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」
(Society5.0等対応カリキュラムの開発・実証)

富山県をモデルとした「モノづくり」現場にIoTを導入する中核的人材育成

〈製造IoT基礎演習　テキスト教材〉

学校法人　浦山学園　富山情報ビジネス専門学校

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」
(Society5.0 等対応カリキュラムの開発・実証)

富山県をモデルとした「モノづくり」現場に
IoT を導入する中核的人材育成

〈製造 IoT 基礎演習　テキスト教材〉

本紙は、文部科学省の生涯学習振興事業委託費による委託事業として
学校法人浦山学園富山情報ビジネス専門学校が実施した平成 30 年度
「専修学校による地域産業中核的人材養成事業」の成果物です。

学校法人浦山学園　富山情報ビジネス専門学校

目　次

製造 IoT 基礎演習 テキスト教材

Appendix【基礎演習】

富山情報ビジネス専門学校

なんでもつながる！！
IoT基礎演習
【製造ＩｏＴ基礎演習】

【演習編】目次

1

Step1　データ整理手法（データベース）

◇このステップでは、基礎編で学んだデータ格納方法の一つとして
データベースを解説します。

◇データベースは、製造IoTに必須の知識です。

◇IoTシステムでは、取得したデータを格納したり、生産計画を立てる際に利用したり、刻々と変化する生産実績を格納したり・・・と、多岐にわたる用途にデータベースを利用しています。

◇本演習では、リレーショナルデータベースシステム Microsoft Access（以下Accessという） を用いて、データベースの利用方法を解説します。他にもSQL Server、Oracle Dtabase、Sybase、MySQL、Postgresql、SQLite・・・ectと、とても多くのデータベースシステムがありますが、Accessの使い方を理解すれば、Accessの画面を通して上記のような他のデータベースを利用することもできます。

◇生産計画と生産実績が納められた生産管理データベースをモデルとして、解説します。

2

Step1　データ整理手法(データベース）

1. 生産計画と生産実績

基礎編で解説したように、生産は生産計画に基づいて行われ、その進捗として生産実績が記録されます。それでは、生産計画と生産実績に最低限必要な項目を、改めて整理しましょう。

※理解しやすいように最低限の管理を行う基本形データベースを考えるため、基礎編の内容とは、項目数や名称などが異なります。

※生産計画を立てた順に計画番号がつけられ、計画番号順に生産が行われることにします。

()内は、情報のデータ型を表します。

【生産計画】

①. 計画番号　：　計画の識別番号です。(文字型)
②. 製品番号　：　計画の対象製品の識別番号です。(文字型)
③. 生産予定数　：　計画した生産数量です。(数値型)
④. 計画日　：　この生産計画を立てた日付です。(日付/時刻型)
⑤. 開始日　：　生産を開始した日付です。(日付/時刻型)
⑥. 終了日　：　生産が終了した日付です。(日付/時刻型)

日々作られる生産計画を計画番号で識別して、同じ入れ物に格納します。
1件の生産計画を【レコード】、この入れ物を【テーブル】と言います。

つまり、1件の生産計画は、【生産計画テーブル】に格納されている
【生産計画レコード】の中の1件と言います。（1行と言う事もあります。）
生産計画レコードは、①～⑥の項目を持ち、これらを【フィールド】
と言います。

①②③④は生産計画作成時に確定しているので、生産計画レコードの
①～④のフィールドには、文字や数値、日付などが格納されています。
⑤⑥は、生産が開始、終了されないと確定されないので、計画時は空欄
として作成されます。この空欄の状態（なにも入っていない状態）を
Null（ヌル）と言います。
Nullは、空白とは異なる状態で、空白（スペース）と厳密に区別されます。

⑤⑥は、計画が進むごとに、上書きされます。

★ここで中間整理すると、
生産管理DBには、1つの【生産計画テーブル】があり、その中に
複数の【生産計画レコード】が存在し、生産計画レコードは①～⑥の
【フィールド】を持っていることになります。

次に生産実績を考えます。
【生産実績】
①. 計画番号 ： 計画の識別番号です。（文字型）
②. 製品番号 ： 計画の対象製品の識別番号です。（文字型）
③. 生産実績数 ： 計画した生産数量です。（数値型）
④. 実績更新日 ： この生産実績を更新した日付です。（日付/時刻型）

①②④は、生産開始時に確定しますので、初めから値が格納されますが、
③は生産が進まないと確定されないため、始めは0が格納されています。
※この0もNullとは厳密に区別されます。

生産が開始されると同時に、【生産計画レコード】の【開始日】が記録され、
この【生産実績テーブル】に【生産実績レコード】が作られます。
【生産実績レコード】には、①～④のフィールドがあり、③のフィールドは、
最初 0 ですが、生産が進むにつれて+1づつカウントアップされて、生産
予定数に達したら、生産終了となります。
このとき、【生産産計画レコード】の終了日が更新されます。

もし、生産が時間の経過とともに刻々と進む状況を折れ線グラフなどで
表示したい場合は、生産実績レコードの作り方を変えて、1個製品が完成
するたびに、【生産実績レコード】を1件作ります。そして、実績更新日には、
日付だけでなく、時刻も記録します。

★ここでまた整理すると、
生産管理DBには、1つの【生産計画テーブル】があり、その中に
複数の【生産計画レコード】が存在し、生産計画レコードは①～⑥の
【フィールド】を持ち、さらに1つの【生産実績テーブル】が存在し、
その中に 複数の【生産実績レコード】があり、生産実績レコードは

①～④ のフィールドを持っていることになります。
また、フィールドには格納するデータの内容に応じたデータ型を設定する必要があります。

【生産計画テーブル】や【生産実績テーブル】の様に複数のテーブルを持つAccessのファイルをデータベースファイルと言います。
※AccessのVBA機能で、プログラムが作り込まれるデータベースファイルは、Accessの新規作成メニューで空のデータベースを作成 してからプログラムを作り込みます。テーブルの構造やその中のデータ、VBAによるプログラムなども含めて、1つのファイルで管理されます。

◇改めて、データベースの構造を考えると、次のようになります。

データベースファイルの中に
テーブルがあり、その中に
レコードがあり、その中に
フィールドがあり、その中は、
データ型が設定されていて（数値型や文字型や日付型）
そこに実データが格納される。

◇データ型について
例えば製品番号を1,2,3・・・と進む、シーケンス番号（数値）で格納することにすると、番号では製品の種類を識別することが出来ませんが、文字型にしておくと、製品名称の頭文字などを製品番号に利用できて、製品の種類も類推できるようになります。計画番号なども、西暦日付＋シーケンス番号（20190105001→2019年1月5日1番目）の様にすれば、生のデータを見ただけでも理解しやすくなります。

データ型には、次のものがあります。フィールドに格納するデータの内容に応じて適切なデータ型を設定します。

①. 短いテキスト　：氏名や部署名などの255文字以下の文字列。郵便番号や電話番号などの計算対象としない数字
②. 長いテキスト　：備考や説明などの長い文字列
③. 数値型　：数量や重量などの数値
④. 日付/時刻型　：受注日や訪問日時などの日付や時刻
⑤. 通貨型　：単価や金額などの通貨データ。正確な計算が必要な実数
⑥. オートナンバー型　：自動的に割り振られる固有のデータ（編集不可）
⑦. Yes/No型　：配偶者の有無や送付済みかなどの2者択一のデータ
⑧. OLEオブジェクト型　：画像やExcel、Wordなどのデータ

⑨. ハイパーリンク型　：Webページ のURLやメールアドレス、ファイルパス等
⑩. 添付ファイル型　：画像やExcel、Wordなどのファイル
⑪. 集計　：テーブル内のフィールドの値を使用して計算するフィールド

※一般によく利用されているのは、①～⑦のデータ型です。

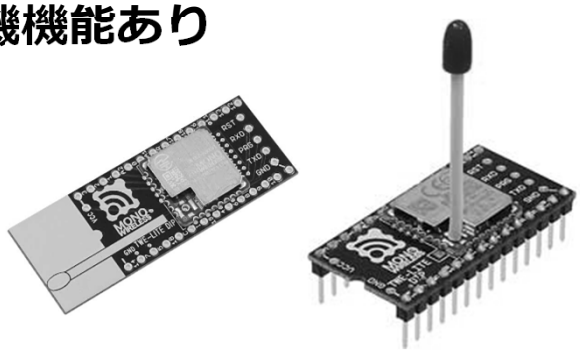
2. 生産管理データベースの作り方

それでは、1で設計した生産管理データベースに2つのテーブルを作りましょう。

Accessで生産管理データベースを作る手順を、
【Appendix A：生産管理データベースの作り方】で詳細に記述しました。
必ず参照して実際に作成してください。

Step2　無線マイコン デジタルI/O

◇TWE-Lite – トワイライト
◇出荷時に標準アプリ書き込み済み
◇配線するだけで使える
◇中継機機能あり



3

Step2　無線マイコン　デジタルI/O

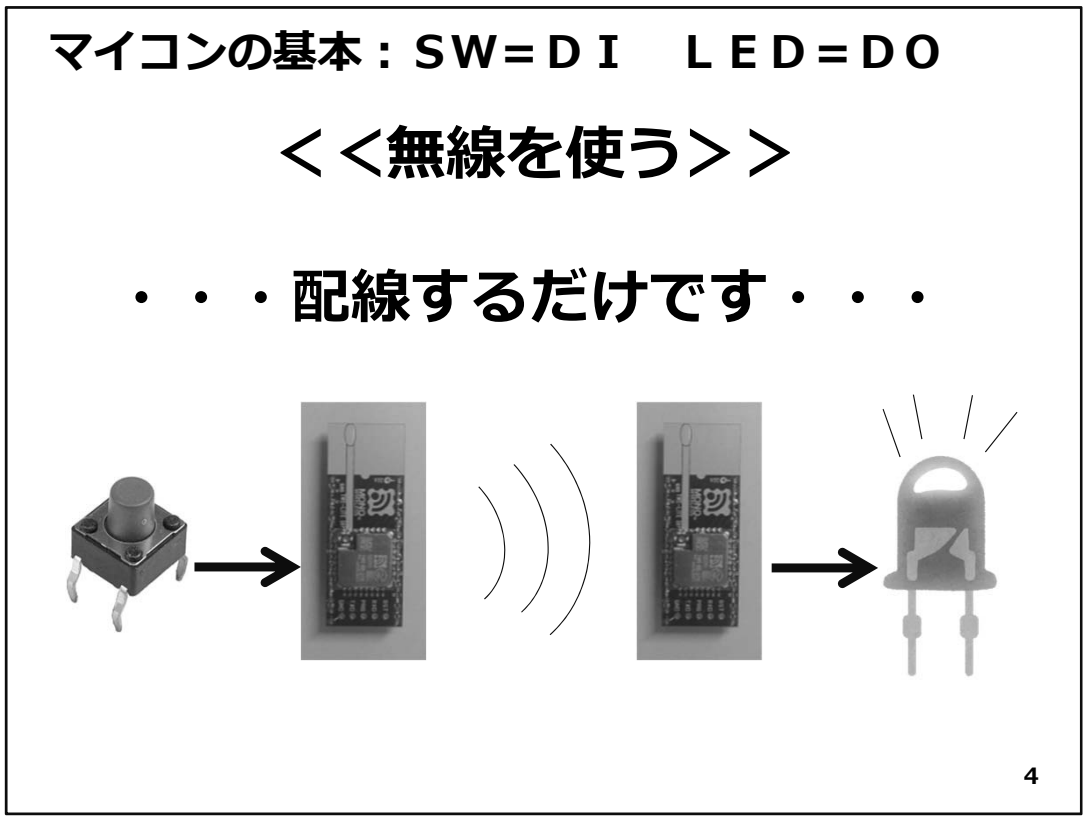
上の図は、TWE-Lite（と書いてトワイライトと読む）無線マイコンモジュールです。金属のパッケージ内部に無線ユニットとマイコンユニットを内蔵しているので、このモジュール1台で無線通信のできるマイコンとして機能するように設計されています。

左側のものは、白くプリントされた部分の裏側にアンテナパターンが配置されているので、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。

どのモジュールも出荷時に**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）。

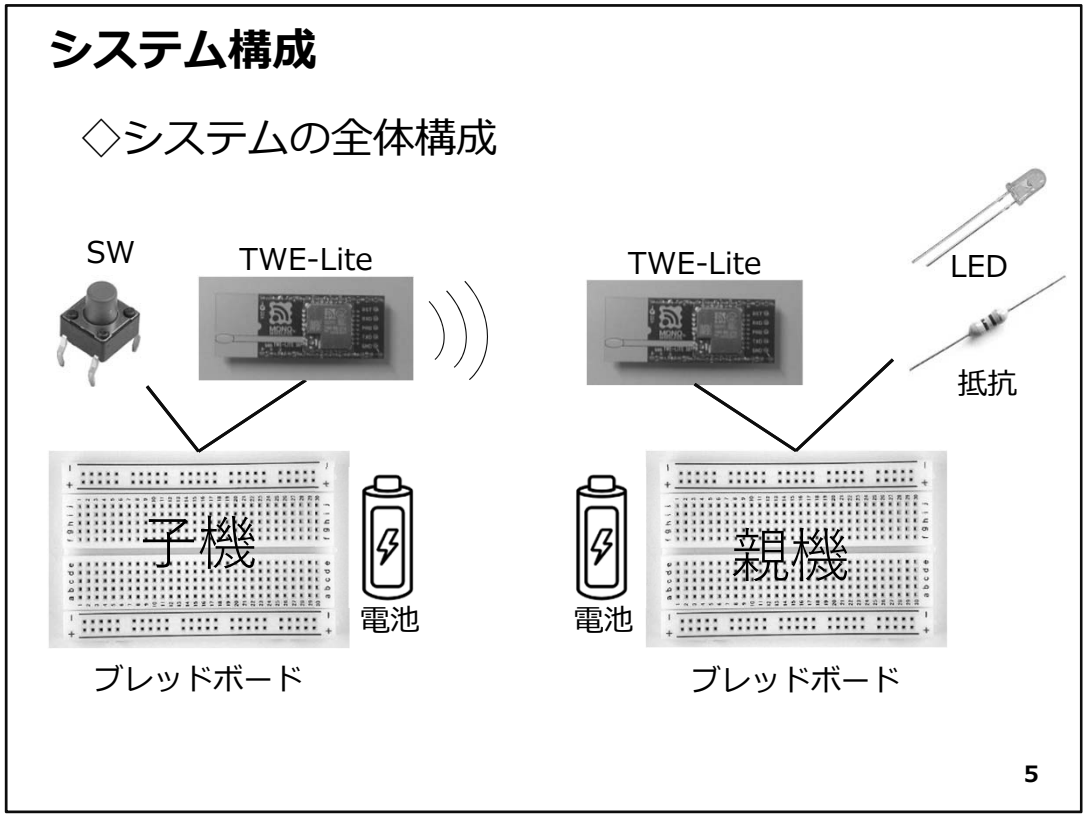
無線というと送信機と受信機の間での通信を行うわけですが、このモジュールは設定を行えば、送信機と受信機の間に入って**中継器**として働き、**全体での通信距離が伸ばせる**という機能があります。中継機としては、送信機と受信機の間に2台まで設定できる設計となっています。

もちろん、マイコンですから自由にアプリケーションを開発して、マイコンに書込み専用システムを開発することもできます。



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。上の図のように、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。



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。

ピン配置（標準アプリ）

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
モード設定ビット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	モード設定ビット3
26	2	M2	モード設定ビット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

6

ピン配置

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、基板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。

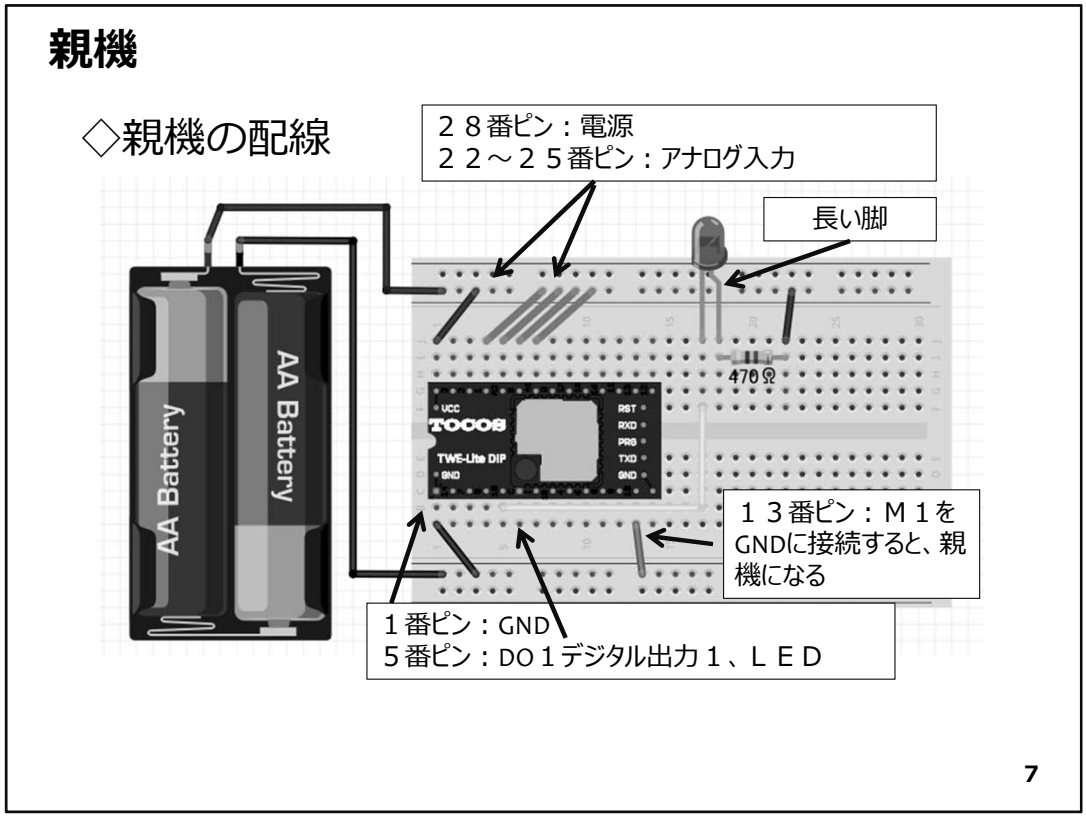
20番：UART速度設定。このモジュールのシリアル通信の通信速度は内部設定で変更することができます。設定変更した場合、この信号をGNDに接続することでその設定が有効になります。解放している場合は、通常の速度設定で動作します。

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。この演習では電池の＋（3V）に接続します。

図のピン配置表は、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。



◇まず親機を配線します。

◇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、親機を上の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。上の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。

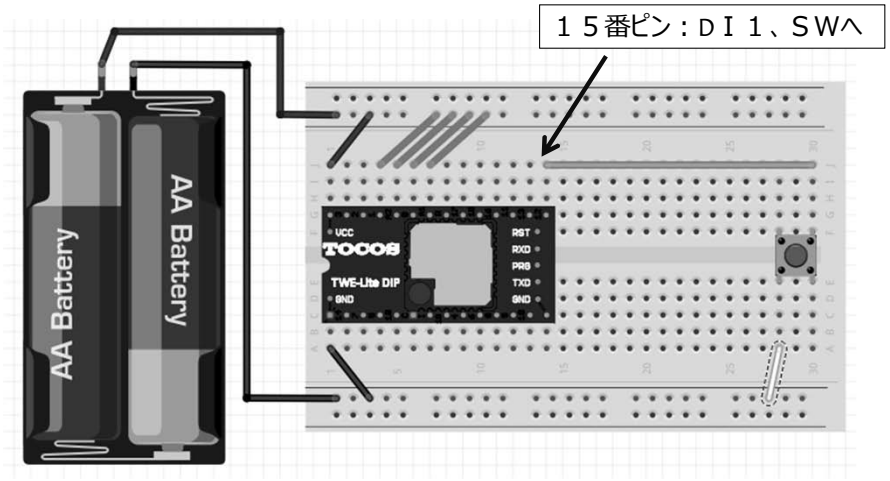
◇1番ピンは、電源のGND（－）に、28番ピンは電源の（＋）に接続します。22～25番ピンは電源の（＋）に接続しておきます。このマイコンモジュールはアナログ信号の変化により状態が変わると、対向機に状態が変化したことを知らせる無線通信が発生してしまうため、これを防ぐ目的で、アナログ入力のレベルを固定しておくための配線です。

◇ 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。

子機

◇子機の配線



8

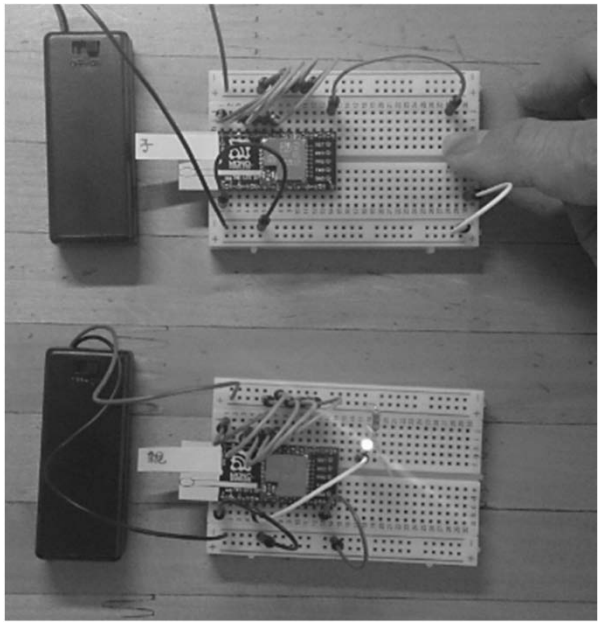
◇次に子機の配線を行います。

◇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。

動作確認



9

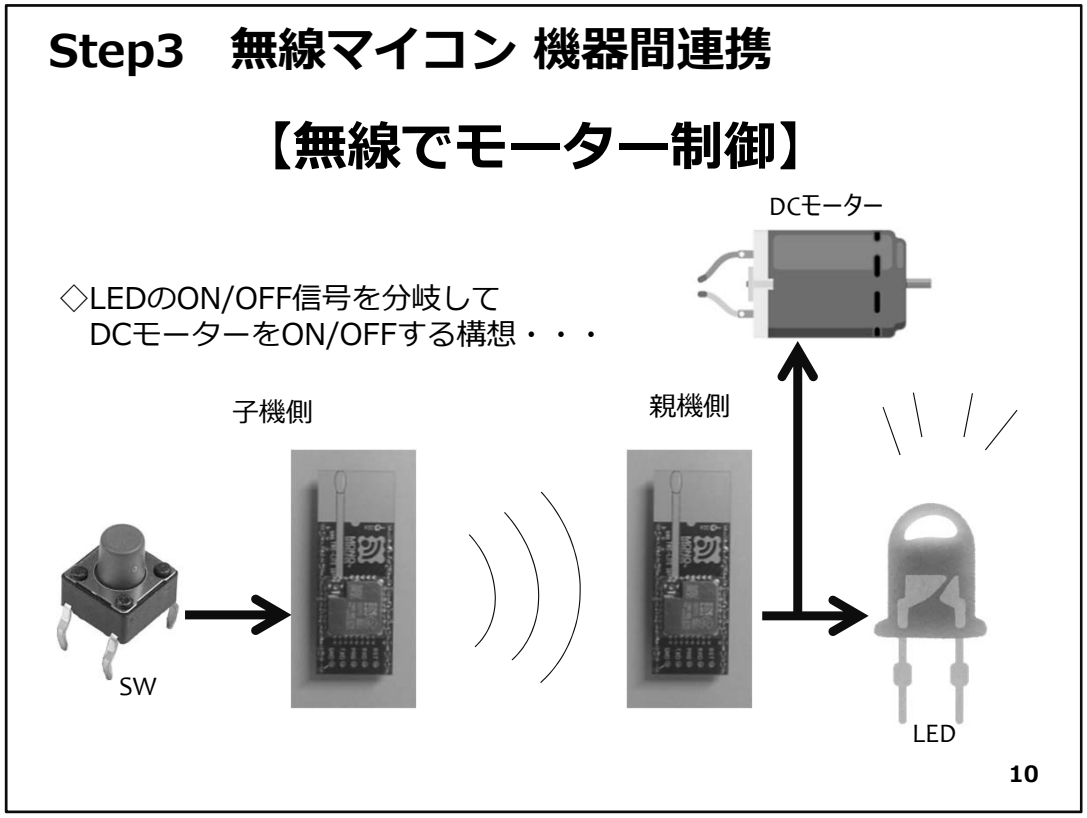
◇配線の確認ができたら、いよいよ動作確認です。

◇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。

◇うまく動作しない様でしたら、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。

【注意】:ここで作成した回路は、次のStepで利用します。可能であれば、そのまま保存してください。



Step3　無線マイコン 機器間連携（外部機器の駆動）

◇Step2で開発したシステムの応用です。LEDのON/OFF信号を利用して、外部機器（モーター）を駆動してみましょう。

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

モーター基板:

1. モーター×1個
2. ブレッドボード×1個
3. 電池（単4乾電池×３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）×2個
7. トランジスタ（2SA1015）×1個
8. ダイオード（1N4148）×1本

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。モーターは、親機に接続しますが、基板が込み入ってしまうので、別にモーター基板として配線します。

◇無線マイコンの　I/O特性

無線マイコン（TWE-Lite）のデータシートの一部を図に示します。ここで利用するDO1（デジタル出力）は通常4mAの電流が流せます。このマイコンのDOは、ON状態でLowになり、電流を吸い込むようになっています。そのままでは、LEDの制御信号を分岐しても、DCモータの無負荷時電流370mAには耐えられませんので、電流を増幅する必要があります。

◇また、DIO Lo出力は、0～0.4Vとなっていて、Low状態でも完全な0Vにはなりません。

◇電流増幅には、トランジスタを使います。

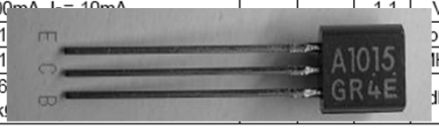
トランジスタには投入する電流を増幅するNPN型と、吸い込む電流を増幅するPNP型がありますが、ここではPNP型を使います。マイコンがDO＝ON時にLowで電流吸い込み型のDOだからです。（ON時HighのDOであれば、NPN型を使用します。）

モーターとトランジスタ

◇モーター：電源電圧：1.5Vから駆動可能
無負荷時電流：200mA
◇マイコンOがLow時の電流4mAを右の
トランジスタで200mA以上に増幅する

■ **ELECTRICAL CHARACTERISTICS** (Ta=25°C, unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Collector-Base Breakdown Voltage	BV_{CBO}	I_C=-100μA, I_E=0	-50			V
Collector-Emitter Breakdown Voltage	BV_{CEO}	I_C=-10mA, I_B=0	-50			V
Emitter-Base Breakdown Voltage	BV_{EBO}	I_E=-10μA, I_C=0	-5			V
Collector Cut-off Current	I_{CBO}	V_{CB}=-50V, I_E=0			-100	nA
Emitter Cut-off Current	I_{EBO}	V_{EB}=-5V, I_C=0			-100	nA
DC Current Gain	h_{FE1}	V_{CE}=-6V, I_C=-2mA	120		700	
	h_{FE2}	V_{CE}=-6V, I_C=-150mA	25			
Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	I_C=-100mA, I_B=-10mA		-0.1	-0.3	V
Base-Emitter Saturation Voltage	$V_{BE(SAT)}$	I_C=-100mA, I_B=-10mA			1.1	V
Output Capacitance	C_{OB}	V_{CB}=-1				pF
Current Gain Bandwidth Product	f_T	V_{CE}=-1				MHz
Noise Figure	NF	V_{CE}=-6 R_G=1k				dB



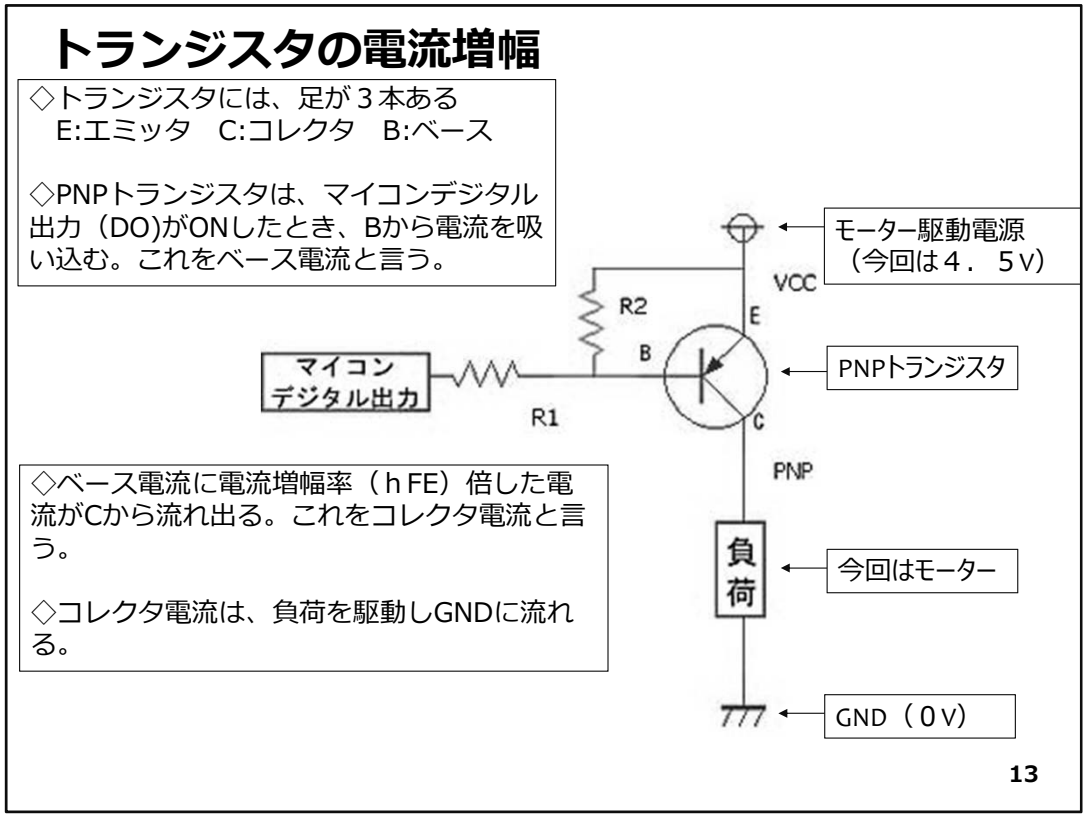
■ **CLASSIFICATION OF h_{FE1}**

RANK	Y	GR	BL
RANGE	120-240	200-400	350-700

12

◇モータの無負荷時電流が200mAなので、マイコンのDOがONの時流れる電流4mAを200mA以上に増幅します。

◇図に掲載したのは、PNPトランジスタ（2SA1015）のデータシートの一部です。
このトランジスタの電流増幅率（hFE1）は、トランジスタのGRグレードで200～400倍です（赤枠）ので、300倍と考えると、4×300＝1200mAとなって、十分モーター駆動に使えます。



◇トランジスタは、電流を増幅します。

◇図のPNPトランジスタの場合は、マイコンのデジタル出力端子を抵抗（R1）を介してベースに接続します。
R1は電流が流れすぎないようにして、不用意にベースラインに接続している半導体や、トランジスタ自体が破損するのを防ぐ役目をしています。

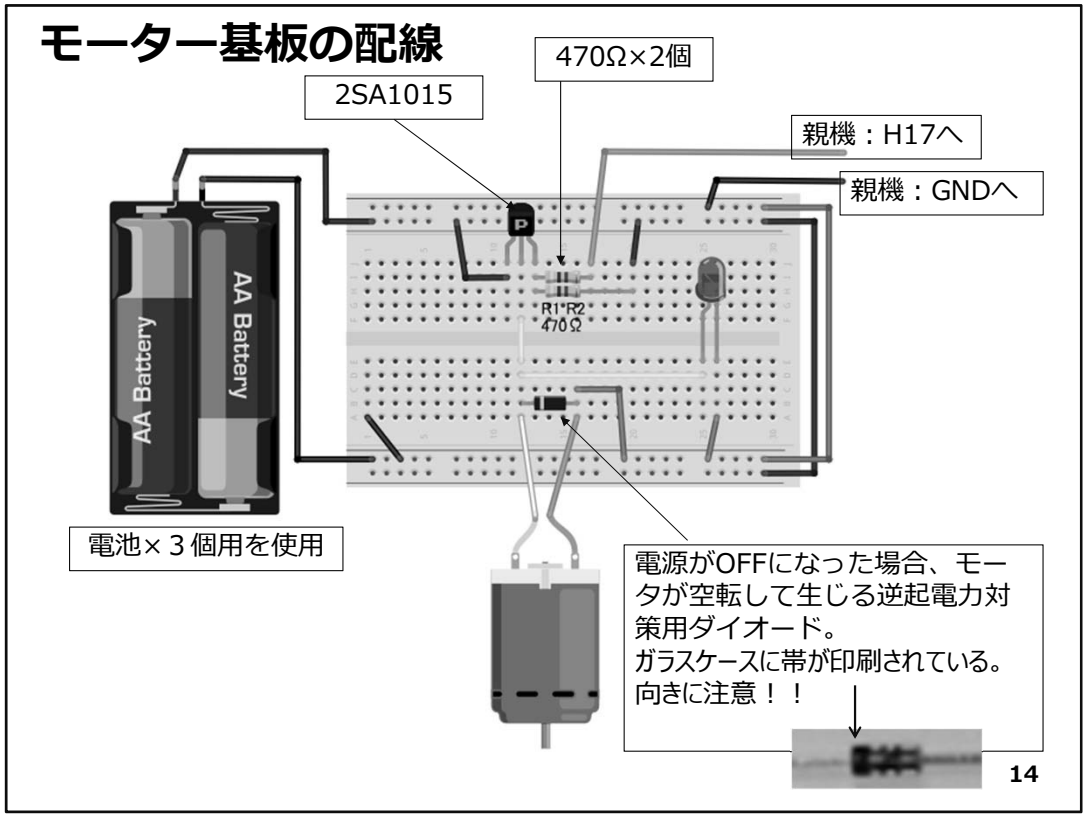
◇無線マイコンのデジタル出力がONすると、DO端子はLowになり、ベースから電流をい込みます。この時ベースに流れる電流をベース電流と言います。

◇トランジスタごとに決まっている電流増幅率があり、データシートなどでは、記号でhFEと表されてます。

◇ベース電流に上のhFEを掛けた電流が、コレクタから流れ出ます。コレクタ電流は、負荷を駆動してGND（0V)に流れ出ます。

◇R2の役割：トランジスタは、ベース電圧が0Vのときにも、コレクタからベースに向かって僅かに電流が流れています。（コレクタ遮断電流）この電流が原因でトランジスタがONしてしまうのを防ぐためにR2を使用します。

