変抵抗器です。つまり抵抗の値が変えられる（可変）抵抗器ということでVariable Resistor、略してVRです。第5回目は、このVRで電圧を変化させて、それを計測してみましょう。

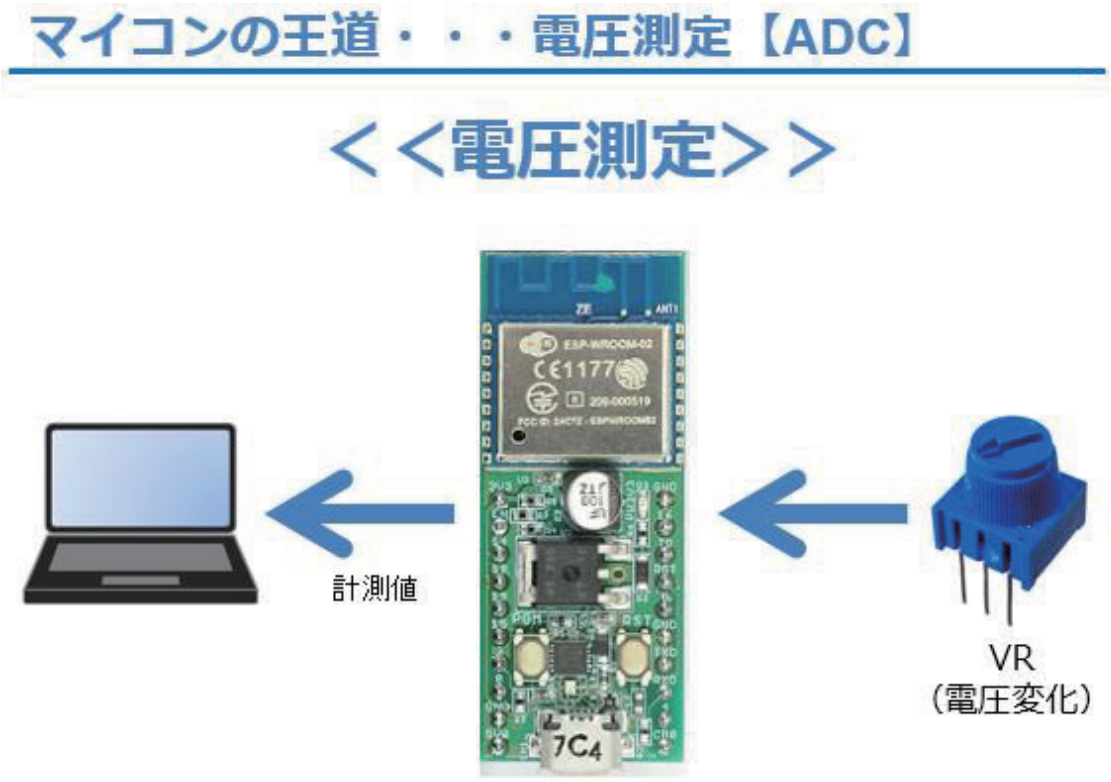


図　206

VRを使った電圧変化を測定してPCに送信します。第3回シリアル通信【送信】の最後で、送信するメッセージの内容を計測値で編集すれば、マイコンが計測した情報を監視する用途にシリアル通信が使えると解説しました。まさにその実証です。

◇全体構成とパーツ

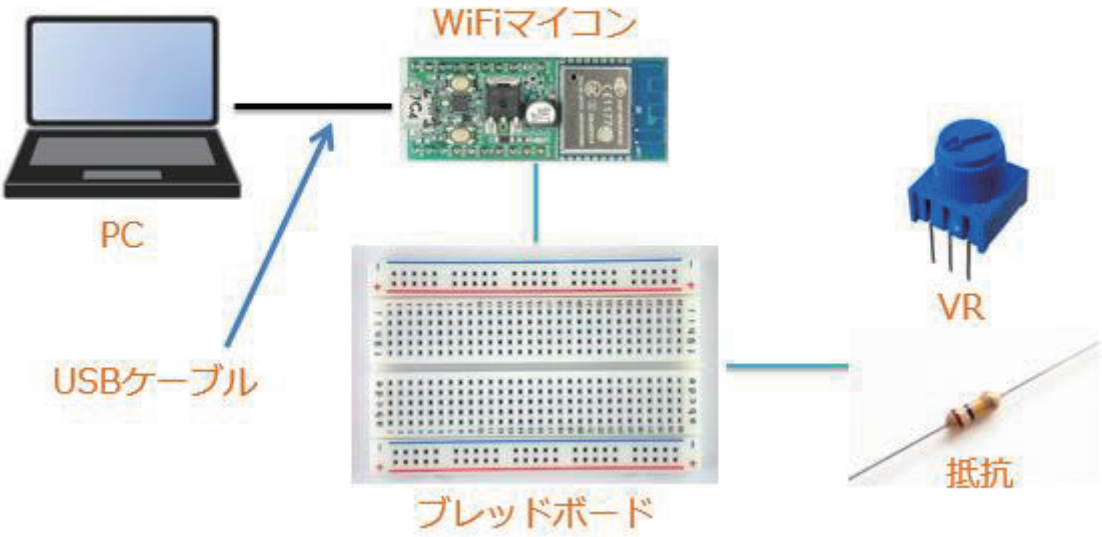


図 207

システムの全体構成を上図に示します。必要な機材・パーツは、下記です。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. VR（可変抵抗器：50KΩ）×1 個
7. 抵抗器（100KΩ）×1 個

ここで、VR と固定抵抗器について解説します。

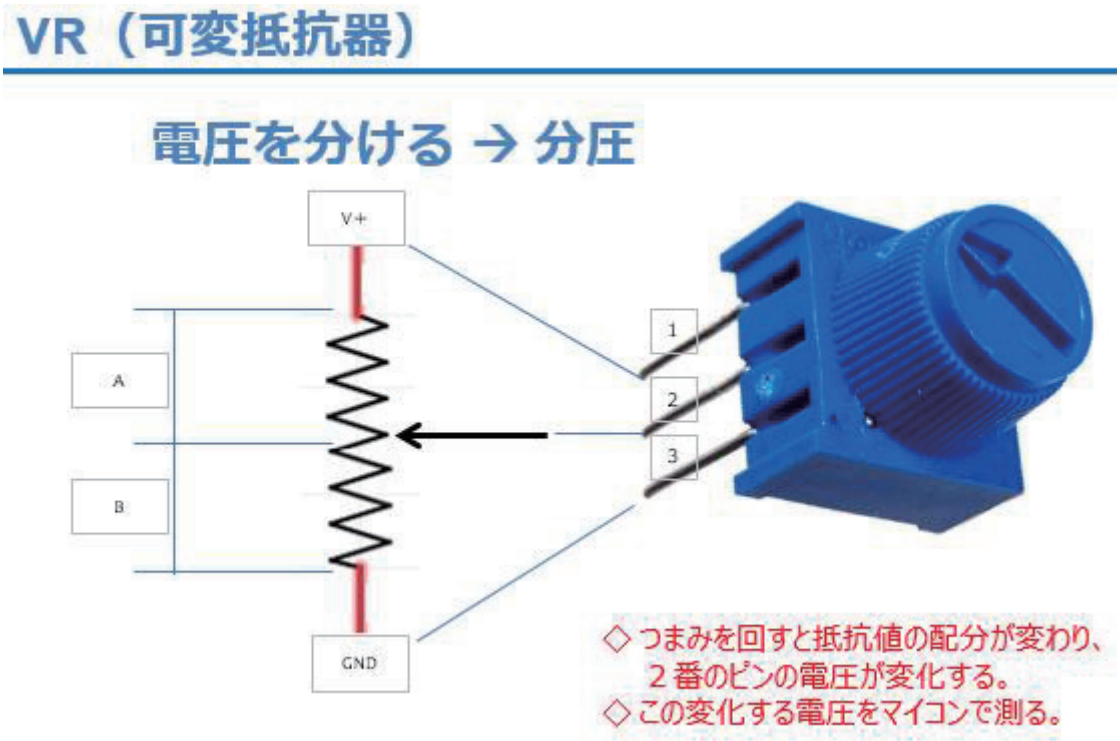
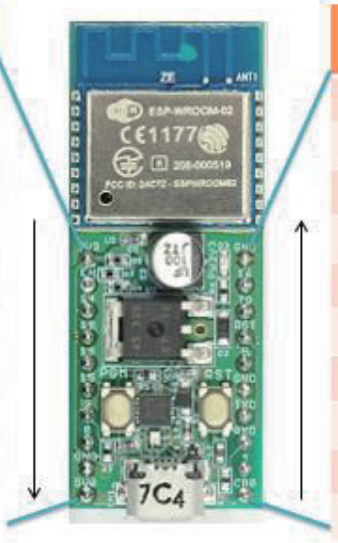


図 208

上図は、今回使用する VR の写真と内部構造を表しています。VR にはつまみ（矢印が刻印されていて周囲がギザギザになっている部分）が付いています。脚ピンが 3 本あり、便宜的に写真上から 1,2,3 番とします。1 番と 2 番ピンの間は、50KΩの抵抗値となっています。つまみは内部の接点（図左側の矢印部分）と連動しています。つまみを右左に回すと、図の矢印部分が上下に移動する仕組みです。ここで、1 番ピンを V+（電源の+側）に、3 番ピンを GND に接続して、つまみをいっぱいに回して、図の矢印が V+に一番近くなった時、2 番ピンと 3 番ピンの間には、V+の電圧がかかっていることになります。反対に矢印部分が一番下がったときは、矢印部分は GND と接続していることになり、2 番ピンと 3 番ピンの間の電圧は 0V になります。このように、VR は、2 番ピンの矢印の位置(つまみの回し具合)によって、図の A と B の部分に電圧を分けてくれる役割を果たします。このことを分圧と云います。

この分圧で得られる電圧値は、GND から V+の間の大きさになります。VR の 2 番ピンから出力される電圧を計測するのが今回の目的です。

番号	名称	機能
1	3V3	3.3V(In/Out)
2	EN	イネーブル端子
3	14	GPIO14
4	12	GPIO12
5	13	GPIO13
6	15	GPIO15
7	2	GPIO2
8	0	GPIO0
9	GND	GND
10	5V0	5V In



番号	名称	機能
20	GND	GND
19	16	GPIO16
18	TO	ADC（max1V）
17	RST	リセット
16	5	GPIO5
15	GND	GND
14	TXD	TXD
13	RXD	RXD
12	4	GPIO4
11	CB0	通信モニタ用

図 209

ここで、上図（再掲）を見てください。この図は第 1 回で説明した WiFi マイコンモジュールのピン配置です。上図の 18 番ピン TO には ADC(max1V)と記載があります。VR の 2 番ピンと接続します。ADC とは Analog to Digital Converter のことで、アナログ値である電圧をデジタル値に変換する便利な機能です。電圧計測の手段は ADC の機能そのものです。

ところで、max1V と記載があります。これは、1V までしか測れないことを意味していますので、VR からの出力が 1V になるように調整しなければいけませんので、少し工夫が必要です。

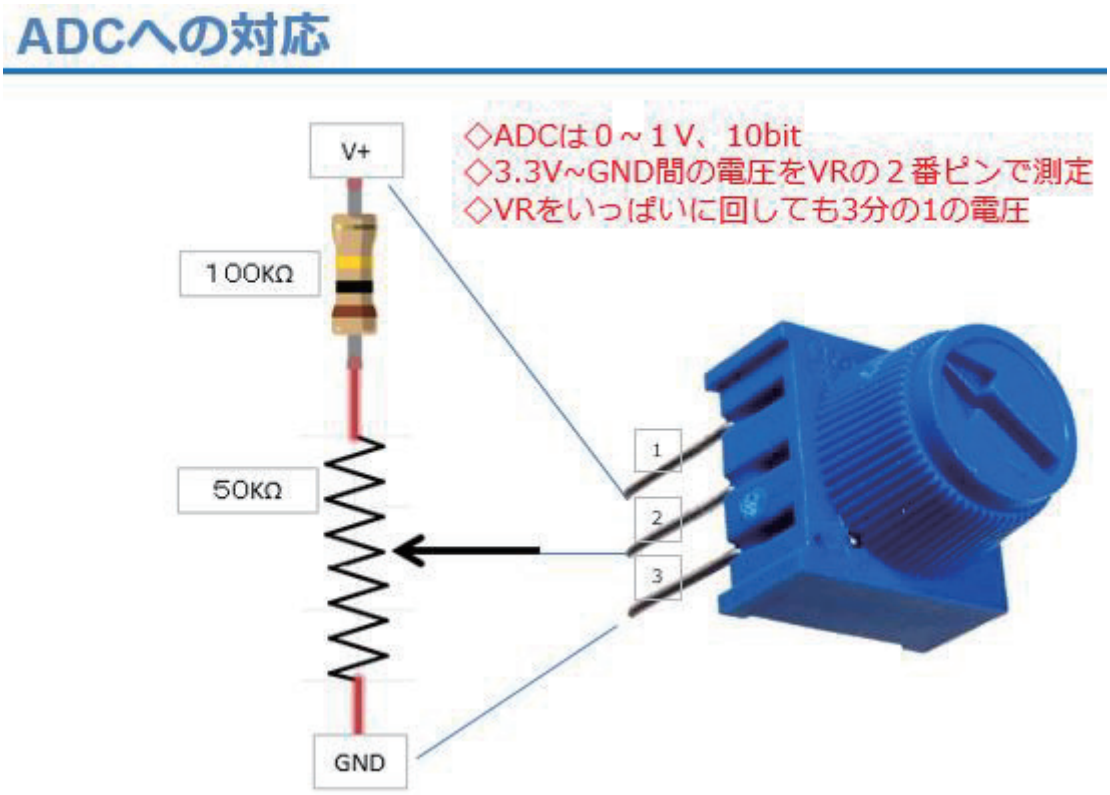


図 210

今回は、上図に示すように、50KΩの VR に 100KΩの抵抗を直列に接続して、VR をいっぱいに回しても 2 番ピンの電圧が V+の 1/3 になるようにします。V+に 3.3V を加えると 2 番ピンは最大で 1.1V の電圧になります。これは 1V よりも少し大きいのですが、本格的な計測で精度や信頼性に重点を置く場合は、この部分を丁寧に設計します。