◇図にはありませんが、負荷がモーターの場合、トランジスタがOFFしてもモーターはすぐに停止できずに慣性で空転します。すると、モーターが発電機となって、これまで供給している向きと反対の逆起電力が発生するので、この電圧による逆向きの電流をモーターコイルに流して（還流）解消するために、モーターと並列に通常の電流の流れを妨げる向きにダイオードをつなぎます。



◇親機・子機とは別にモーター基板を作成します。図に従い、配線を行って下さい。

◇図は、電池2個の電源となっていますが、演習キットに含まれている電池×３個用のケースを使用して配線します。
基盤外へのジャンパー線は、次の様に配線します。
緑 → 親機：H17へ（H17と言うのは基板上のアドレスです）
黒 → モーター基板：GNDへ

◇使用するトランジスタ（2SA1015）は、左の脚ピンからE,C,Bとなっています。

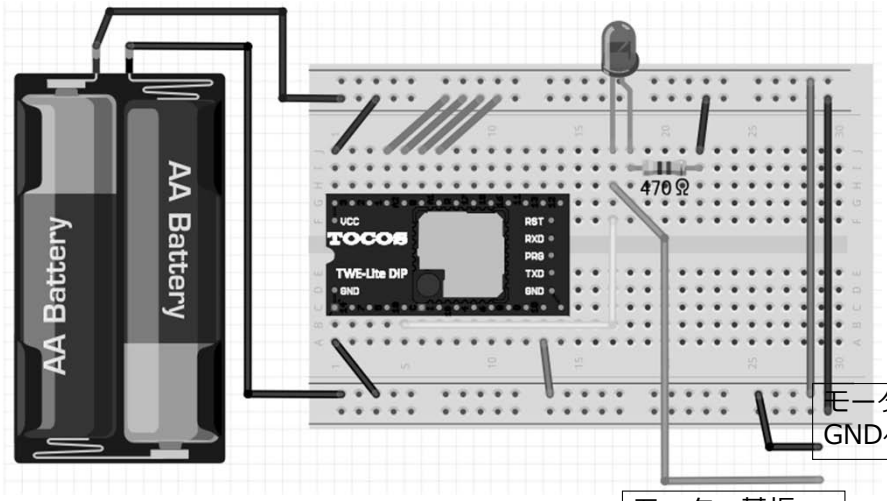
◇前の図で解説したR1とR2は、同じ値（470Ω）を使用して配線します。

◇モーターは、リード線を色分けしていますが、反対に配線しても問題ありません。
回転の向きが反対になるだけです。
※モータの主軸の回転を目で見やすくするために、紙片などを貼っておくと良いでしょう。

◇電源がOFFになった場合、モータが空転して生じる逆起電力対策用ダイオードがモーターに並列接続されています。

向きに注意して配線してください。

親機の配線

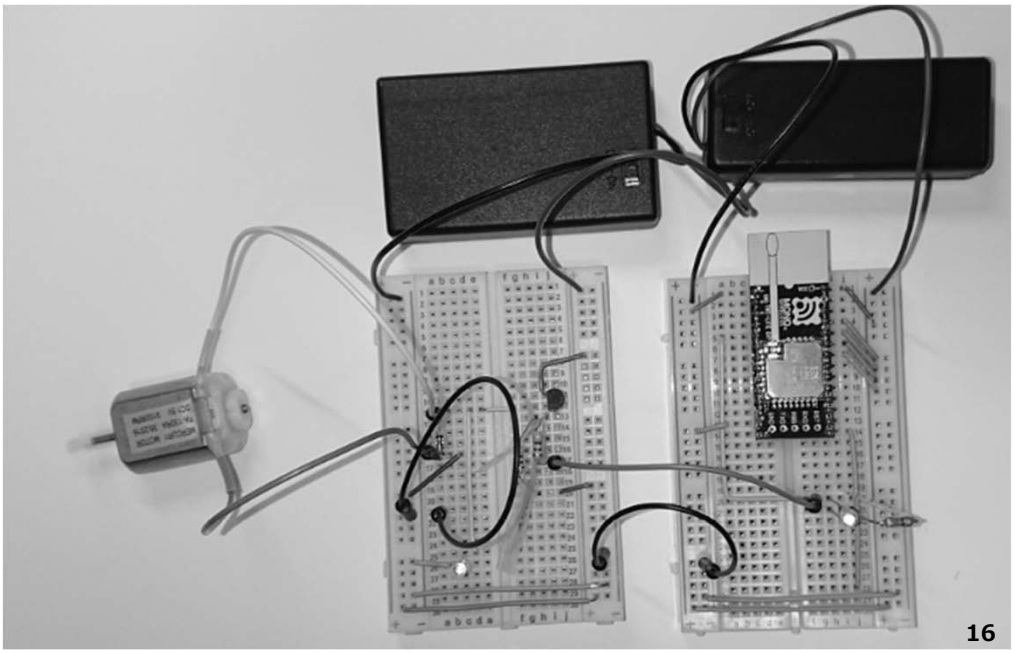


15

◇子機は前回の物をそのまま使用します。

※新たに配線をされる方はStep2を参照して、子機の配線を行って下さい。

作製した回路と動作確認

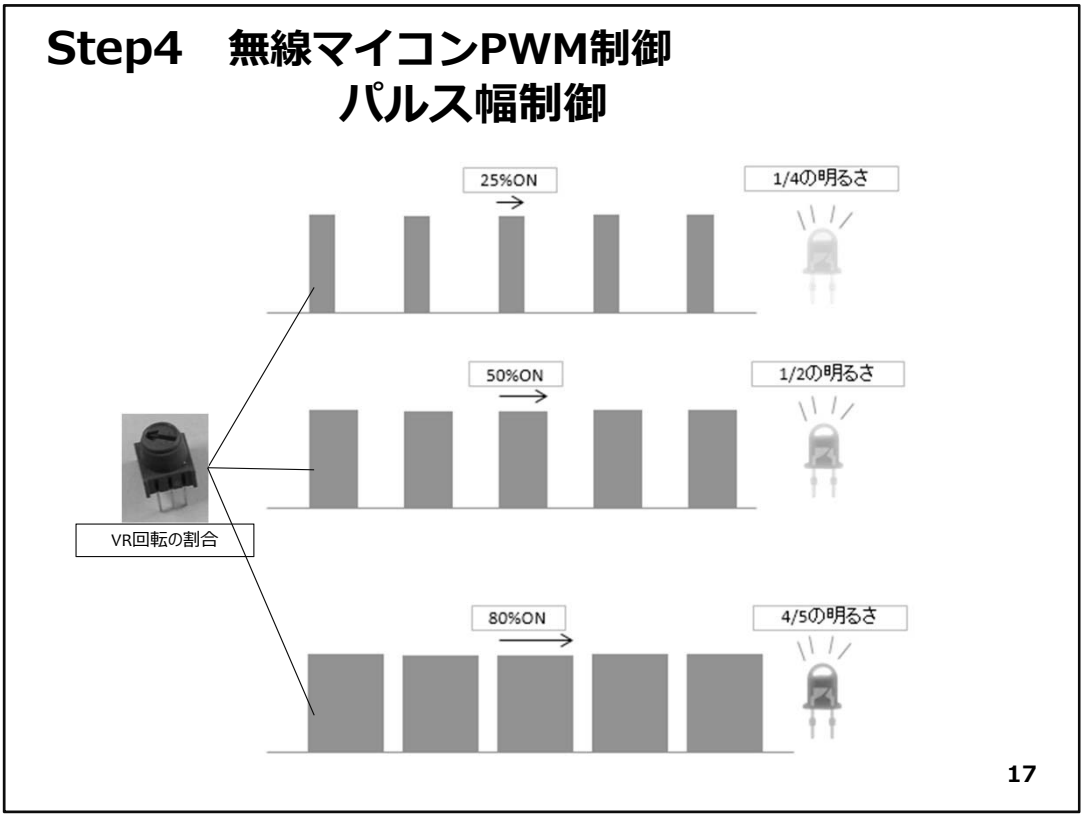


◇作成した回路と動作確認
図は、作成した回路です。左側が、モーター基板です。

◇動作確認は、全ての基板の電源SWをONにスライドして、子機のSWを押してください。
親機のLEDが点灯して、同時にモーターが回転を始めます。
確認のために取り付けたモーター基板のLEDもモーターの回転と共に、明るく点灯することが確認できます。

・・・いかがでしょうか。LEDの点灯が出来ると、その信号を取り出して、外部機器の駆動を行うこともできることが分かりました。流れる電流が小さいときは、トランジスタで増幅すればよいことも分かりました。

・・・次はPWM制御です。



Step4　無線マイコンPWM制御

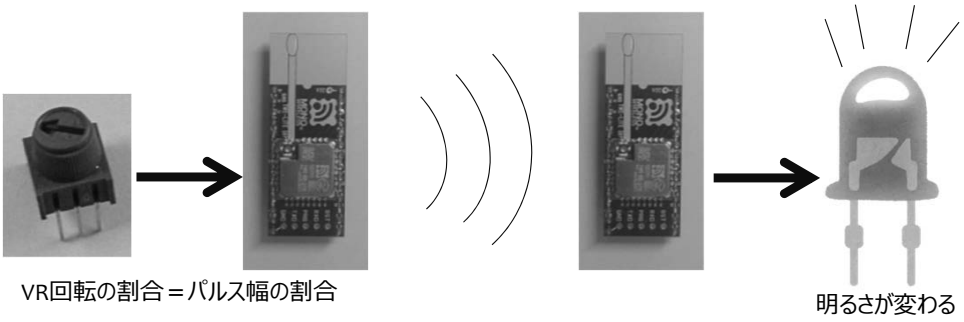
PWM制御という言葉をお聞きになったことがありますか。これは、Pulse Width Modulation の頭文字をとった言葉で、パルス幅変調制御というものです。上の図を見てください。

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・・・と言うように、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。

PWMによるLEDの明るさ制御

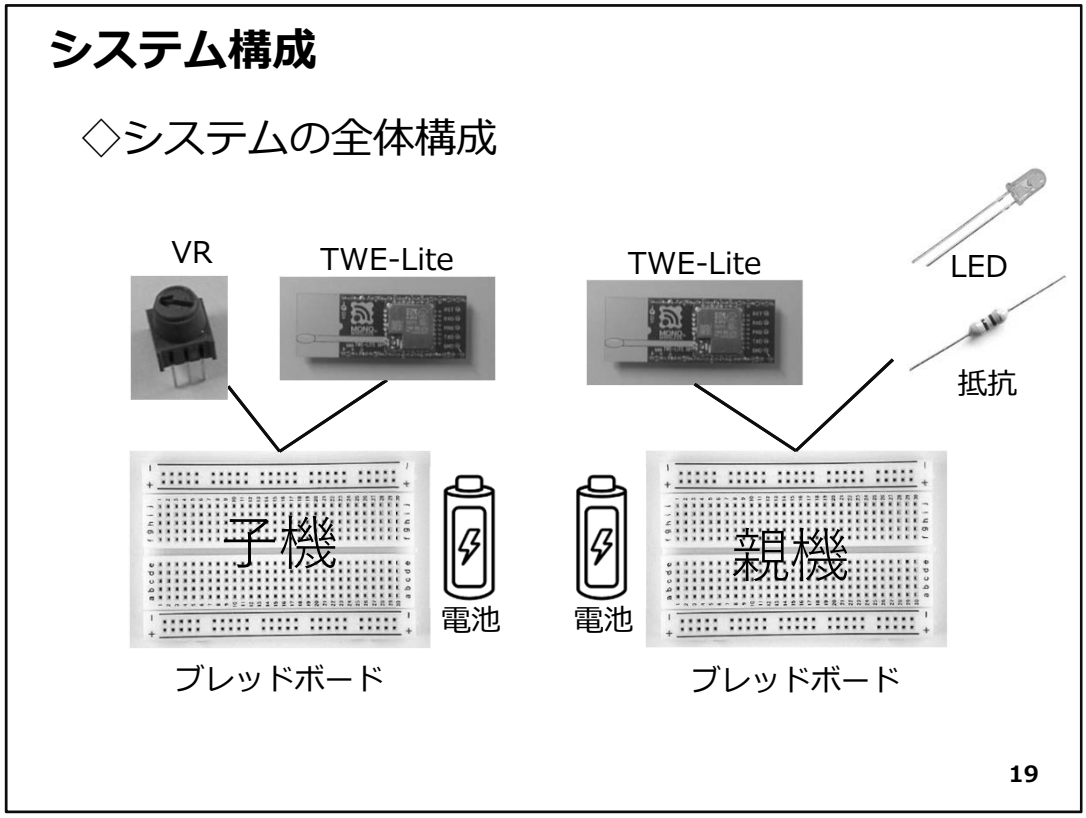
＜＜VRで明るさが変化＞＞

・・・配線するだけです・・・



18

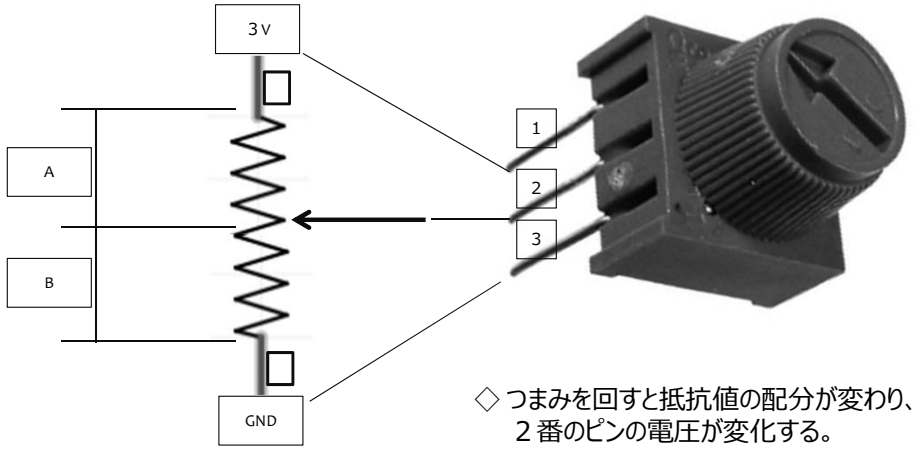
◇上図左側が子機です。VRの両端に電圧を加えてVRを回転させると、回転の割合に応じた電圧がVRから取り出せます。この可変電圧を無線マイコンモジュールのアナログ入力ピンに接続すると、電圧が変化したとき、子機は無線通信でその状態を右側の親機に伝えます。親機はこの無線で通知されたアナログ入力ピンに加えられた電圧に対応するPWM信号をPWM出力ピンに出力しますので、ここにLEDを接続しておくと、子機のVRの回転の割合に対応して親機のLEDの明るさが変化するという具合です。



◇上の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（次の図）。

VR（可変抵抗器）の役割

電圧を分ける → 分圧



◇ つまみを回すと抵抗値の配分が変わり、
２番のピンの電圧が変化する。
◇ この変化する電圧をマイコンで測る。

20

◇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。

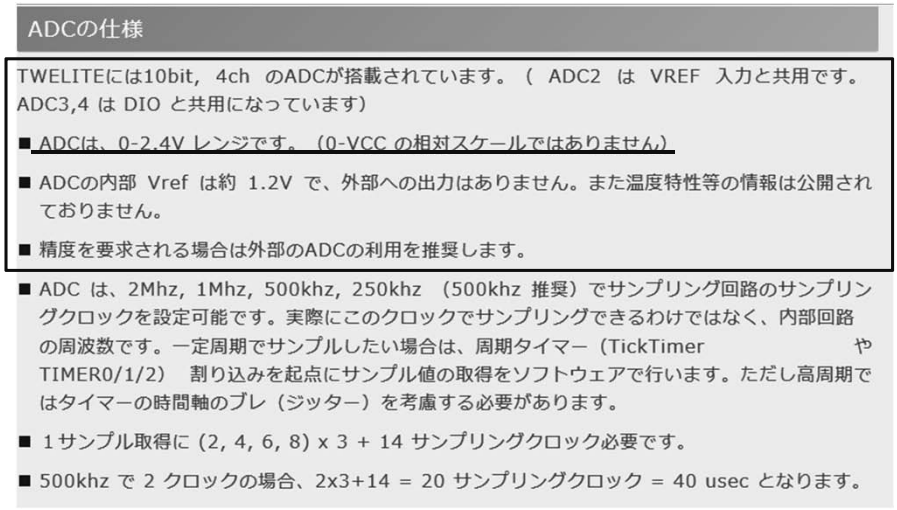
◇1番ピンに3V、3番ピンにGNDを接続すると、2番ピンからは、VRの回転の割合に応じて内部の抵抗がAとBの部分に分割されて、それに対応する電圧が取り出せます。

◇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**と言います。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。

VRの電圧を測るADCの仕様

◇アナログ入力はAD変換器（ADC)



ADCの仕様

TWELITEには10bit, 4ch のADCが搭載されています。（ ADC2 は VREF 入力と共用です。ADC3,4 は DIO と共用になっています）

- ADCは、0-2.4V レンジです。（0-VCC の相対スケールではありません）
- ADCの内部 Vref は約 1.2V で、外部への出力はありません。また温度特性等の情報は公開されておりません。
- 精度を要求される場合は外部のADCの利用を推奨します。
- ADC は、2Mhz, 1Mhz, 500khz, 250khz （500khz 推奨）でサンプリング回路のサンプリングクロックを設定可能です。実際にこのクロックでサンプリングできるわけではなく、内部回路の周波数です。一定周期でサンプルしたい場合は、周期タイマー（TickTimer や TIMER0/1/2） 割り込みを起点にサンプル値の取得をソフトウェアで行います。ただし高周期ではタイマーの時間軸のブレ（ジッター）を考慮する必要があります。
- 1サンプル取得に (2, 4, 6, 8) x 3 + 14 サンプリングクロック必要です。
- 500khz で 2 クロックの場合、2x3+14 = 20 サンプリングクロック = 40 usec となります。

※メーカー資料抜粋

21

◇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、

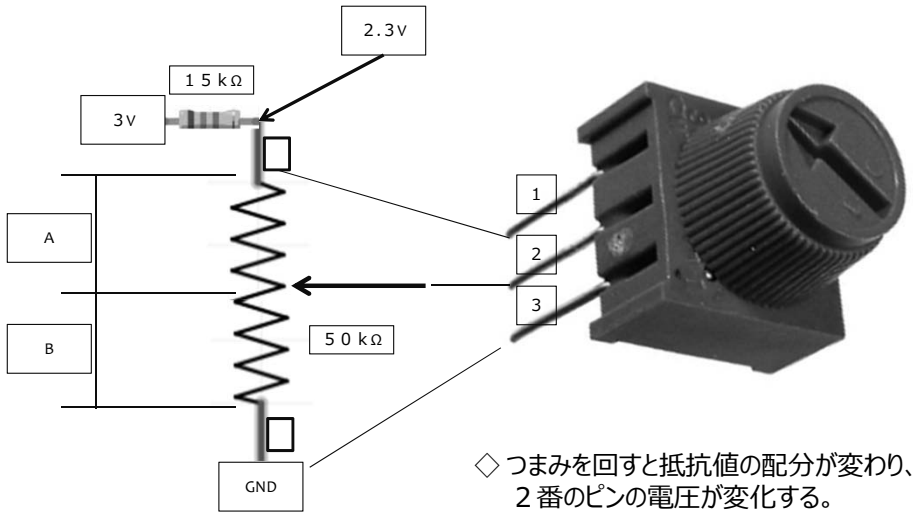
$$\frac{50}{50+15} \times 3 \approx 2.3 \text{〔V〕}$$

となります。
◇15kΩの抵抗をVRの電源側に接続します。

22

◇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は次の図のようにします。

VR（可変抵抗器）の回路



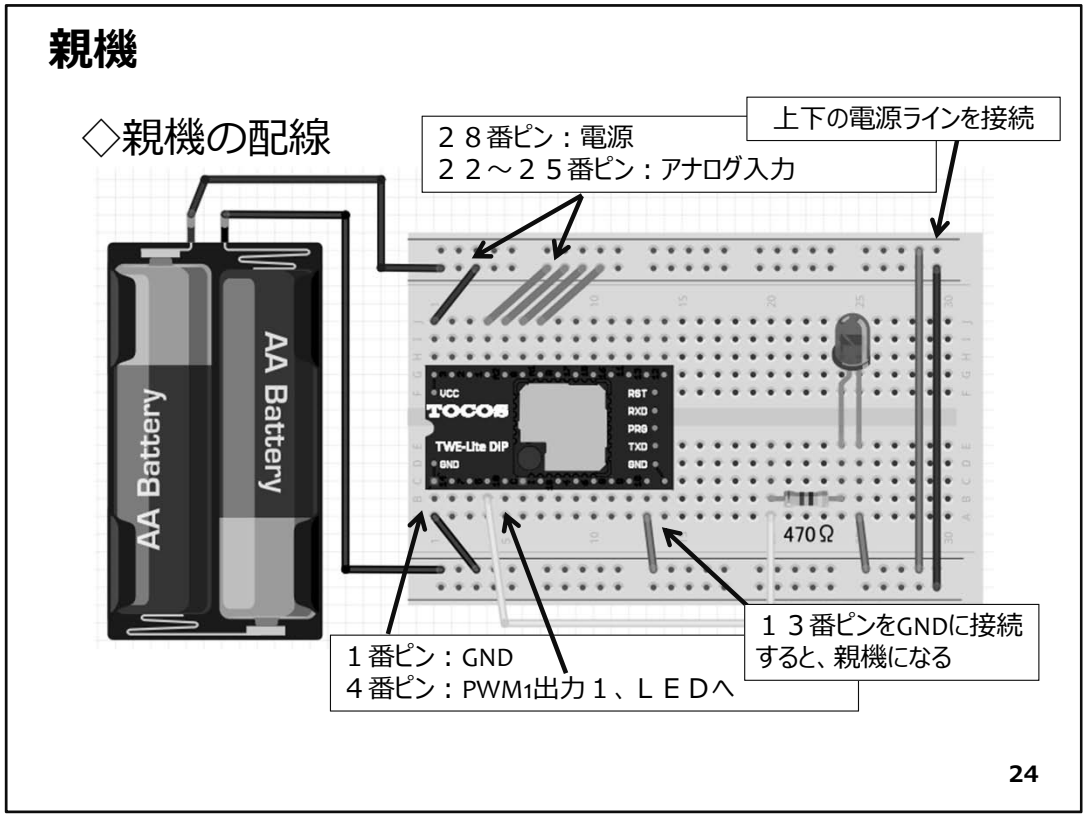
◇ つまみを回すと抵抗値の配分が変わり、
２番のピンの電圧が変化する。
◇VRを目いっぱい回しても、１５：５０で
分圧されて、出力が２.４V未満になる。

23

◇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

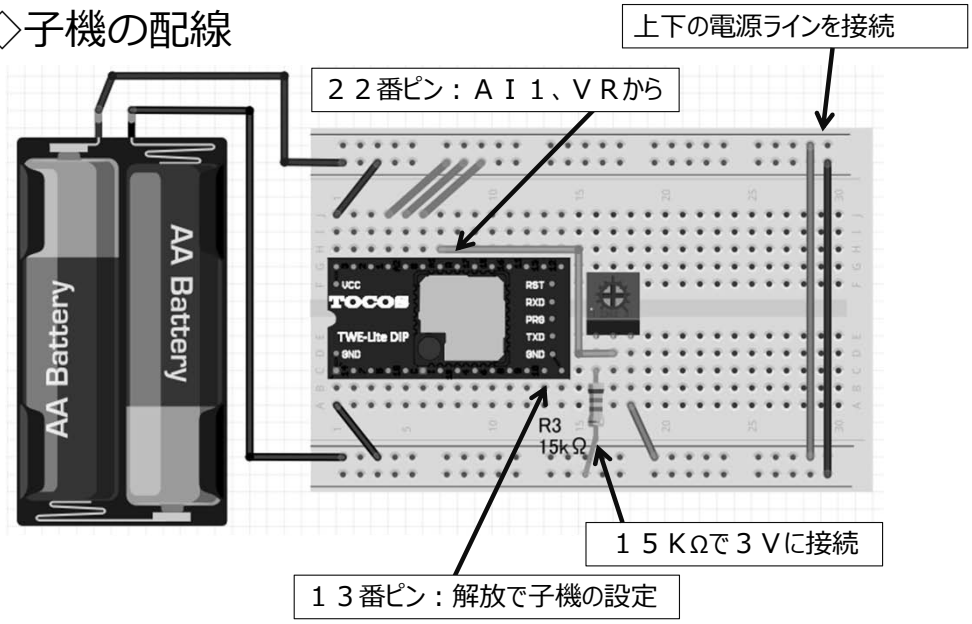
親機側:
1. 無線マイコンモジュール×1台
2. ブレッドボード×1個
3. 電池（単4乾電池×2個＋SW付電池ケース×1個）×1セット
4. 配線用ジャンパー線
5. LED×1個
6. 抵抗器（470Ω）×1個



◇では親機の配線を上の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、これまでの演習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。今回の演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。

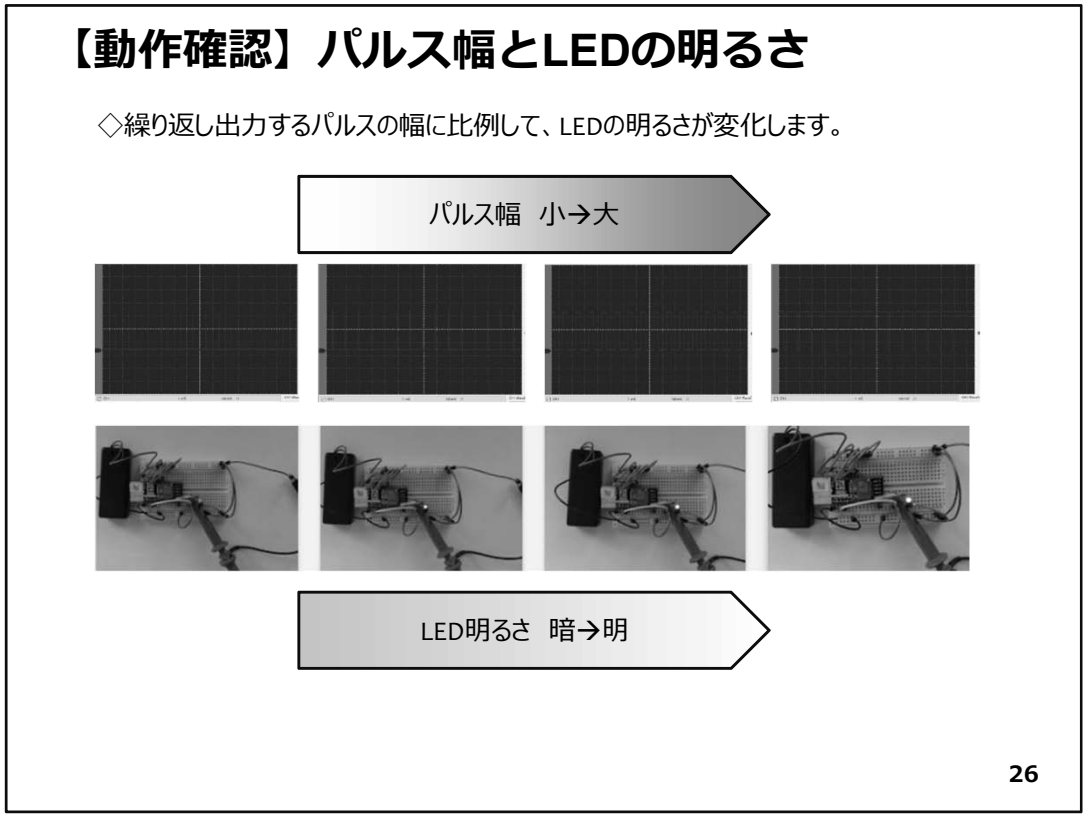
子機

◇子機の配線



25

◇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。



◇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。

上の写真は、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。

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。

Step5　無線マイコン PC間通信

◇IoTに有効なツール、【温度センサー】を使います。

◇温度を測定、無線で送り、PCで処理.
◇PC側の処理プログラムを作る.
◇処理系 → Python 使用.
◇PC側 開発環境の説明あり.

【重要】 先に進む前に、必ず以下をお読みください。
【Appendix D：A/D変換と温度センサー】
【Appendix E：Python開発環境の準備】

27

Step5　無線マイコン PC間通信

◇Step5では、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。主な内容は上の通りです。

◇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。

【重要】
このステップでは、
1．温度センサー
2．ADC（アナログ・デジタル変換器）
3．Python

を使用します。これらについては、編集の都合で、【Appendix】で詳しい解説を行いました。先に進む前に、必ず以下をお読みください。
【Appendix D：A/D変換と温度センサー】
【Appendix E：Python開発環境の準備】

システム概要

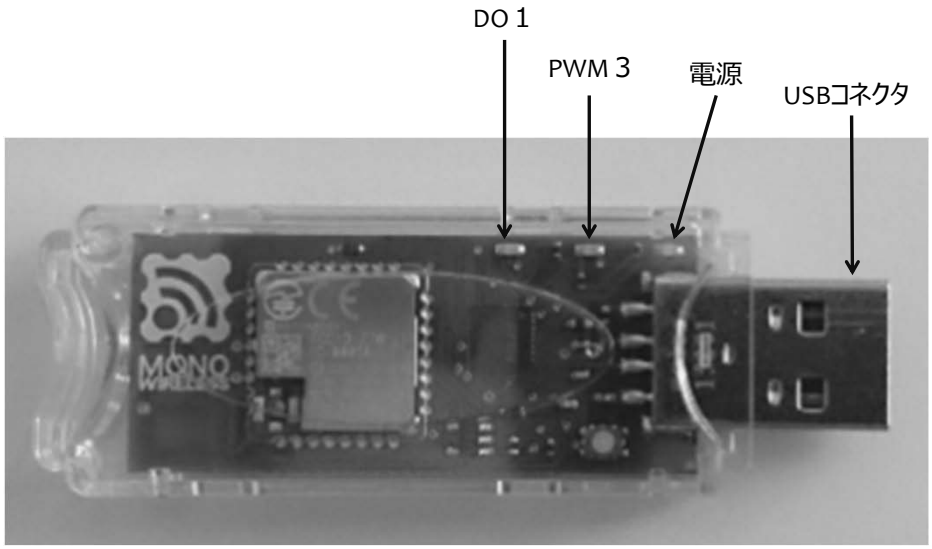
◇温度を無線通信 + PC表示

28

◇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、演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。

TWE-Lite ＭｏＮｏＳＴＩＣＫ



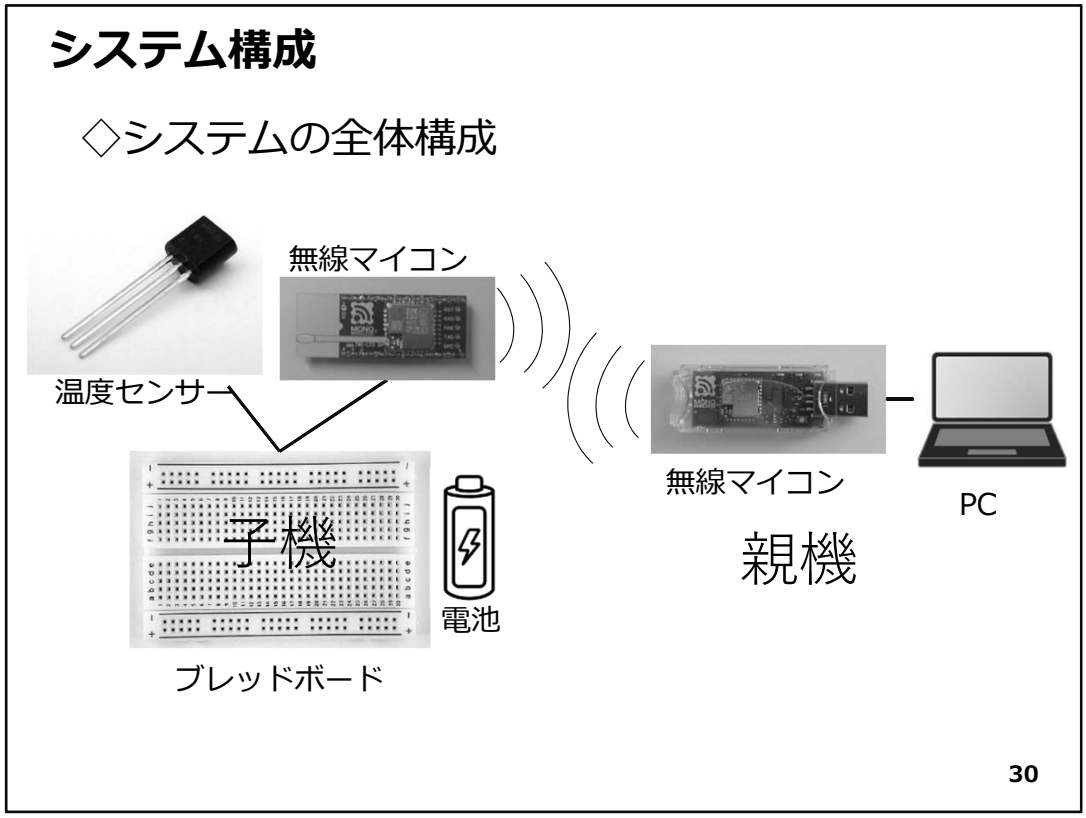
【重要】Windows PCで認識される、COMポート番号を確認しておくこと。
後にソースコードに記述する必要がある。

29

◇この演習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PCやスマートフォンに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PC・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辺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。

【重要】 MoNoSTICKをPCのUSBコネクタに差し込むと、Windowsでは仮想COMポートが設定されます。Windowsのデバイスマネージャで、このCOM番号を確認してください。
　COM番号は、後でPythonソースコードを入力する際に、必要になります。確認したら、メモを取って保存しておきます。