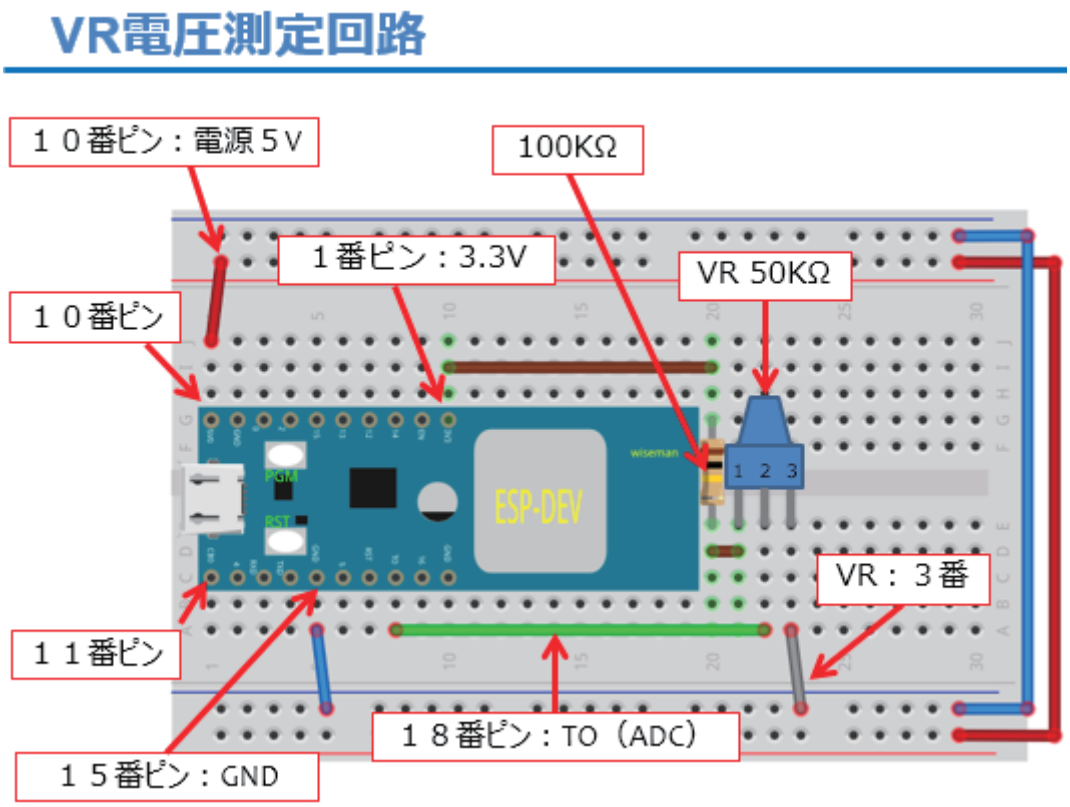


図　211

図に従い、配線をします。実際に配線したものが下の写真です。

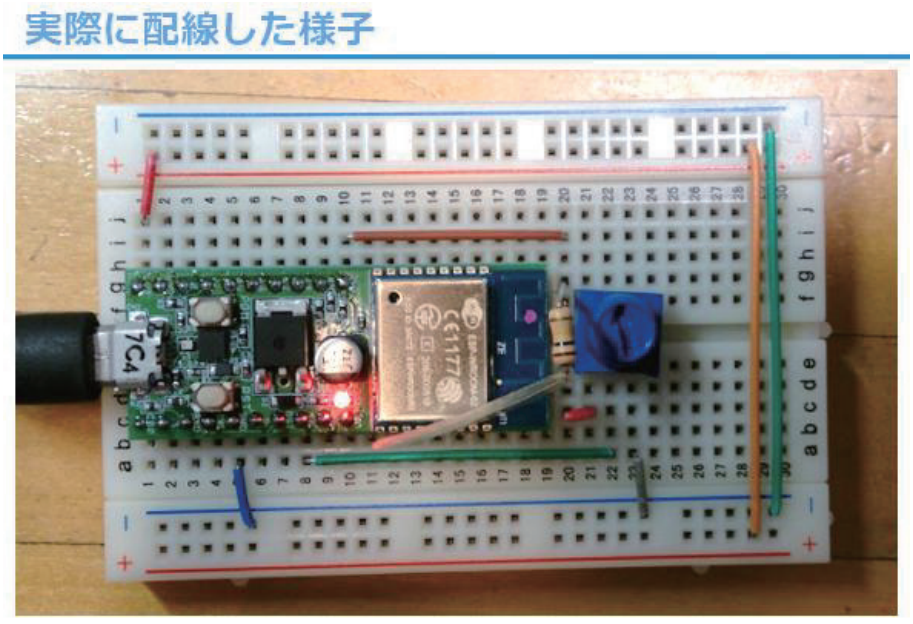


図　212

プログラムを書く

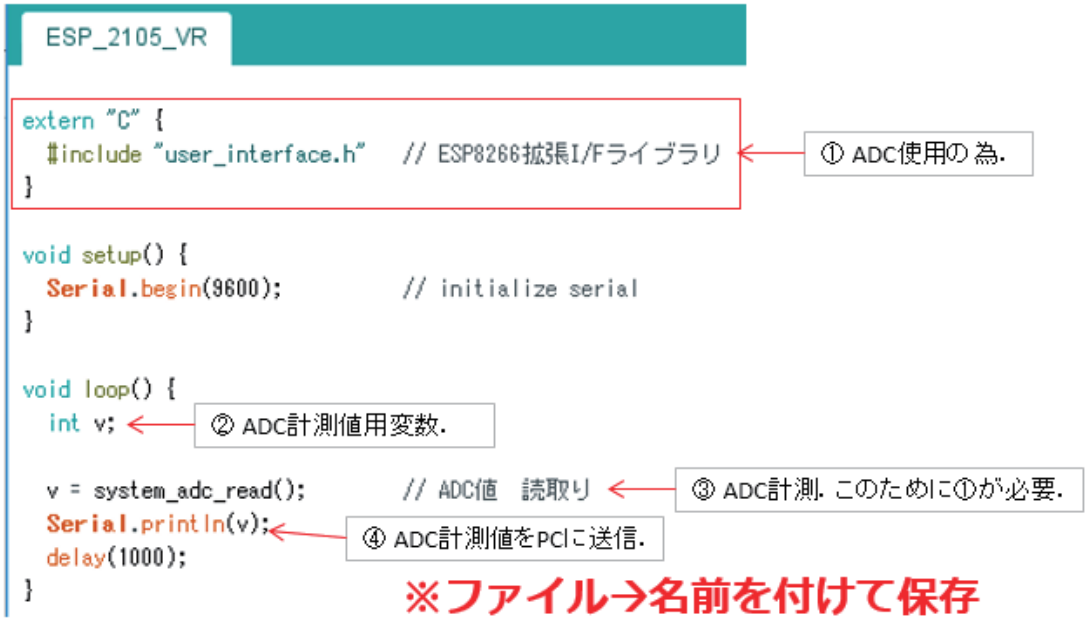


図 213

IDEを開き、新しいスケッチとして上図のソースコードを入力します。

①ADC利用のためヘッダ取り込み.

②ADCにより電圧の直読値を格納する変数

③実際にA/D変換を行い、計測巣する関数

④計測した値をそのままシリアル通信で送信（改行付き）

このプログラムの動作としては、loop()の最後にあるように、1秒間隔で電圧計測を行い、その値をPCにシリアル通信で送信するという動作を繰り返します。

ソースコードを入力したら、名前を付けて保存しておきましょう。

ここから、WiFiマイコンへの書込みまで一気に進めましょう。

以下の手順でプログラムのコンパイルから WiFi マイコンへの書き込みを行います。

①PC と WiFi マイコンの接続：USB ケーブルで接続

②シリアルポート（COM 番号）の確認：デバイスマネージャで確認

③シリアルポート（COM 番号）の設定：IDE で設定

④WiFi マイコンへの書込み準備：WiFi マイコンの SW 操作

⑤スケッチ（プログラム）のコンパイルから書込み：IDE で操作

書込みが終了すると書込み完了のメッセージが IDE に表示される

※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDE の下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなく WiFi マイコンの SW 操作を行い、もう一度 IDE の右向き矢印ボタンをクリックしてください。

書込みが終了すると、WiFi マイコンに Reset が掛かり、プログラムの実行が始まります。

動作確認

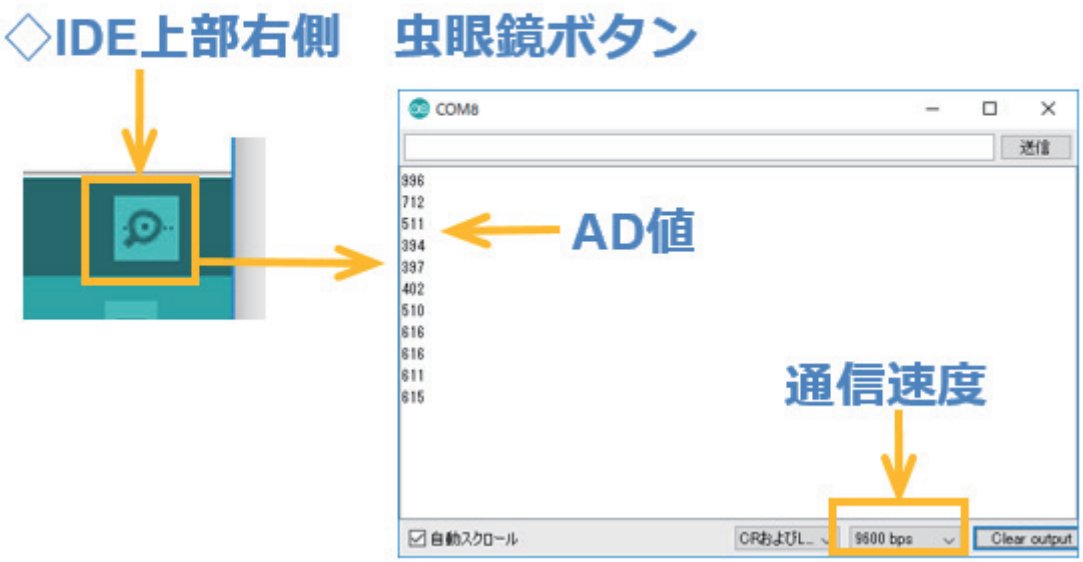


図　214

動作確認をしましょう。IDE の虫眼鏡マークのボタンをクリックしてシリアルモニターを起動します。**通信速度**を Serial.begin()で設定した速度に合わせます。すると、約 1 秒ごとに数値が表示され続けます。

これが VR によって分圧された電圧の A/D 変換値です。VR を回転してみると、表示される値が変化することが分かります。

ここで表示されるのは数値だけですので、【電圧であるという実感】が湧きませんね。そこで次の測定を行いました。

◇TO端子電圧とAD値の関係（実測）

V	AD
0.000	9
0.085	89
0.157	159
0.286	278
0.400	387
0.500	482
0.599	579
0.701	670
0.802	773
0.905	872
0.994	950

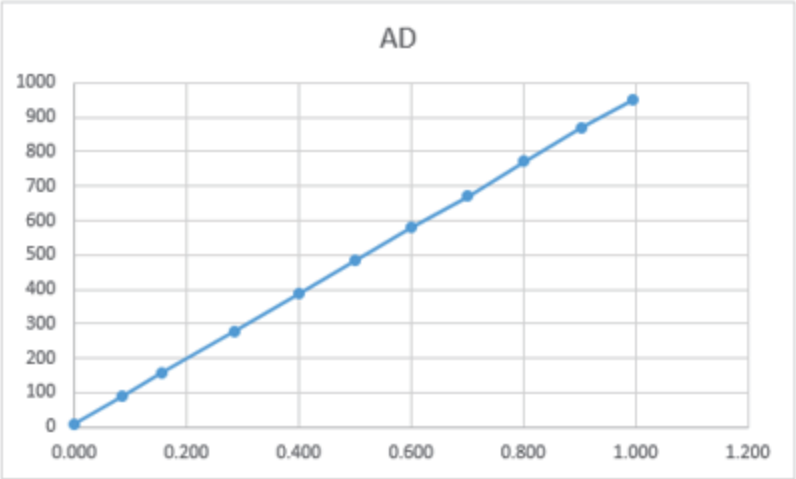


図 215

上図は、VR の 2 番ピンと GND 間の電圧をテスターで測りながら、その時シリアルモニターに表示される値を記録した表(図左)です。

その値をグラフにしたものが図右です。当然ですが直線になっていて【A/D 変換値とテスターで測った電圧が比例関係】にあることが分かります。

電圧への換算

◇10bit：1023＝1Vとして換算

V ＝ AD値÷1023

※使用する抵抗の精度や電源電圧が結果に影響します

図 216

【重要】

このA/D変換器は10bit仕様です。ADC直読値から電圧値を求めるためには、上図のような変換が必要になります。今回のシステムは、ADC直読値の表示だけでしたが、計算を行って電圧を表示するようにプログラムを改善すると電圧計になります。電圧が計測できるようになると、様々なセンサーを利用することができるようになります。

次の講座では、そのことを実証します。

第6回　温度センサー(アナログ)

センサー活用はIoT必須の要素です。温度センサーを利用した環境計測では【温度を測ること】は【分圧した電圧を測ること】と同じことです。電圧測定ではA/D変換値をそのままPCに送信していましたが、今回はA/D変換値を電圧に変換し、さらに温度にして通知します。

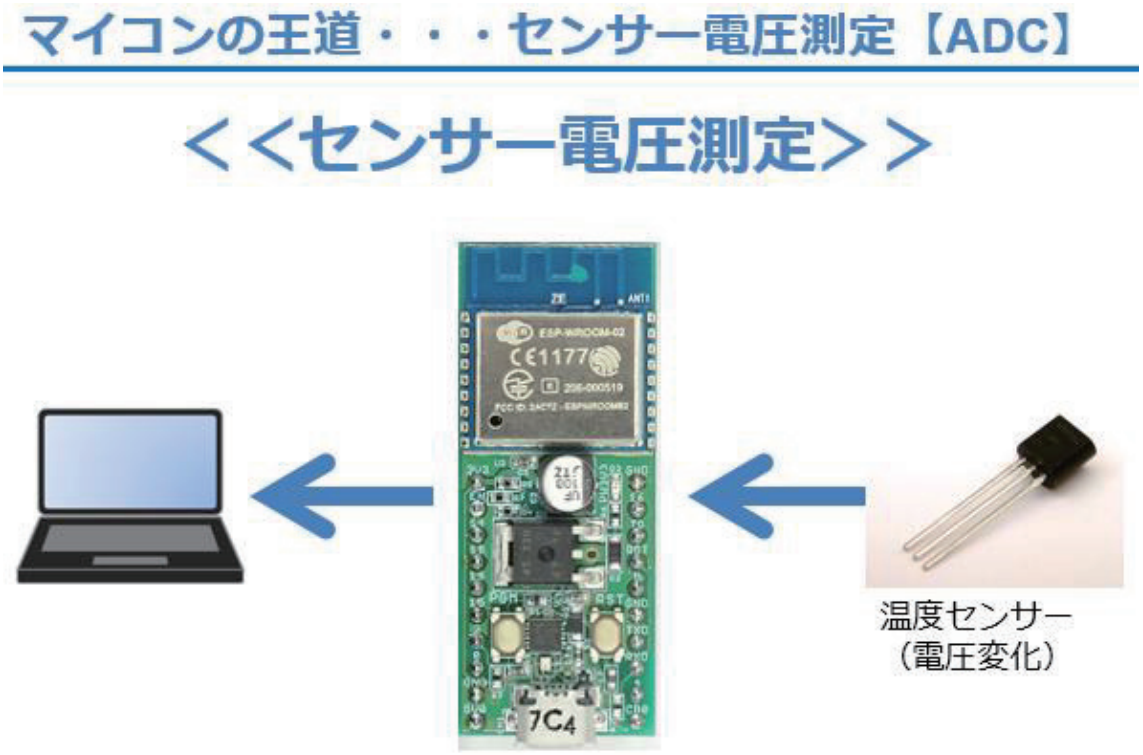


図　217

上図を見れば明らかなように、システムの外観としてはVRと温度センサーを取り換えたシステムを作ります。

◇全体構成とパーツ

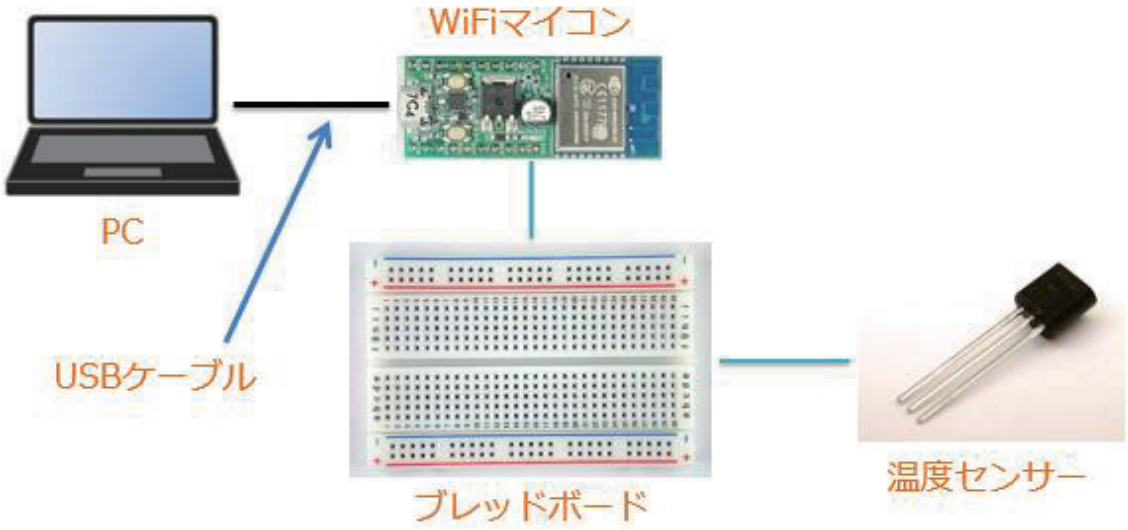


図 218

システムの全体構成を上図に示します。必要な機材・パーツは、下記です。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. 温度センサー（LM61CIZ）×1 個

温度センサー

◇LM61CIZ　リニアな特性

◇測定範囲：－30℃～100℃
－30℃＝300mV
～
0℃＝600mV
～
100℃＝1600mV

◇温度係数：＋10mV／℃

◇動作電圧範囲：＋2．7～＋10V

温度センサー（LM61CIZ）

図　219

使用する温度センサーの Data Sheet を抜粋したものが上図です。測定範囲は-30℃から 100℃と広範囲ですが、今回の実験では気温程度が測定できれば良いので十分すぎる測定範囲です。温度係数が+10mV/℃で温度と出力電圧の間に比例関係がありますので、センサー出力電圧から温度を求めるには、次のような計算を行えば求まります。

温度＝（センサー出力電圧－600mV）÷１０mV

図　220

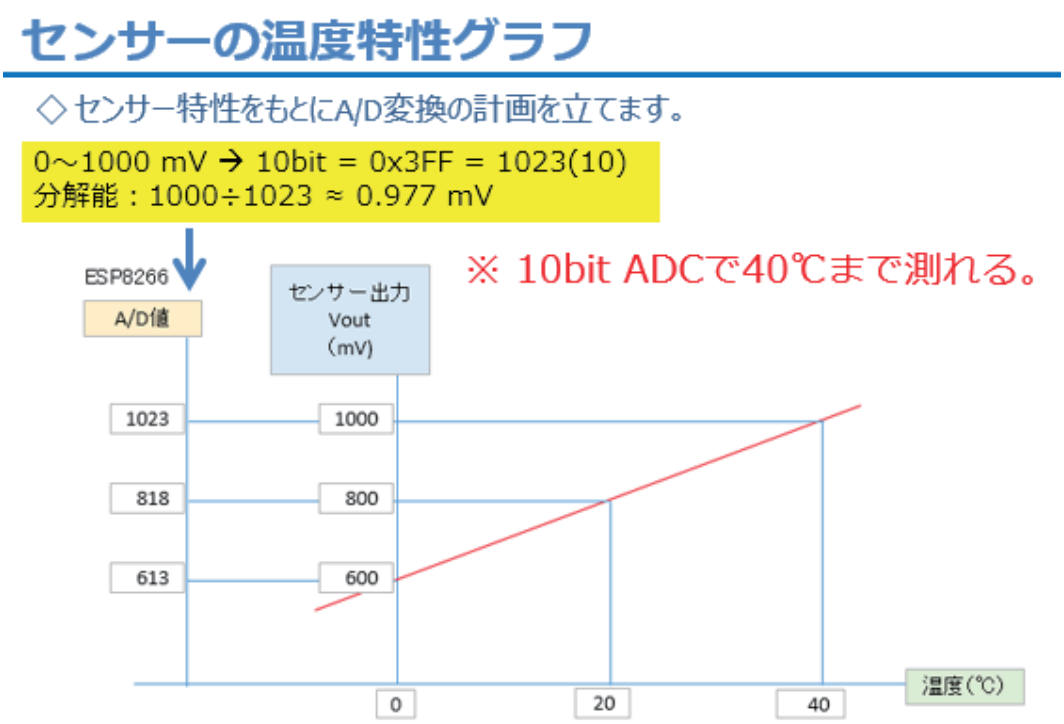


図　221

センサーの温度特性をグラフにすると上図のようになります。センサーの計測範囲はもっと広いのですが、今回の用途は気温計測なので、高くても 40℃です。室内で実験しますので氷点下にはなりませんので、上図の範囲で十分です。この時のセンサー出力電圧は、600~1000mV で、WiFi マイコンの ADC の仕様にちょうどマッチします。ADC の仕様から 1bit 当たりの分解能は、およそ 0.977mV です。これを基にして A/D 変換値から温度を求めるには次の計算を行います。

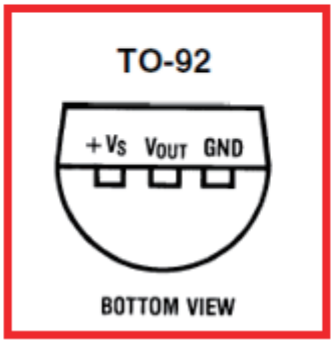
（AD値×分解能 ☆1 － 600mV ☆2）÷10 ＝ 温度（℃）

☆1　0.977mV
☆2　0℃のときの出力電圧

図　222

データシート

ピン配置で配線が分かる



+Vs：3.3V
Vout：出力
GND：0V

図 223

Data Sheet には、上図の様にピン配置が描かれています。BOTTOM VIEW と記載のあることで分かるように、センサーを脚ピン側から見た図です。蒲鉾を逆さにしたような形の平らな部分に型番などが印刷されています。図の左から順に

+Vs：電源--->3.3V 駆動が可能です。

Vout：出力--->WiFi マイコンの TO（18 番ピン）に接続します。

GND：0V（GND）

このピン配置と前回使用した VR とを比較したものが下図です。

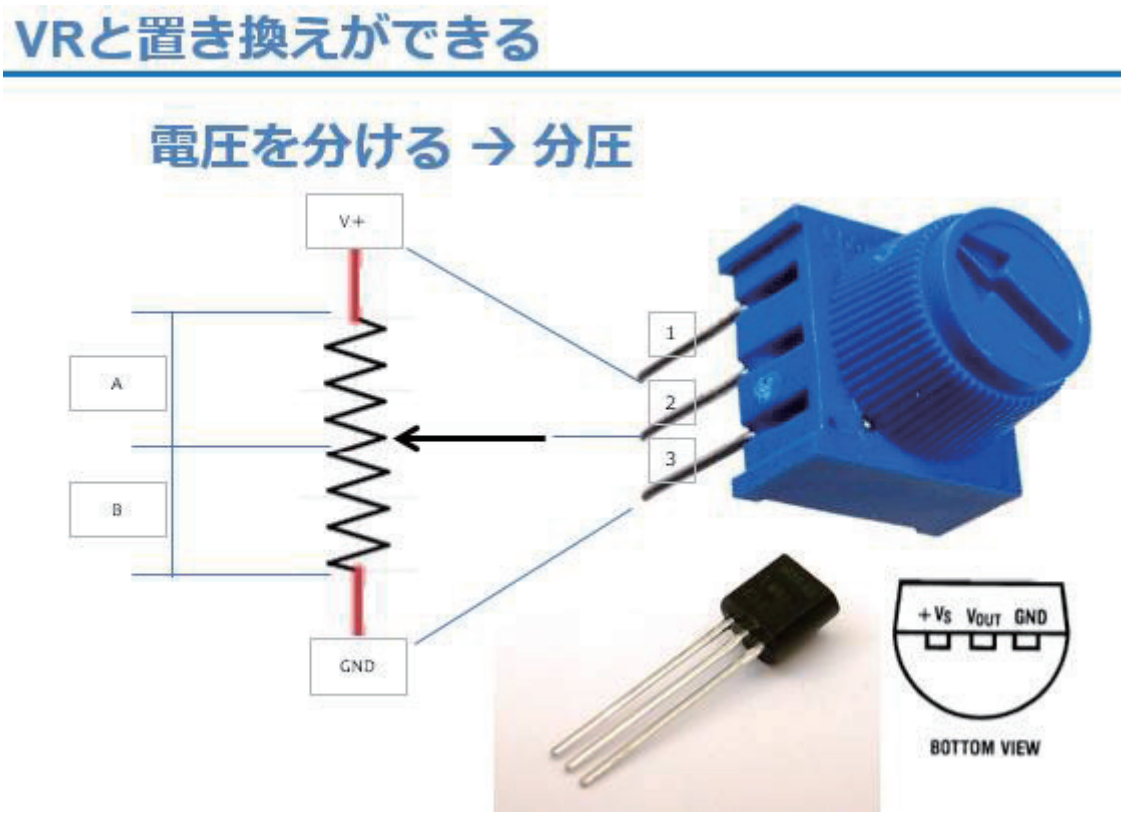


図　224

この温度センサーは、上図に示すように、前回使用した VR とピン配置が同じなので、差し替えが可能です。(向きに注意) そして計測対象の温度範囲では、出力電圧が 1V に収まるので、前回使用した 100KΩ の固定抵抗も必要ありません。直接 WiFi マイコンに接続できます。

センサー電圧測定回路

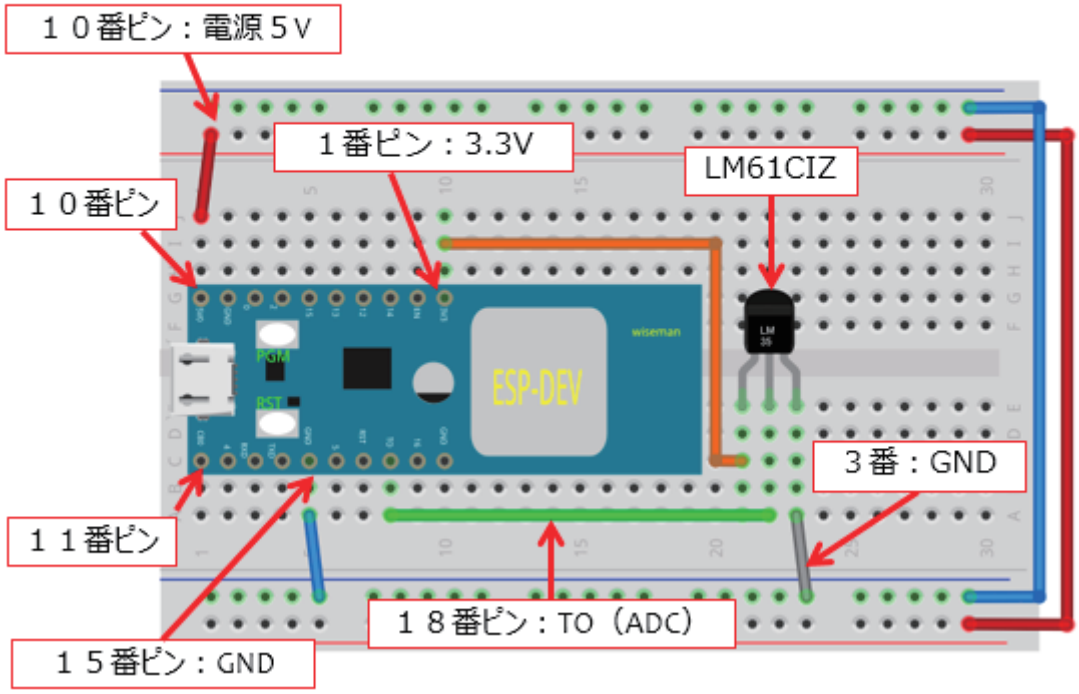


図　225

図に従い配線をします。

実際に配線した様子

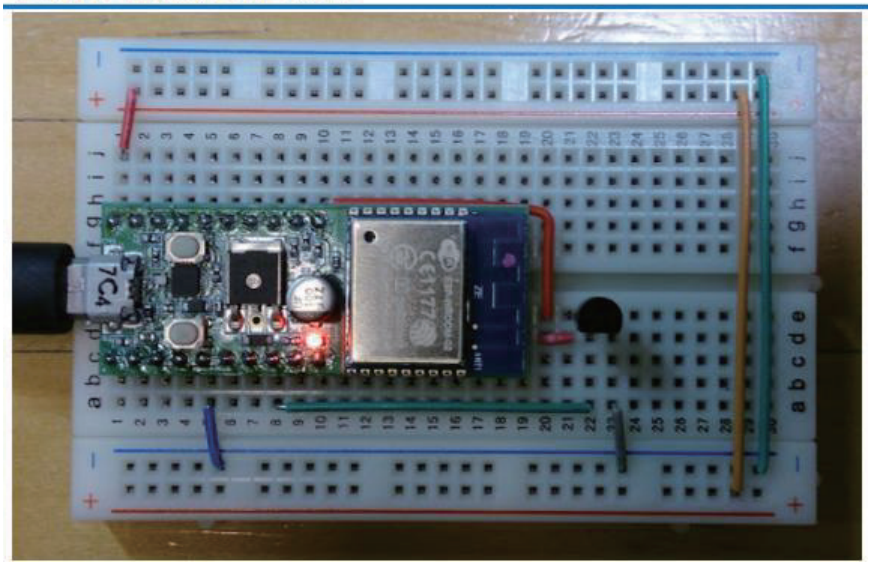


図　226

プログラムを書く

ESP_2106_Temp_Senser

```
extern "C" {
  #include "user_interface.h"    // ESP8266拡張I/Fライブラリ
}

void setup() {
  Serial.begin(9600);           // initialize serial
}

void loop() {
  int ad;
  float t;
  ad = system_adc_read();         // ADC値　読取り
  t = (ad * 0.977 - 600.)/10.;    // ADC値--->温度(℃)
  Serial.println("temp--->"+String(t));  // 温度表示
  delay(1000);
}
```

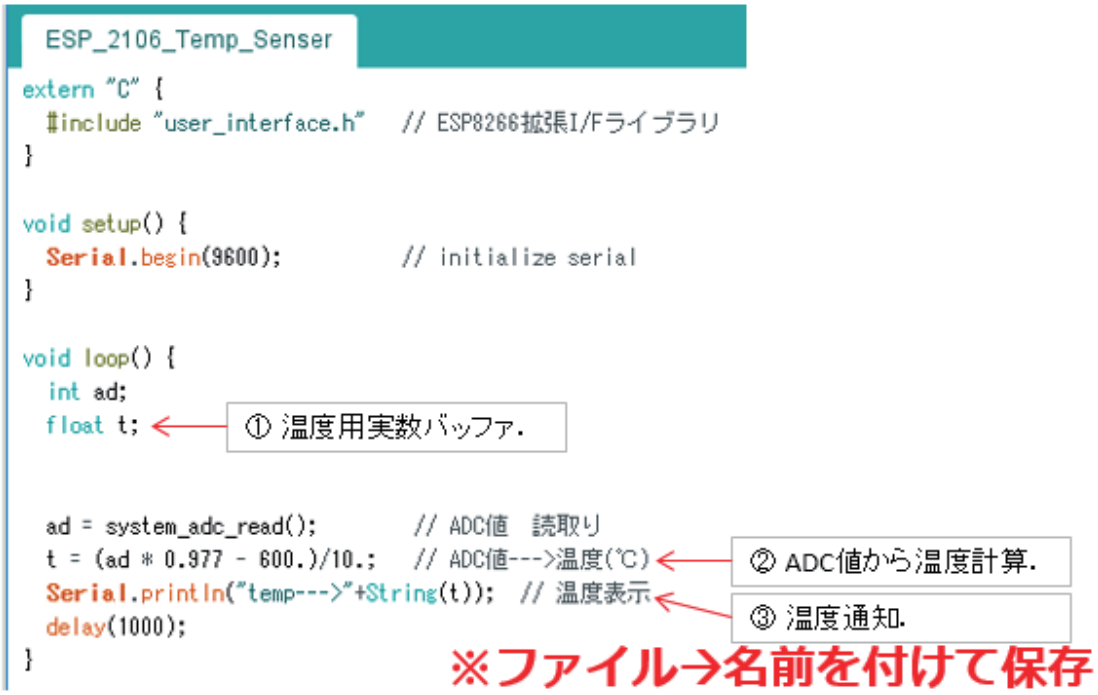


図　227

IDE を使い、ソースコードを入力します。

特に解説する部分は以下の通りです。

①温度は小数点以下まで求めているので実数型変数とする

②温度換算計算式は前に示した

③温度を文字列に変換して通知する

ソースコードを入力したら、名前を付けて保存してください。

ここから、WiFiマイコンへの書込みまで一気に進めましょう。

以下の手順でプログラムのコンパイルからWiFiマイコンへの書き込みを行います。

①PCとWiFiマイコンの接続：USBケーブルで接続

②シリアルポート（COM番号）の確認：デバイスマネージャで確認

③シリアルポート（COM番号）の設定：IDEで設定

④WiFiマイコンへの書込み準備：WiFiマイコンのSW操作

⑤スケッチ（プログラム）のコンパイルから書込み：IDEで操作

書込みが終了すると書込み完了のメッセージがIDEに表示される

※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。

書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作確認

図　228

IDE からシリアルモニターを起動して、**通信速度**を合わせます。モニターに温度が 1 秒間隔で表示されています。今回使用した温度センサーは精度があまり高くありません。ですから、実際に使用する場合は何度か計測して平均値表示したり、小数部の桁数を考慮したりするなどの工夫をすると良いでしょう。プログラムを改善してみてください。ここまで進んだ皆さんなら、容易にできるはずです。

次回のテーマも温度計測ですが、使用する温度センサーの使い方が全く異なる、デジタル温度センサーを使います。

第7回　温度センサー(デジタル)

前回使用した温度センサーは出力が電圧でした。出力される温度に対応した電圧はアナログ値なのでアナログセンサーと呼ばれたりもします。温度センサーの出力電圧を計測出来る A/D 変換器は、WiFi マイコンでは、18 番ピンだけなので、1 個のセンサーしか使えません。それでは、マイコン 1 個で 1 計測となってしまいます。

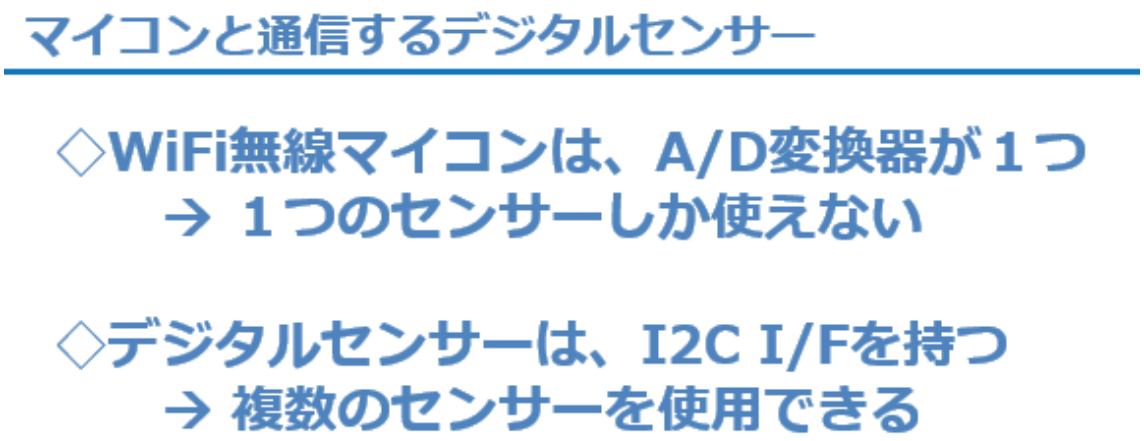


図　229

今回使用するデジタルセンサーは、I2C I/F を持つ温度センサーです。I2C は、日本ではアイ・ツー・シーと呼ぶことが多いのですが、正式には Inter-Integrated Circuit の略で、I-squared-C（アイ・スクエアド・シー）のことです。I2C I/F は、フィリップス（オランダ）という会社が開発したものです。シリアルデータ（SDA）とシリアルクロック（SCL）という 2 本の信号線に、複数（同じ種類でも異なる種類でも可）のデバイスを接続できる、シリアル通信の仲間です。これを利用すれば、1 つのマイコンで多数のセンサーが使えます。