◇システムの全体構成は、上の図の通りです。左上がアナログ温度センサーです。

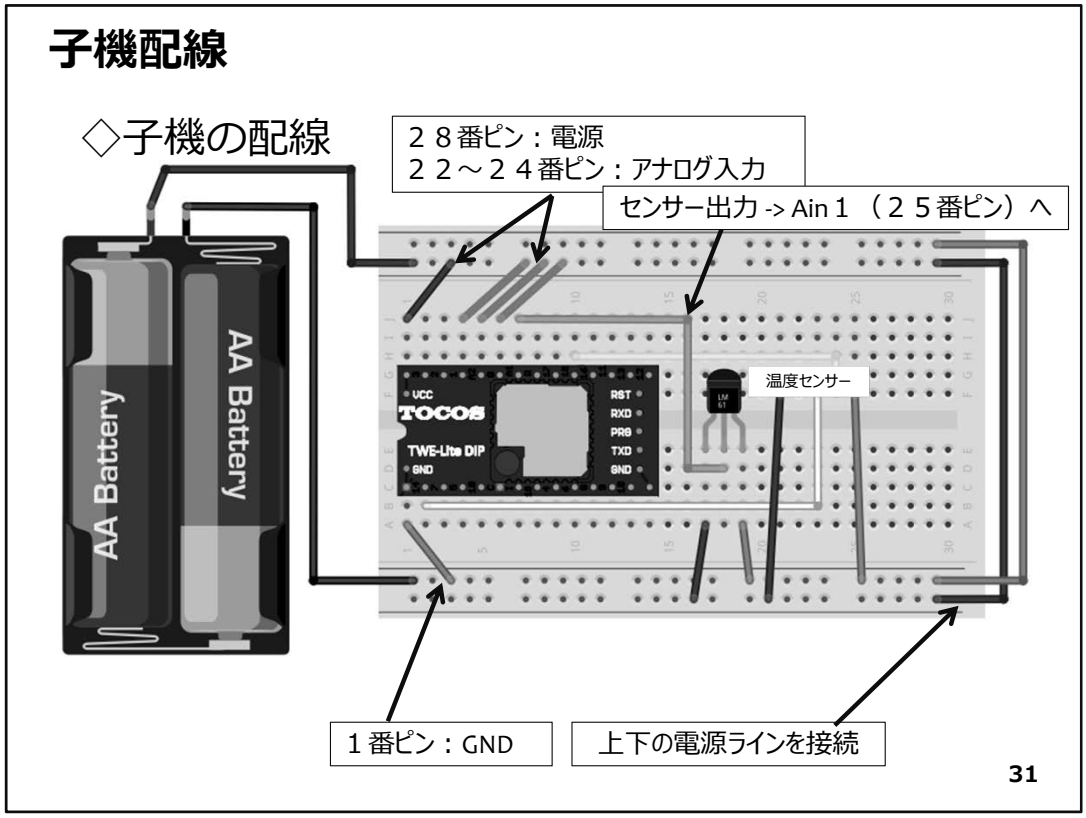
◇今回の演習で使用するパーツは次の通りです。

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. MoNoSTIC×１台
2. PC×1台（Windows10＋Internet接続）
3. Python開発・実行環境×1セット　→ Python環境はAppendixを読み、準備します。



◇子機配線

◇パーツの準備が出来たら上の図に従い子機を配線します。

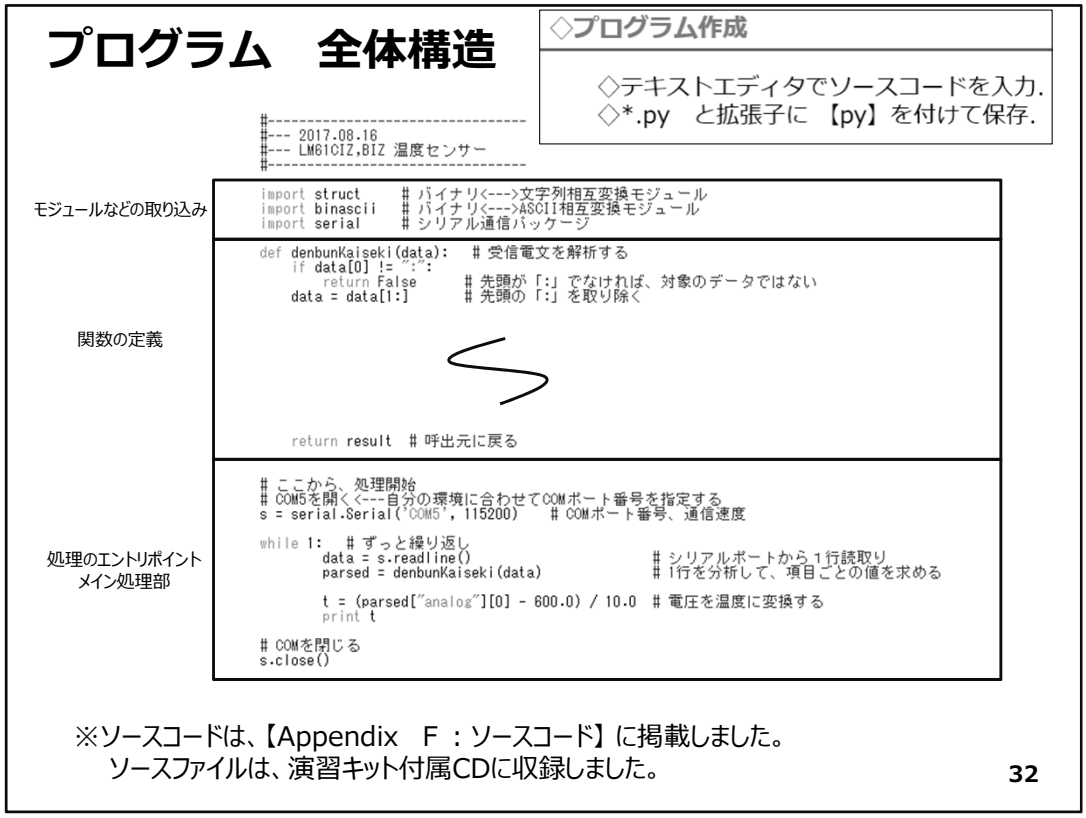
◇これまで、25番ピンに接続されていたアナログ入力を安定させる配線に代えて、温度センサーからの出力に接続します（図では緑色）。

◇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。温度センサーの電源とGNDを反対向きに配線すると、大変高温になり、回復不可能なダメージを生じて、正常に温度を計測出来なくなりす。

◇配線は容易だと思いますが、一発でうまく動作するように、念入りに確認しておきましょう。

◇子機の配線が確認できたら、次は、プログラムの作成です。

.



◇テキストエディタでソースコードを入力します。ソースコードは、Appendixに掲載していますので、参照して入力して下さい。また、ファイルとしては、演習キット付属CDに収録されています。

◇この演習のPythonのプログラムは大まかに図のような3つの部分に分かれた構造をしています。

①. 最初にモジュールなどの取り込み
②. 次に関数定義の部分
③. 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。

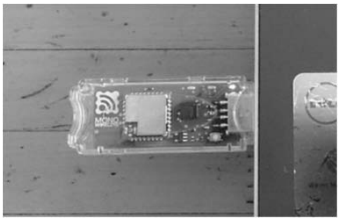
【＃より右側はコメントです。】

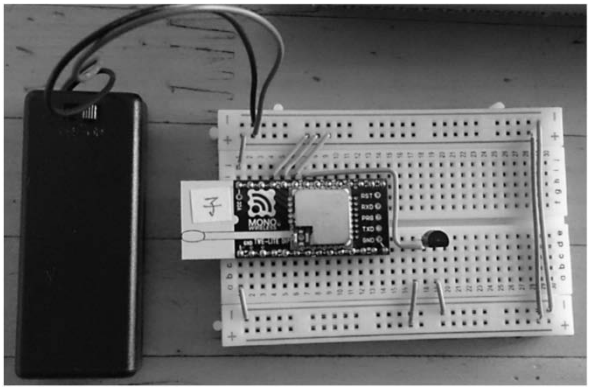
◇*.py　と拡張子に【py】を付けて保存。
Appendixにも書きましたが、ソースコードの入力が終了しましたら、*.py　と拡張子に【py】を付けて保存してください。

※ソースコードは、演習キット付属のCDに収録しています。ほぼすべての行にコメントを付けました。

動作確認：親機と子機の準備

◇MonoStickをPCにセットする.
◇子機の電源をON！！

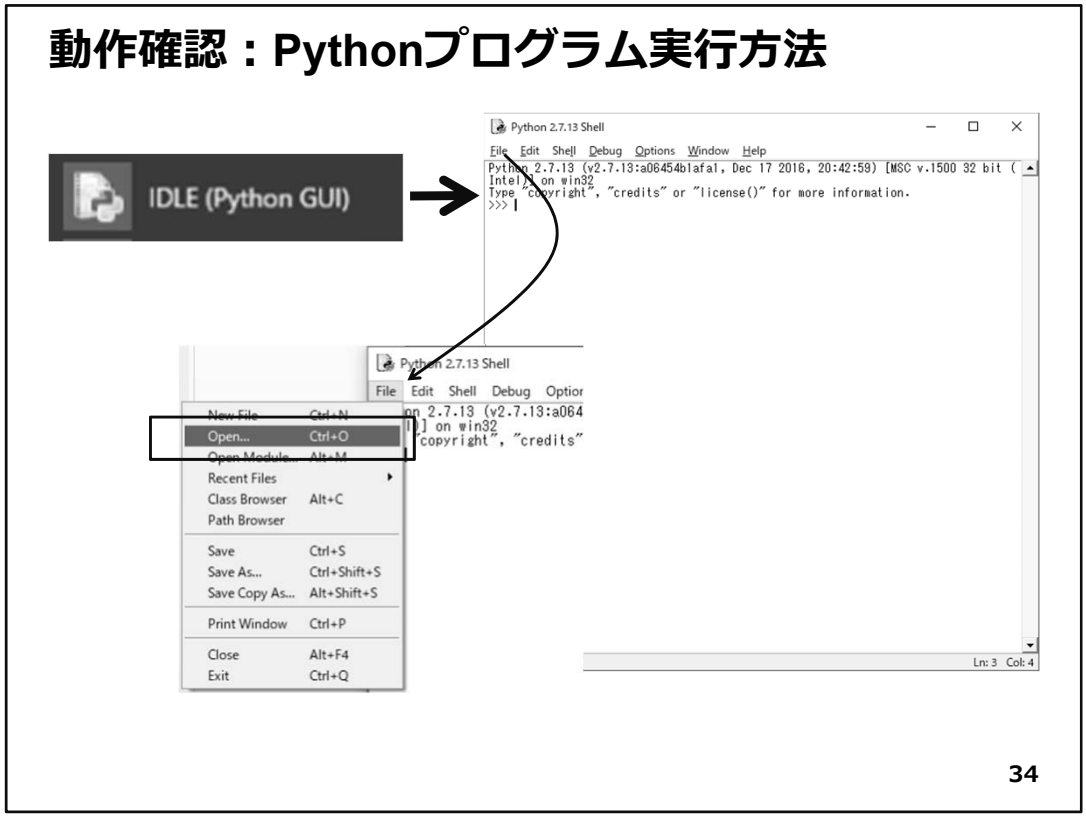




33

◇動作確認：準備

①. MonoStickをPCにセットします。（USBコネクタに差込みます。）
②. 子機の電源SWをONにスライドします。
③. Pythonプログラムを実行します。実行方法は、次で説明します。

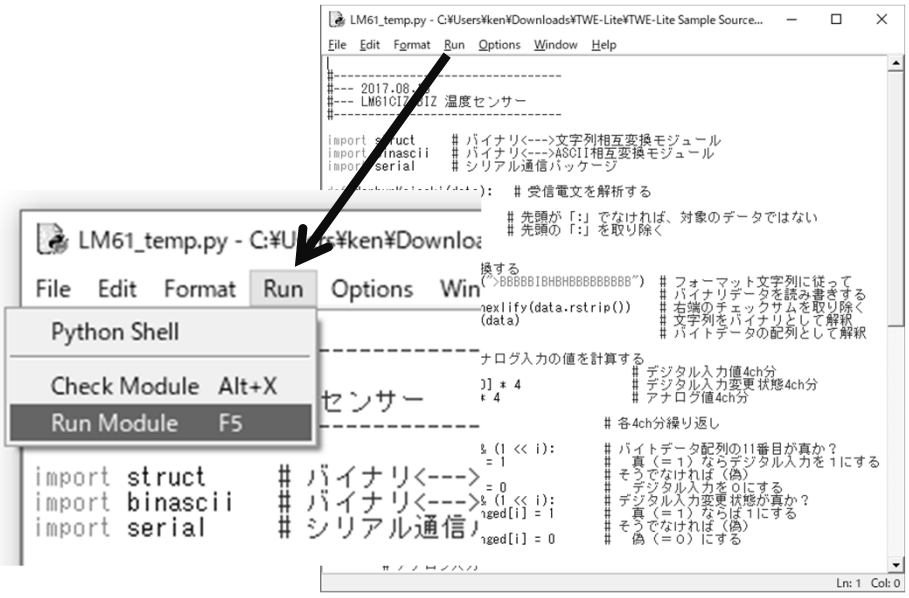


◇Python PGの実行方法

以下の手順で、Pythonのプログラムを実行します。

①. Python環境の準備でインストールされている【IDLE】を起動する
②. 該当するPythonのソースファイルを開く

Pythonプログラム実行方法

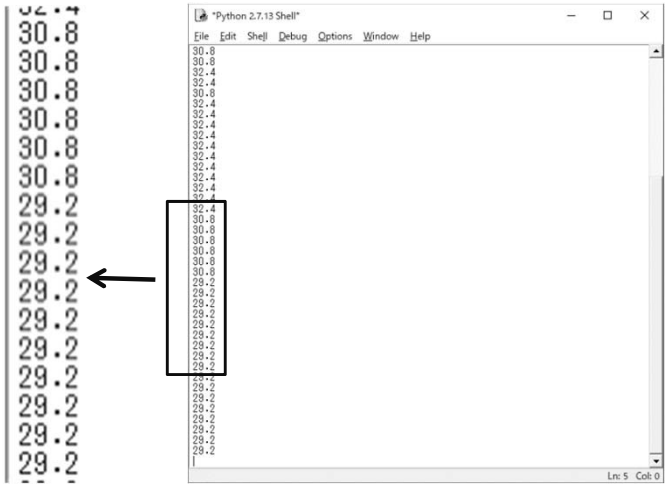


35

③. 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。

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.



36

◇動作結果

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。

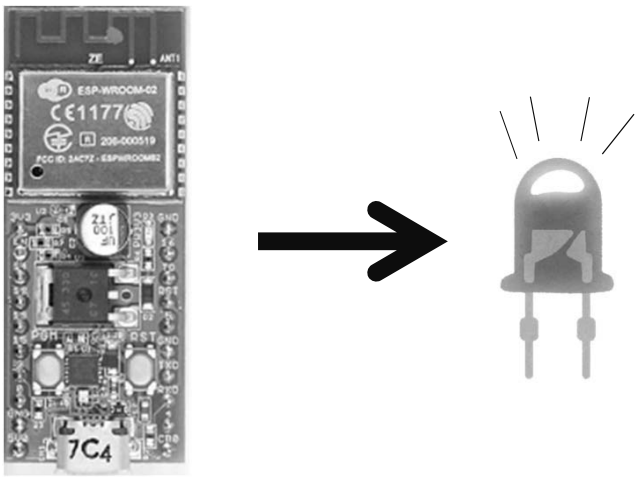
子機は電池駆動で親機とは無線通信をしていますから、測定場所を移動することが容易にできます。これが無線マイコンモジュールの便利なところです。

今回の演習では、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が、表示器が使えると、この点が改善できます。

後のWiFiマイコンを使用した演習で液晶表示器（LCD）の使い方を学びます。

Step6　WiFiマイコン デジタルI/O

＜＜LEDを点滅する＞＞

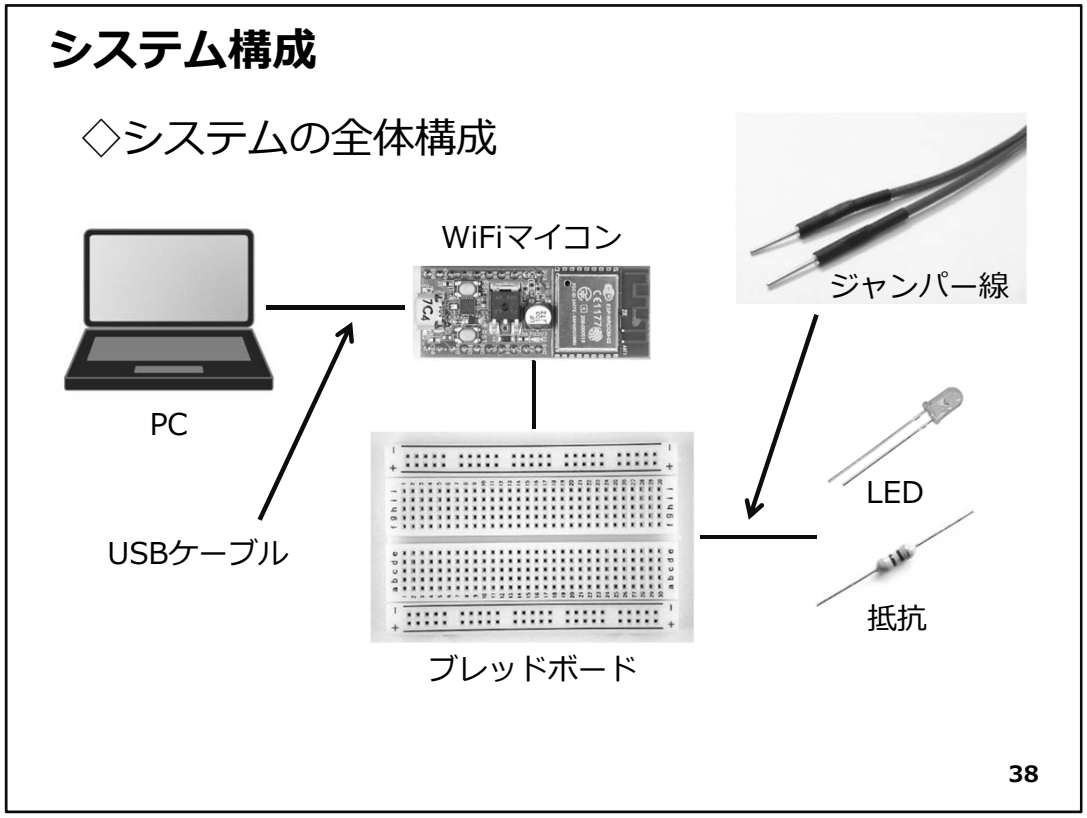


37

Step6　WiFiマイコン　デジタルI/O

◇これから使用するマイコンは、WiFi機能を持つマイコン（ESP8266）で、そのまま単独でアクセスポイントに接続できます。これからの一連の演習では、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。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。



◇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。

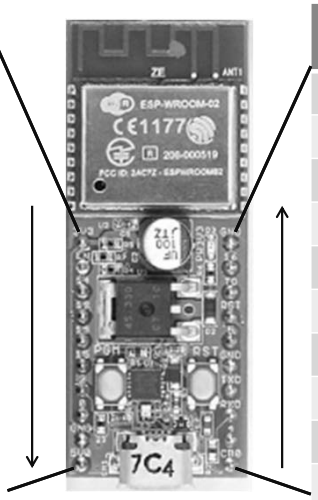
1. WiFiマイコン（ESP8266）×1台
2. PC（プログラム開発・書込）×1台
3. USBケーブル（マイコンとの接続）×1本
4. ブレッドボード×1個
5. 配線用ジャンパー線×適宜
6. LED×1個
7. 抵抗器（470Ω）×1個
8. SW×1個 （後半のデジタル入力で使用）

※LEDは、次の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かりにくいので、長い方の脚を曲げて表現しています。

WiFiマイコン　ピン配置

ESP-WROOM-02

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

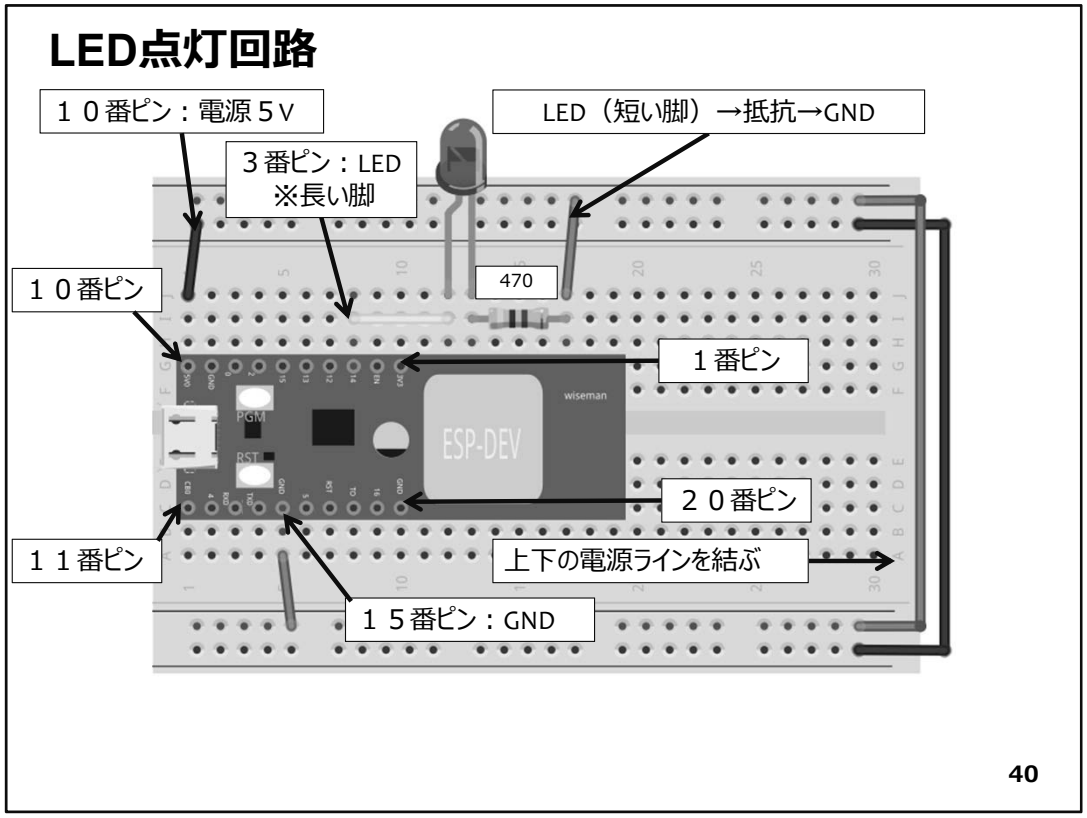


番号	名称	機能
20	GND	GND
19	16	GPIO16
18	TO	ADC（max1V）
17	RST	リセット
16	5	GPIO5
15	GND	GND
14	TXD	TXD
13	RXD	RXD
12	4	GPIO4
11	CB0	通信モニタ用

39

◇上の図は、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。

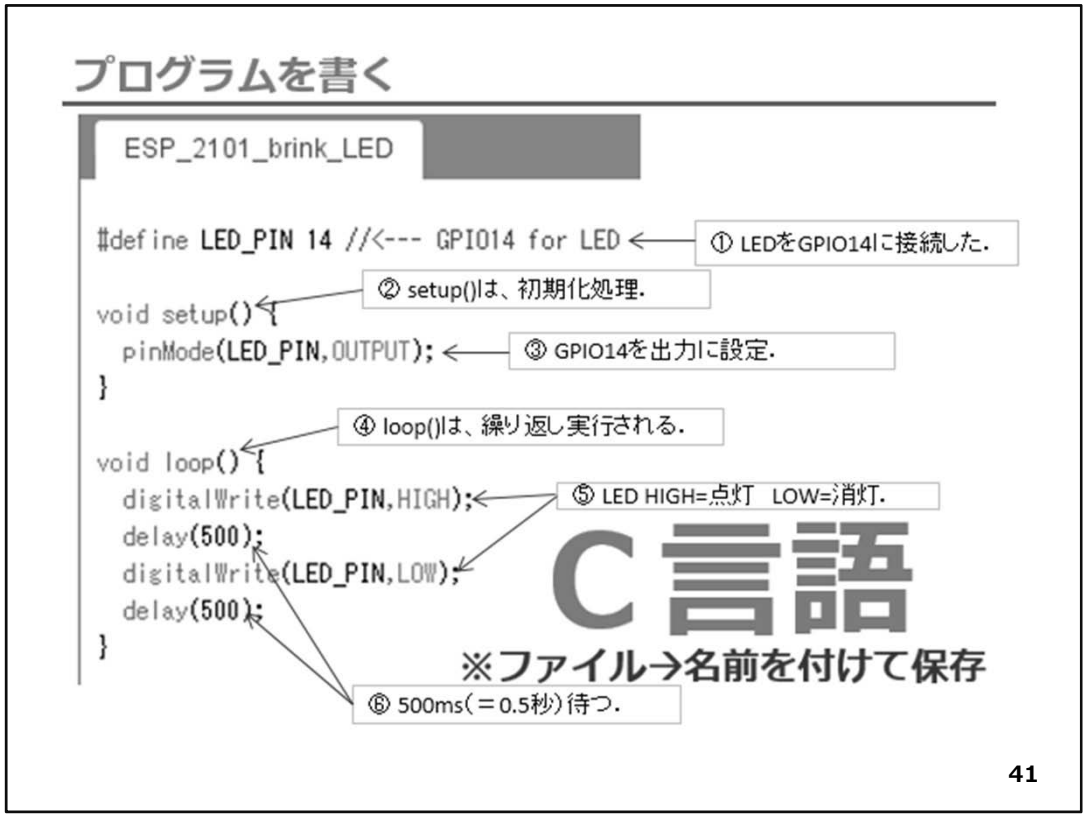


◇いよいよ無線マイコンモジュールの実験回路を配線します。**配線は必ずUSBケーブルを外した状態で行います。**配線と言っても線の数や使用するデバイスの数が少ないので、じっくりと確認しながら行ってもすぐに終わります。各パーツには、くれぐれも余計な力がかからないようにして、回路の配線を行ってください。手順としては、半田付けする基板の場合は、**【高さの低いものから順に基板に取り付けてゆく】**のが王道ですが、ブレッドボードとジャンパー線による配線なので、作業しやすいと思った順に行えばよいでしょう。回を重ねれば、自ずと最適な作業手順が身についてくると思います。

◇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**、次の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。

◇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。



◇**基礎編のAppendix**で準備した、**ArduinoIDE（WiFiマイコン開発環境）**を起動して、ソースコードを入力します。

◇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、上の図の通り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、この演習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

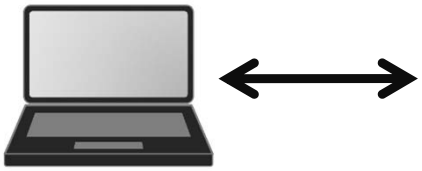
が点灯(GPIO=1)したり消灯（GPIO=0）したりします。⑥deley()は、指定の時間、プログラムを一時停止します。単位はms（1/1000秒）です。

◇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。

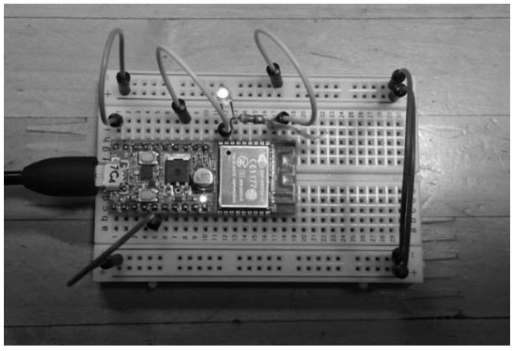
◇ソースコードを入力したら、名前を付けて保存してください。以後このIDEでは、ソースコードは**【スケッチ】**という名称で管理されます。

プログラムの書込み

PCと接続
【USBケーブル使用】



PC



42

◇プログラムの書込み:

WiFiマイコンに、プログラムを書込みます。USBケーブル経由でIDEの機能を使い、マイコン内部のフラッシュメモリにプログラムを書込みます。

まず、プログラムを保存してください。そして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】

上の図右の写真では、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

【参考】
COMポートドライバがインストールされていない場合は、FTDI社のHP（下記）にあるVCPドライバをインストールして下さい。
http://www.ftdichip.com/Drivers/VCP.htm

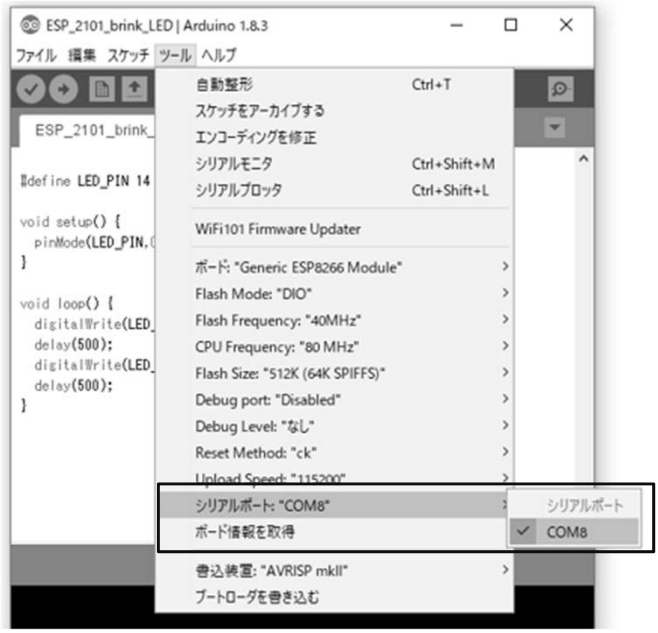
43

◇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。

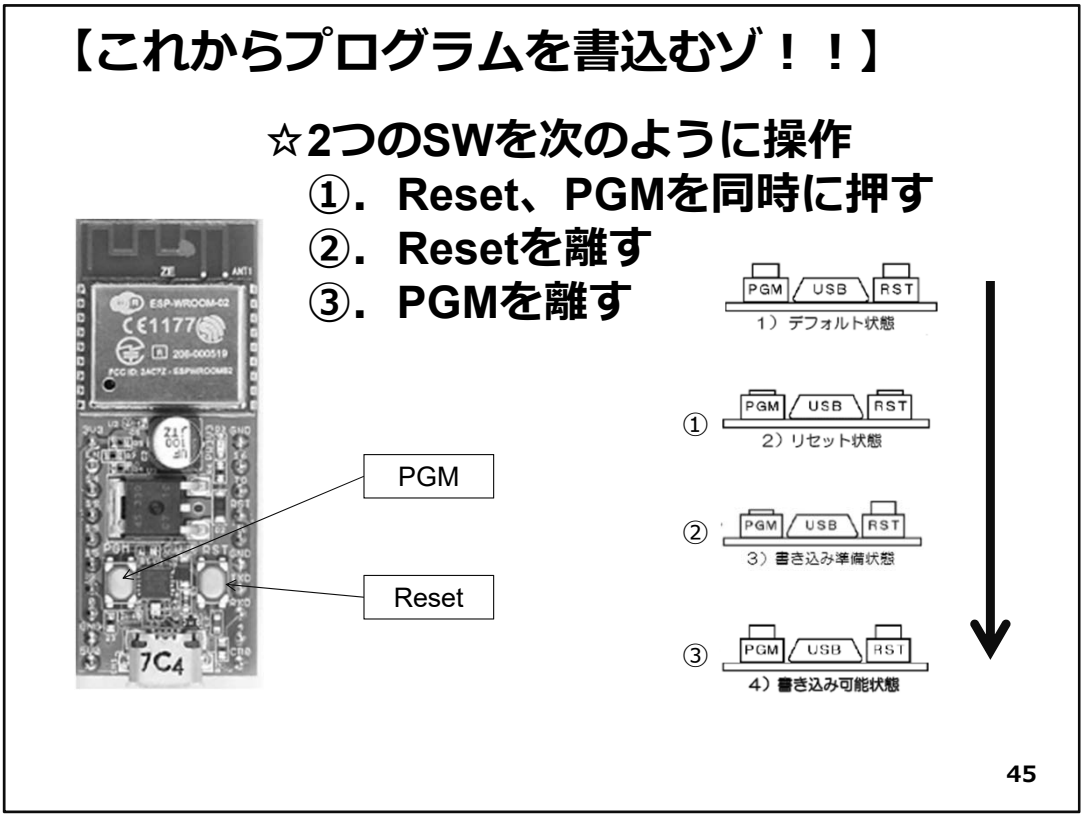
シリアルポートの設定



44

◇【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。



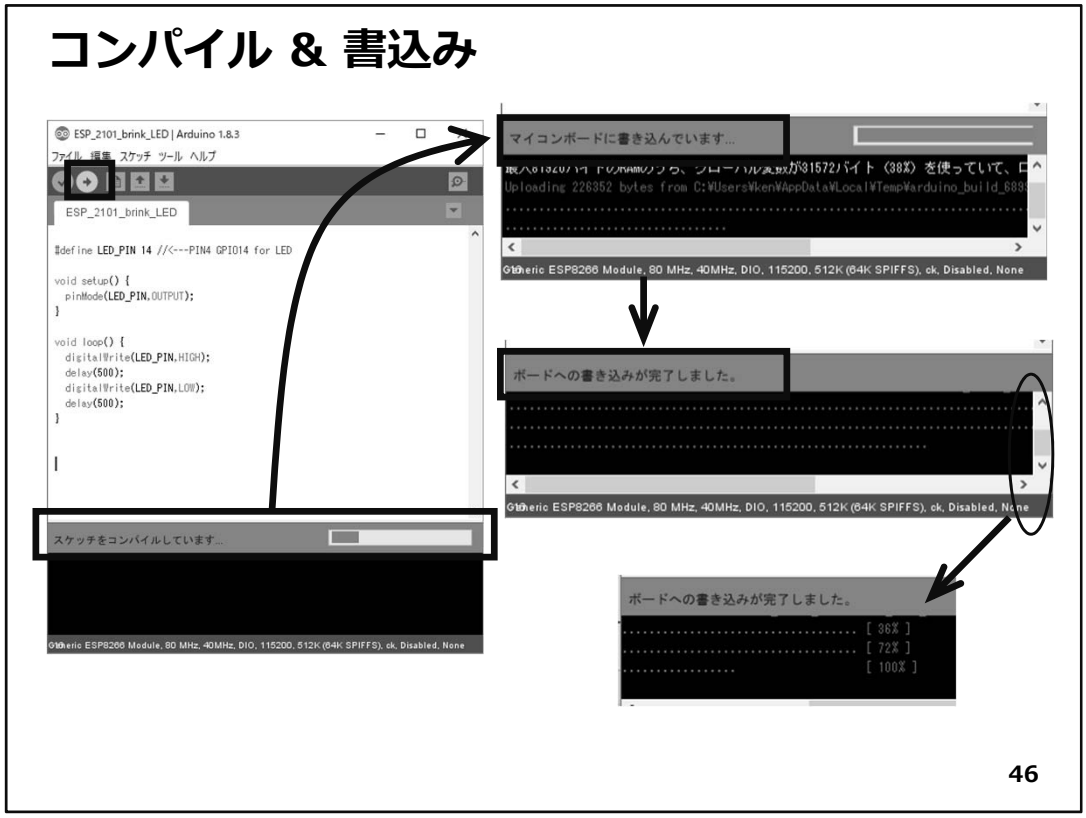
◇**【これからプログラムを書込むゾ！！】**

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。



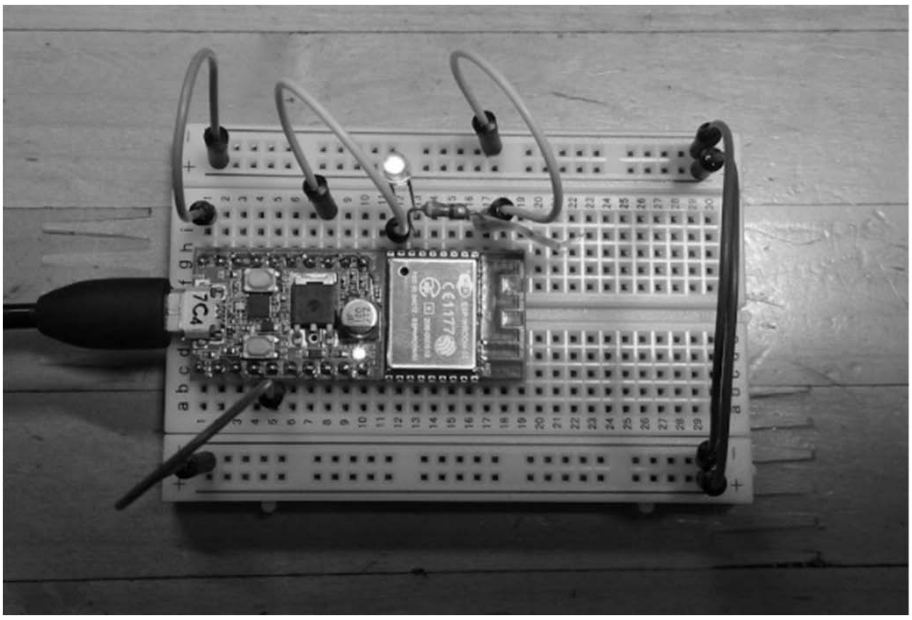
◇これから、プログラムのコンパイル・リンクとWiFiマイコンモジュールへの書込みを行います。IDEの左上にある、【右向き矢印ボタン】をクリックしてください。