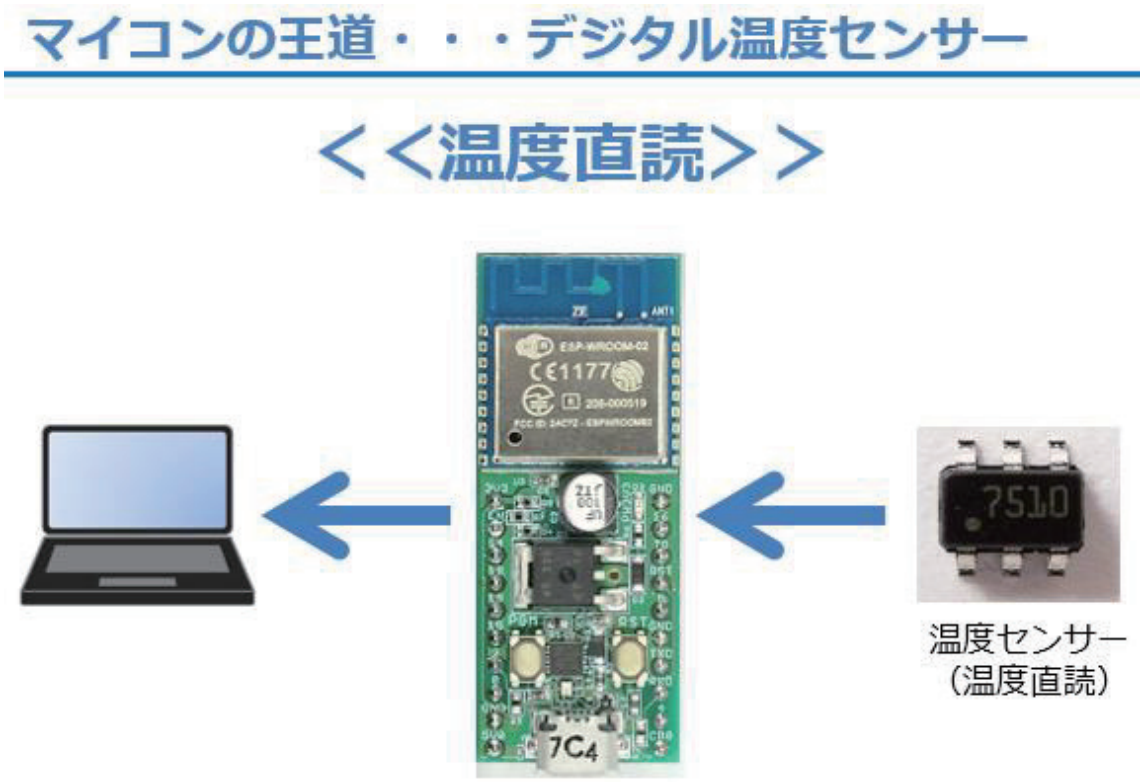


図　230

このシステムで利用するデジタル温度センサーは、図右の様に、足が6本あります。アナログ温度センサーと違い、直接温度を読むことができるので、電圧から温度への換算などは行わなくてよいという利点があります。このデジタル温度センサーから温度を直接読み込んで、PCに通知するものです。大がかりなデジタル温度計と考えて良いと思います。

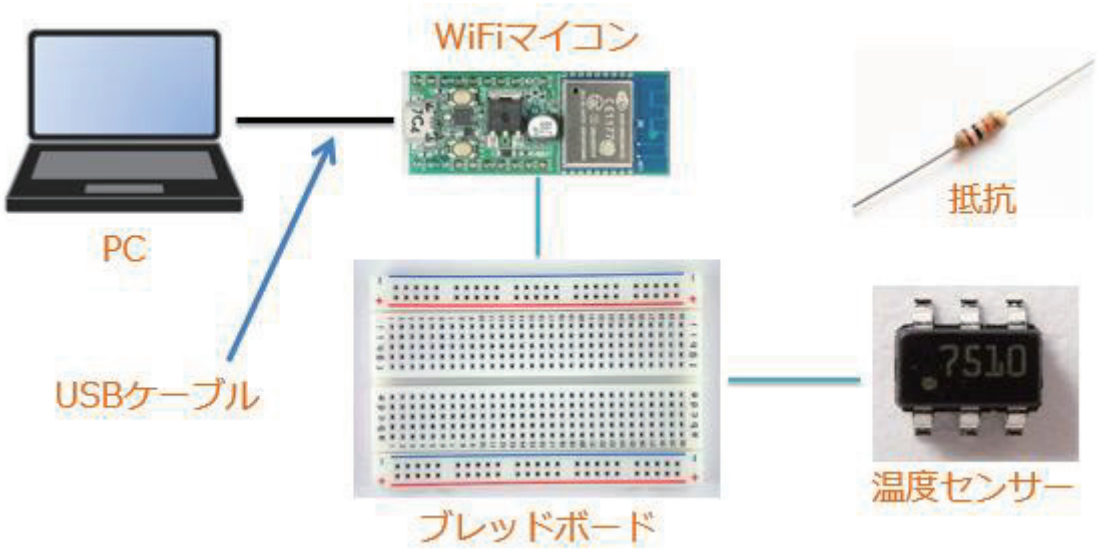


図 231

システムの全体構成を上図に示します。必要な機材・パーツは、下記です。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×1 個
5. 配線用ジャンパー線×適宜
6. デジタル温度センサー（STTS751）×1 個
7. 抵抗（10KΩ）×2 個（I2C I/F の Pull Up に使用）

以下、デジタルセンサー：STTS751 を少し詳しく説明します。

デジタル温度センサー　STTS751

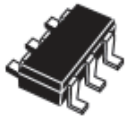
STTS751

2.25 V low-voltage local digital temperature sensor

Features

- Operating voltage 2.25 V to 3.6 V
- Operating temperature –40 °C to +125 °C
- Programmable
  - 10 different conversion rates
    0.0625 to 32 conversions/sec.
    1 conversion/sec. - default
  - 4 different resolutions
    9-bit (0.5 °C/LSB) to12-bit (0.0625 °C/LSB)
    10-bit (0.25 °C/LSB) - default
- Low supply current
  - 50 µA (typ) for 8 conversions/sec.
  - 20 µA (typ) for 1 conversion/sec.
  - 3 µA (typ) standby
- Accuracy
  - ±1.0 °C (typ) 0 °C to +85 °C
  - ±2.0 °C (typ) –40 °C to +125 °C
- One-shot mode for power saving
- Fast conversion time 21 ms (typ) 10-bit
- Pull-up resistor value allows single pin to select one of four slave addresses
- Supports 400 kHz serial clock

UDFN-6L



SOT23-6L

- SMBus 2.0 compatible
  - SMBus ALERT (ARA) support
  - SMBus timeout
- RoHS/green

Applications

- Solid state drives
- Portable electronics
- Notebook computers
- Smart batteries
- Servers
- Telecom

図　232

上図は、今回使用するデジタル温度センサー（STTS751）の Data Sheet です。駆動電圧は 2.25～3.6V で WiFi マイコンの 3.3V を利用することができます。また計測範囲は-40℃～+125℃と広範囲です。プログラムによって 4 種類の分解能が選択できます。また、1 秒間に 8 回の計測で流れる電流は 50μA で、とても小さな値です。

デジタル温度センサー　STTS751

Figure 2.　Pinout - SOT23-6L and UDFN-6L

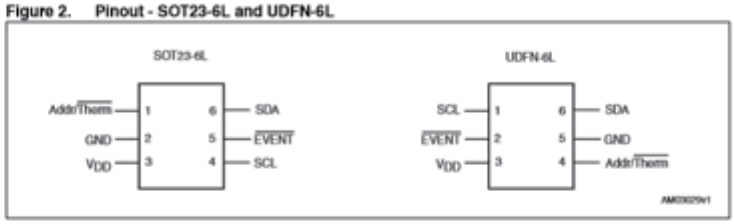


Table 3.　Pin descriptions

Pin		Name	Description
SOT23-6L	UDFN-6L		
1	4	Addr/Therm	Open-drain output that can be used to turn on/off a fan or throttle a CPU clock in the event of an overtemperature condition. The pin at power-up determines the SMBus slave address according to the pull-up resistor value as shown in *Table 1*. This pin must have a pull-up resistor connected to the same voltage as V_{DD} or tied to GND (pin cannot float). Total capacitance on this pin must be <100 pF. Note: By tying Addr/Therm to ground, the device functions as one address device only. The Therm functionality is then not available. The address for device STTS751-0 is 72h and the address for device STTS751-1 is 76h.
2	5	GND	GND
3	3	V_{DD}	Power supply V_{DD}
4	1	SCL	SMBus clock
5	2	EVENT	Open-drain interrupt output. Output supports the SMBus Alert (ARA). Note: This pin may not float.
6	6	SDA	SMBus data input/output

マイコン
16Pin
12Pin

図　233

今回使用するセンサーのパッケージは上図左のものです。センサーの4番ピン（SCL：I2Cのクロック）と6番ピン（SDA：I2Cのデータ）をそれぞれWiFiマイコンの16番ピン、12番ピンに接続します。

Figure 4.　Application hardware hookup

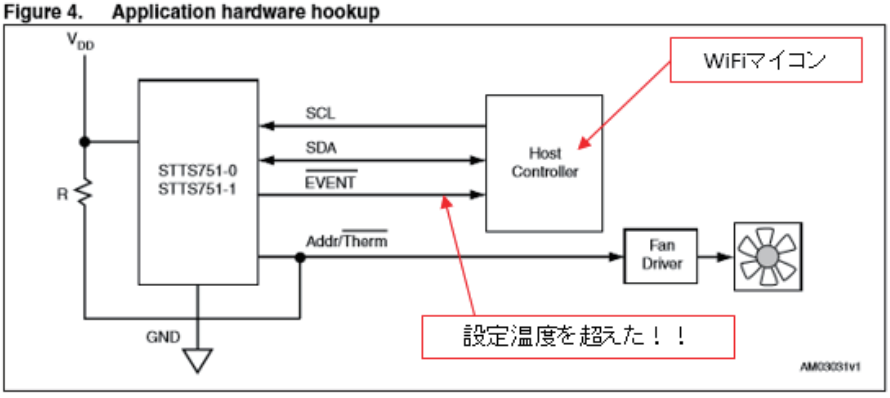


図　234

設定温度に対する判断の信号(EVENT)も準備されています。（上図）

デジタル温度センサー　STTS751

4　STTS751 register summary

The STTS751 uses 8-bit registers. Variables longer than 8 bits are managed in byte pairs. For example, when reading a 10-bit temperature value (10 bits is the default resolution.) the application must read two registers and then concatenate the upper byte with the 2 most significant bits of the lower byte.

Table 9 below summarizes the register map for the device. Accessing any invalid address results in indeterminate data.

Table 9.　Registers/pointers

Address pointers (h)	STTS751 register map			Power-up default values binary (dec)
	Device registers name	Size	Type	
00	Temperature value high byte	8	R	undefined
01	Status	8	R	undefined
02	Temperature value low byte	8	R	undefined
03	Configuration	8	R/W	0000 0000
04	Conversion rate	8	R/W	0000 0100

図　235

このセンサーはレジスタが 5 つありますが、そのうち次の 3 つを使います。

00 番：温度の整数部を取り出すレジスタです。

02 番：温度の小数部を取り出すレジスタです。

03 番：設定用レジスタ。初期化で使用します。

デジタル温度センサー　STTS751

4.2 Temperature register format

The temperature data is a 12-bit number and is stored in two's complement format spanning the high byte and low byte registers as shown in *Table 11*.

Table 11. Temperature register (two's complement)

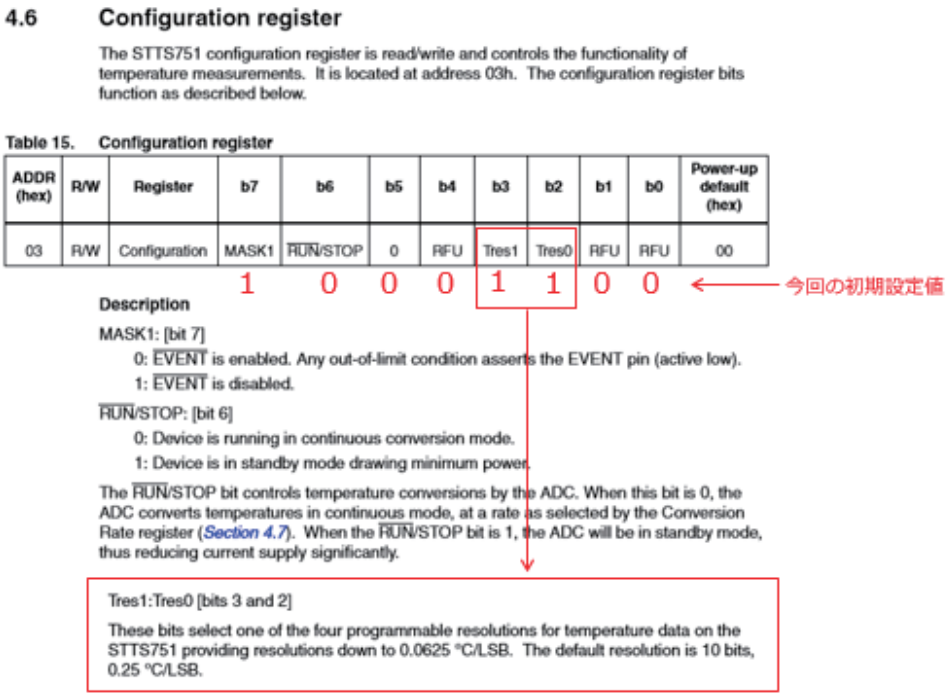
ADDR (hex)	R/W	Register	b7	b6	b5	b4	b3	b2	b1	b0	Power-up default (hex)
00	R	Temperature - high byte	sign	64 °C	32 °C	16 °C	8 °C	4°C	2 °C	1 °C	00
02	R	Temperature - low byte	½ °C	¼ °C	$^{1}/_{8}$ °C	$^{1}/_{16}$ °C	0	0	0	0	00

The integer portion of the temperature is stored in the high byte, and the fractional portion in the low byte. The lower four bits of the low byte will always read 0. At power-up, the STTS751 defaults to 10-bit resolution. Thus, bits b5 and b4 of the lower byte will also read 0 until the device is configured to a higher resolution (via the Tres bits in the configuration register).

図　236

00 番レジスタは、1bit あたり 1℃の温度を通知してくれますので、読んだ値をそのまま使用できるのですが、最上位の b7 が sign bit となっていて、two's complement（2 の補数）ですから、氷点下の場合は、そのことに配慮が必要ですが、今回の計測対対象は室温ですので、直読値をそのまま利用できます。

02 番レジスタは、小数部の温度を通知してくれますが、下位 4bit が 0 となっているので、読み込んだのち右に 4bit シフトして、LSB（Least Significant Bit：最下位 bit または分解能を意味する）の 1/16℃を掛け算して、小数点以下の温度を求める処理が必要です。



4.6 Configuration register

The STTS751 configuration register is read/write and controls the functionality of temperature measurements. It is located at address 03h. The configuration register bits function as described below.

Table 15. Configuration register

ADDR (hex)	R/W	Register	b7	b6	b5	b4	b3	b2	b1	b0	Power-up default (hex)
03	R/W	Configuration	MASK1	RUN/STOP	0	RFU	Tres1	Tres0	RFU	RFU	00
			1	0	0	0	1	1	0	0	← 今回の初期設定値

Description

MASK1: [bit 7]

0: EVENT is enabled. Any out-of-limit condition asserts the EVENT pin (active low).

1: EVENT is disabled.

RUN/STOP: [bit 6]

0: Device is running in continuous conversion mode.

1: Device is in standby mode drawing minimum power.

The RUN/STOP bit controls temperature conversions by the ADC. When this bit is 0, the ADC converts temperatures in continuous mode, at a rate as selected by the Conversion Rate register (Section 4.7). When the RUN/STOP bit is 1, the ADC will be in standby mode, thus reducing current supply significantly.

Tres1:Tres0 [bits 3 and 2]

These bits select one of the four programmable resolutions for temperature data on the STTS751 providing resolutions down to 0.0625 °C/LSB. The default resolution is 10 bits, 0.25 °C/LSB.

図　237

Configuration レジスタは、センサーの使い方を指示するものです。図に今回の設定値を書いておきました。1 にする bit を説明します。

b7：MASK1 は、1 にすると、EVENT 信号（指定した温度を超えたかどうかを判断する信号）を使わない設定です。今回は使用しません。

（※実は配線しなければ 0 でも 1 でもどちらでも良いのですが。）

b3,b2：Tres1:0 となっています。この bit は、プログラムで設定可能な分解能 4 つの内の 1 つを選ぶと書いてあります。この bit を 11 とするとどうなるかは次の資料を見てください。

Table 16.　Conversion resolution

Tres1:Tres0	Temperature resolution	LSB step size (°C)
00	10 bits (default)	0.25
01	11 bits	0.125
11	12 bits	0.0625
10	9 bits	0.5

Table 11.　Temperature register (two's complement)

ADDR (hex)	R/W	Register	b7	b6	b5	b4	b3	b2	b1	b0	Power-up default (hex)
00	R	Temperature - high byte	sign	64 °C	32 °C	16 °C	8 °C	4°C	2 °C	1 °C	00
02	R	Temperature - low byte	½ °C	¼ °C	1/8 °C	1/16 °C	0	0	0	0	00

図　238

Tres1:0 を 11 にセットすると、図の下に示すように 12bit の分解能となります。（Tres は、Temperature Resolution の短縮形らしい）

これらにより温度処理は次の様になります。

◇12bit で温度測定
◇温度は、整数部と小数部に分かれている
　→整数部と小数部は別に処理
◇小数部は、b7~b4 の4bit
◇小数部は、 1 LSB＝1/16℃（0.0625℃）
　→読取り後、4bit右シフトして×0.0625する

① 11110000　→　00001111
② 00001111×0.0625＝℃（小数部）

図　239

【重要】I2C I/F では、多くのデバイスが同じ信号線につながるので、通信するデバイスを識別するために、【スレーブアドレス】が決められます。このアドレスをプログラムで使用します。

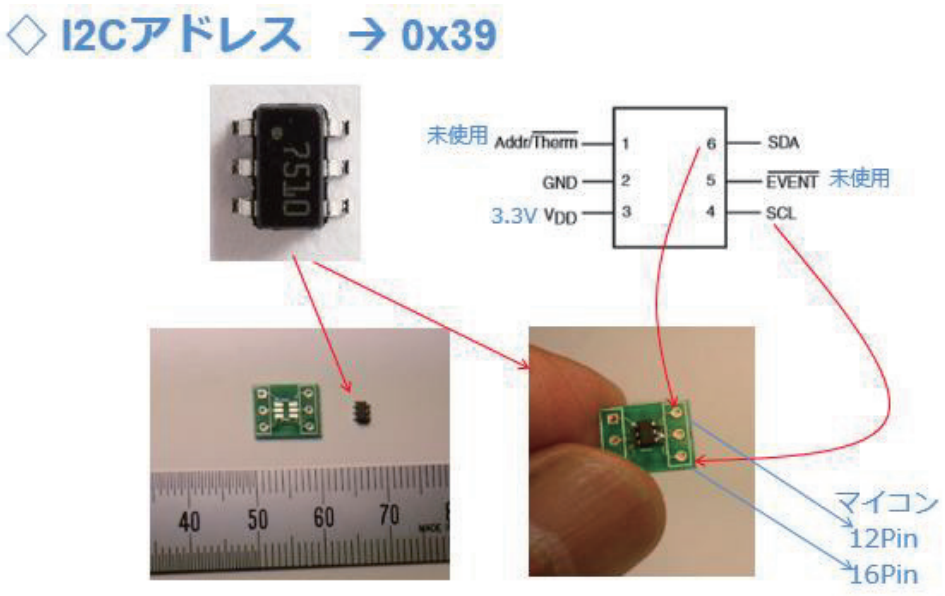


図 240

実際のデジタル温度センサー（STTS751）は図のような小さなものです。単体では 2×1mm 程度の大きさで、そこに 6 本の端子が出ていますが、そのままではブレッドボードでは使えませんので、図右下のように小さな基板に半田付けしたものを使用します。よく見るとセンサーの背面左上に小さな丸印があり、基板の左上にも白い丸印があるので、これを合わせて使用します。回路を作成する際も、この小さな丸印を基準にピン配置を考えながら配線します。

【重要】次に示す配線図では、I2C I/F の SCL と SDA の信号は、10KΩの抵抗で 3.3V に Pull Up する部分があります。見落とさないでください。Pull Up というのは、第 2 回で解説した Pull Down の反対です。今回は常時 High レベルにしておきたい信号に使用しています。

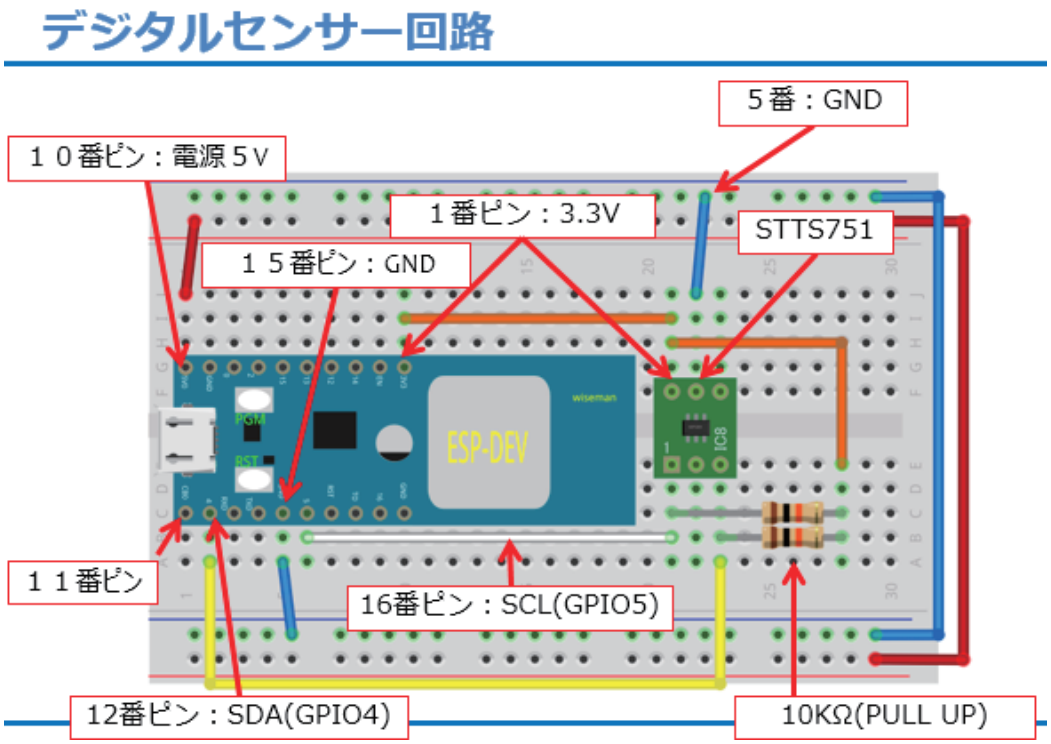


図　241

上図を見て回路を作成してください。完成したものが下図です。

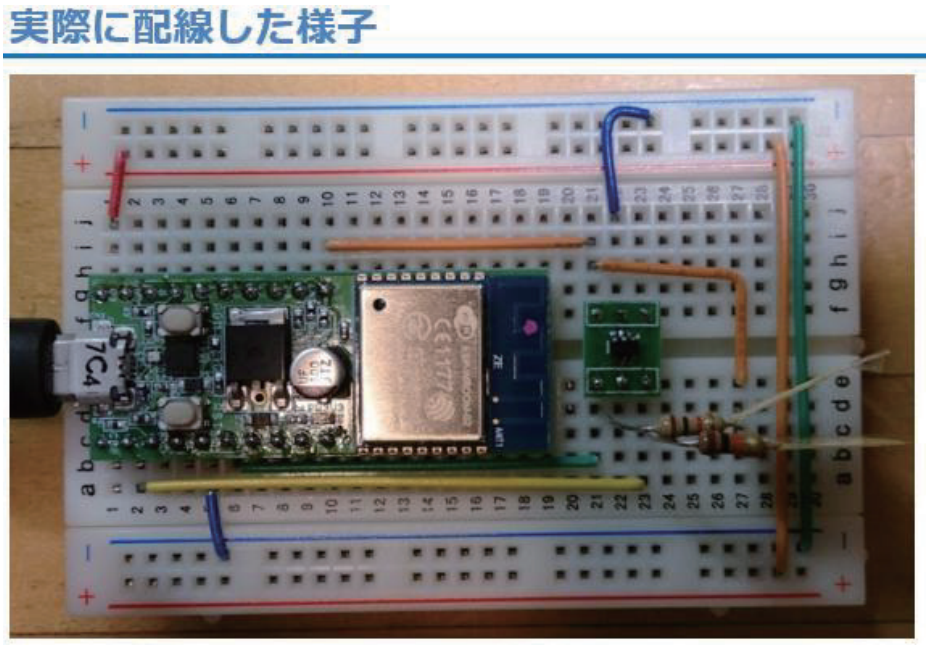


図　242

プログラムを書く

ESP-2107_Temp_Sensor_STTS751

```
#include <Wire.h>                          ← ① I2Cデバイス通信のライブラリヘッダ.

#define STTS751_ADRS  0b0111001            ← ② センサーのスレーブアドレス.

void setup()
{
    Serial.begin(115200);                  ← ③ 今回は早めのシリアル通信速度(対PC).
    Wire.begin();                          ← ④ I2C I/Fの初期化.
    Wire.beginTransmission(STTS751_ADRS);  ← ⑤ I2C 通信開始.
    Wire.write(0x03);                      // config reg.   ← ⑥ 03レジスタ指定.
    Wire.write(0b10001100);                // EVENT停止、12bit分解能指定
    Wire.endTransmission();                ← ⑦ 初期設定.
}                                          ← ⑧ I2C通信終了.
```

図 243

IDE にプログラムを入力して下さい。

①I2C でバイスとの通信に必要なライブラリを利用するためのヘッダ

②使用するセンサーの I2C スレーブアドレス（0x39）

③シリアル通信初期化(速度 115200bps)

④I2C I/F 初期化

⑤センサー初期化のため I2C 通信開始

⑥03 レジスタ設定

⑦初期設定

⑧I2C 通信終了

【重要】

※上の様に I2C デバイスと通信を行う際は、初めに I2C スレーブアドレスを指定して通信を開始します。

プログラムを書く　loop()　前半

```
void loop()
{
    byte valueHigh,valueLow; ←① 温度用変数.
    int16_t temp; ←② 小数部温度処理用変数.

    // 整数部をセンサーから取得
    Wire.beginTransmission(STTS751_ADRS);←③ I2C通信開始.
    Wire.write(0x00);         // high byte of Temp.←④ 温度整数部レジスタ指定.
    Wire.endTransmission();←⑤ I2C通信終了.
    Wire.requestFrom(STTS751_ADRS, 1);←⑥ I2C通信にて1byte読込リクエスト.
    valueHigh = Wire.read();←⑦ 温度(整数部) 読込.

    // 整数部をＰＣへ出力
    Serial.print(valueHigh);←⑧ 温度整数部をPCに通知.
    Serial.print(".");←⑨ 小数点.
```

図　244

loop()での処理は 2 つに分けて説明します。

①温度を整数部と小数部別々に読み込むための変数

②小数部を PC に通知する際、1 桁ごとに処理するための変数

③I2C 通信開始。スレーブアドレス指定。

④00 番レジスタ指定（温度整数部）

⑤I2C 通信終了

⑥I2C 通信で 1byte（温度整数部）読込リクエスト

⑦I2C 通信で 1byte 読込

⑧整数部温度を PC に通知（改行無し）

⑨続けて小数点を通知（改行無し）

プログラムを書く　loop()　後半

```
    // 少数部をセンサーから取得
    Wire.beginTransmission(STTS751_ADRS);
    Wire.write(0x02);          // low byte of temp.
    Wire.endTransmission();
    Wire.requestFrom(STTS751_ADRS, 1);
    valueLow = Wire.read();

    // 少数部をＰＣへ出力
    temp = (valueLow >> 4) * 625;  // LSB:0.0625
    Serial.print(temp/1000);
    temp %=1000;
    Serial.print(temp/100);
    temp %=100;
    Serial.print(temp/10);
    temp %=10;
    Serial.print(temp);
    Serial.print("¥r¥n");
    delay(1000);
}
```

① I2C通信開始.
② 温度小数部レジスタ指定.
③ I2C通信終了.
④ I2C通信にて1byte読込リクエスト.
⑤ 温度(少数部)読込.
⑥ 小数部データの加工.
⑦ 少数第1位通知.
⑧ 少数第2位以下取り出し.
⑨ 少数第4位通知.
⑩ 改行.

※ファイル→名前を付けて保存

図　245

上記で特に説明をする部分は、少数部を PC へ出力している部分です。

⑥小数部温度のデータを 4bit 右シフトして分解能 625 を掛ける

⑦少数第 1 位出力

⑧余りを採って以下の桁の処理を続ける

⑨少数第 4 位出力

⑩改行を出力

ソースコードを入力したら、名前を付けて保存してください。

ここから、WiFiマイコンへの書込みまで一気に進めましょう。

以下の手順でプログラムのコンパイルからWiFiマイコンへの書き込みを行います。

①PCとWiFiマイコンの接続：USBケーブルで接続

②シリアルポート（COM番号）の確認：デバイスマネージャで確認

③シリアルポート（COM番号）の設定：IDEで設定

④WiFiマイコンへの書込み準備：WiFiマイコンのSW操作

⑤スケッチ（プログラム）のコンパイルから書込み：IDEで操作

書込みが終了すると書込み完了のメッセージがIDEに表示される

※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。

書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作確認

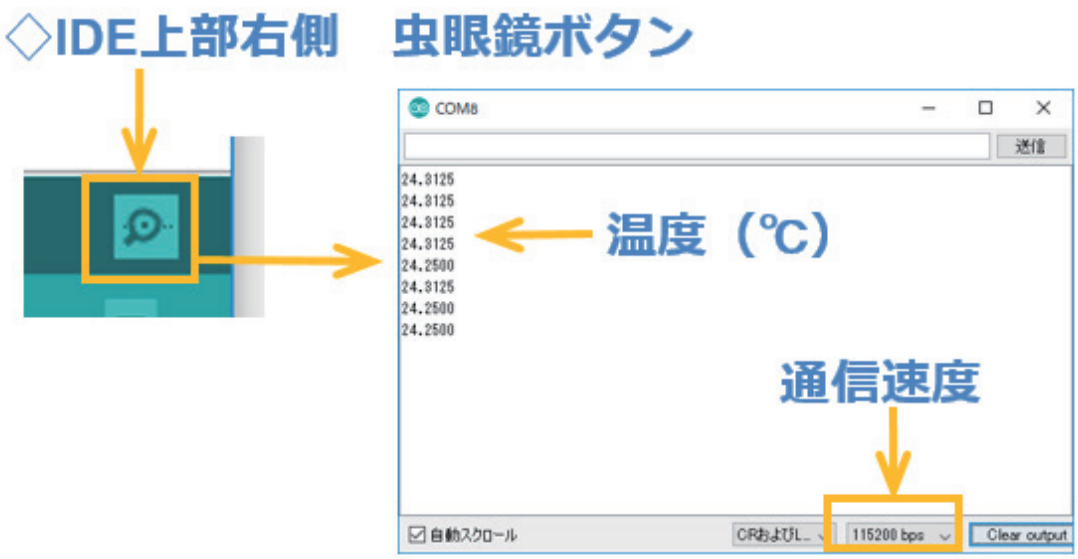


図　246

シリアルモニターを起動して、通信速度を合わせてください。今回は少し早めの 115200bps としました。通信速度を合わせると、1 秒毎に温度が表示されていきます。温度センサーの背中に軽く指を触れてみてください。温度が変化することが分かるでしょう。

今回は、センサーの分解能である 1/16℃まで表示していますが、実際に使用する場合は、必要な表示桁数などの検討をしてください。

【重要】

このセンサーは、デジタルセンサーなので、I2C I/F 上に複数デバイスを接続できます。次の講座では、I2C I/F で制御する液晶表示器を同じ I/F に接続します。

第8回　液晶表示器

これまで、WiFiマイコンからの表示装置としてPCのシリアルモニターを使っていたわけですが、これからはWiFiマイコン単体でも表示ができるように、液晶表示器（LCD：Liquid Crystal Display）を使ってみましょう。前回使用したデジタル温度センサーはI2C I/Fを使用して温度を読み込む入力デバイスでしたが、今回は表示データを書込んで目に見える文字情報を表示する出力デバイスとして液晶表示器を使います。

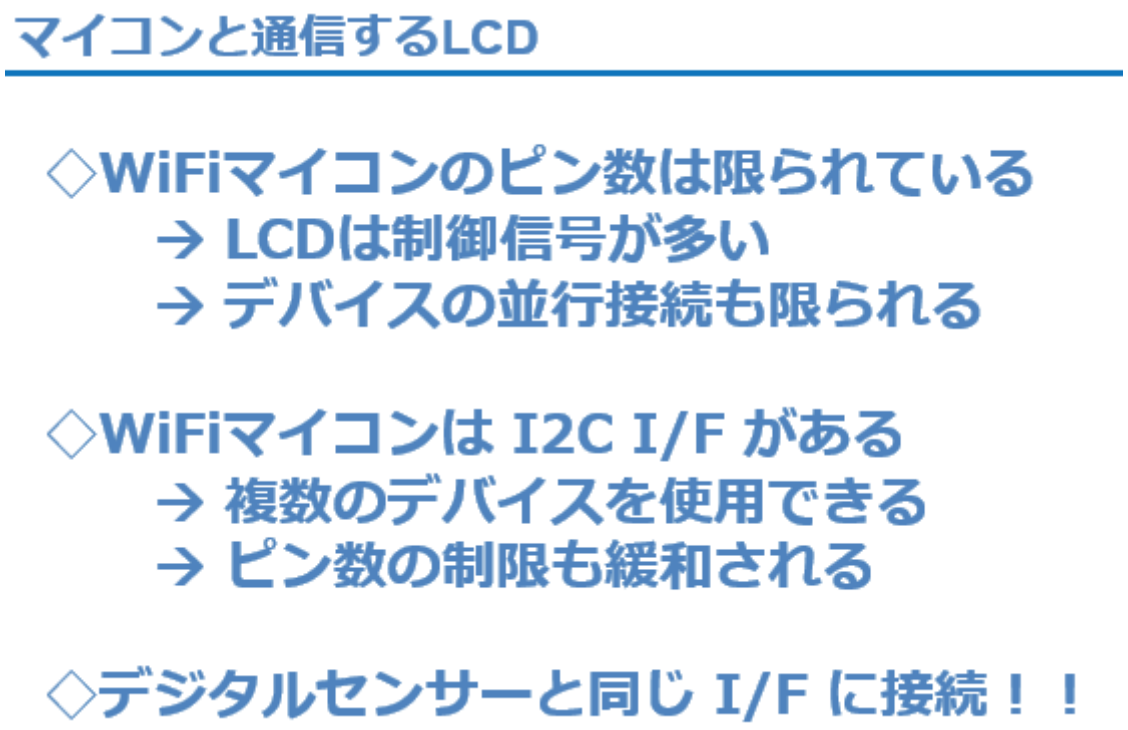


図　247

マイコンの王道・・・表示機能【LCD】

＜＜LCD＞＞

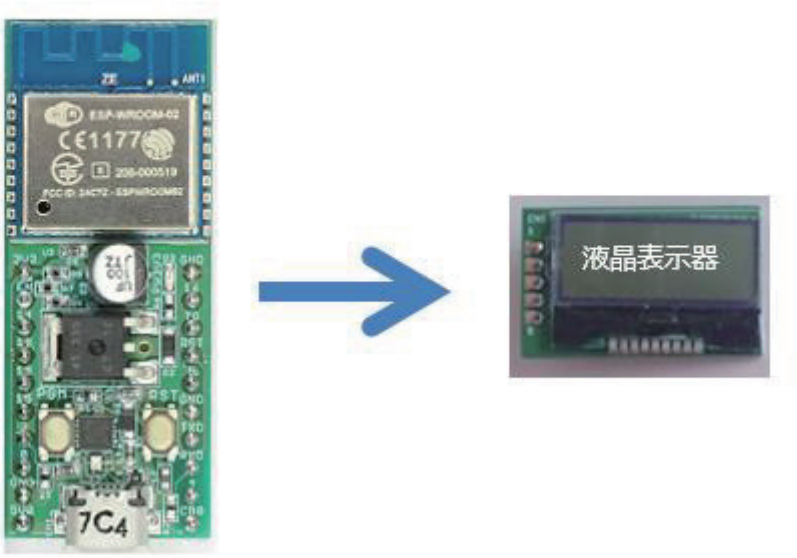


図　248

このシステムで利用する液晶表示器は、図右の様なものです。今回は使い方を学ぶことに重点を置いて、まずは固定の文字列を表示してみましょう。シリアル通信の【送信】と同じ考え方です。固定の文字列が表示できれば、標示する文字列をダイナミックに変化するデータで編集することで、【色々な情報を外部に伝える重要な仕事】が行えるようになります。