◇ボタンを押すと上の図の様に、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。

【重要】

【これからプログラムを書込むゾ！！】からコンパイル、書込みまでは、今後の各Stepで必ず行う事なので、自然に覚えてしまうと思いますが、もし手順に不安があったら、この部分を参照して、実行してください。

実行確認：動作の様子



47

◇実行確認:

◇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、WiFiマイコン第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。

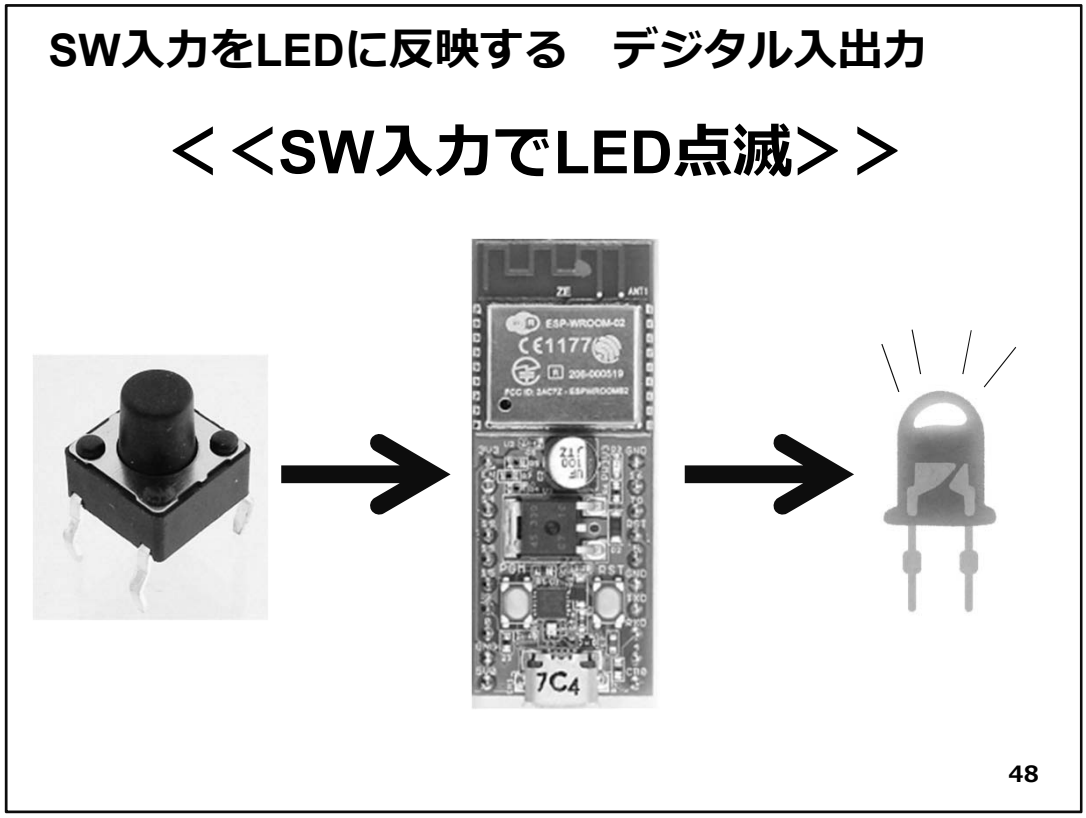
----------<< WiFiマイコン開発手順 >>----------

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。

◇続けて、SWによる入力を行います。



◇DI/DOの実験

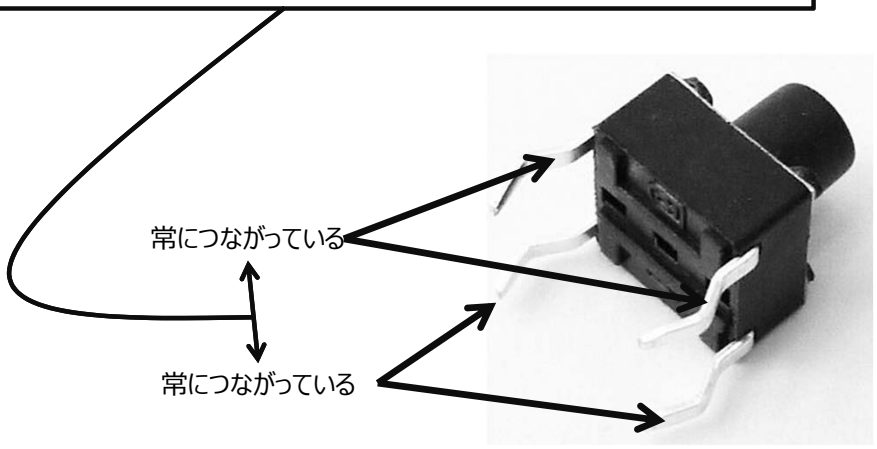
LEDのON/OFF信号がデジタル出力（DO）であったのに対して、SWからの入力はデジタル入力（DI）になります。システムが動作するとSW操作にWiFiマイコンが反応しているように見えます。

次に、先ほど作った回路にSWを追加して、SWの入力（DI)をLED（DO）に反映してみます。

SW（タクトスイッチ）

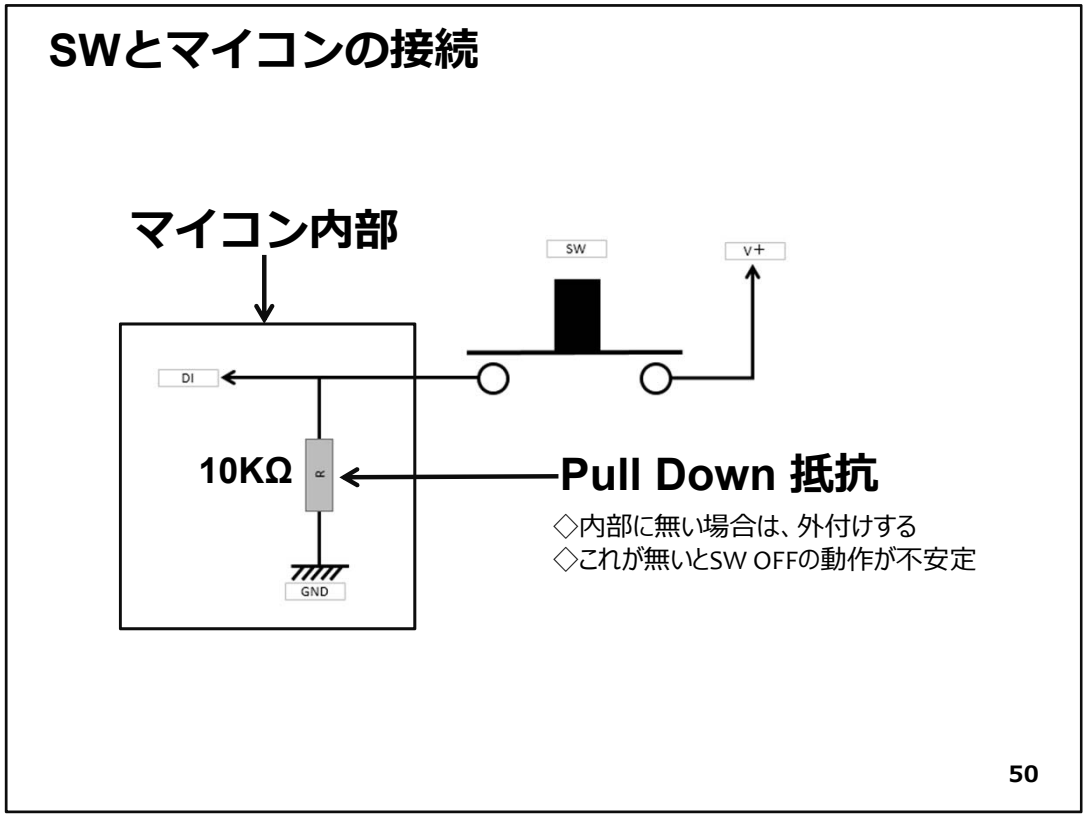
◇動作：

押したとき接点がつながり、放すと切れる



49

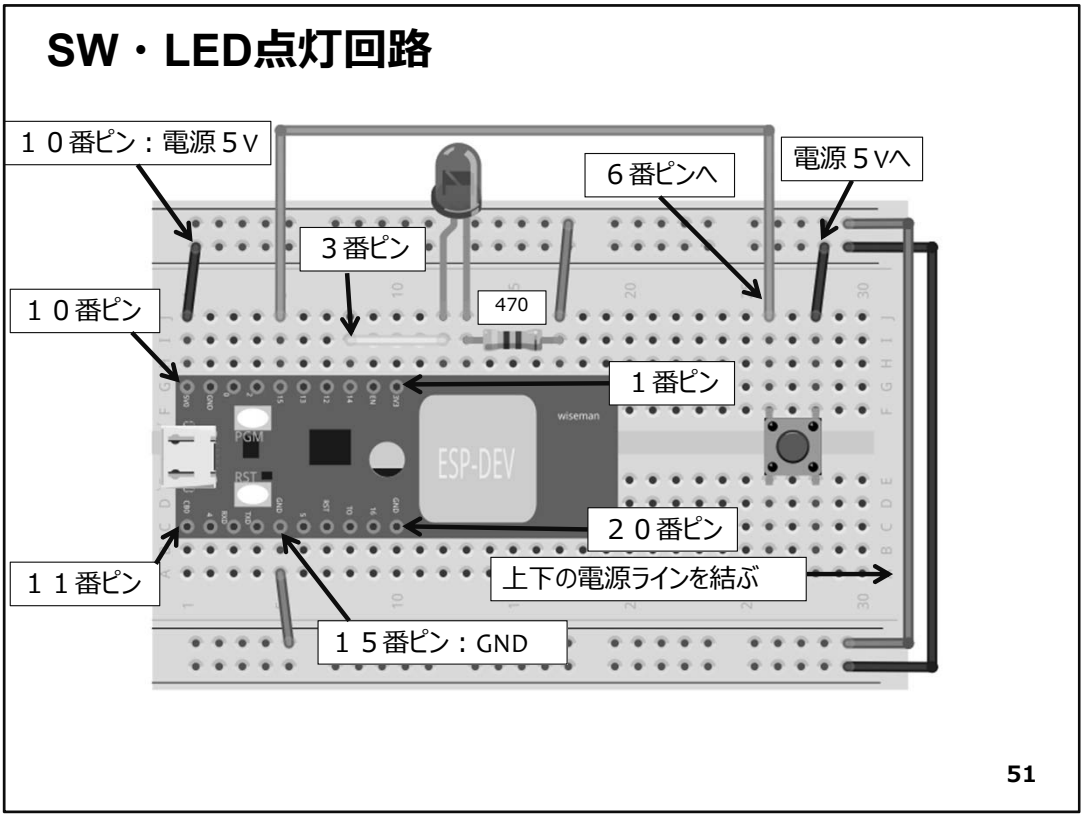
◇今回使用するSWは【タクトスイッチ】という名称の小さなSWです。動作は単純で、押したときに接点が繋がり、放すと切れるというものです。上図で湾曲した脚ピンがありますが、湾曲している向かい合った対になるピンが内部で接続されています。これが2対あり、ボタンを押すと、その2対が内部で繋がる仕組みです。



◇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下の〇印です。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)抵抗と言います。



【重要】

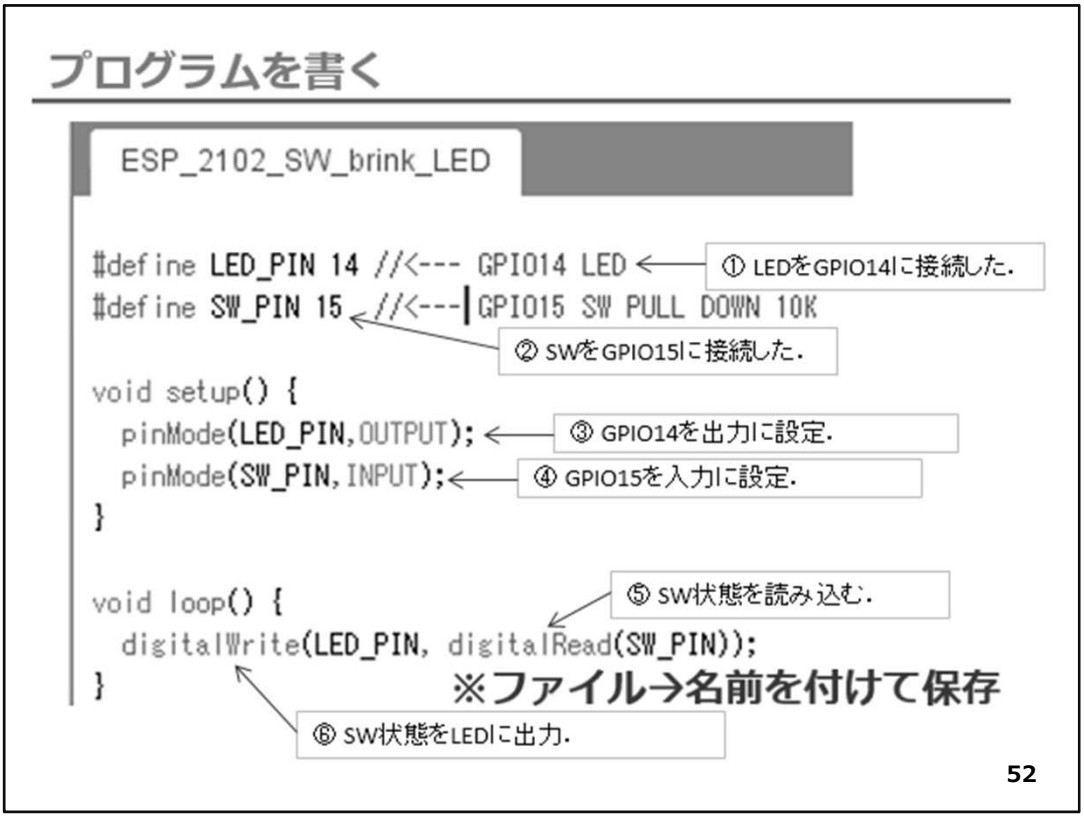
配線を行う際は、必ずUSBケーブルを外して下さい。接続したまま配線を行ってショートが発生すると、PCのUSBポートを壊してしまうことがあります。以後の演習も同様です。

◇まず、上の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。追加する配線は僅かですが、ショートなどしない様に慎重に行ってください。

◇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。

【注意】

この回路は、次回もそのまま使用します。保管が可能であれば、そのままの状態で保管してください。



◇プログラムを書く

先のソースコードに追加して、図の様なプログラムを書きます。

◇新規に作成する場合は、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、白い部分に上の図のソースコードを入力してください。

以下、ソースコードの説明です。

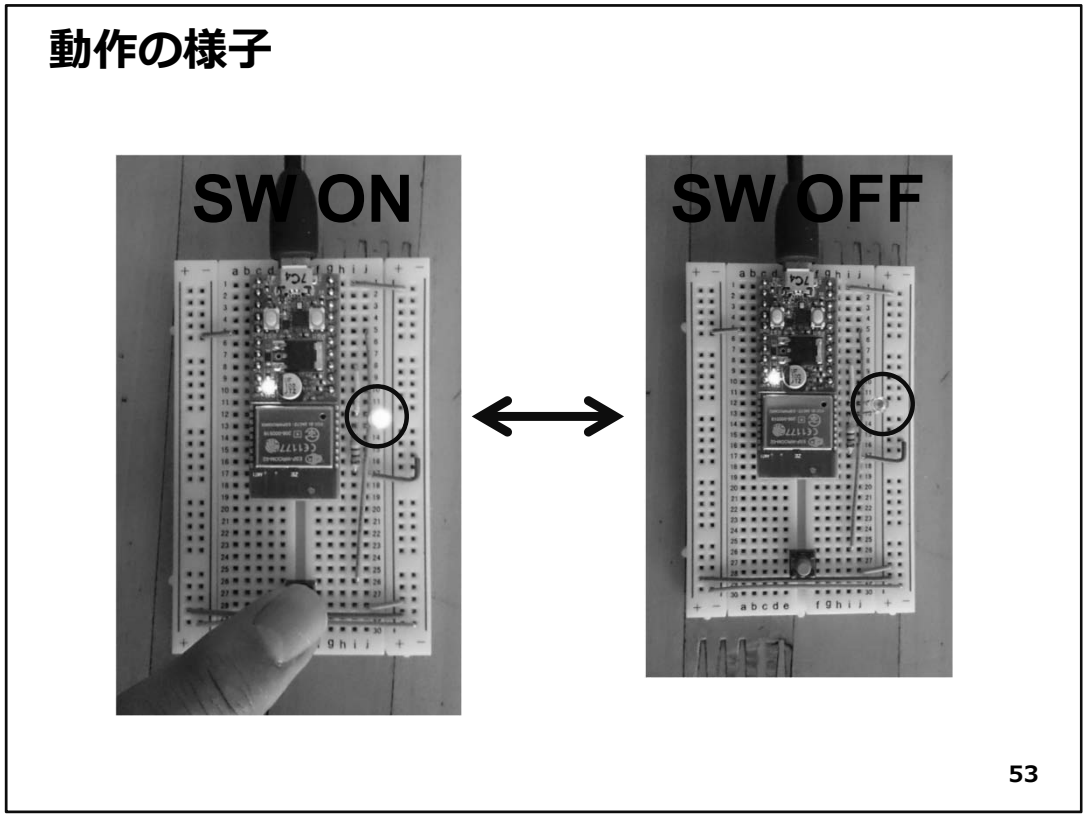
①LEDをGPIO14に接続したので14を定義しています。
②SWはGPIO15に接続しました。
③GPIO14を出力に設定。
④GPIO15を入力に設定。
⑤SW状態を読み込みます。状態はdegitalRead()の戻り値として返されますが、次のdigitalWrite ()でLEDの接続されているGPIO14にそのまま出力します。

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。配線の際にはUSBケーブルを外しているはずなので、改めて接続します。

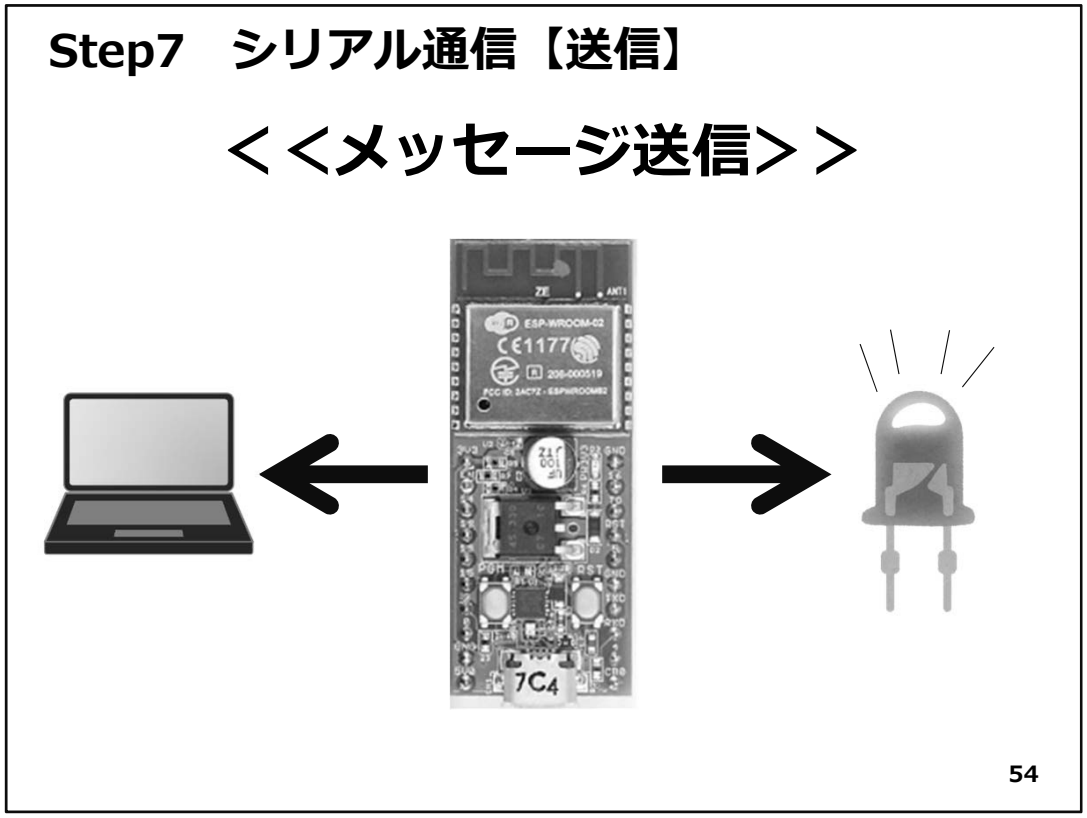
◇**【これからプログラムを書込むゾ！！】**以下の手順を振り返って、ソースコードのコンパイルからマイコンへの書込みまでを行います。

◇書込みが完了すると、マイコンはリセットが掛かり、プログラムの実行が始まっています。



◇動作確認

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。今回の演習の中で解説した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。



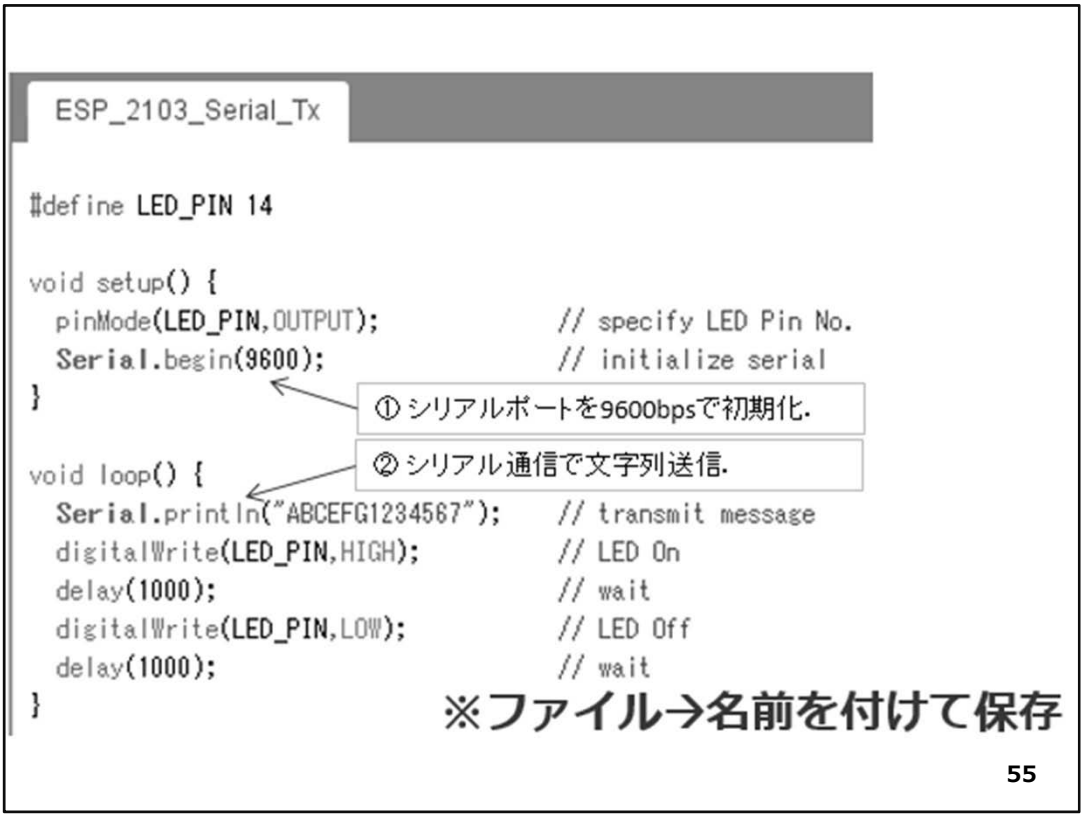
Step7　WiFiマイコン　シリアル通信

**　次は、シリアル通信によるPCへの送信を行い、その後、PCからの受信に応答する仕組みを開発しましょう。　まずは【送信】です。**

◇ 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。

◇回路は、前回作成したものがそのまま利用できます。もし、新たに回路を作成される場合は、Step6を参照して配線を行ってください。また、保管しておいた方は、配線の緩みやはずれがないか、よく確認を行って下さい。

◇従って、配線についての説明は省略します。



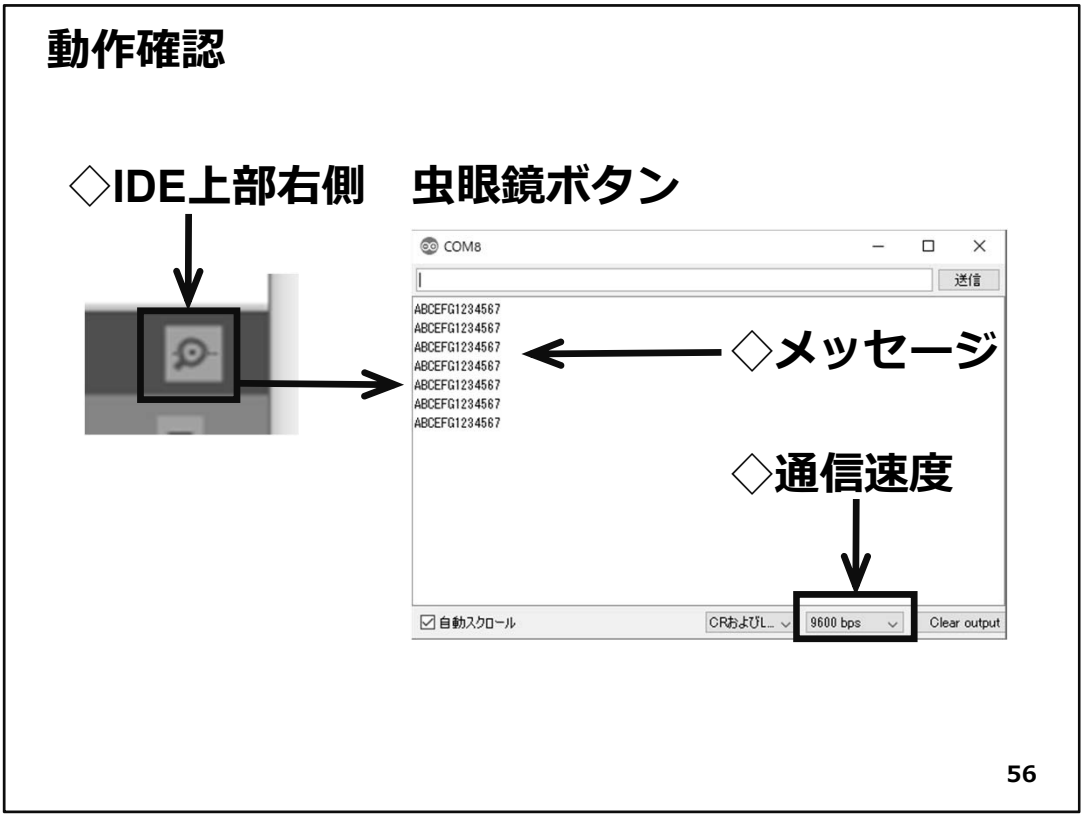
◇IDEに上のソースコードを入力してください。前Stepと比べて新たな部分は次の2点です。

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、Step6**【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。

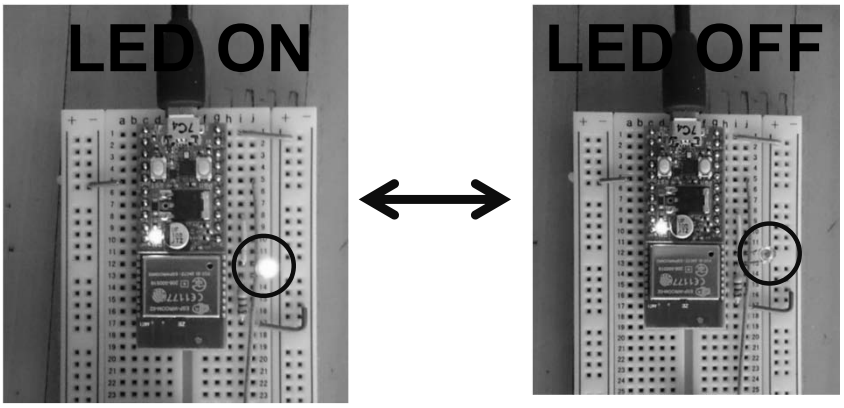


◇IDEウインドウの右上にある**虫眼鏡マーク**のボタンをクリックすると、シリアルモニターのウインドウが開きます。シリアルモニターウインドウの下部にある通信速度のプルダウンで、setup()関数で設定した9600bpsを選択します。通信速度が合っていないと、何も表示されなかったり文字化けを起こしたりしてしまいますので、忘れない様にしてください。

◇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。

LED動作の様子

◇メッセージ表示と同期してLED点滅

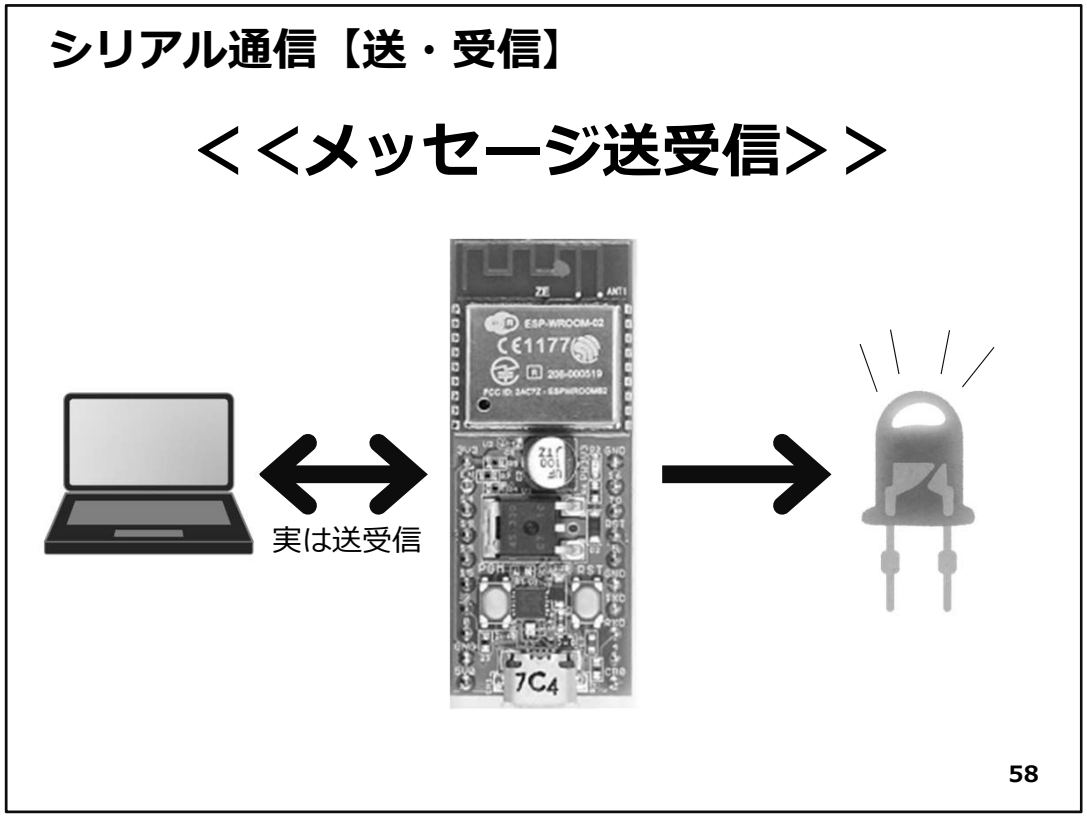


57

◇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、次は受信を行ってみましょう。



◇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。

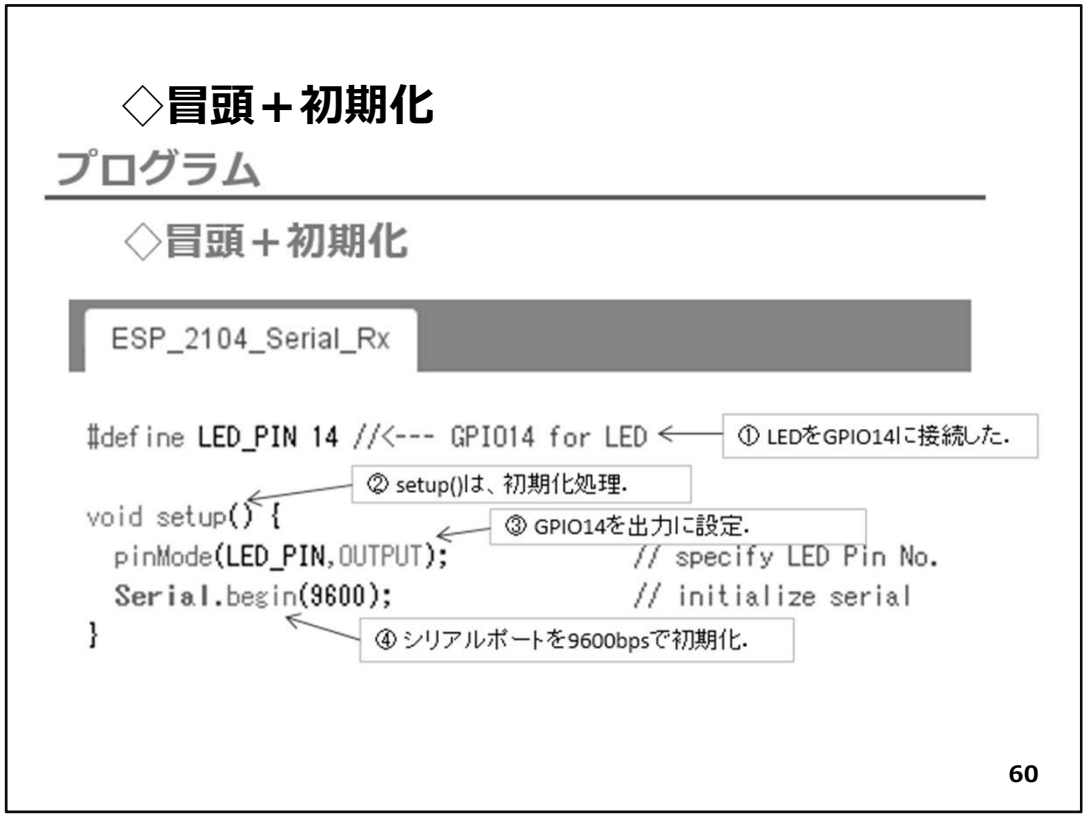
◇上の図は、前回と同じように見えますが、PCとWiFiマイコンが双方向の矢印で結ばれています。今回のテーマは、シリアル通信の【受信】ですが、せっかく送信もできるようになっているので、**欲張って双方向の通信をしてみましょう**。PCから送信するメッセージの電文内容によって、WiFiマイコンの外部に配線したLEDを制御してみます。またその際のWiFiマイコンの振舞いをPCに送信します。

電文設計：通信電文と動作

◇先頭に‘０’→LED消灯＋メッセージ
◇先頭に‘１’→LED点灯＋メッセージ
◇上記以外→LED消灯＋メッセージ
◇電文終端は ‘¥n’（改行）
◇改行コードを受信したとき、
　電文の解析を行い、対応する動作をする

59

◇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。上の図を見てください。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。



◇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

```
void loop() {
  int i=0;
  char c;
  char buf[10];

  while (1) {                          // 受信処理
    if(Serial.available()){            // 受信データあり？
      c = Serial.read();               // 1文字 Read
      buf[i++] = c;                    // 受信した文字を格納
      if(c == '¥n'){                   // 改行ならば電文の終端
        ここに、電文の解析処理を書く！！
      }
    }
  }
}
```

① i はメッセージ用バッファ配列のインデックス.
② c はシリアル受信用バッファ.
③ buf はメッセージ用バッファ.

④ 受信処理.

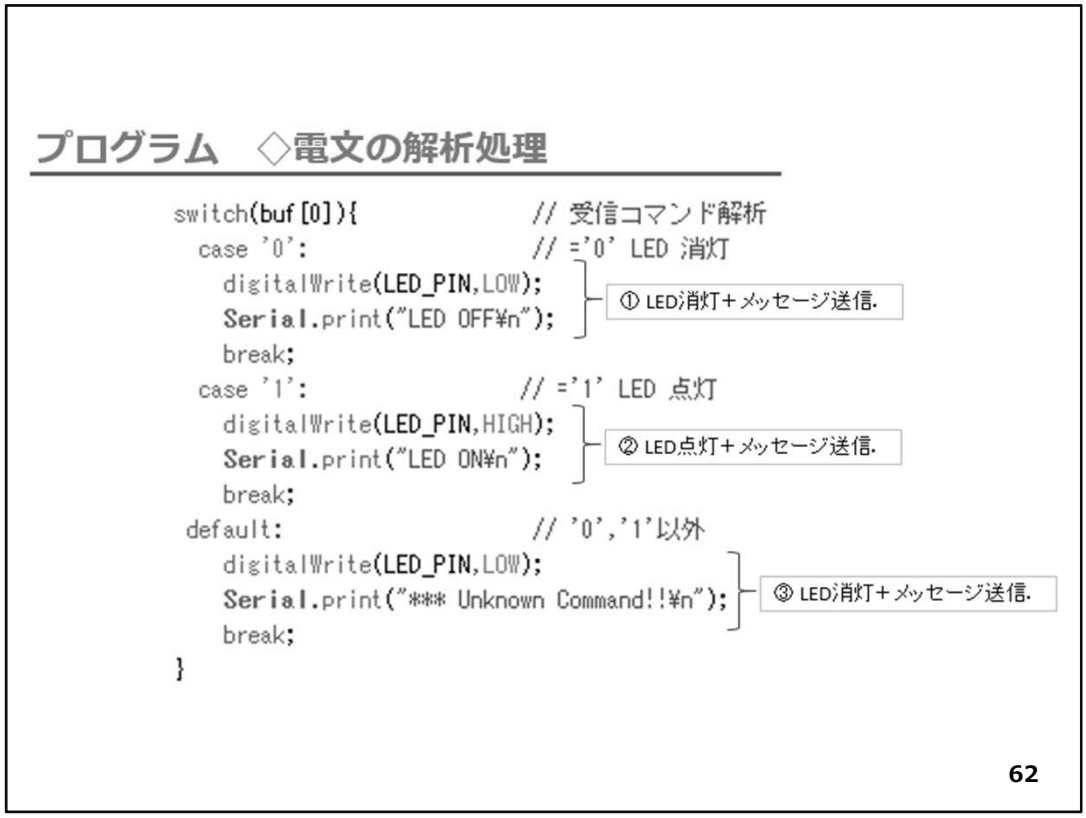
⑤ 解析処理は次で解説.

{ } の対応に注意してください！！

61

◇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。



◇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。

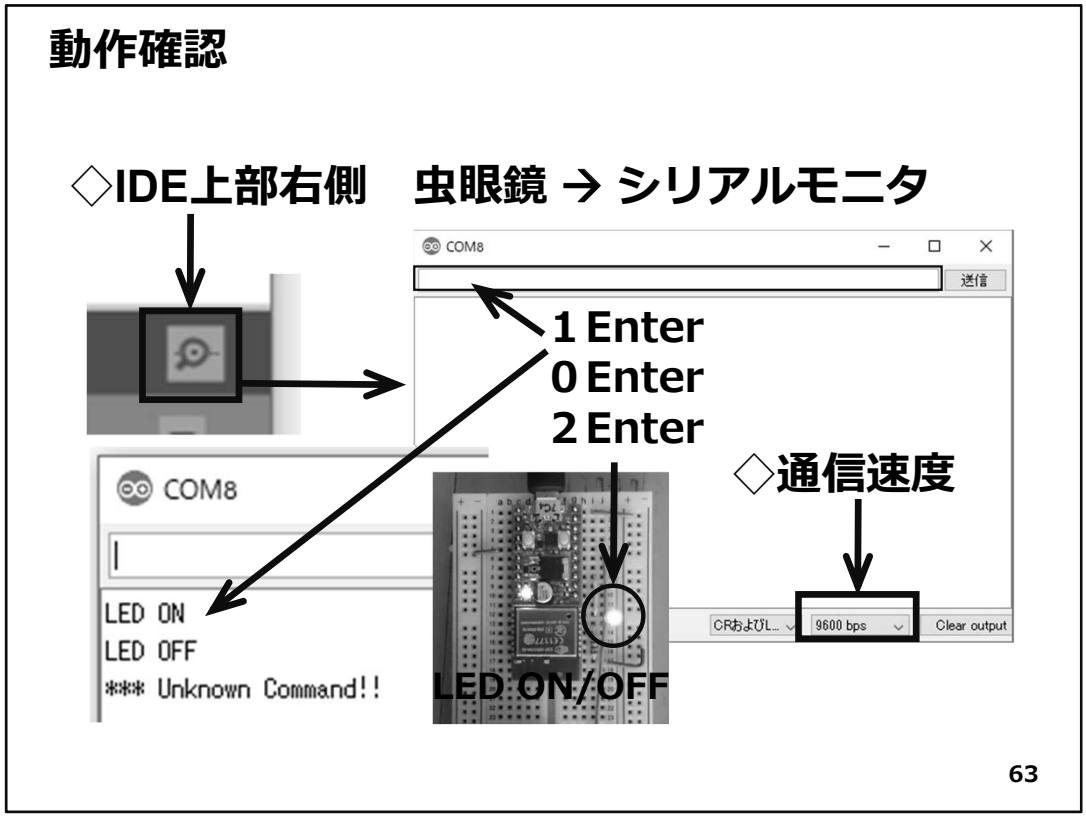
◇【ソースコードは、演習キット付属のCDに収録しました。】

◇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

◇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。

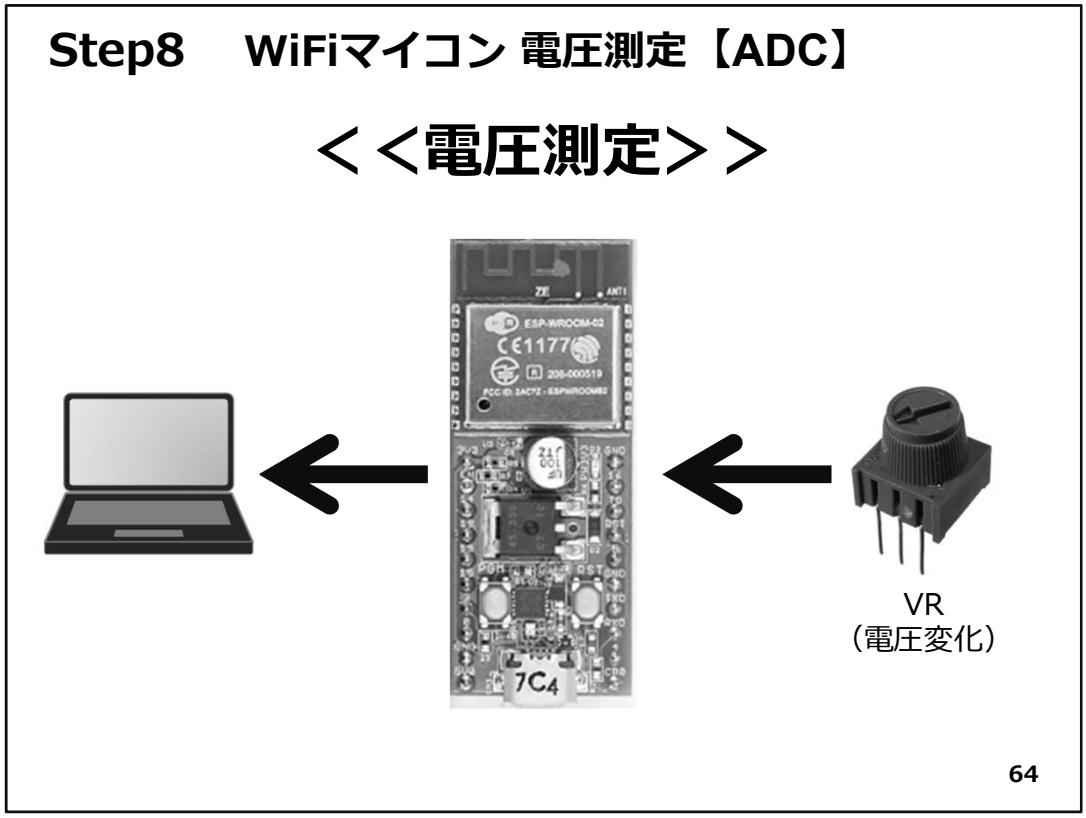
◇次に送信データに’0’を入力してEnterキーを押下するか、送信ボタンをクリックして下さい。すると、LEDが消灯してLED OFFのメッセージがシリアルモニターに表示されます。

◇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。

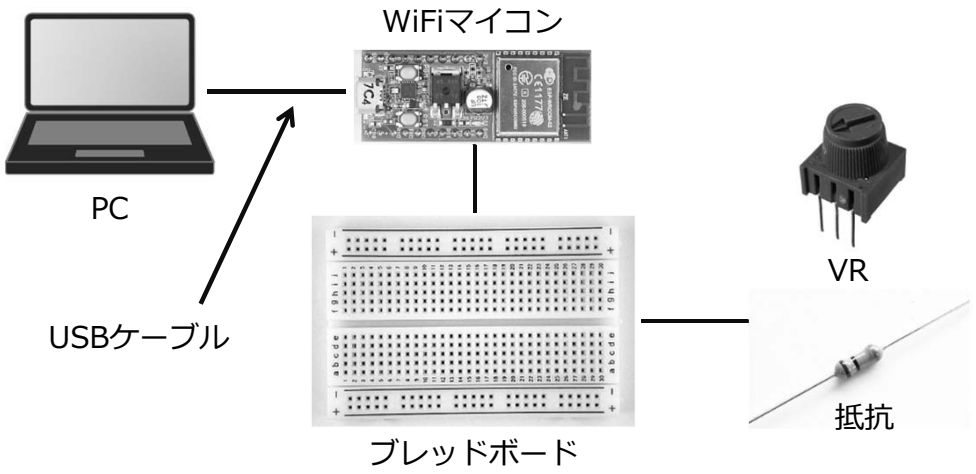


Step8　WiFiマイコン　電圧測定

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。 Step8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。

システム概要

◇全体構成とパーツ



65

◇ VRを使った電圧変化を測定してPCに送信します。Step7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。

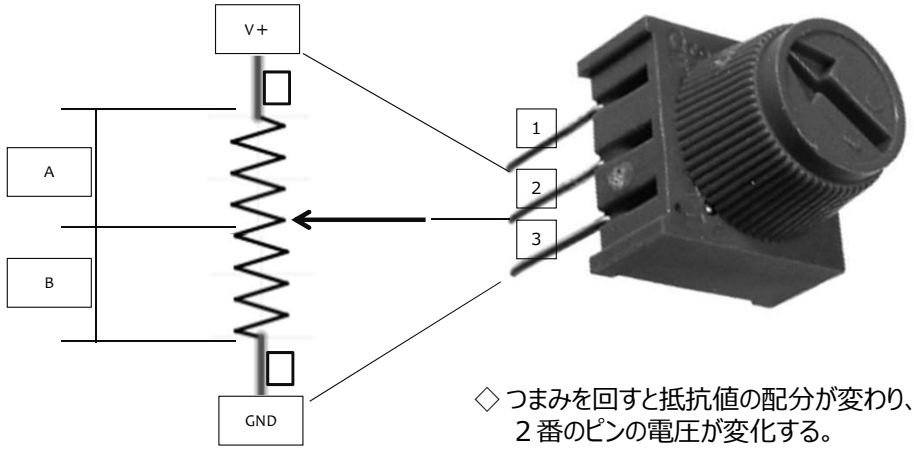
◇システムの全体構成を上図に示します。必要な機材・パーツは、下記です。

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。

VR（可変抵抗器）

電圧を分ける → 分圧



◇ つまみを回すと抵抗値の配分が変わり、
２番のピンの電圧が変化する。
◇ この変化する電圧をマイコンで測る。

66

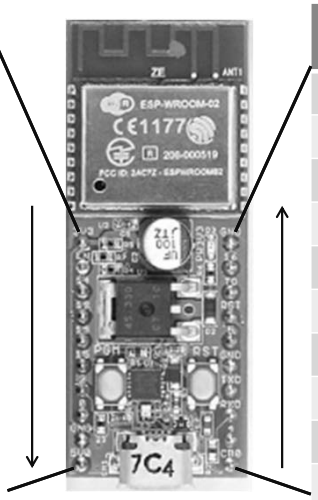
◇上図は、今回使用するVRの写真と内部構造を表しています。VRにはつまみ（矢印が刻印されていて周囲がギザギザになっている部分）が付いています。脚ピンが3本あり、便宜的に写真上から1,2,3番とします。1番と2番ピンの間は、50KΩの抵抗値となっています。つまみは内部の接点（図左側の矢印部分）と連動しています。つまみを右左に回すと、図の矢印部分が上下に移動する仕組みです。ここで、1番ピンをV+（電源の+側）に、3番ピンをGNDに接続して、つまみをいっぱいに回して、図の矢印がV+に一番近くなった時、2番ピンと3番ピンの間には、V+の電圧がかかっていることになります。反対に矢印部分が一番下がったときは、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**と言います。

◇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。

WiFiマイコン　ピン配置

ESP-WROOM-02

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

67

◇Step6で説明した、WiFiマイコンモジュールのピン配置です（再掲）

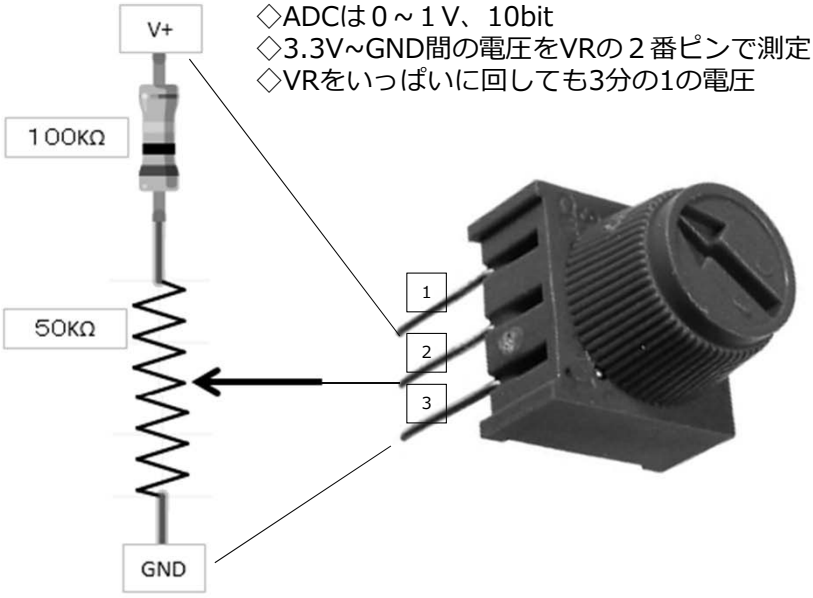
◇ WiFiマイコンモジュールのアンテナパターンを上に向けて、左上のピンから
1,2,3,4・・・と番号が付いています。
----------<< 各ピンに配置されている信号 >>----------
1番：3.3Vの電源で駆動するときは、このピンに直接3.3Vを接続します。USB経由または、外部5Vで駆動する場合は、このピンから3.3Vの電源を取り出すことができます。
2番：イネーブル端子で、このモジュール全体の有効・無効を設定します。通常は解放で有効です。無効（Disable）とするときは、GNDに接続します。
3～8番：汎用入出力端子です。デジタル入出力を取り扱います。
12,16,19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。

◇ピン配置図の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。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。

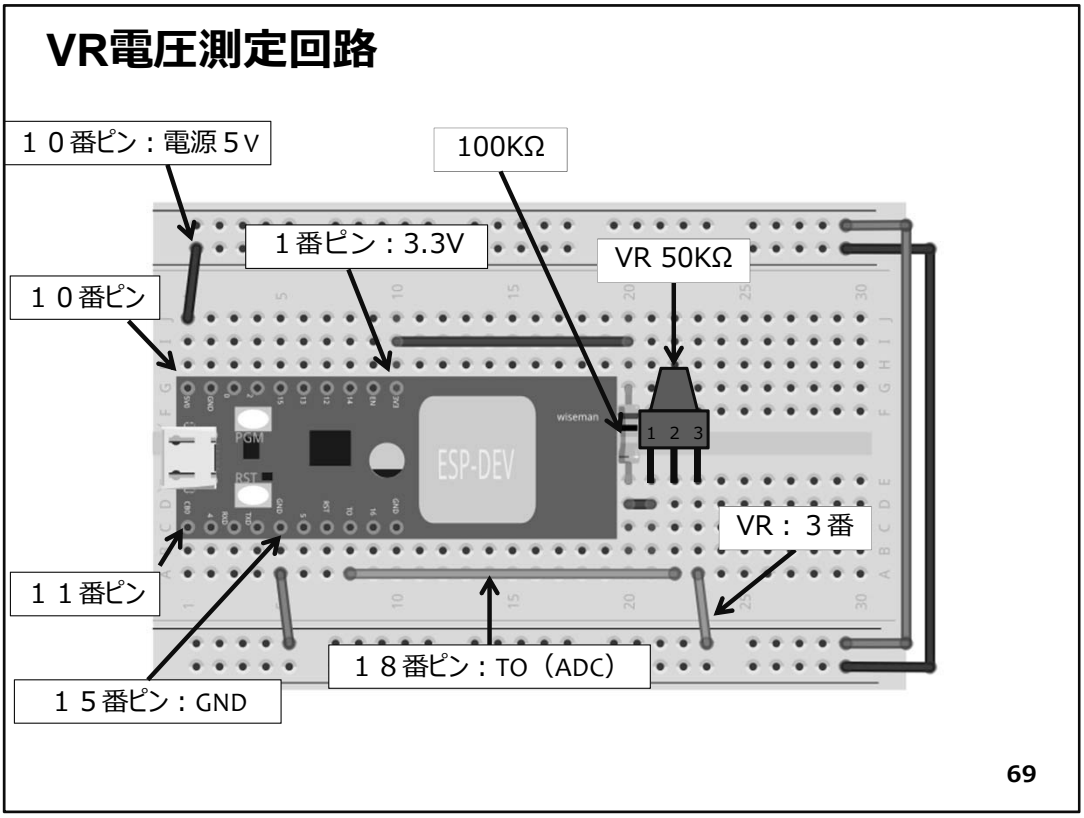
ADCへの対応



◇ADCは０～１V、10bit
◇3.3V~GND間の電圧をVRの２番ピンで測定
◇VRをいっぱいに回しても3分の1の電圧

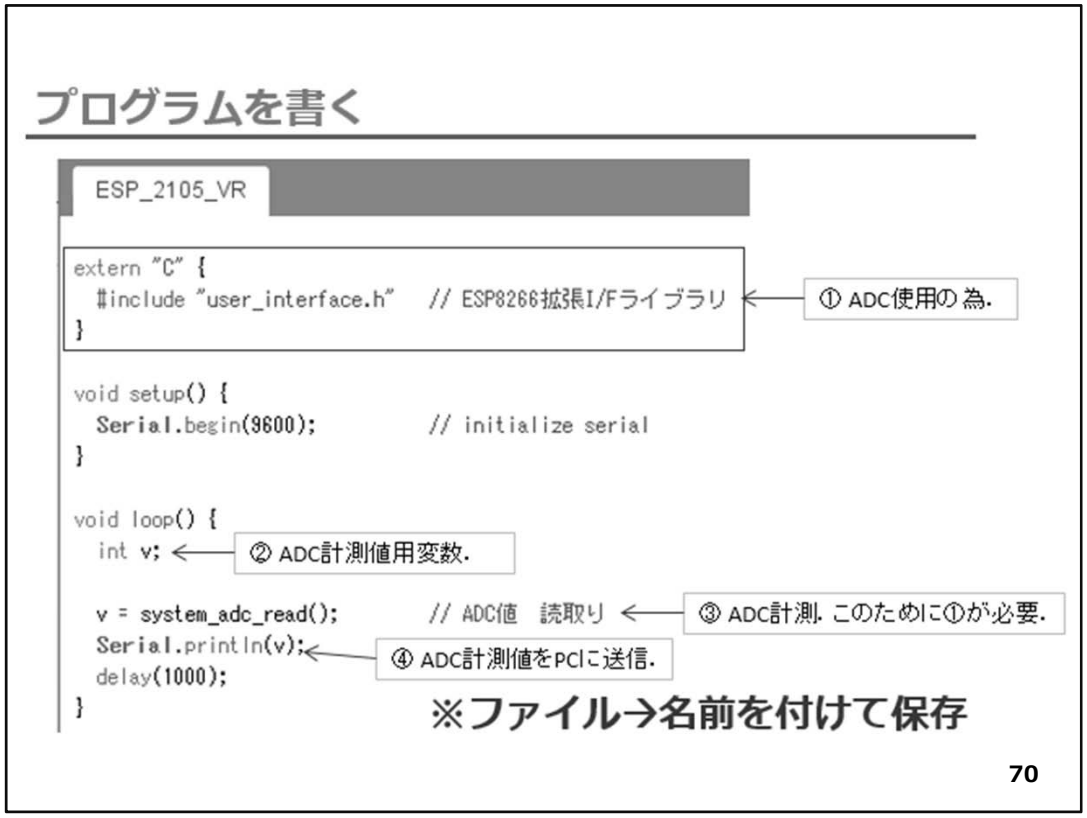
68

◇今回は、上の図に示すように、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。



◇図に従い、配線をします。配線を行うときは、USBケーブルを外してください。マイコンの回路もショートすると壊れてしまうことがありますが、PC側も壊れてしまいます。ご用心！ご用心！！

◇もう皆さんは、配線にもすっかり慣れたことでしょう。しかし、その頃が一番失敗し易い時期でもあります。ここは、ぐっと、気持ちを落ち着けて、しっかり配線を確認してください。



◇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）

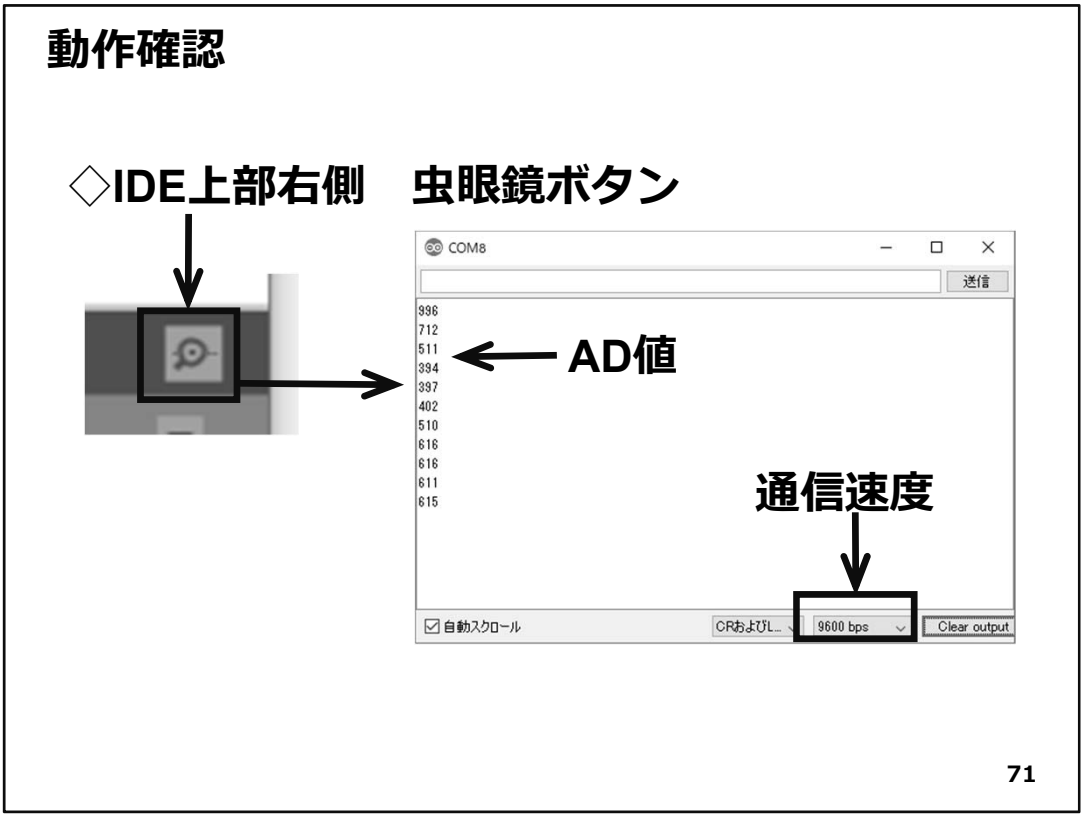
◇このプログラムの動作としては、loop()の最後にあるように、1秒間隔で電圧計測を行い、その値をPCにシリアル通信で送信するという動作を繰り返します。

◇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。

◇ USBケーブルを接続して、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

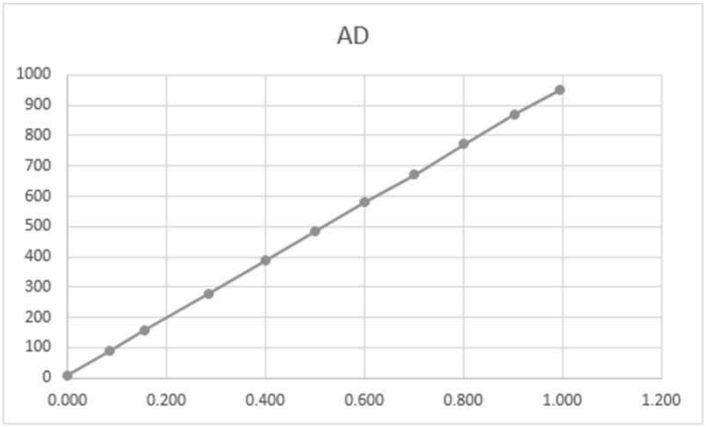
動作確認をしましょう。IDEの虫眼鏡マークのボタンをクリックしてシリアルモニターを起動します。**通信速度**をSerial.begin()で設定した速度に合わせます。すると、約1秒ごとに数値が表示され続けます。これがVRによって分圧された電圧のA/D変換値です。VRを回転してみると、表示される値が変化することが分かります。

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。

動作確認

◇TO端子電圧とAD値の関係（実測）

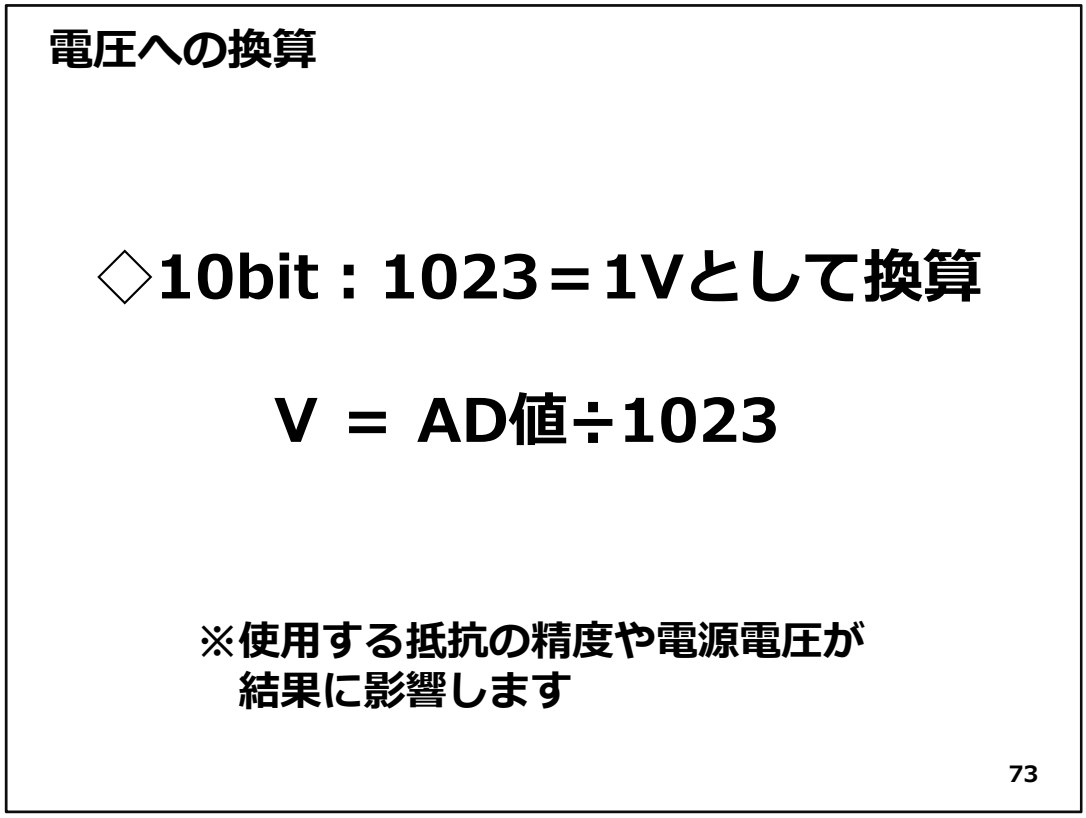
V	AD
0.000	9
0.085	89
0.157	159
0.286	278
0.400	387
0.500	482
0.599	579
0.701	670
0.802	773
0.905	872
0.994	950



72

◇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。



【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。

10bit ＝ 1023 というのは・・・分かりますよね ~ V(^^)V

問題：作成したプログラムを改善して、温度が表示されるシステムにしてください。

Step9 WiFiマイコン デジタル温度センサー

＜＜マイコンと通信するデジタルセンサー＞＞

◇WiFi無線マイコンは、A/D変換器が１つ
→ 一つのアナログセンサーしか使えない

◇デジタルセンサーは、I2C I/Fを持つ
→ 複数のセンサーを使用できる

74

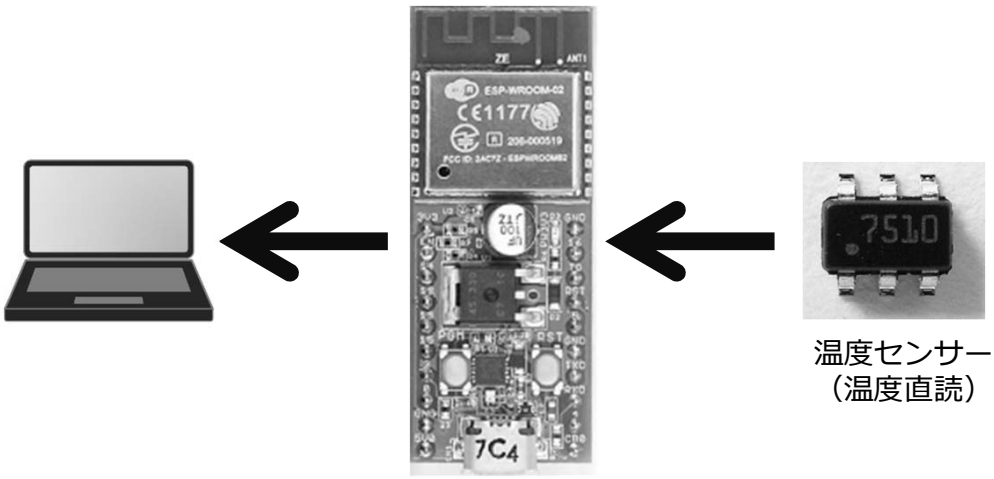
Step9　WiFiマイコン　デジタル温度センサー

無線マイコンで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。

今回はデジタル温度センサーを使用します。このセンサーは、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。

マイコンの王道・・・デジタル温度センサー

＜＜温度直読＞＞

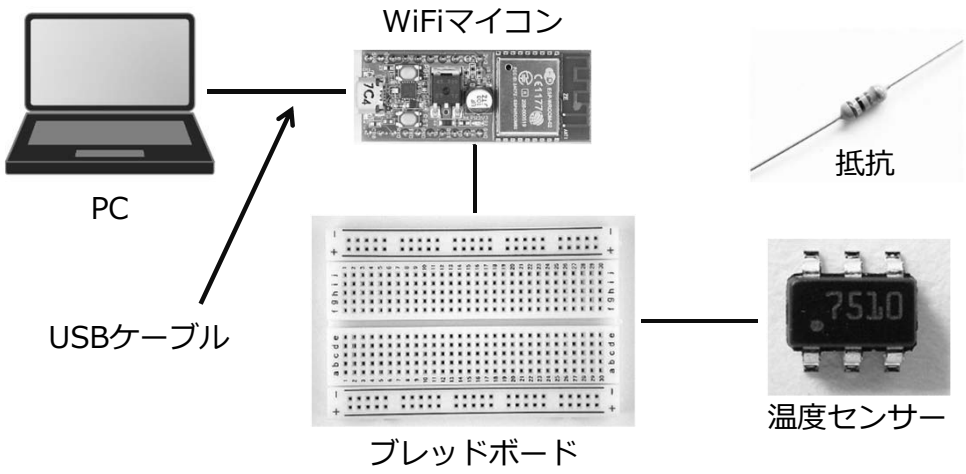


75

◇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。

システム概要

◇全体構成とパーツ



76

◇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

77

◇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

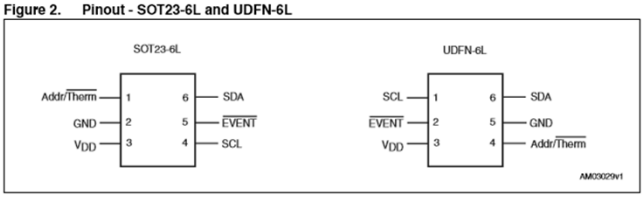


Table 3.　Pin descriptions

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 ⟷ SCL
12Pin ⟷ SDA

78

◇今回使用するセンサーのパッケージは図上左のものです。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。

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

79

◇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).