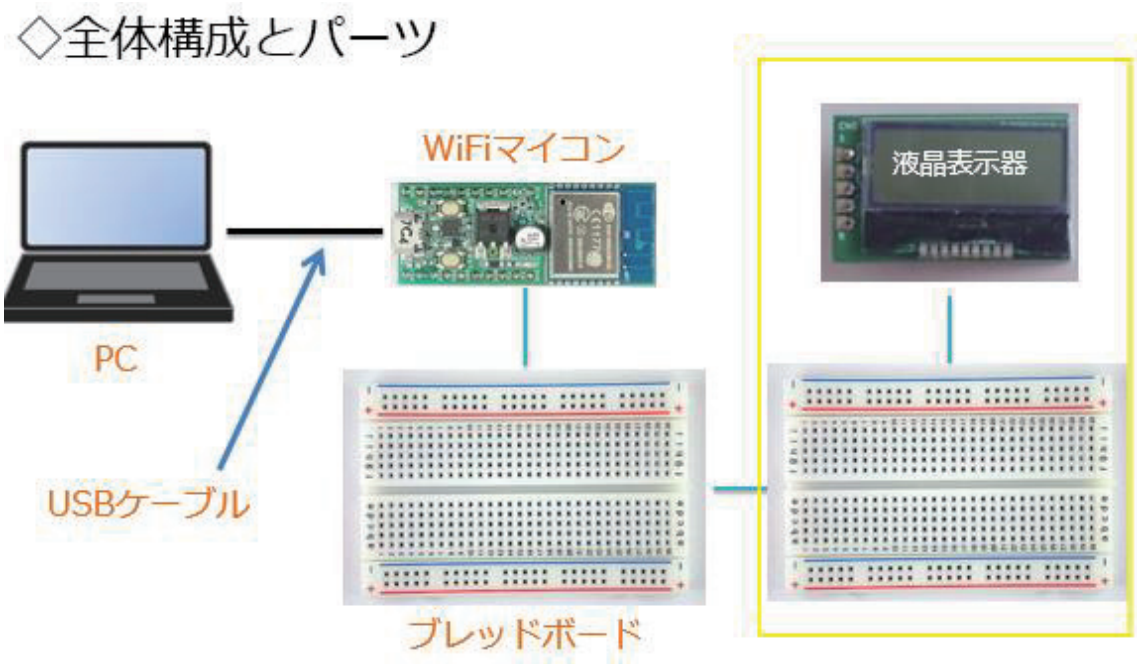


図 249

システムの全体構成を上図に示します。前回使用したデジタル温度センサーの回路を残したままま、液晶表示器を取り扱います。ブレッドボードが混雑するので、**液晶表示器は別のブレッドボードに液晶表示器ユニットとして作成することにしました。**

必要な機材・パーツは、下記です。

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×2 個
5. 配線用ジャンパー線×適宜
6. LCD（液晶表示器）×1 個
7. デジタル温度センサー（STTS751）×1 個（前回のまま）

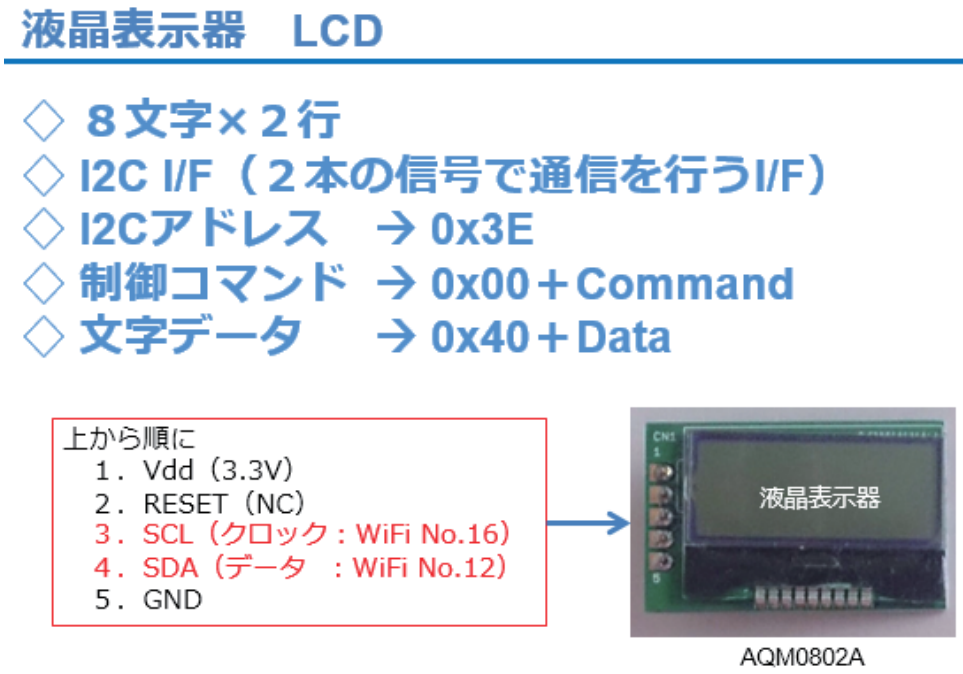


図　250

使用する LCD は、AQM0802A という、8 文字×2 行の表示器です。デジタルセンサーと同様の I2 I/F を用いるので、スレーブアドレス（0x3E）が設定されています。基板の左端には 5 本のピンが取り付けられていて、上から順に 3.3V、RESET、SCL、SDA、GND となっていますが、このうち RESET 信号は使用しません。LCD の初期化は、WiFi マイコンのプログラムで行います。

I2C の通信は 2 種類あって、初期化や消去などの制御コマンドを送る場合と、表示する文字データの送信に分かれています。制御コマンドの場合は、最初に 0x00 を送り、続けてコマンドコードを、文字データの場合は最初に 0x40 を送り、続けて文字コードを送る仕組みです。これらはプログラムで処理します。

デジタルセンサー回路

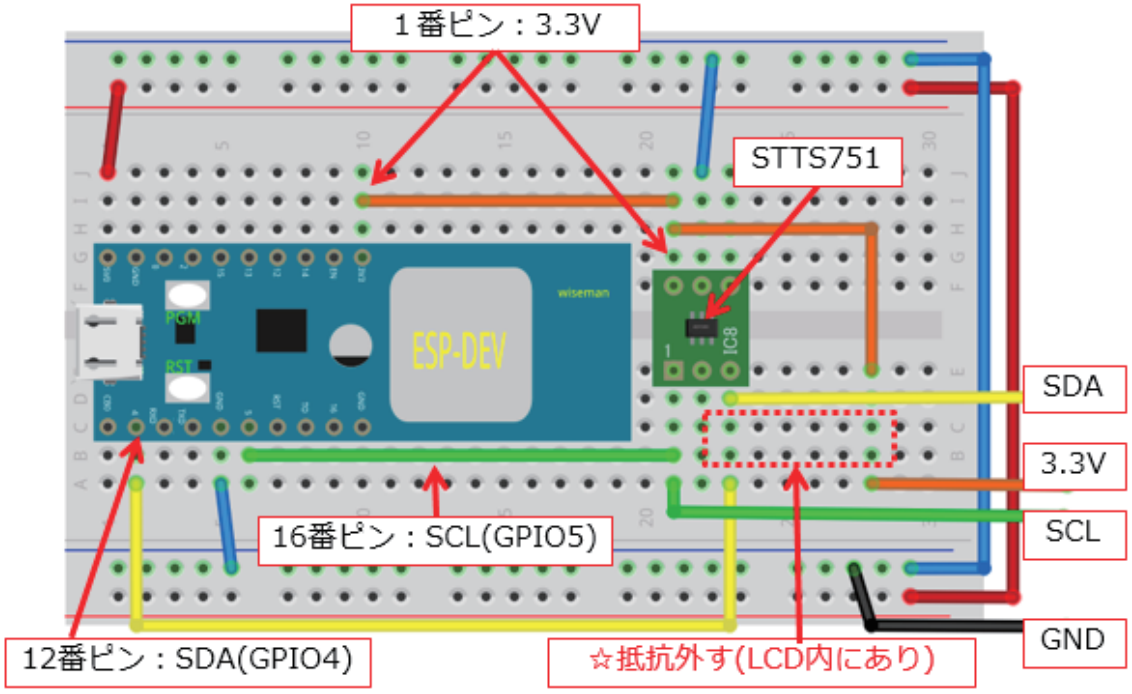


図　251

回路図は WiFi マイコン側と LCD ユニット側に分かれています。

WiFi マイコン側は、前回のデジタル温度センサー回路を流用します。少し変更する点があります。**【10KΩ の Pull Up 抵抗は外します。】**なぜかというと I2C I/F を持つ **LCD の内部に同じ目的の抵抗が内蔵されているので、その他の Pull Up が不要**になるからです。LCD ユニットには図の右側に出ている SDA、3.3V、SCL、GND の 4 本のジャンパー線で接続します。

【重要】

この Pull Up 抵抗を取り外さなくても動くと思います。が、さらに Pull Up 抵抗を内蔵している I2C デバイスを増やすと、抵抗の並列接続となり I2C I/F に流れる電流が多くなって、I/F が不安定になります。

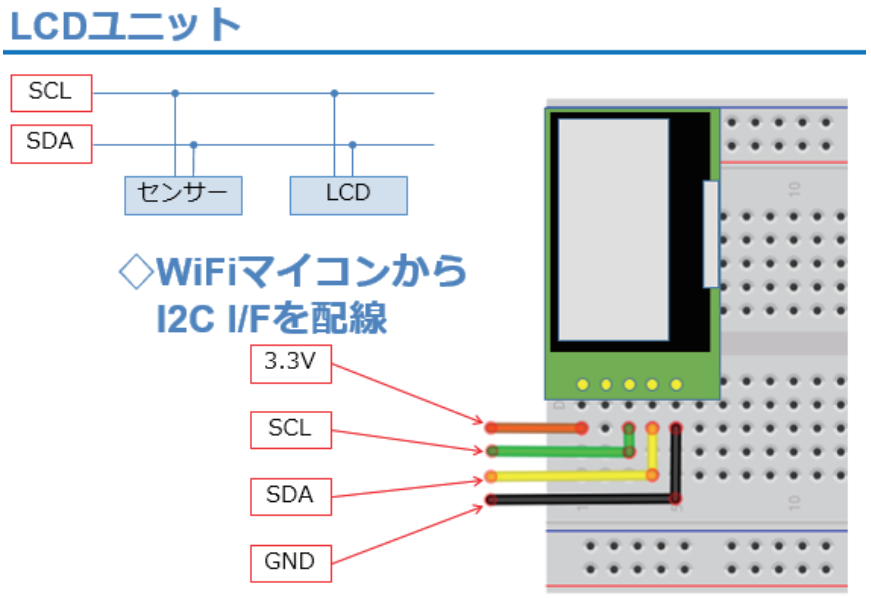


図　252

LCD ユニットは上図の様に配線してください。今回の回路では、デジタルセンサーも併設しているので、図左上に描いたように SCL,SDA に温度センサーと LCD がぶら下がっている形になります。

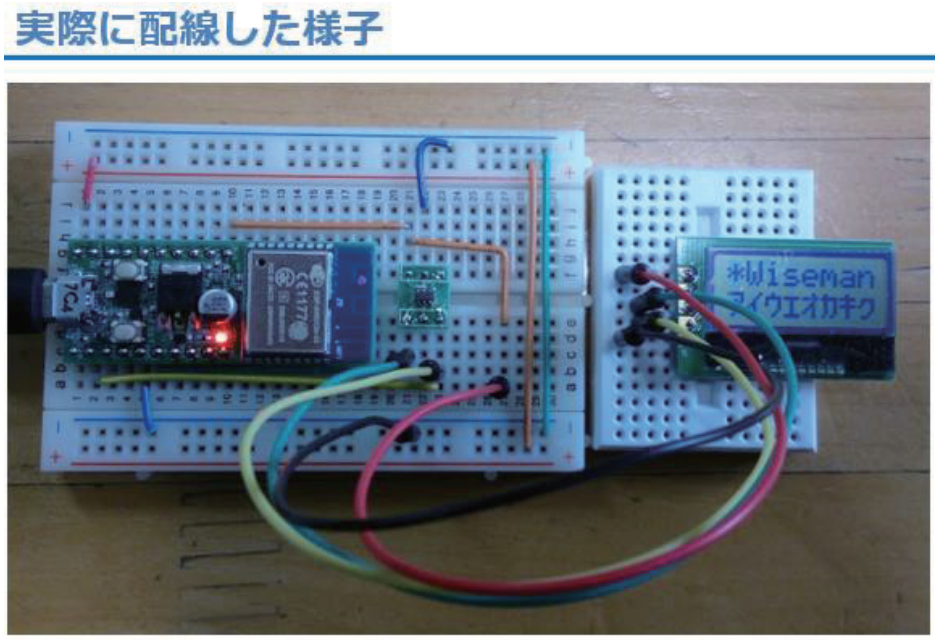


図　253

プログラムを書く

ESP_2108_LCD_8X2

```
#include <Wire.h>          ← ① I2Cデバイス通信のライブラリヘッダ.
#define LCD_ADRS 0x3E      ← ② LCDスレーブアドレス.
char moji[]="*Wiseman";    ← ③ 表示文字列.

//SCL=16:LCD No.3  SDA=12:LCD No.4

void setup() {
  Wire.begin();            ← ④ I2C I/F初期化.
  init_LCD();              ← ⑤ LCD初期化.
}
```

図　254

①I2C 通信ライブラリ用ヘッダ

②LCD の I2C スレーブアドレス

③LCD に表示する固定文字列バッファ（8 文字）

④I2C I/F 初期化

⑤LCD 初期化。関数の中身は後で説明

プログラムを書く　loop()

```
void loop() {
  for(int i=0; i<8; i++) {
    writeData(moji[i]);          ← ① 1文字表示.
  }
  writeCommand(0x40+0x80);       ← ② 標示位置指定.   //2nd Line top
  for(int i=0; i<8; i++) {
    writeData(i+0xb1);           //0xb1=(katakana) ア
  }                              ← ③ カタカナ1文字表示.
  while(1){}
}
```

図　255

①バッファ内文字列を１文字表示

②LCD の２行目先頭を指定（LCD 内の表示メモリアドレスを指定）

③１文字表示（0xb1 は【ア】）アイウエオ…

表示は１回行えばよいので、最後に while(1)で永久ループとします。

プログラムを書く　文字送信・コマンド送信

図　256

①I2C 通信開始

②表示文字通知指定

③文字コード送信

④I2C 通信終了

⑤I2C 通信開始

⑥コマンド通知指定

⑦コマンドコード送信

※コマンドコードは沢山種類があります。詳細はデータシートを入手して参照してください。

⑧I2C 通信終了

プログラムを書く　LCD初期化

```
void init_LCD() {
  delay(100);
  writeCommand(0x38);  // Function set
  delay(20);
  writeCommand(0x39);  // Function set
  delay(20);
  writeCommand(0x14);  // OSC Freq. set
  delay(20);
  writeCommand(0x70);  // Contrast set
  delay(20);
  writeCommand(0x56);  // 3.3V, ICON,Contrast
  //writeCommand(0x52);  // 5V, ICON,Contrast
  delay(20);
  writeCommand(0x6C);  // Follwer Control
  delay(20);
  writeCommand(0x38);  // Function set
  delay(20);
  writeCommand(0x01);  // Clear Display
  delay(20);
  writeCommand(0x0C);  // Display ON/OFF control
  delay(20);
}
```

図　257

上記関数 init_LCD()は、writeCommand()で LCD 初期化のためのコマンドコードを連続して送信して、LCD 内部を初期化します。初期化用コマンドは、デバイスのデータシートに記載があります。サンプルコードが提供されることもあります。この LCD は AQM0802A という型番です。WEB で検索すると多くの情報とサンプルがヒットします。調べてみてください。