80

◇ 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。

デジタル温度センサー　STTS751

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	$\overline{\text{RUN}}$/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: $\overline{\text{EVENT}}$ is enabled. Any out-of-limit condition asserts the EVENT pin (active low).

1: $\overline{\text{EVENT}}$ is disabled.

$\overline{\text{RUN}}$/STOP: [bit 6]

0: Device is running in continuous conversion mode.

1: Device is in standby mode drawing minimum power.

The $\overline{\text{RUN}}$/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 $\overline{\text{RUN}}$/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.

81

◇ 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、4つのプログラムで設定可能な分解能の内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。

デジタル温度センサー　STTS751

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

82

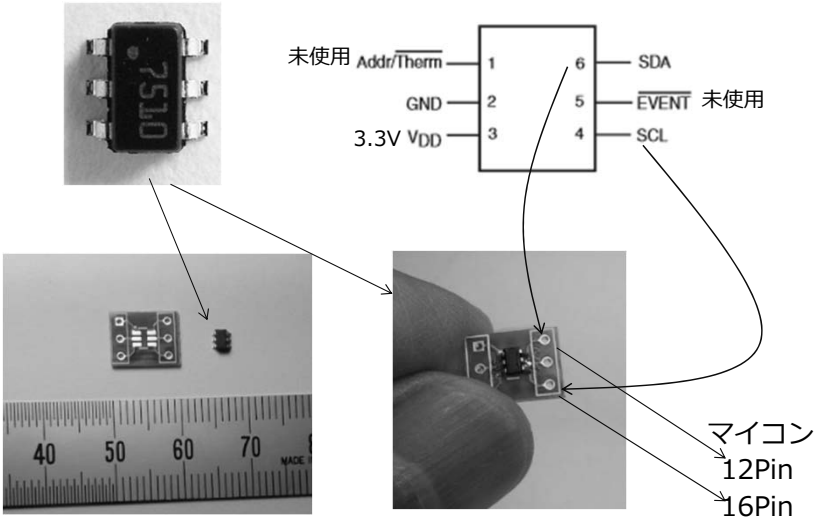
◇ Tres1:0を11にセットすると、図の下に示すように12bitの分解能となります。（Tresは、Temperature Resolutionの短縮形らしい）
これらにより温度処理は次の様になります。

★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、1LSB＝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）

デジタル温度センサー　STTS751

◇ I2Cアドレス　→ 0x39【スレーブアドレス】



83

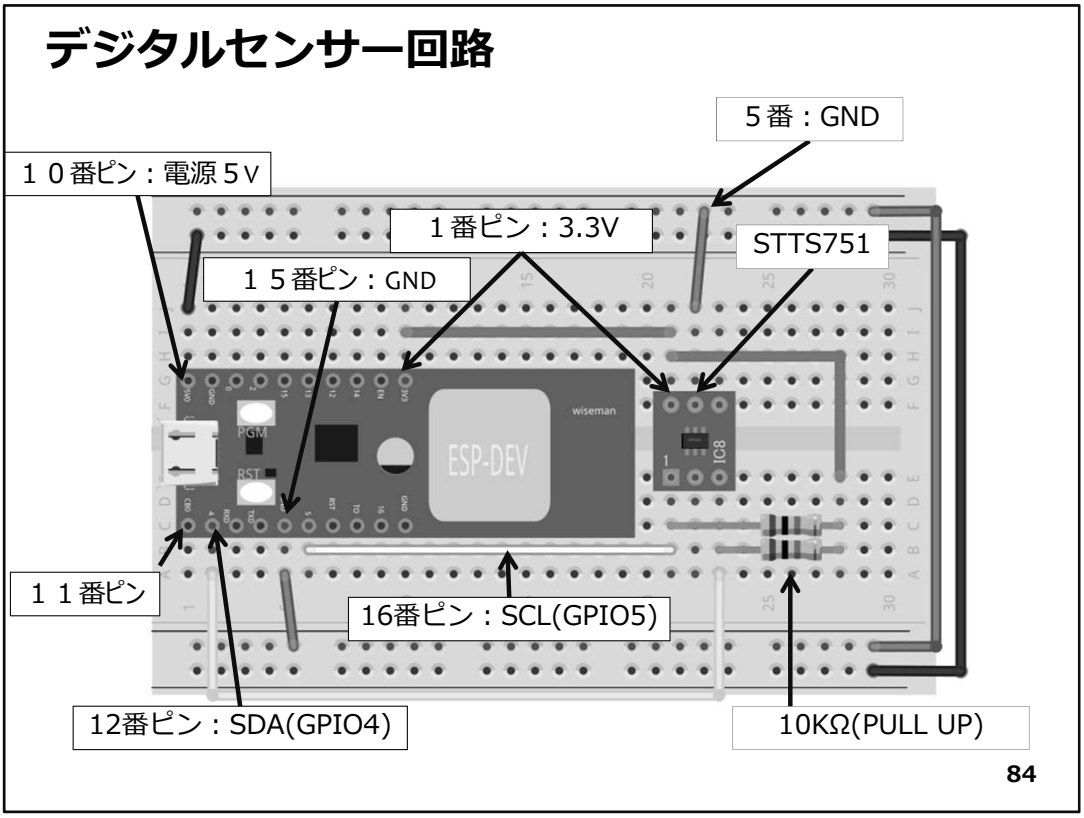
【重要】

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。

◇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】

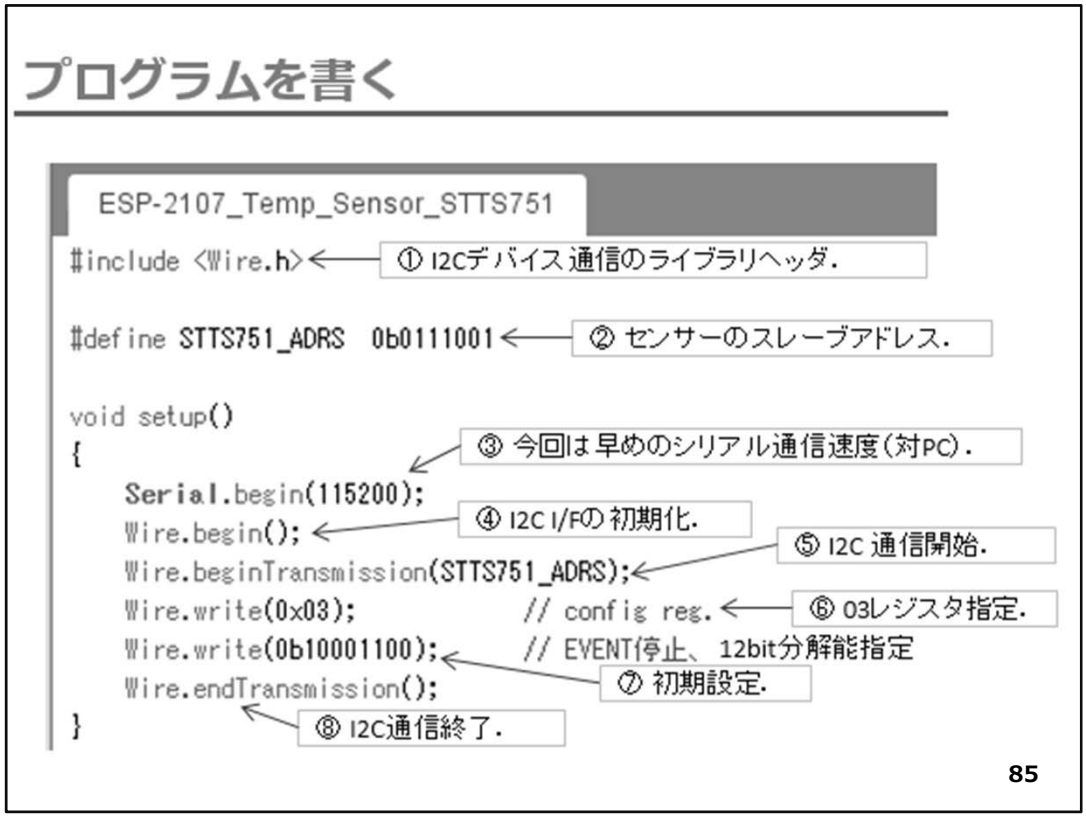
次に示す配線図では、I2C I/FのSCLとSDAの信号は、10KΩの抵抗で3.3VにPull Upする部分があります。見落とさないでください。Pull Upというのは、第2回で解説したPull Downの反対です。今回は常時Highレベルにしておきたい信号に使用しています。



◇ 上図を見て回路を作成してください。**USBケーブルを外すことをお忘れなく！！**

【重要】

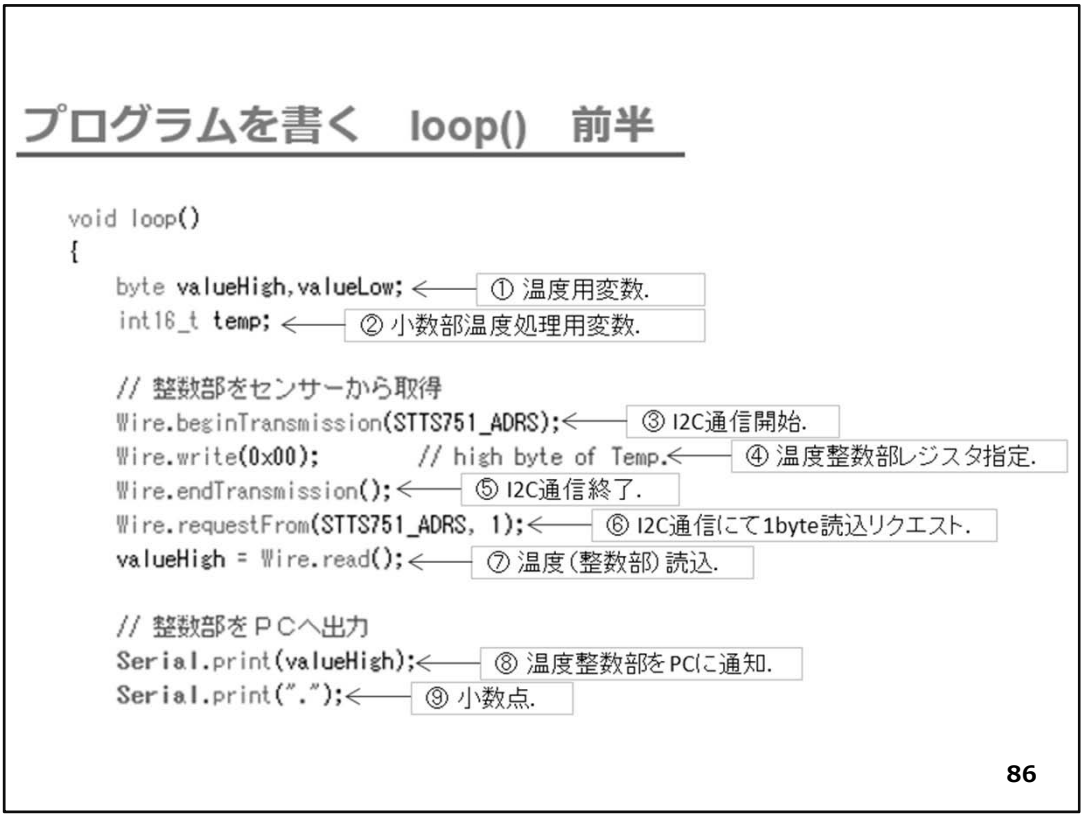
この回路は次回以後も継続して使用しますので、実験が終わっても可能であれば保存してください。



◇IDEにプログラムを入力して下さい。

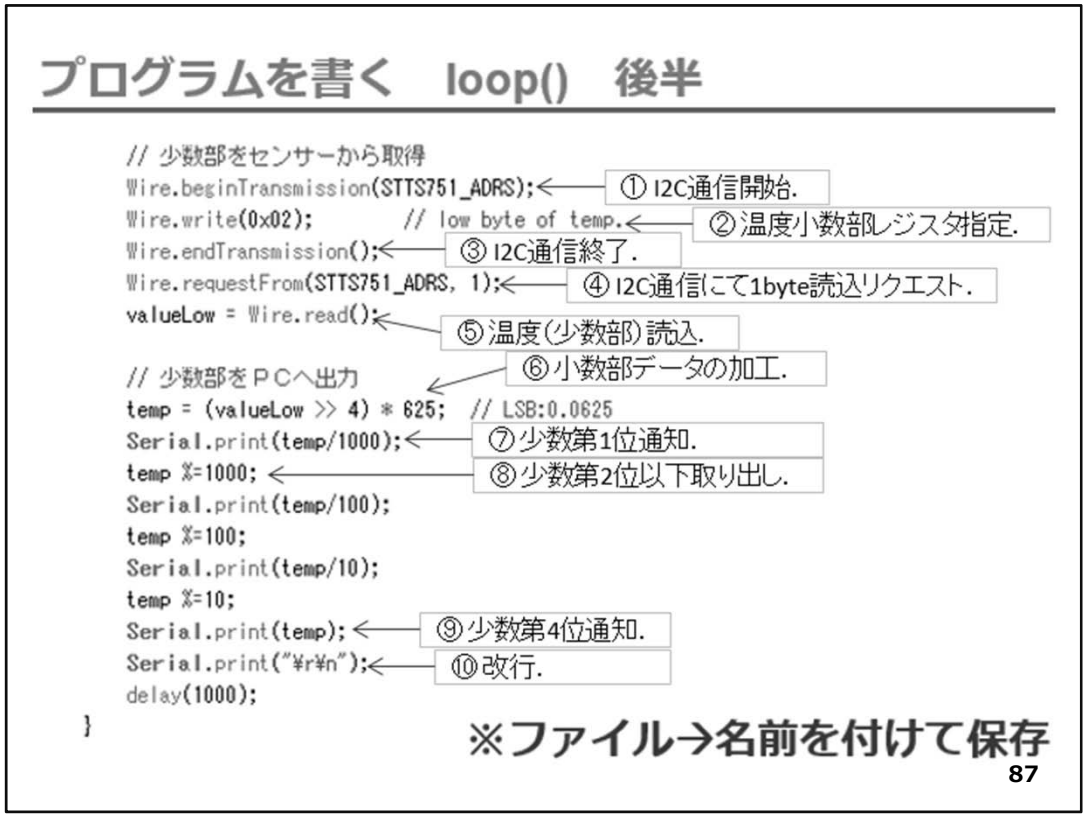
①I2Cでバイスとの通信に必要なライブラリを利用するためのヘッダ
②使用するセンサーのI2Cスレーブアドレス（0x39）
③シリアル通信初期化(速度115200bps)
④I2C I/F初期化
⑤センサー初期化のためI2C通信開始
⑥03レジスタ設定
⑦初期設定
⑧I2C通信終了

【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。



◇ 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）



◇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余りを取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

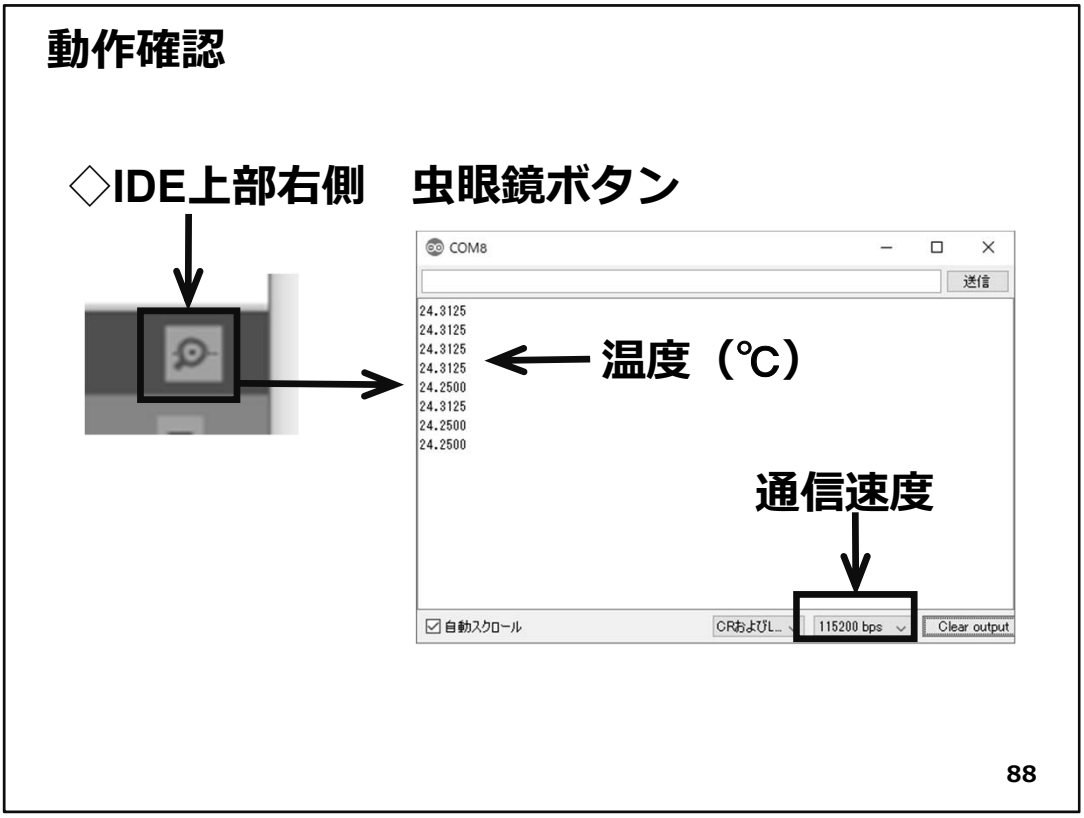
◇ソースコードを入力したら、名前を付けて保存してください。

◇USBケーブルを接続して、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。



◇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Step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。

このStepで作成した回路は、次回以後も継続して使用しますので、実験が終わっても可能であれば保存してください。

Step10　WiFiマイコン 液晶表示器（LCD）

＜＜マイコンと通信するLCD＞＞

◇WiFiマイコンのピン数は限られている
→ LCDは制御信号が多い
→ デバイスの並行接続も限られる

◇WiFiマイコンは I2C I/F がある
→ 複数のデバイスを使用できる
→ ピン数の制限も緩和される

◇デジタルセンサーと同じ I/F に接続！！

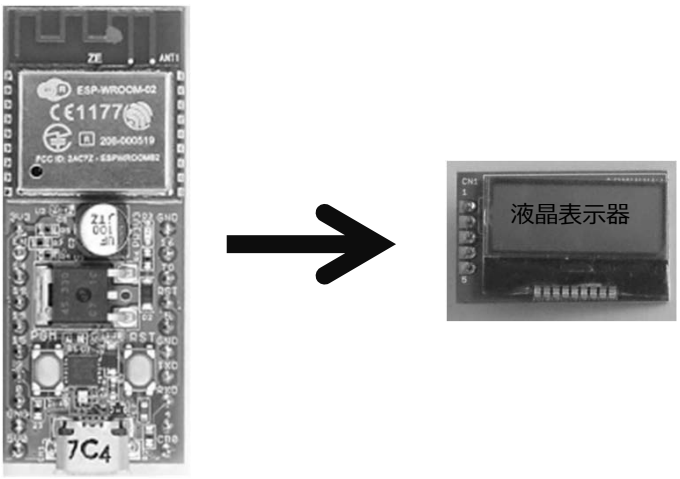
89

Step10 WiFiマイコン 液晶表示器(LCD)

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。

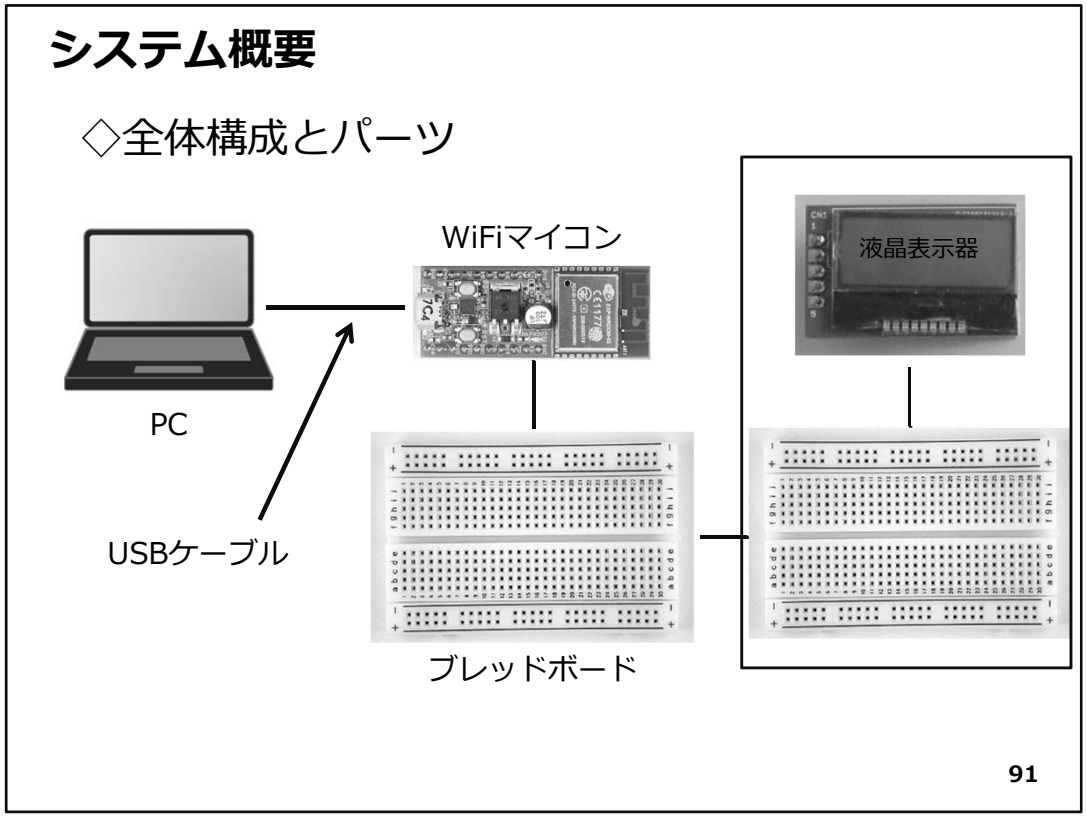
マイコンの王道・・・表示機能【LCD】

＜＜ LCD ＞＞



90

◇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、表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。



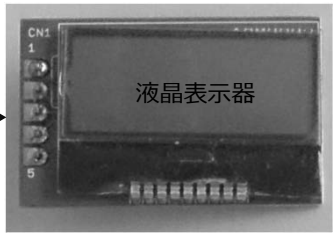
◇システムの全体構成を上図に示します。前回使用したデジタル温度センサーの回路を残したまま、液晶表示器を取り扱います。ブレッドボードが混雑するので、**液晶表示器は別のブレッドボードに液晶表示器ユニットとして作成することにしました。**
必要な機材・パーツは、下記です。

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）

液晶表示器　LCD

◇ ８文字×２行
◇ I2C I/F（２本の信号で通信を行うI/F）
◇ I2Cアドレス　→ 0x3E
◇ 制御コマンド　→ 0x00＋Command
◇ 文字データ　　→ 0x40＋Data

上から順に
1．Vdd（3.3V）
2．RESET（NC）
3．SCL（クロック：WiFi No.16）
4．SDA（データ　：WiFi No.12）
5．GND

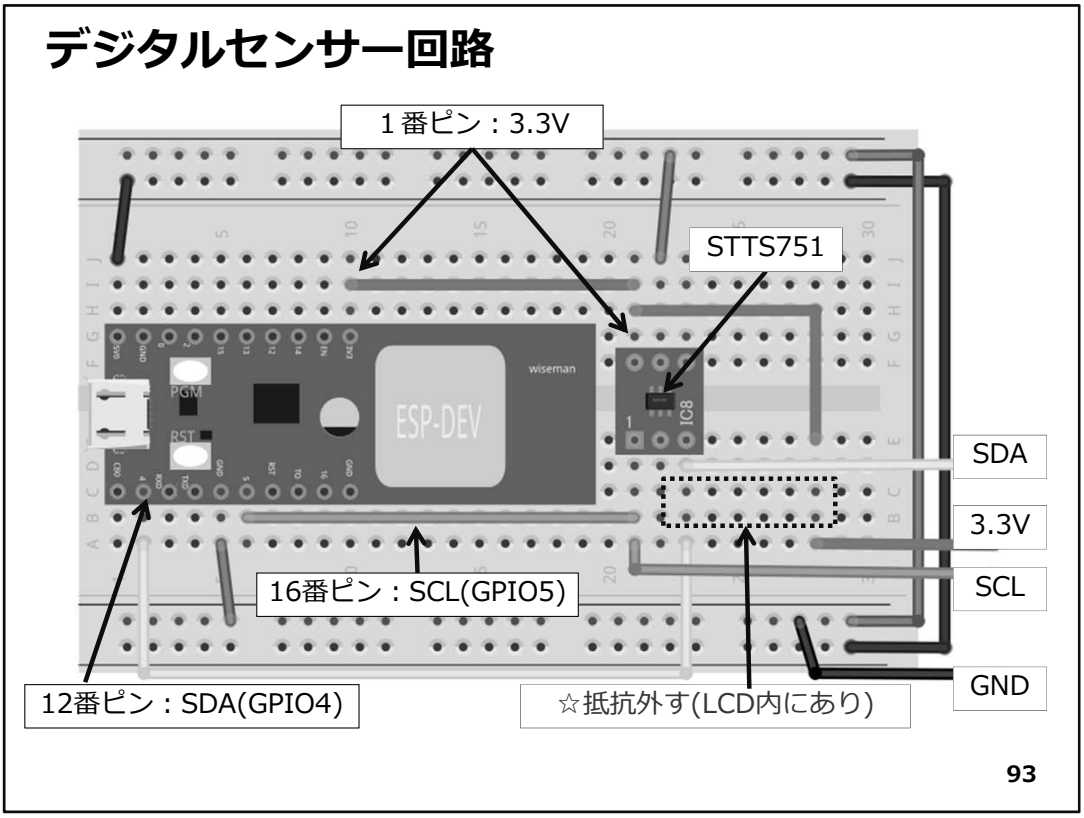


AQM0802A

92

◇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。



◇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。

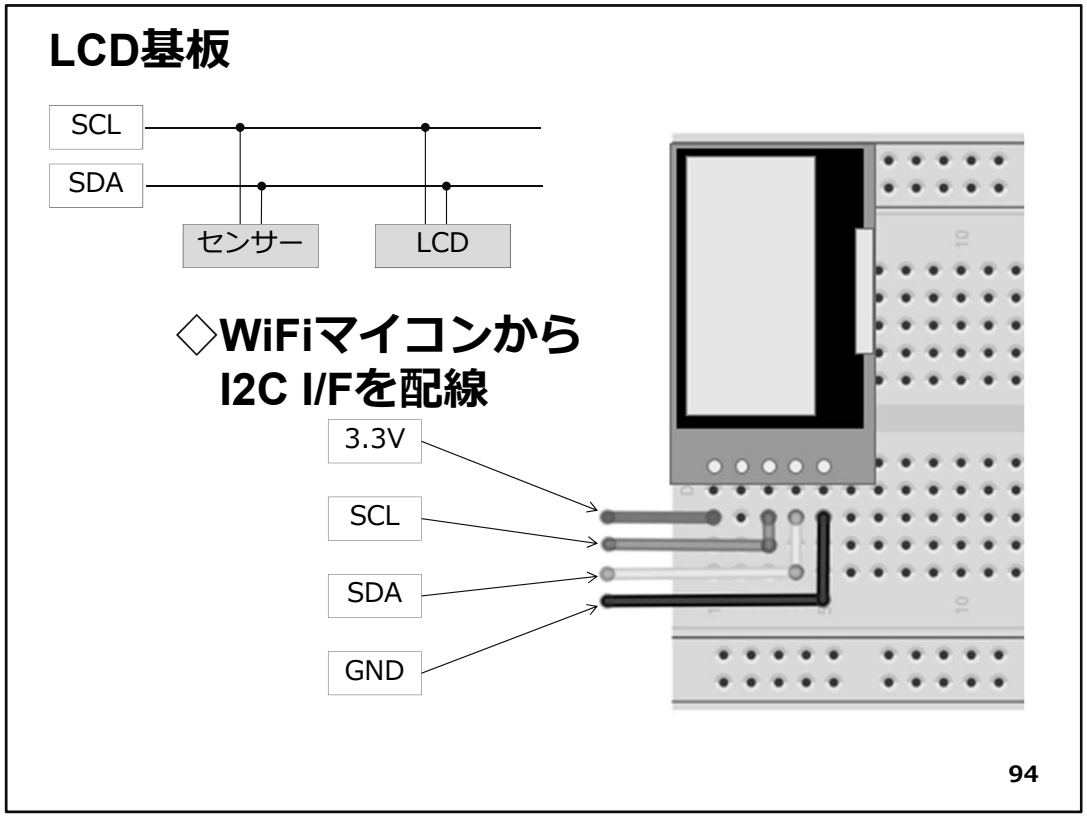
◇ WiFiマイコン側は、前回のデジタル温度センサー回路を流用します。少し変更する点があります。**【10KΩのPull Up抵抗は外します。】**

◇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。

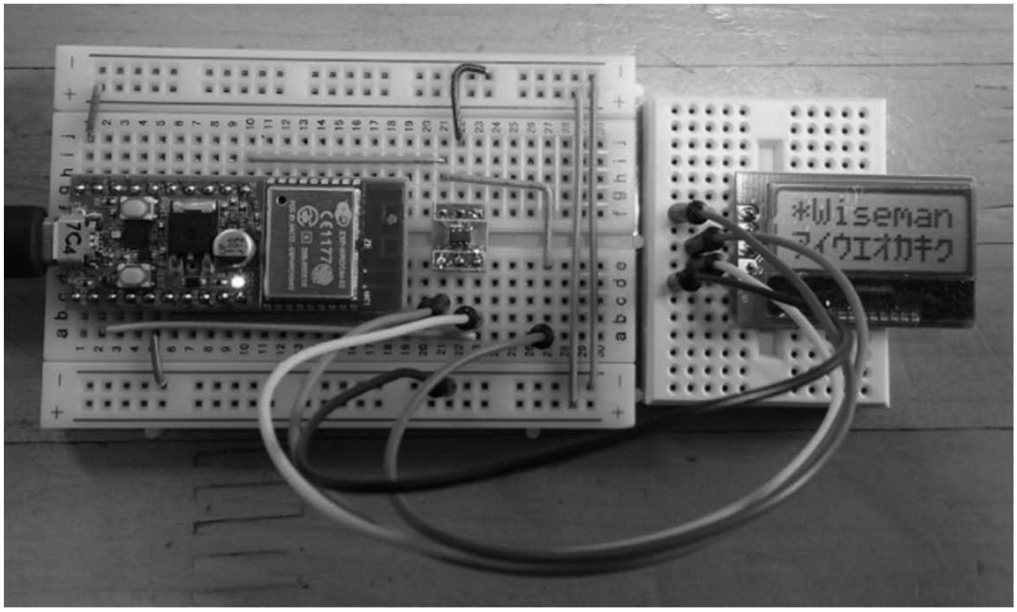


◇LCDユニットは上図の様に配線してください。今回の回路では、デジタルセンサーも併設しているので、図左上に描いたようにSCL,SDAに温度センサーとLCDがぶら下がっている形となります。

【重要】

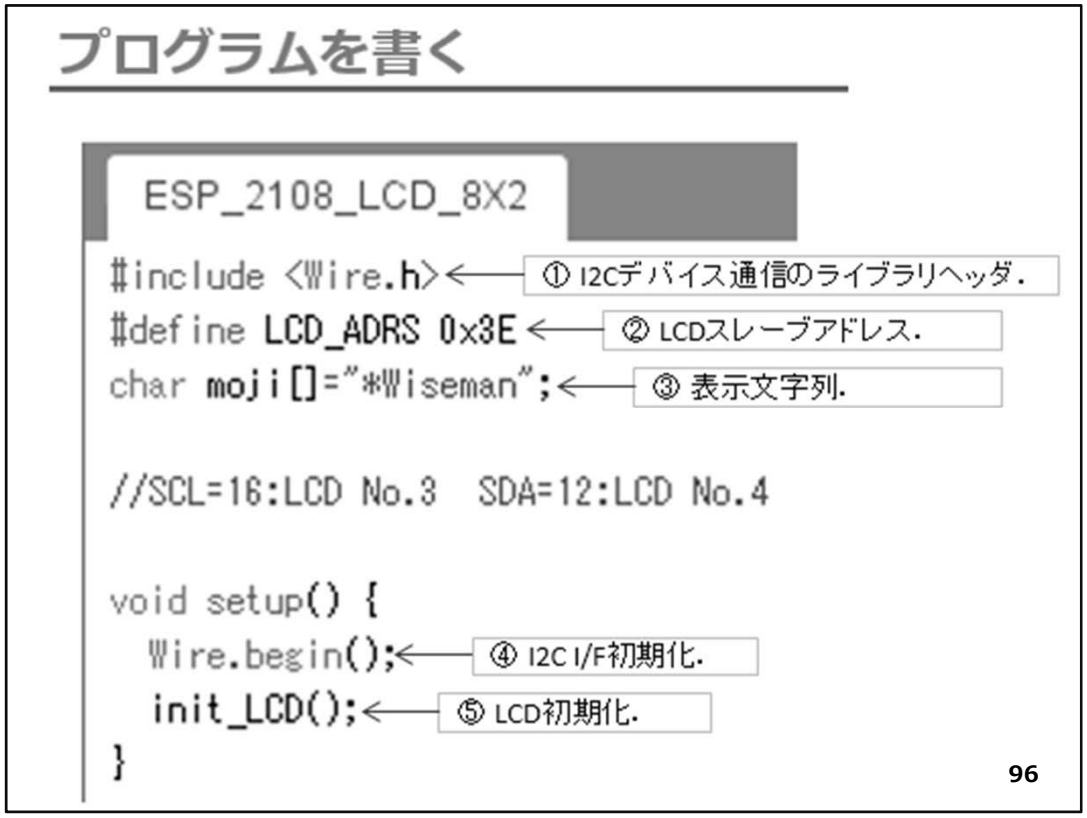
このStepで作成した回路は、次回以後も継続して使用しますので、実験が終わっても、可能であれば保存してください。

実際に配線した様子

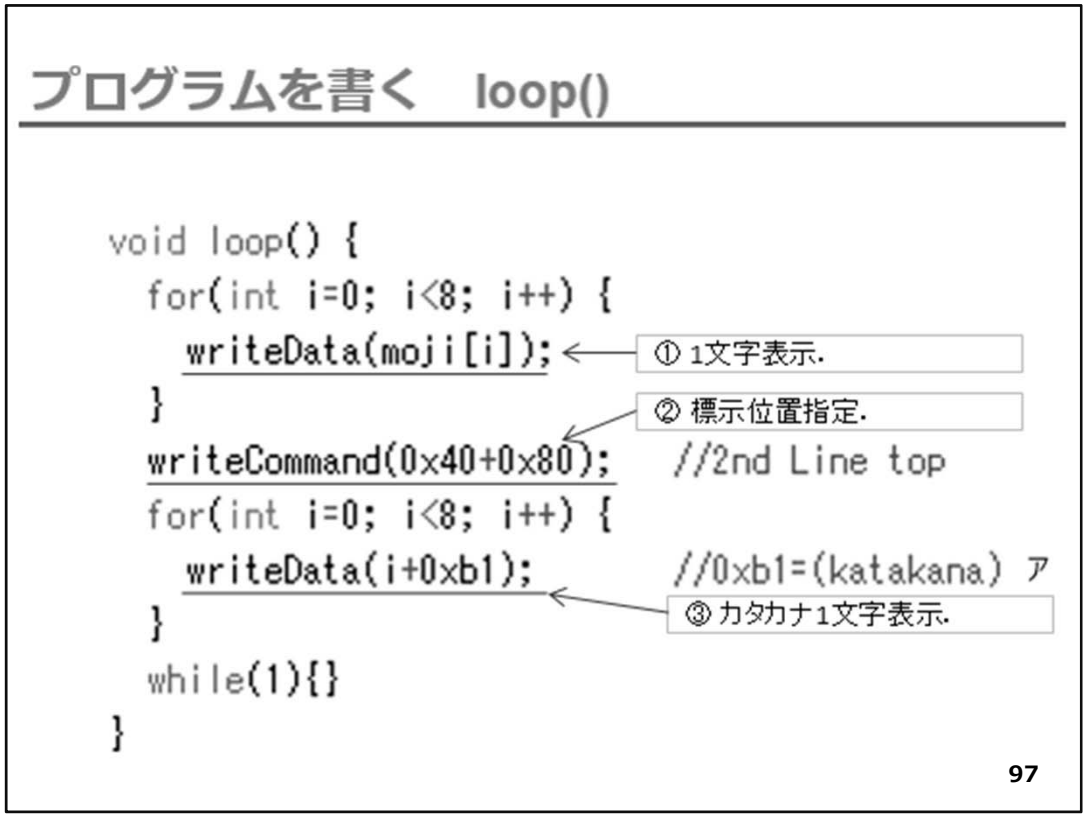


95

◇2つのユニットに分けて作成した様子です。写真では、LCDは小さいブレッドボードで配線していますが、演習キットには、同じサイズのブレッドボードが複数枚セットされています。

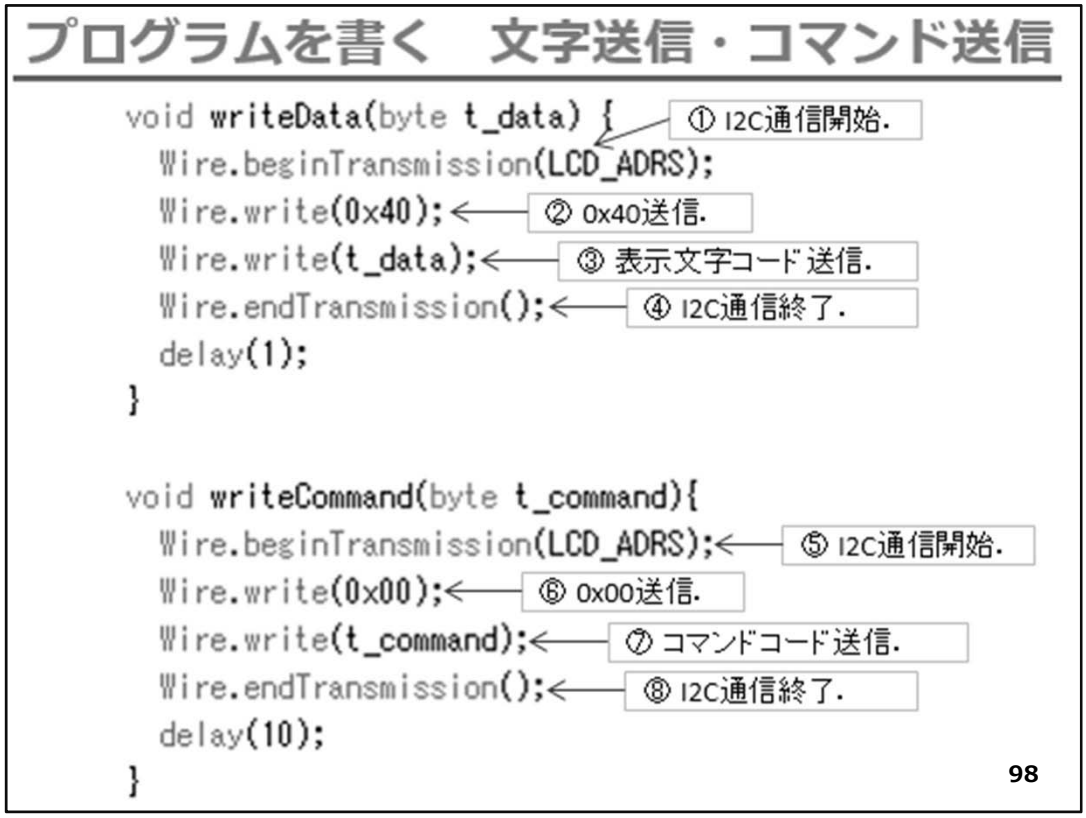


①I2C通信ライブラリ用ヘッダ
②LCDのI2Cスレーブアドレス
③LCDに表示する固定文字列バッファ（8文字）
④I2C I/F初期化
⑤LCD初期化。関数の中身は後で説明



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。



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);
}
```

※ファイル→名前を付けて保存

99

◇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

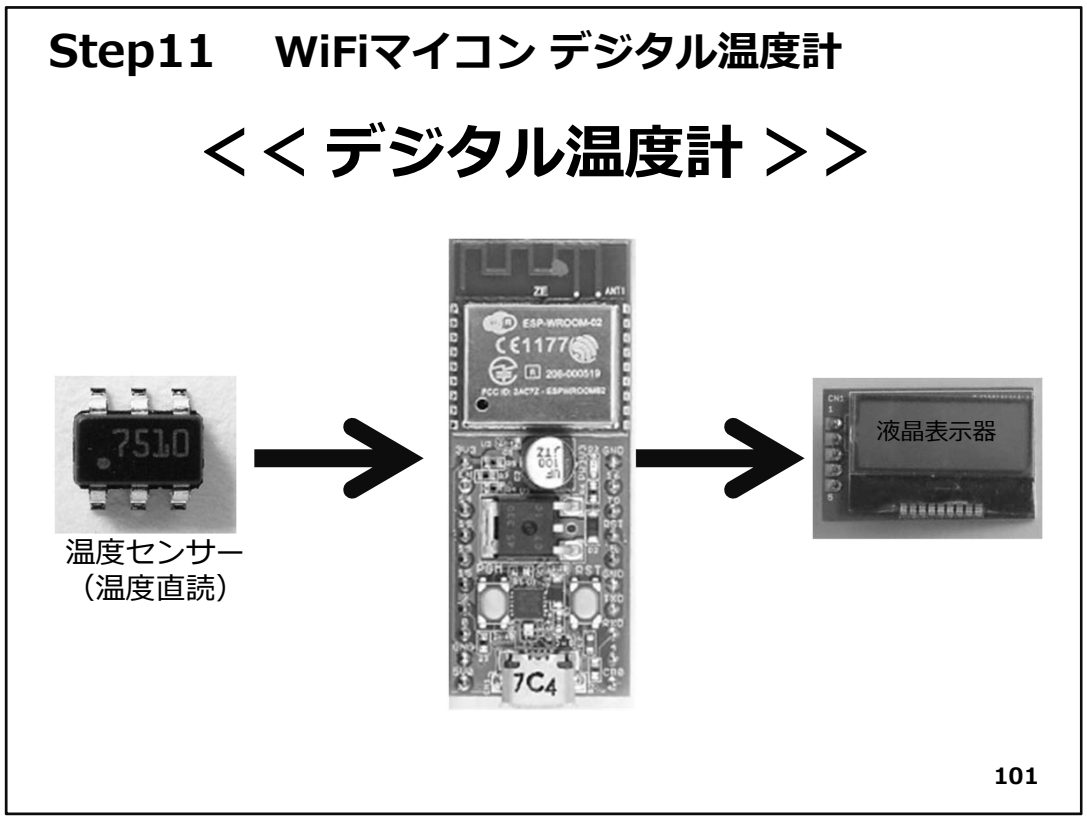
※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作確認

100

◇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。

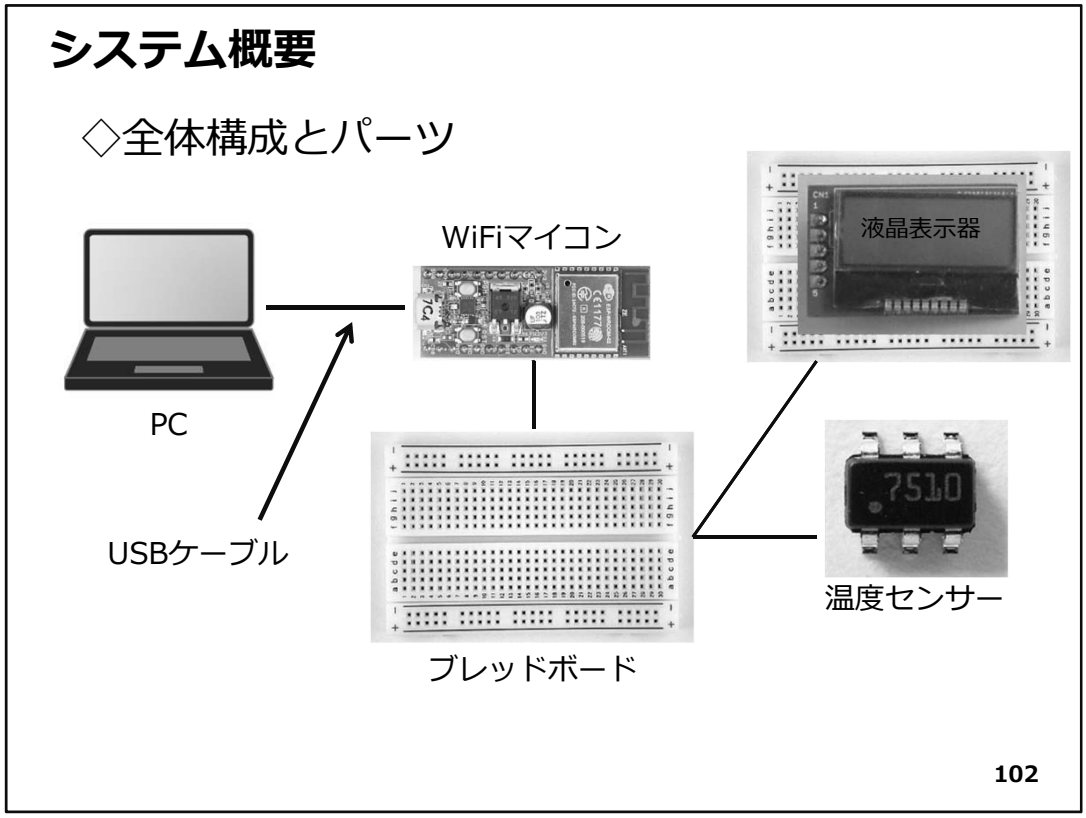
◇次回は、デジタル温度センサーとLCDを同時に使用して、デジタル温度計を開発します。



Step11 WiFiマイコン　デジタル温度計

Step9ではデジタル温度センサー、Step10では液晶表示器（LCD）を使ってみたわけですが、Step11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。

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。

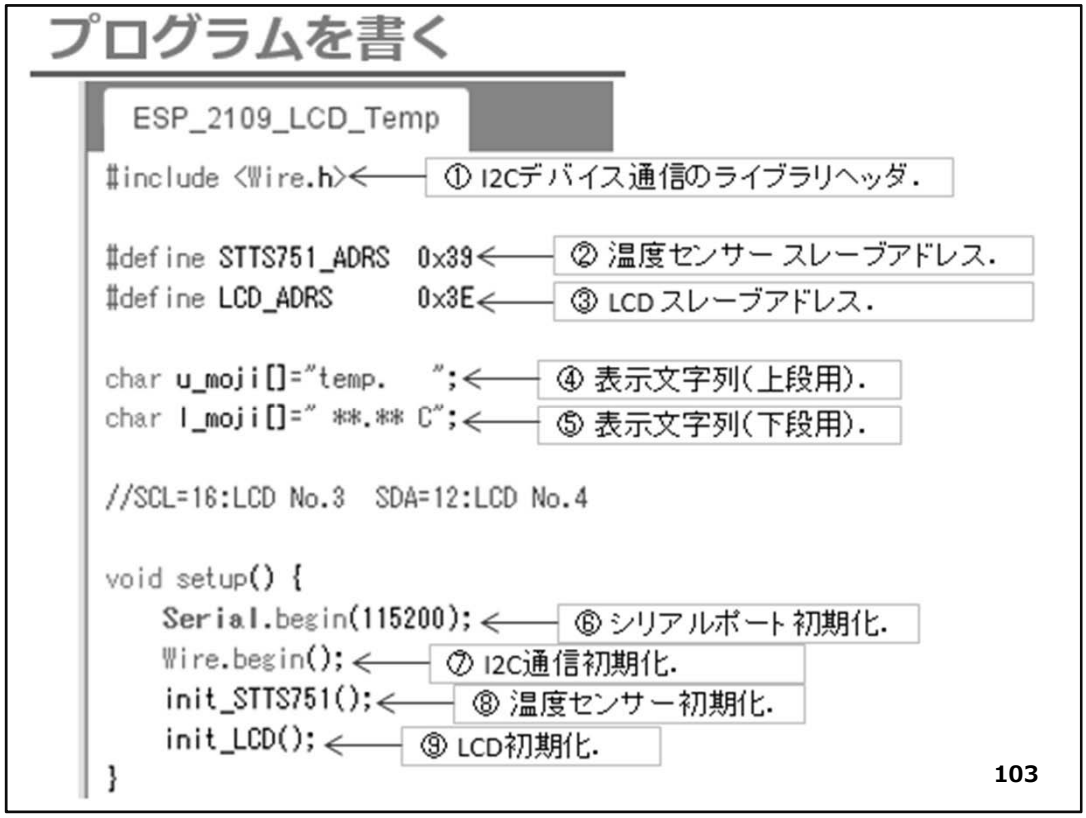


◇図で分かるように、直前2回の演習で使用したデバイスを使用します。
そのために、前回もデジタル温度センサーを残したままの回路としました。
必要な機材・パーツは、下記です。（前回のまま）

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。回路も同じものです。保存してある方はそのまま利用できます。

◇デジタル温度センサー、LCDについては、前2回で詳しく解説しました。回路図も同様ですので、このStepでの説明は省きます。



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。)

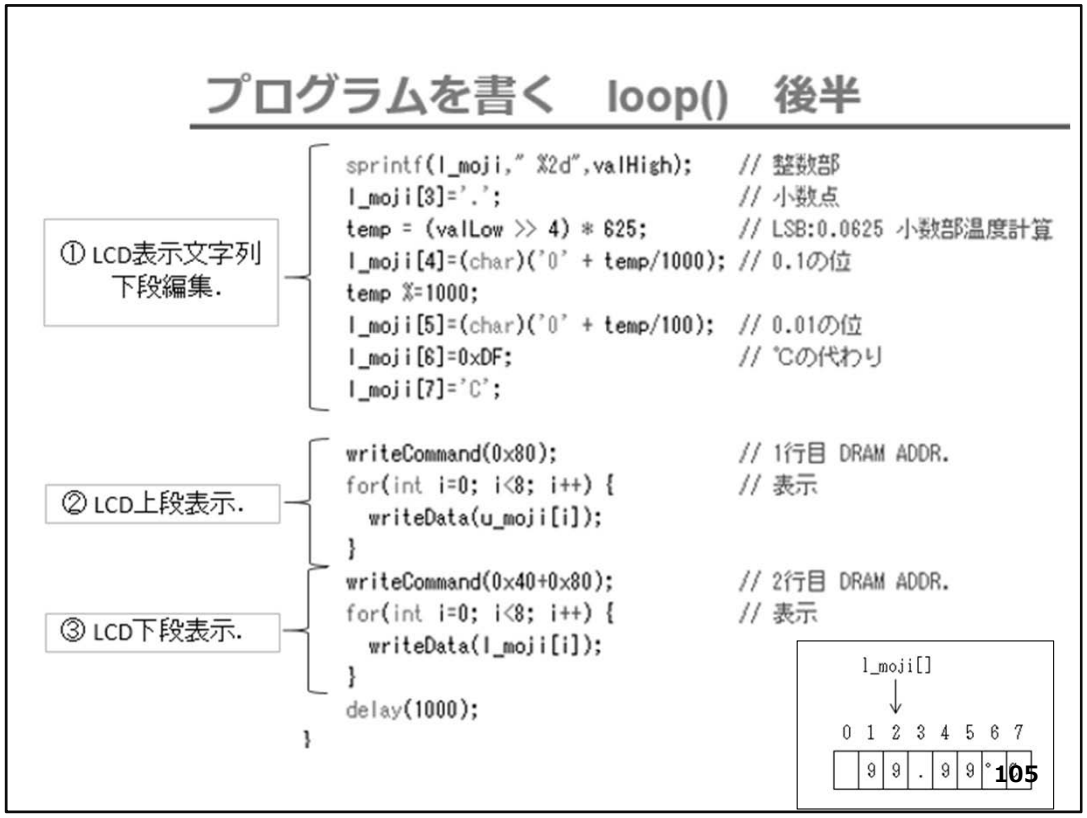
プログラムを書く　loop()　前半