ソースコードを入力したら、名前を付けて保存してください。

ここから、WiFiマイコンへの書込みまで一気に進めましょう。

以下の手順でプログラムのコンパイルからWiFiマイコンへの書き込みを行います。

①PCとWiFiマイコンの接続：USBケーブルで接続

②シリアルポート（COM番号）の確認：デバイスマネージャで確認

③シリアルポート（COM番号）の設定：IDEで設定

④WiFiマイコンへの書込み準備：WiFiマイコンのSW操作

⑤スケッチ（プログラム）のコンパイルから書込み：IDEで操作

書込みが終了すると書込み完了のメッセージがIDEに表示される

※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。

書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作確認

図 258

動作確認は、LCD を見れば一目瞭然です。プログラムで設定した文字列が 1 行目に表示され、2 行目はアイウエオ・・・とカタカナが並びます。

次回は、デジタル温度センサーと LCD を同時に使用して、デジタル温度計を開発します。

第9回　デジタル温度計

第７回ではデジタル温度センサー、第８回では液晶表示器（LCD）を使ってみたわけですが、第９回は、その両者を合体して、単体で温度を計測・表示するデジタル温度計を開発します。

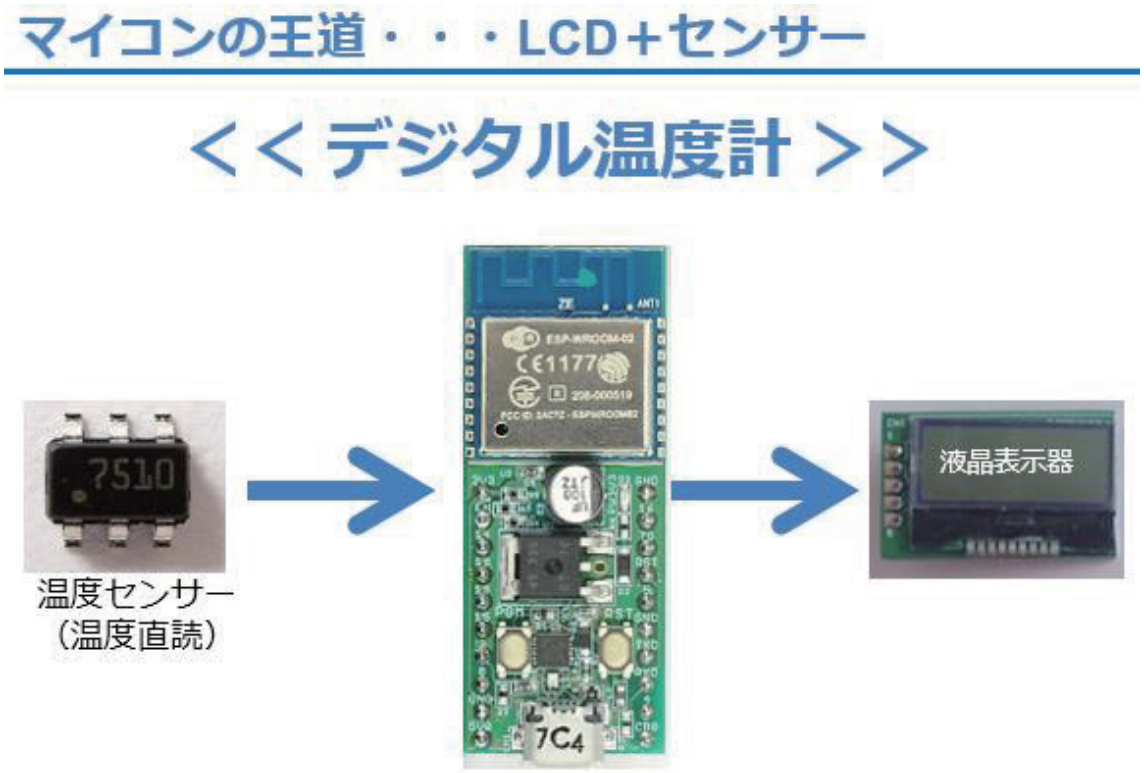


図　259

デジタル温度センサー（STTS751）は、最小計測温度が 1/16℃という、相当精度の高い温度センサーでした。計測した温度を文字列にして、PC に通知していましたが、その文字列を LCD 用に編集してあげることで、デジタル温度計が開発できることは、容易に想像できます。PC にも通知しながら、LCD にも表示するデジタル温度計を作りましょう。

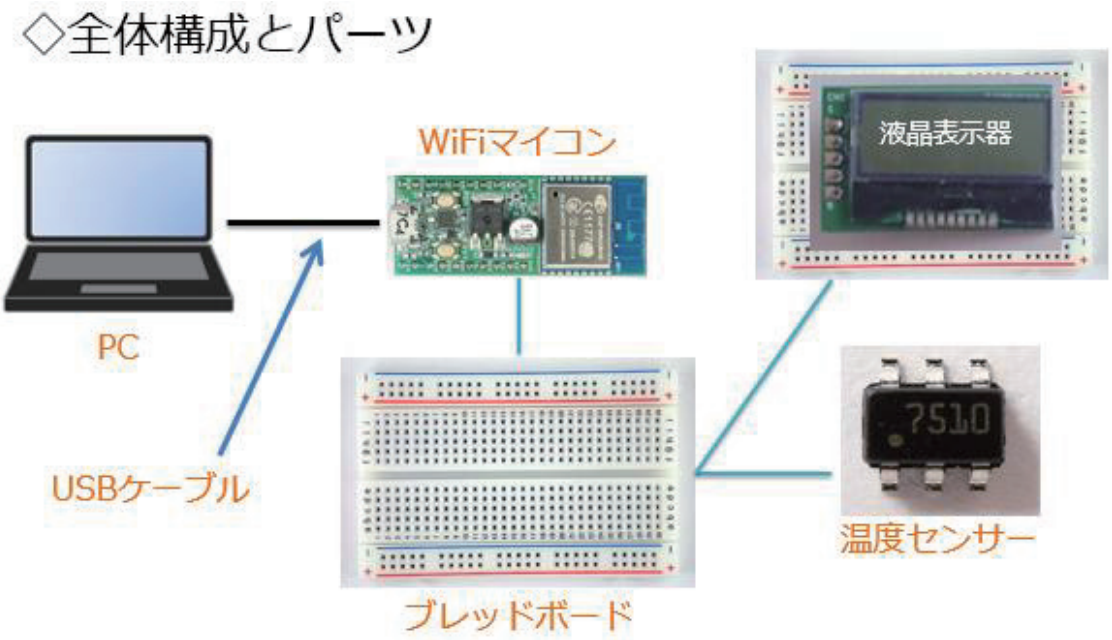


図 260

図で分かるように、直前 2 回の講座で使用したデバイスを使用します。そのために、前回もデジタル温度センサーを残したままの回路としました。

必要な機材・パーツは、下記です。（前回のまま）

1. WiFi マイコン×1 台
2. PC（プログラム開発・書込）×1 台
3. USB ケーブル（マイコンとの接続）×1 本
4. ブレッドボード×2 個
5. 配線用ジャンパー線×適宜
6. LCD（液晶表示器）×1 個
7. デジタル温度センサー（STTS751）×1 個（前回のまま）

使用する LCD とデジタル温度センサーの仕様については、直前 2 回の内容を参照してください。回路も同じですが再掲します。

デジタルセンサー回路

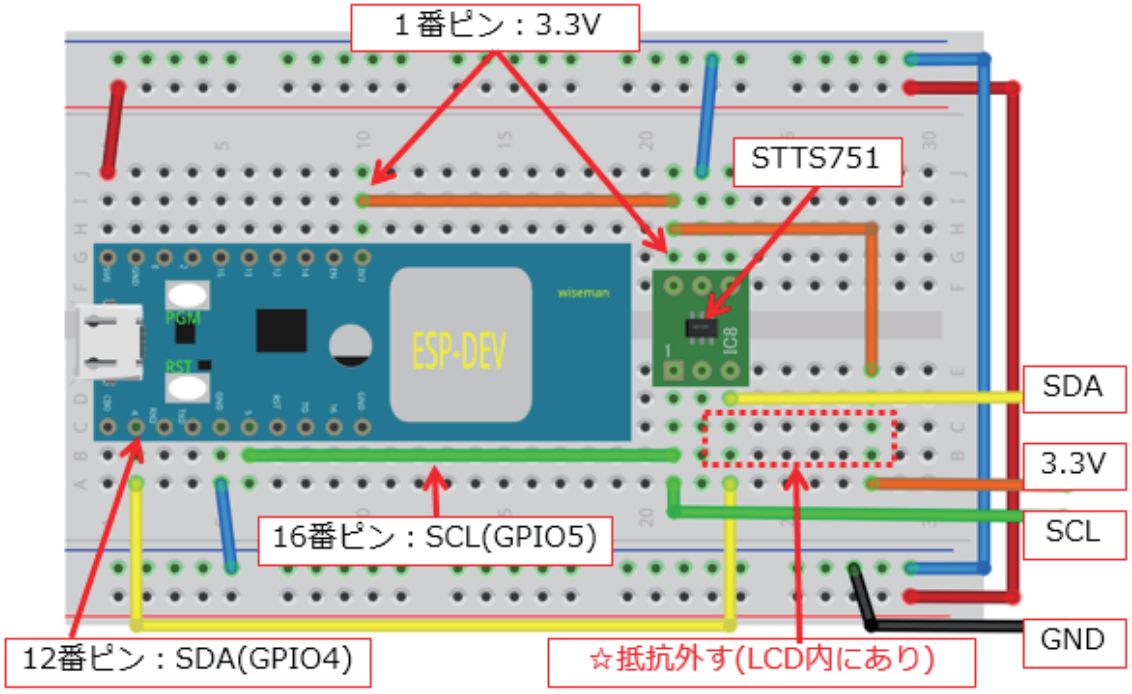


図　261

LCDユニット

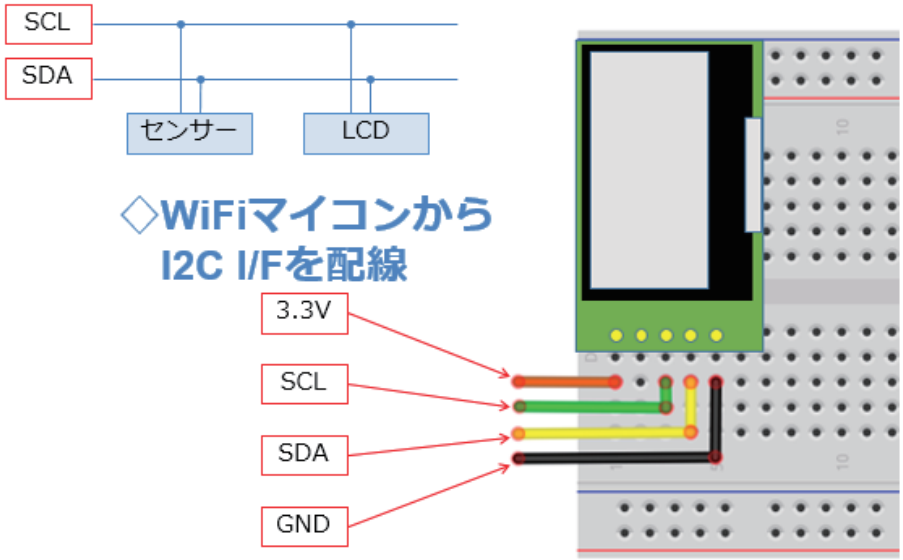


図　262

実際に配線した様子

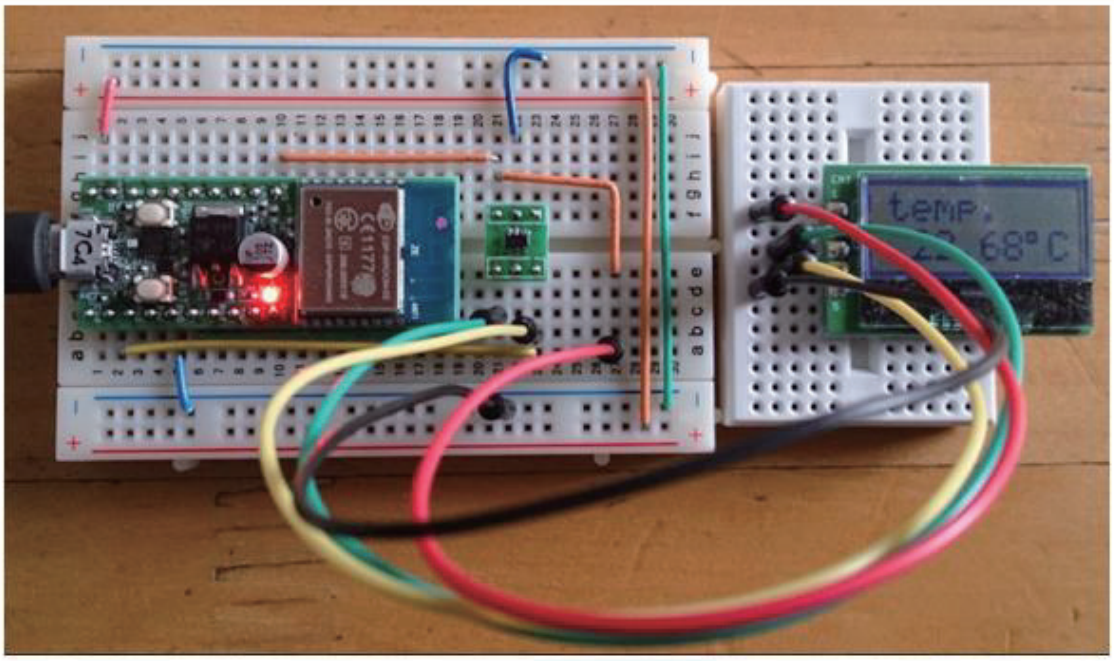


図 263

上の写真はすでに動作をしている様子です。今回は既に 1 度、回路も完成しているのでソフトウエア主体の開発です。しかし、開発と言っても直前 2 回分のソースコードを合体させて、LCD 表示文字列の編集部分のみ作成すればよいので、後に示すソースコード全体量で感じるほどの新規開発分はありません。ほんの僅かです。

【重要】

【これら】のことは、しっかりとした基礎技術を身に付ければ、長く・有効に活用できて、応用開発も楽だということを示唆しています。

※【これら】とは、どのようなことでしょう？

次にプログラムを作成しますが、既に説明したようにデジタル温度センサーのプログラムと、LCD のプログラムを合体したものになるので、少し長めのソースコードになります。

プログラムを書く

ESP_2109_LCD_Temp

```
#include <Wire.h>                    ← ① I2Cデバイス通信のライブラリヘッダ.

#define STTS751_ADRS  0x39           ← ② 温度センサー スレーブアドレス.
#define LCD_ADRS      0x3E           ← ③ LCD スレーブアドレス.

char u_moji[]="temp.    ";           ← ④ 表示文字列(上段用).
char l_moji[]=" **.** C";            ← ⑤ 表示文字列(下段用).

//SCL=16:LCD No.3  SDA=12:LCD No.4

void setup() {
    Serial.begin(115200);            ← ⑥ シリアルポート 初期化.
    Wire.begin();                    ← ⑦ I2C通信初期化.
    init_STTS751();                  ← ⑧ 温度センサー初期化.
    init_LCD();                      ← ⑨ LCD初期化.
}
```

図　264

①I2C I/F を使用するライブラリ用ヘッダ

②温度センサーの I2C デバイス識別用スレーブアドレス

③LCD の I2C デバイス識別用スレーブアドレス

④表示文字列用文字配列（上段：1 行目）

⑤表示文字列用文字配列（下段：2 行目）

⑥シリアルポート初期化

⑦I2C 初期化

⑧温度センサー初期化（関数内容は後に解説します。）

⑨LCD 初期化（関数内容は後に解説します。）

プログラムを書く　loop()　前半

```
void loop() {
    byte valHigh,valLow;
    int temp;

    valHigh = readUpp();          ①// 整数部をセンサーから取得
    Serial.print(valHigh);          // 整数部を出力
    Serial.print(".");              // 小数点
    valLow = readLow();           ②// 少数部をセンサーから取得
    temp = (valLow >> 4) * 625;     // LSB:0.0625 小数部温度計算
    Serial.print(temp/1000);        // 0.1の位
    temp %=1000;
    Serial.print(temp/100);         // 0.01の位
    temp %=100;
    Serial.print(temp/10);          // 0.001の位
    temp %=10;
    Serial.println(temp);           // 0.0001の位
```

図　265

①温度の整数部をセンサーから取得し valHigh に格納

※readUpp()関数は後で解説

②温度の少数部をセンサーから取得し valLow に格納

※readLow()関数は後で解説

上の 2 つの部分は、第 7 回デジタル温度センサーの講座では、loop()の中に直に記述されていましたが、今回は 2 つの関数に分けました。

プログラムを書く　loop()　後半

① LCD表示文字列
下段編集.

② LCD上段表示.

③ LCD下段表示.

```
  sprintf(l_moji," %2d",valHigh);     // 整数部
  l_moji[3]='.';                      // 小数点
  temp = (valLow >> 4) * 625;         // LSB:0.0625 小数部温度計算
  l_moji[4]=(char)('0' + temp/1000); // 0.1の位
  temp %=1000;
  l_moji[5]=(char)('0' + temp/100);   // 0.01の位
  l_moji[6]=0xDF;                     // ℃の代わり
  l_moji[7]='C';

  writeCommand(0x80);                 // 1行目 DRAM ADDR.
  for(int i=0; i<8; i++) {            // 表示
    writeData(u_moji[i]);
  }
  writeCommand(0x40+0x80);            // 2行目 DRAM ADDR.
  for(int i=0; i<8; i++) {            // 表示
    writeData(l_moji[i]);
  }
  delay(1000);
}
```

図　266

①LCD 下段に表示する温度表示用文字列の編集を行う。

文字列用バッファは、次の様に編集されます。

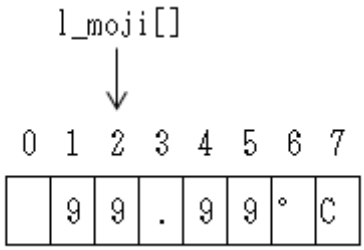


図　267

②LCD の上段（1 行目）に固定文字列を表示

③LCD の下段（2 行目）に温度文字列表示

delay(1000)ですので、1 秒おきに動作が繰り返されます。

プログラムを書く　温度読込

```
byte readUpp() {
    // 整数部をセンサーから取得
    Wire.beginTransmission(STTS751_ADRS);   ← ① I2C通信開始.
    Wire.write(0x00);                       ← ② 温度整数部レジスタ指定.
    Wire.endTransmission();                 ← ③ I2C通信終了.
    Wire.requestFrom(STTS751_ADRS, 1);      ← ④ I2C通信にて1byte読込リクエスト.
    return(Wire.read());                    ← ⑤ 温度(整数部)読込. 戻り値にセット.
}

byte readLow() {
    // 少数部をセンサーから取得
    Wire.beginTransmission(STTS751_ADRS);   ← ⑥ I2C通信開始.
    Wire.write(0x02);                       ← ⑦ 温度小数部レジスタ指定.
    Wire.endTransmission();                 ← ⑧ I2C通信終了.
    Wire.requestFrom(STTS751_ADRS, 1);      ← ⑨ I2C通信にて1byte読込リクエスト.
    return(Wire.read());                    ← ⑩ 温度(少数部)読込. 戻り値にセット.
}
```

図　268

①I2C 通信開始

②00 番（温度整数部）レジスタ指定

③I2C 通信終了

④I2C にて 1byte 読込リクエスト

⑤1byte（温度整数部）読込。そのまま戻り値としてセット

⑥I2C 通信開始

②02 番（温度小数部）レジスタ指定

③I2C 通信終了

④I2C にて 1byte 読込リクエスト

⑤1byte（温度小数部）読込。そのまま戻り値としてセット

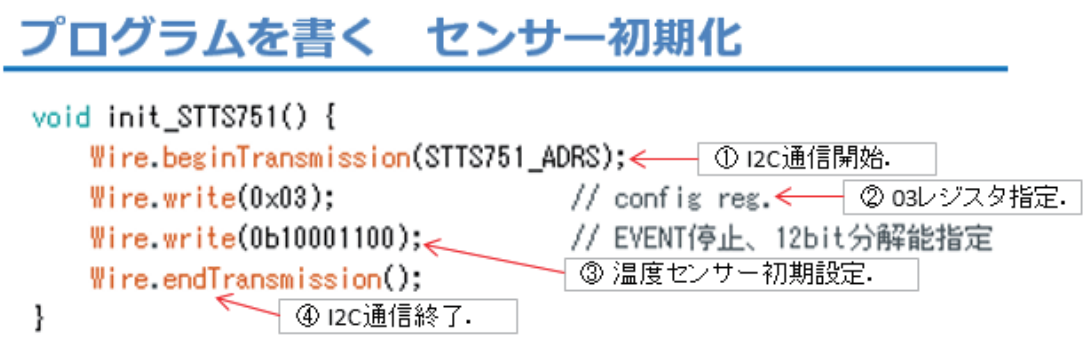


図　269

①I2C 通信開始

②03 番レジスタ(設定レジスタ)指定

③初期設定

④I2C 通信終了

プログラムを書く　文字送信・コマンド送信

図　270

①I2C 通信開始

②表示文字通知指定

③文字コード送信

④I2C 通信終了

⑤I2C 通信開始

⑥コマンド通知指定

⑦コマンドコード送信

※コマンドコードは沢山種類があります。詳細はデータシートを入手して参照してください。

⑧I2C 通信終了

各々の最後の delay()は、I2C I/F が切り替わるための待ち時間です。

プログラムを書く　LCD初期化

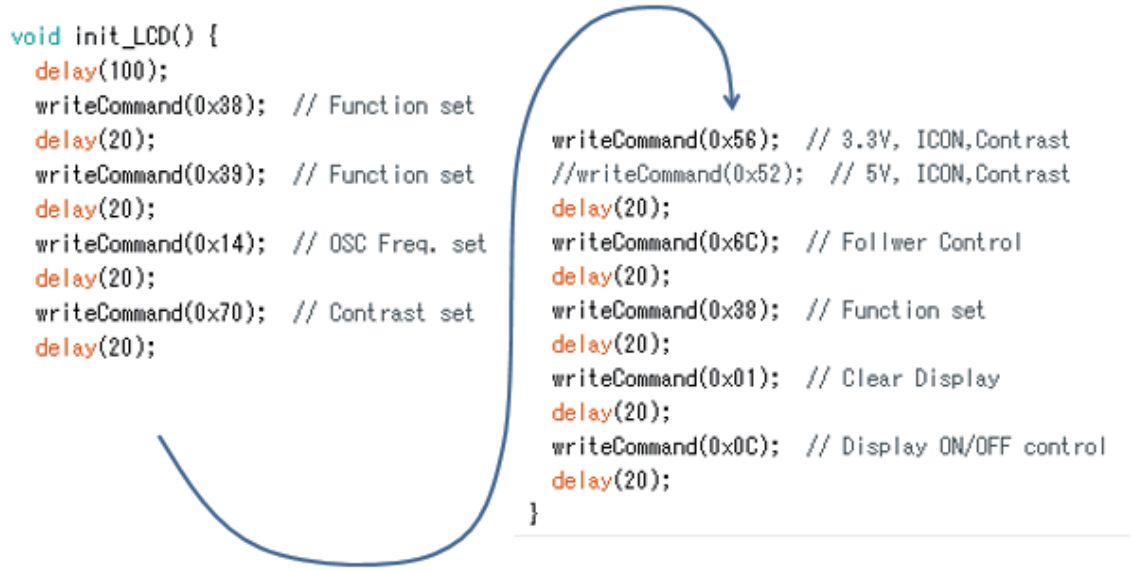


図　271

上記関数 init_LCD()は、writeCommand()で LCD 初期化のためのコマンドコードを連続して送信して、LCD 内部を初期化します。初期化用コマンドは、デバイスのデータシートに記載があります。サンプルコードが提供されることもあります。この LCD は AQM0802A という型番です。WEB で検索すると多くの情報とサンプルがヒットします。調べてみてください。

各コマンドの間にある delay()はコマンドが有効になるための待ち時間です。

ソースコードを入力したら、名前を付けて保存してください。

ここから、WiFiマイコンへの書込みまで一気に進めましょう。

以下の手順でプログラムのコンパイルからWiFiマイコンへの書き込みを行います。

①PCとWiFiマイコンの接続：USBケーブルで接続

②シリアルポート（COM番号）の確認：デバイスマネージャで確認

③シリアルポート（COM番号）の設定：IDEで設定

④WiFiマイコンへの書込み準備：WiFiマイコンのSW操作

⑤スケッチ（プログラム）のコンパイルから書込み：IDEで操作

書込みが終了すると書込み完了のメッセージがIDEに表示される

※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。

書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作確認

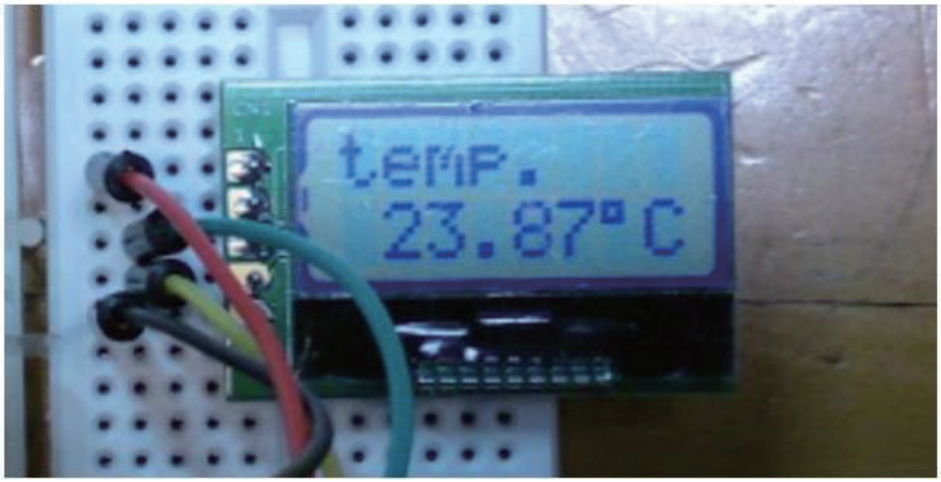


図　272

動作確認です。

まず、LCD に図の様に整数部、小数部、℃の表示が行われます。次に、IDE からシリアルモニターを起動します。

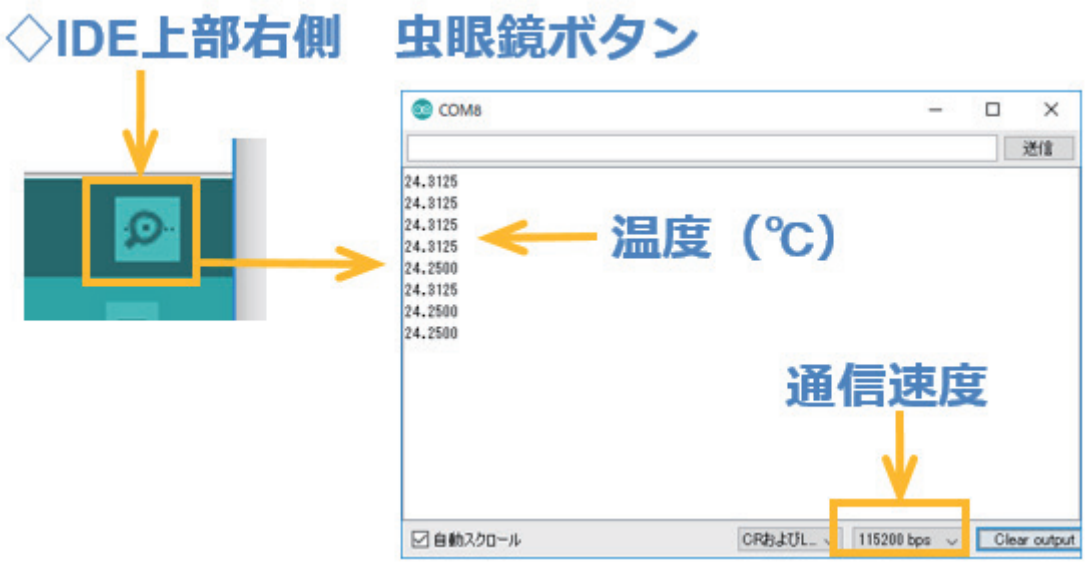


図　273

通信速度を合わせると、温度が 1 秒ごとに表示されます。

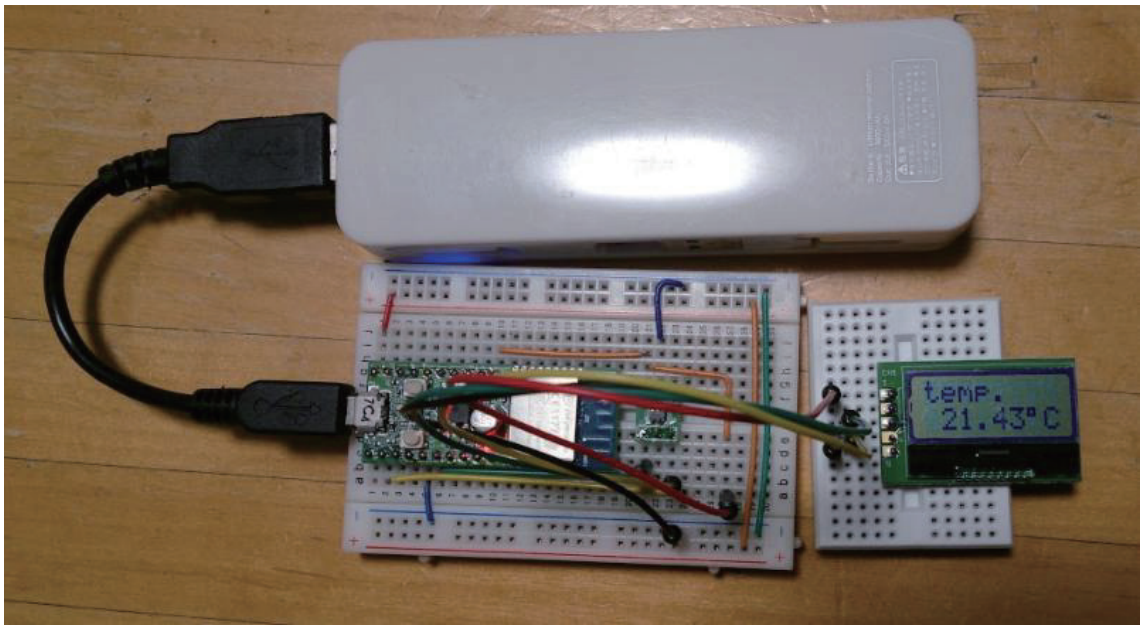


図 274

図の様に、USB ケーブルで電源（写真は携帯電話の充電用）を供給すれば、単独で稼動するデジタル温度計となります。

これまで、沢山の実習をしてきました。これで、WiFi マイコンの【マイコン機能】の使い方は終了です。次回から、いよいよ WiFi 機能を使った WEB 連携システムへの利用を実習します。

Appendix A: 実習キット

2 冊のテキストで使用する、全てのパーツが揃った実習キットを示します。

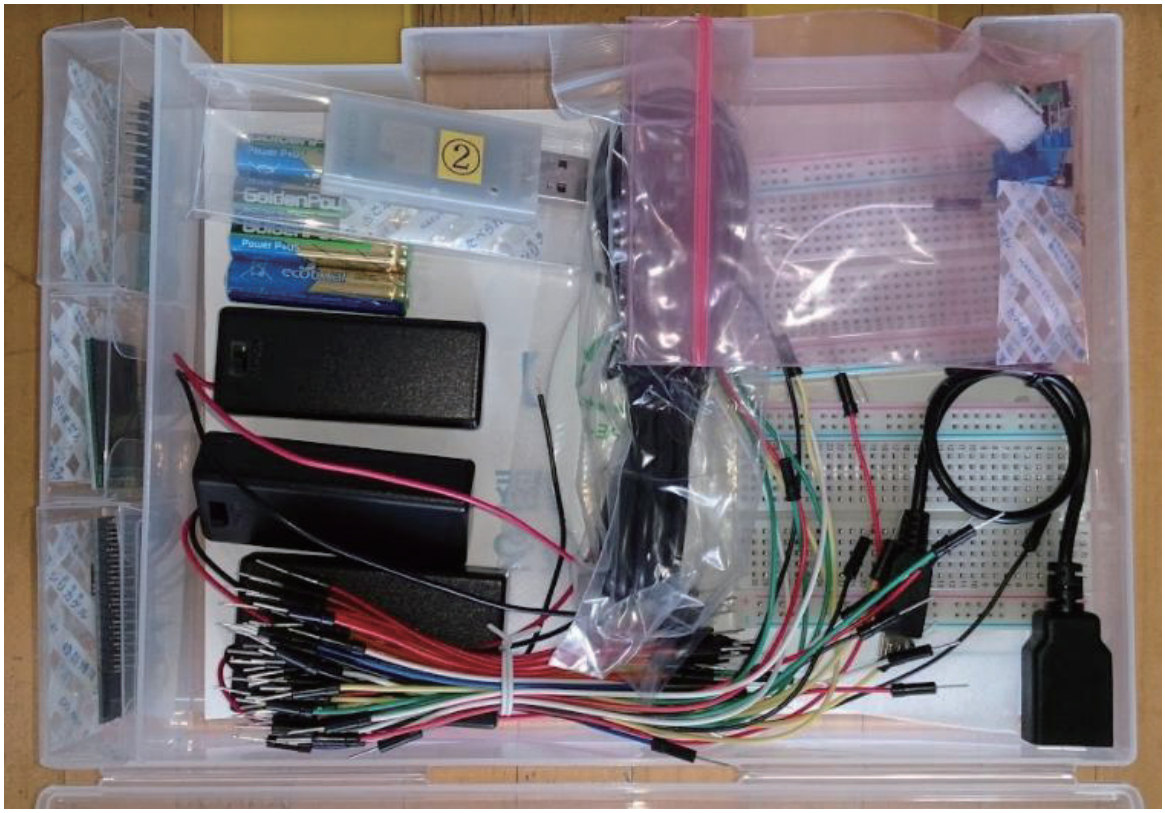


図 275

Appendix B: 使用する全パーツ

図　276

図　277

最後に

最後まで読み進めていただき、ありがとうございます。すべての講座を実習された方は、マイコンの基礎を習得する順番も理解できたと思います。DO・DI・シリアル送信・受信・ADC（AI：アナログ入力）・LCD（表示器）の順が、新しいマイコンを初めて扱うときの王道です。もうマイコン利用の基本編は身に付きました。次はぜひ第二分冊へと進んでください。第二分冊では、このテキストで使用したマイコンを利用して、Internet に接続して、いろいろな WEB サービスを使用してみます。第二分冊を終了するころには、現場で使える IoT の WEB 連携技術が身に付いているでしょう。

平成 30 年度「専修学校による地域産業中核的人材養成事業」
情報通信技術に対応した組込みシステム開発技術者育成のモデルカリキュラム開発と実証事業

【IoT へのドリル１】組込みシステム開発技術

平成 31 年 3 月

一般社団法人全国専門学校情報教育協会
〒164-0003　東京都中野区東中野 1-57-8　辻沢ビル 3F
電話：03-5332-5081　FAX　03-5332-5083