```
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
```

104

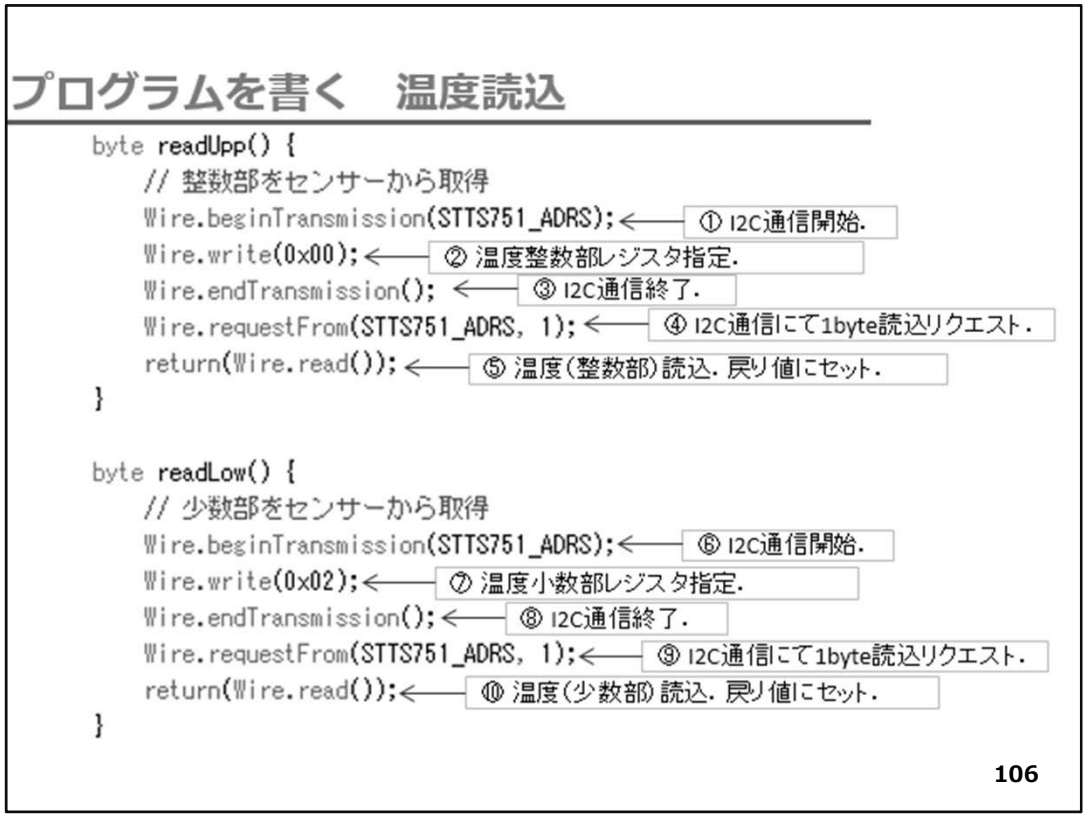
①温度の整数部をセンサーから取得しvalHighに格納
　※readUpp()関数は後で解説
②温度の少数部をセンサーから取得しvalLowに格納
　※readLow()関数は後で解説

上の2つの部分は、第7回デジタル温度センサーの演習では、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。



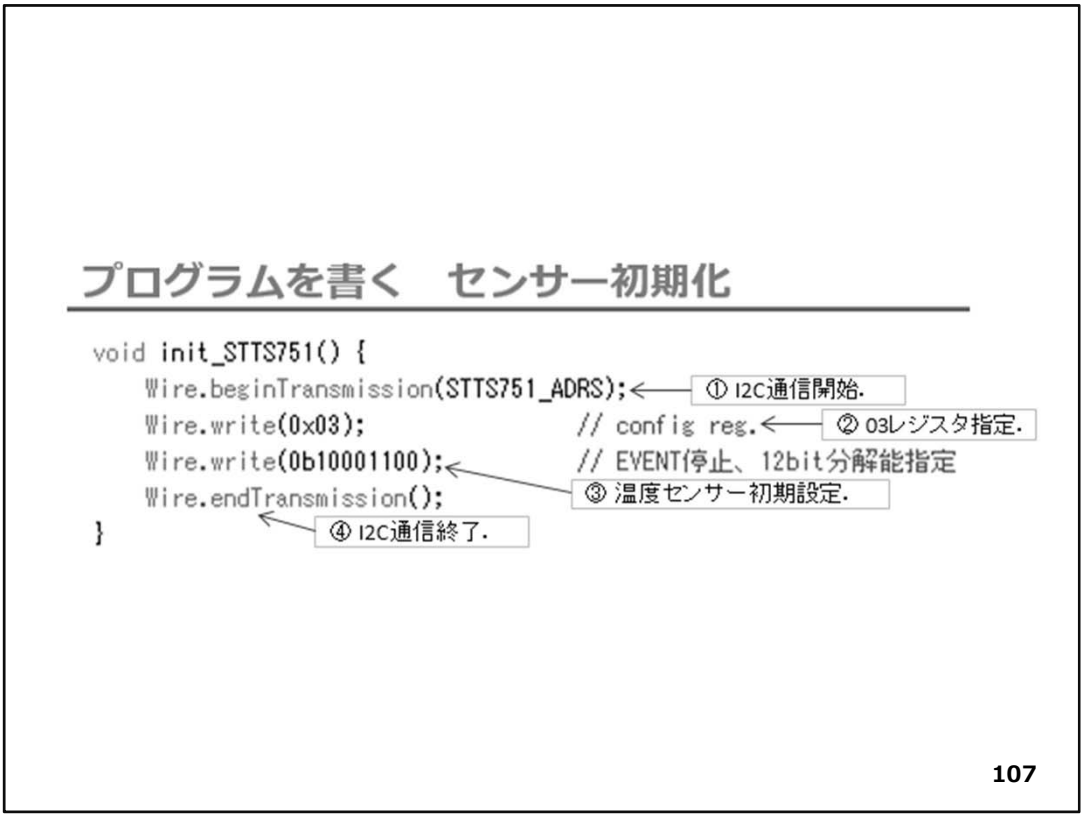
①LCD下段に表示する温度表示用文字列の編集を行う。
　　※文字列用バッファは、図右下の様に編集されます。
②LCDの上段（1行目）に固定文字列を表示
③LCDの下段（2行目）に温度文字列を表示

delay(1000)ですので、1秒おきに動作が繰り返されます。

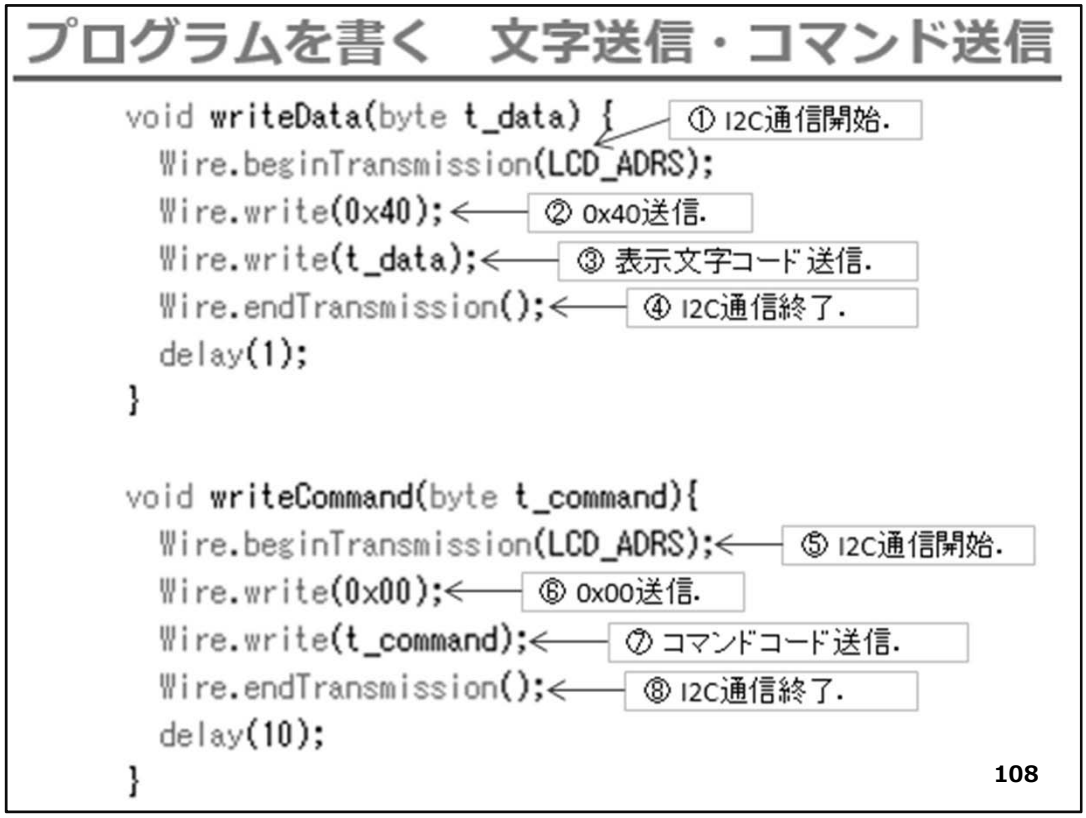


①I2C通信開始
②00番（温度整数部）レジスタ指定
③I2C通信終了
④I2Cにて1byte読込リクエスト
⑤1byte（温度整数部）読込。そのまま戻り値としてセット

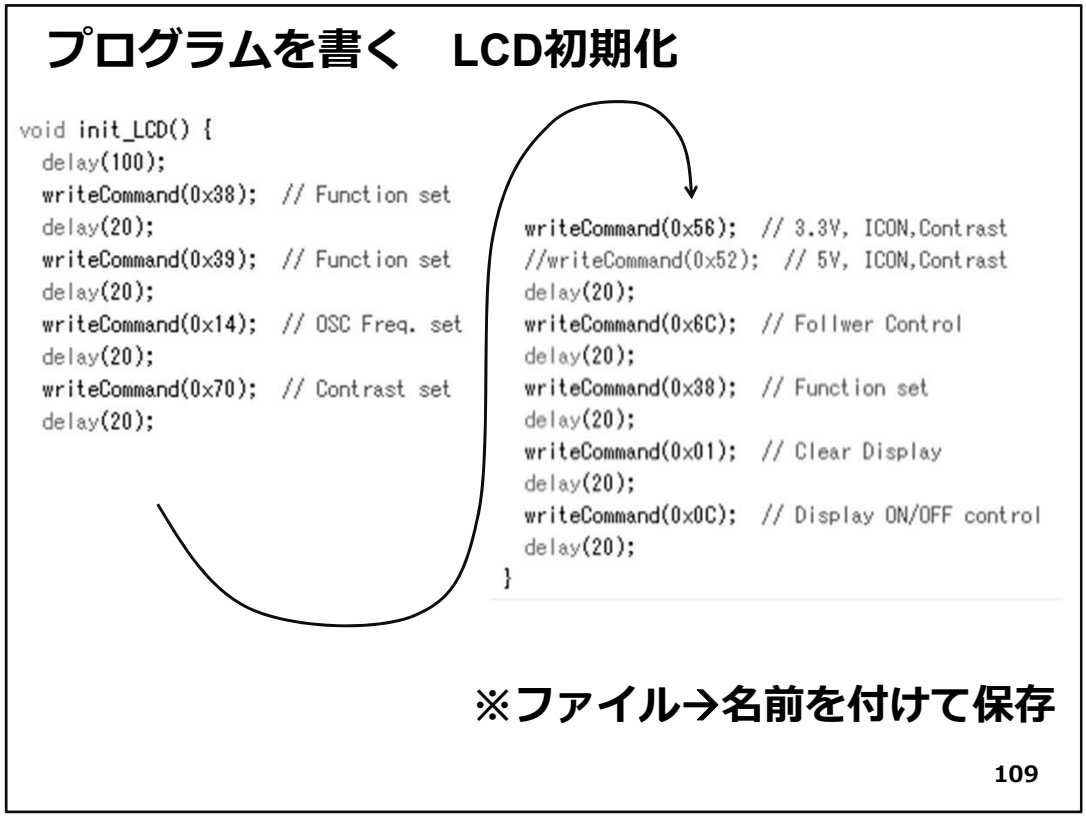
⑥I2C通信開始
⑦02番（温度小数部）レジスタ指定
⑧I2C通信終了
⑨I2Cにて1byte読込リクエスト
⑩1byte（温度小数部）読込。そのまま戻り値としてセット



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



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。



◇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。

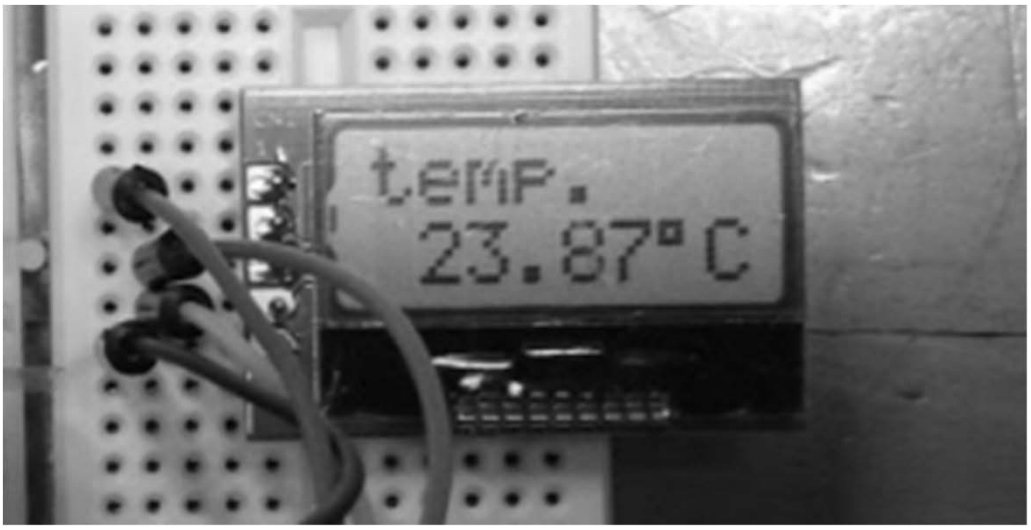
①PCとWiFiマイコンの接続：USBケーブルで接続
②シリアルポート（COM番号）の確認：デバイスマネージャで確認
③シリアルポート（COM番号）の設定：IDEで設定
④WiFiマイコンへの書込み準備：WiFiマイコンのSW操作
⑤スケッチ（プログラム）のコンパイルから書込み：IDEで操作

◇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。

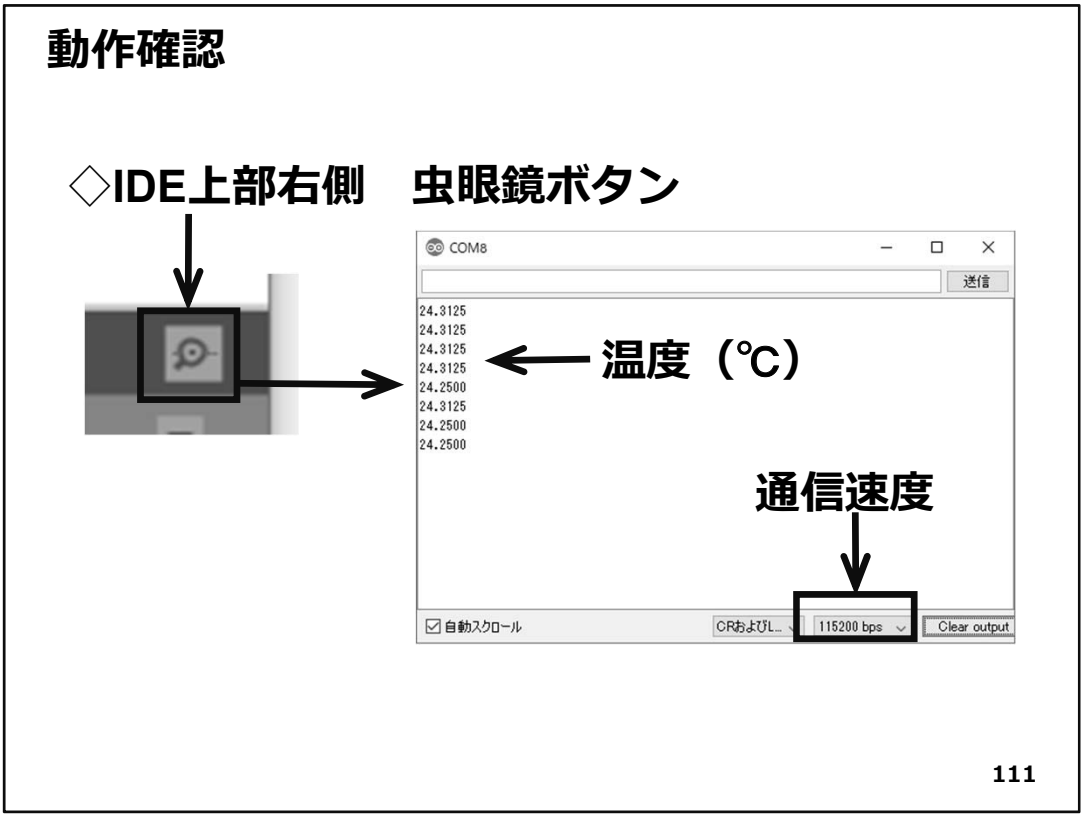
動作確認



110

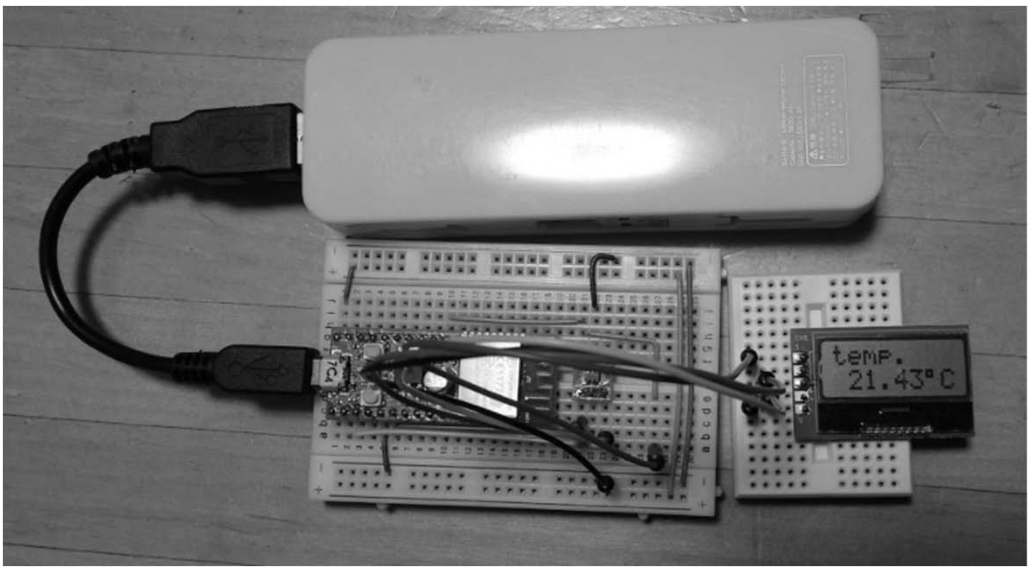
◇動作確認

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。



◇通信速度を合わせると、温度が1秒ごとに表示されます。

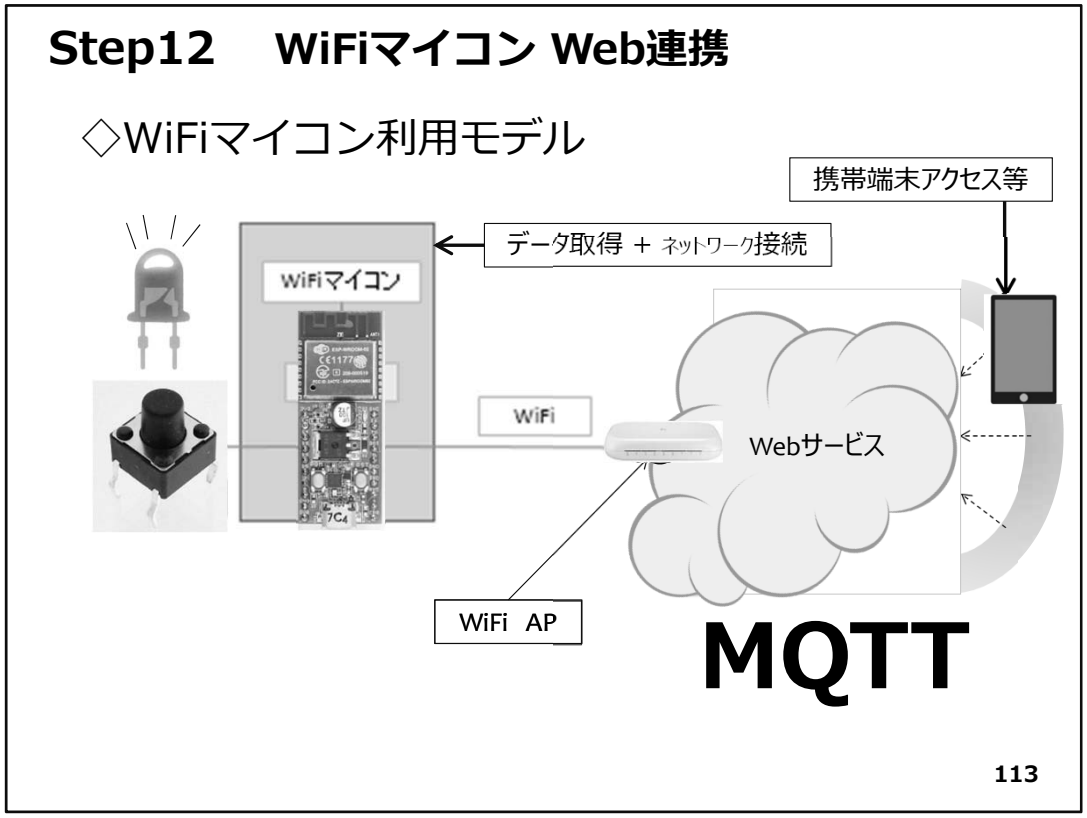
バッテリー駆動の例



112

◇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。

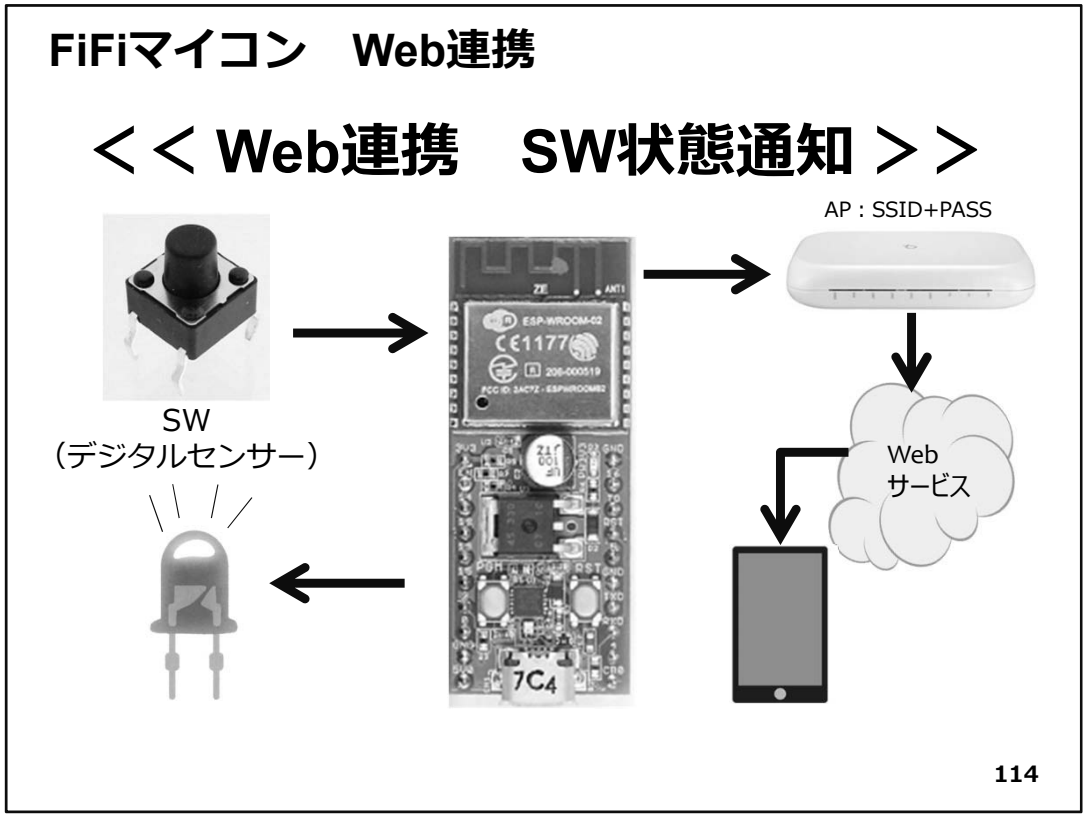
◇これまで、沢山の演習をしてきました。これで、WiFiマイコンの【マイコン機能】の使い方は終了です。次回は、いよいよWiFi機能を使ったWeb連携システムの利用を演習します。



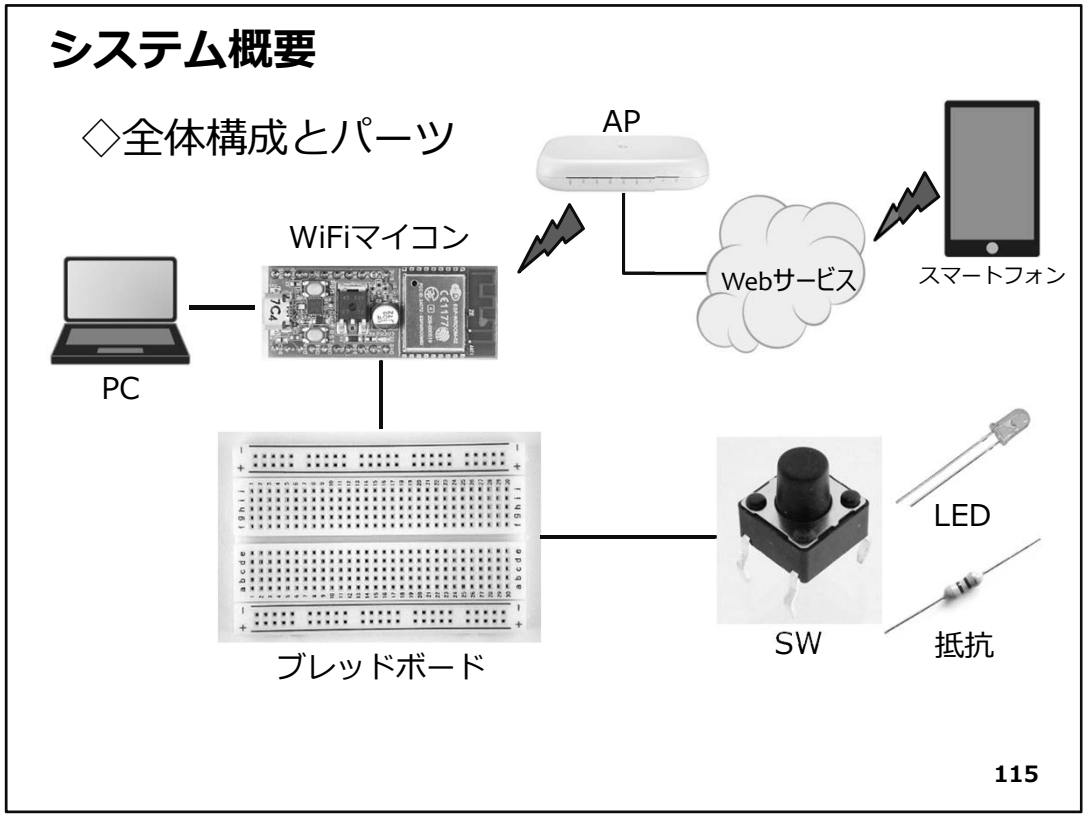
Step12 WiFiマイコン Web連携

今回は、Webサービスを通じてSWの状態を通知するモデルです。図をご覧ください。

WiFiマイコンの外部デバイスとしてSWとLEDを使用します。SWの状態に応じてLEDの点灯・消灯を制御しながら、状態が変化したときだけ、MQTTサービスを利用してメッセージを発行するシステムを作ります。このモデルは、お年寄りの【見守りシステム】などで既に実用化されているものと同様です。工場などで設備の稼働・停止を把握する稼動状況収集システムや、製品が1個1個出来上がっている状況を把握する生産実績収集システム等に応用できます。



◇図に記載しましたが、SWはON/OFF状態を検出する【デジタルセンサー】だと考えれば分かり易いでしょう。LEDでSWのON/OFF状態を表示します。今回は【SWのON/OFF状態が変化した】ことを検出してメッセージを発行することにしましょう。



◇全体の構成は上図の通りです。必要な機材・パーツを下記に示します。

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
9. スマートフォン×1台（アプリインストール）

◇上記の機材・パーツはStep6の開発に使用したものと同じです。
回路はStep6を参照して配線してください。

Webサービス　MQTT

◇IoTに相応しく.
◇遠隔地への通知可能なWebサービス.
◇利用容易で無料枠あり.

Webサービス
MQTT

※Message Queue Telemetry Transport

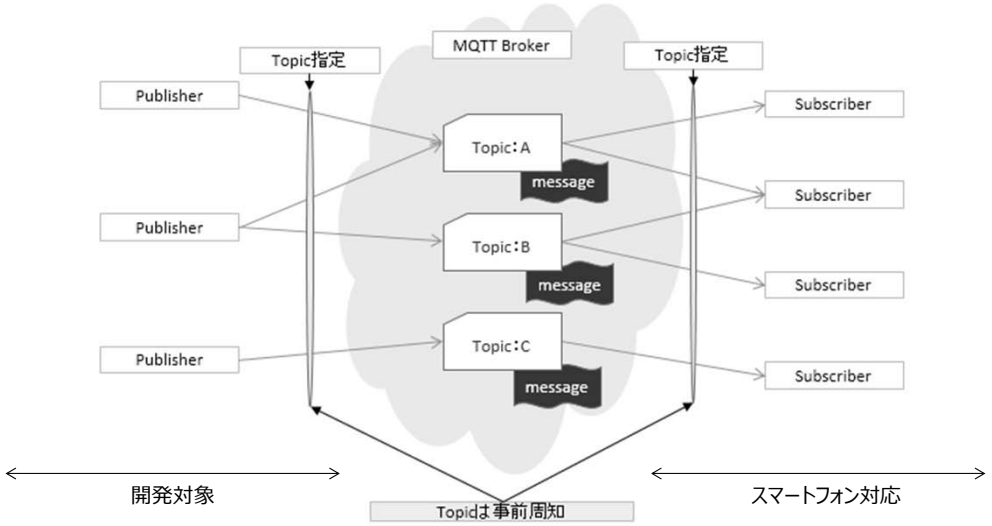
116

◇ MQTTはMessage Queue Telemetry Transportの略で、短いメッセージを頻繁にやり取りすることに特化したサービスです。

Webサービス　MQTT

◇MQTT：短いメッセージの発行と購読

※Message Queue Telemetry Transport



117

◇上の図はMQTTの利用方法を描いたものです。
まず、メッセージの発行者側（Publisher）とそのメッセージの読者側（Subscriber）で、特定のメッセージを交換するためにTopicというキーワードを決めます。読者側は購読したいTopicをあらかじめMQTTサービスに登録しておきます。メッセージ発行者がメッセージをTopicと共にMQTTサービスに送ると、MQTT Brokerは、そのTopicのメッセージ購読希望者に対してメッセージを配信するというものです。実際には、MQTT Brokerが登録されている全てのTopicと購読者に対してメッセージを送るのではなく、購読者側の端末プログラムがMQTT Brokerに問い合わせをかけて、新しいメッセージが発行されると、それを読み出す処理をしているようです。購読者が何人いても良く、これまで実験したところではタイムラグもほとんど感じません。ユーザ登録も不要で無料利用できるMQTT Brokerは、たいへんありがたい存在です。短いメッセージに特化しているからできることなのでしょう。

◇この演習では、メッセージの発行側（Publisher）を開発対象としています。購読者側（Subscriber）はスマートフォンアプリを利用します。

MQTT Broker

◇無料でテストできる MQTT Brokerがある.

１．test.mosquitto.org

２．broker.hivemq.com

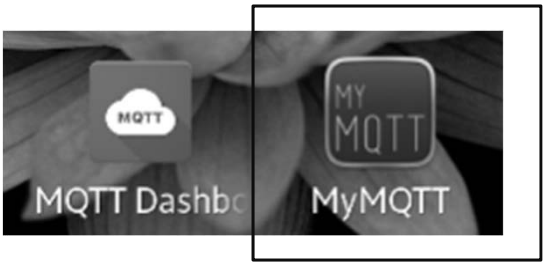
118

◇この演習では、接続が安定している**broker.hivemq.com**というMQTT Brokerを使いました。そして、スマートフォン側は、公開されているMQTTアプリの中で、連続して届くメッセージを見やすいMyMQTTを使用しています。

MQTT 対応スマホアプリの準備

◇公開MQTTアプリ

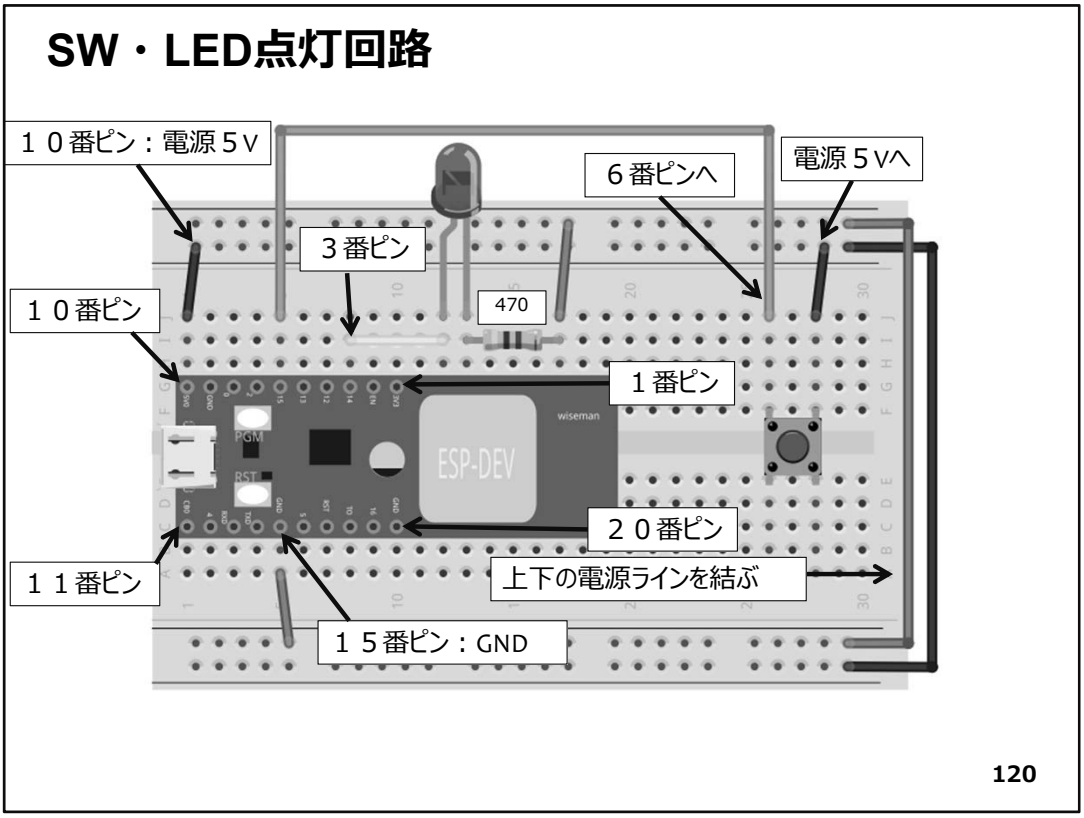
1．MQTT Dashboard（Android)
2．MyMQTT（Android)



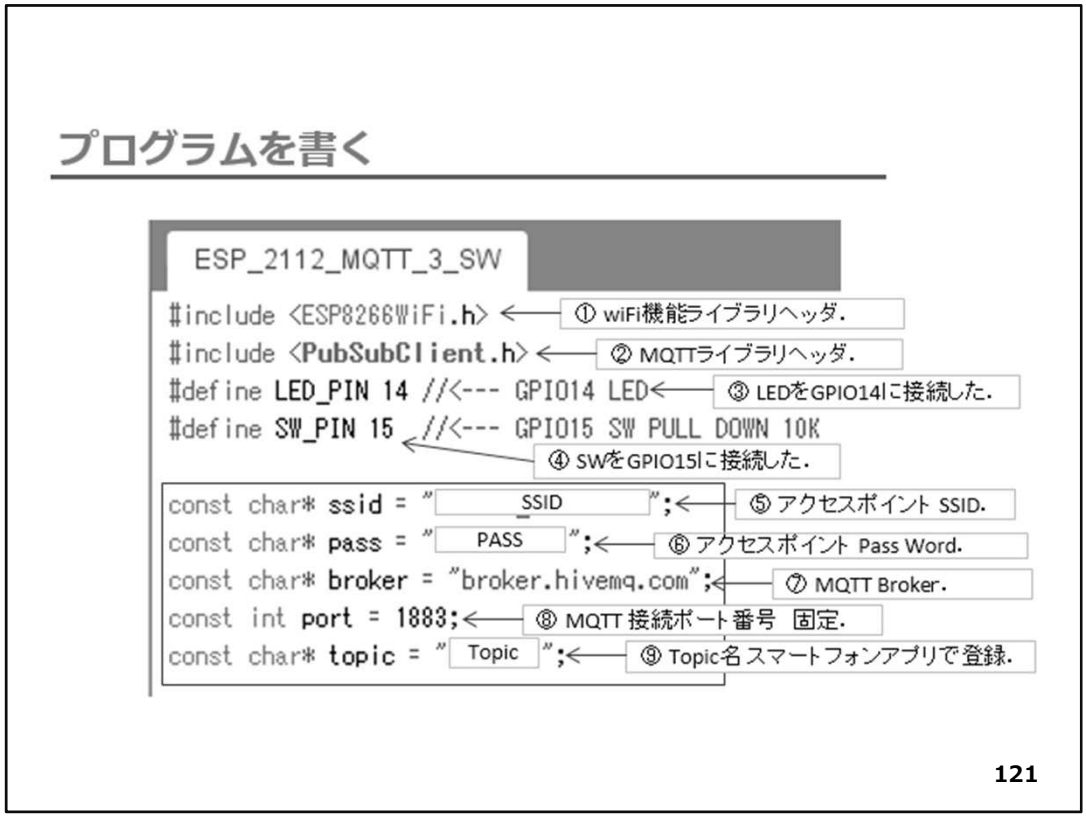
◇MyMQTT（Android版）を使用する.

119

◇ MyMQTTはAndroid版ですが、他の携帯端末をお使いの方は他のOS用アプリも公開されていますので探してみてください。アプリをダウンロードしてあらかじめインストールしておいて下さい。アプリの使い方は**【Appendix G：MQTTアプリの利用】で解説しましたので参照してください。**



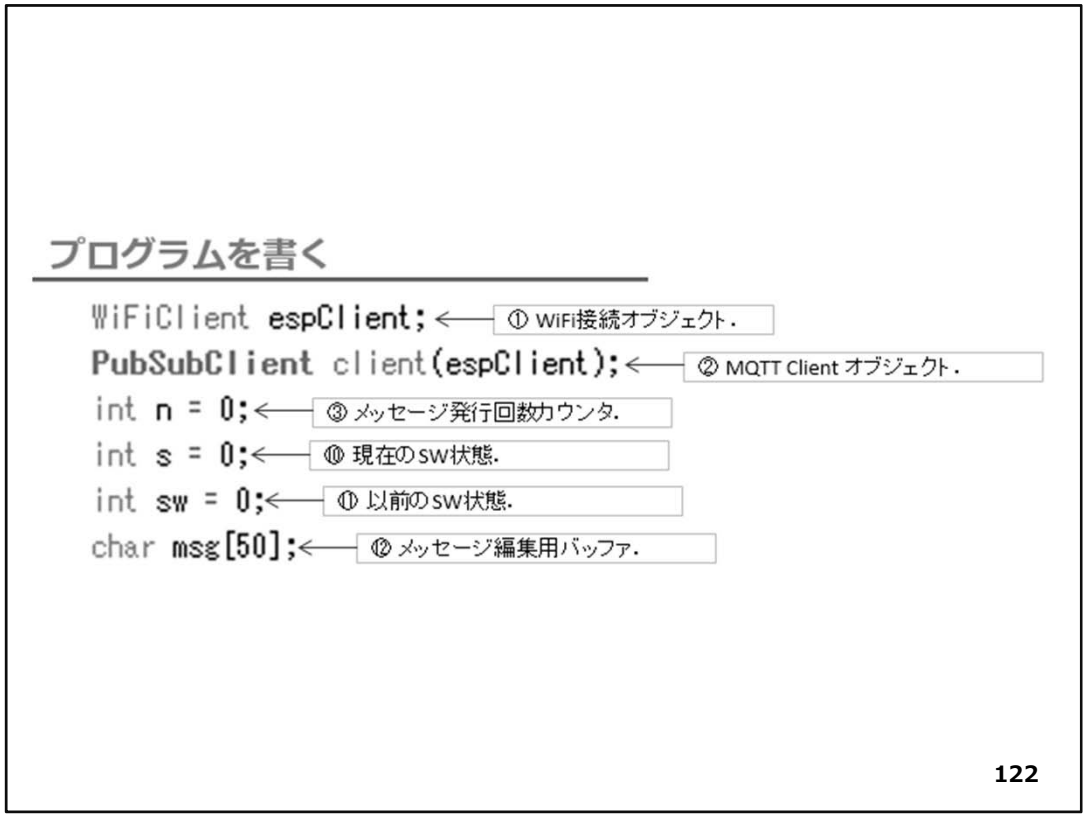
◇回路はStep6を参照して配線してください。回路図を再掲します。



◇ソフトウエア開発にはMQTTライブラリが必要です。
MQTTライブラリ準備の手順は【Appendix H：MQTTライブラリの準備】を参照して用意してください。

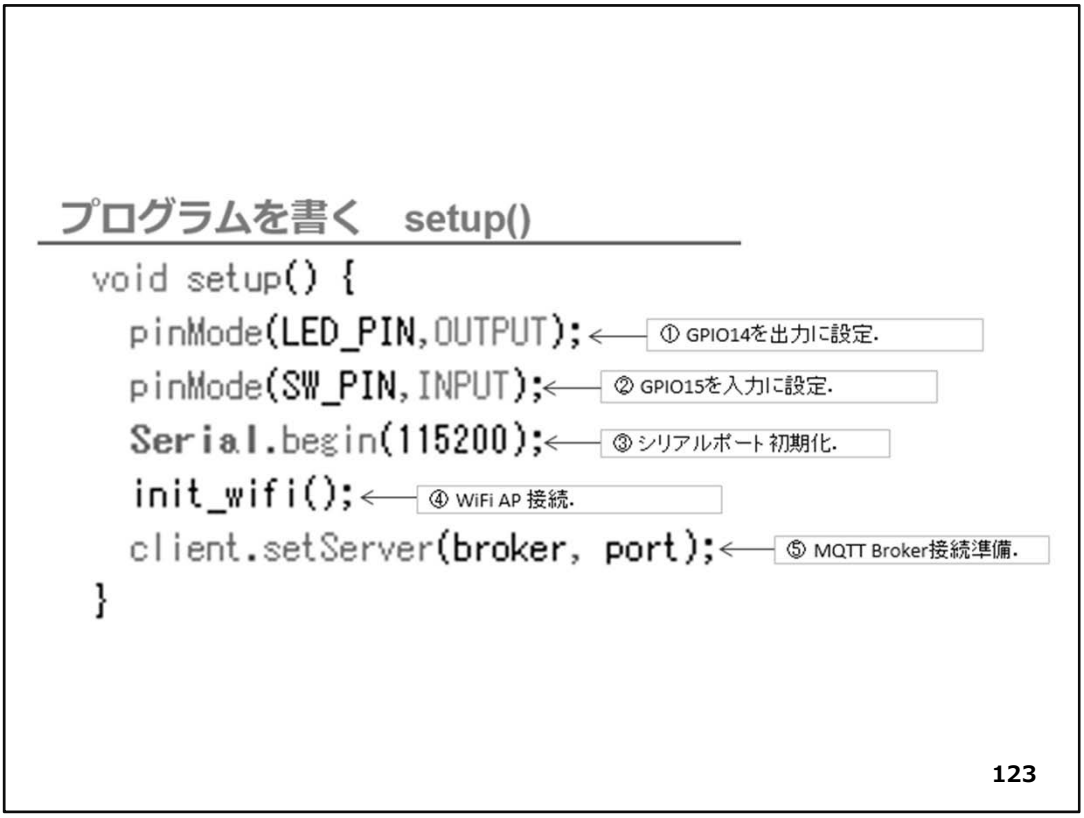
◇プログラムを作りましょう。以後に示すソースコードは、そのままの順番で、IDEに入力してください。既に詳細に説明した部分は解説を省略しています。これまでの演習の解説を参照してください。

①WiFi機能を使用するためのライブラリヘッダ
②MQTT用ライブラリヘッダ
③LEDをGPIO14に接続
④SWをGPIO15に接続
⑤アクセスポイントSSIDを文字列で記述
⑥同PassWordを記述
⑦使用するMQTT Broker
⑧ポート番号は固定で1883.
⑨スマホアプリで設定したTopic名を記述
※Topicはスラッシュ【/】で区切って階層構造化できます。また、ある階層以下を全部指定することも可能です。その時は、親のTopic名に【/+】を付加します。



①アクセスポイント接続用WiFi機能オブジェクト
②MQTT接続用オブジェクト
③メッセージ発行回数用カウンタ

⑩現在のSW状態
⑪以前のSW状態
⑫メッセージ編集用バッファ



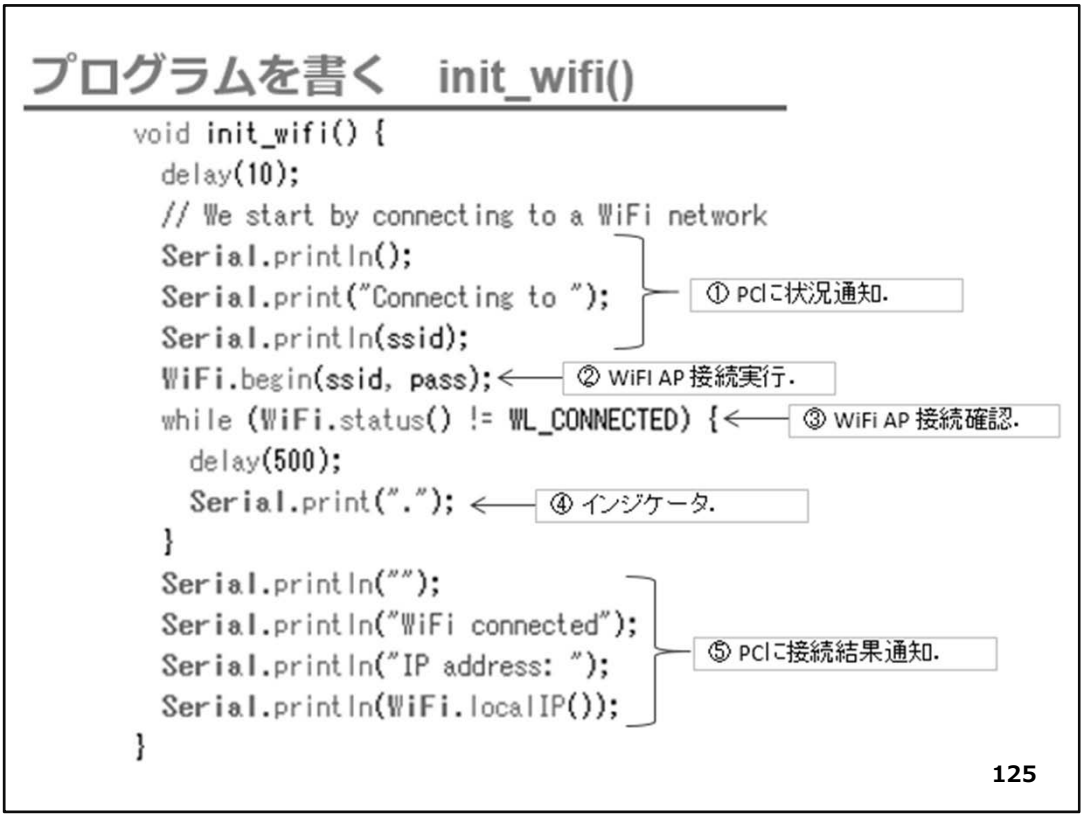
①. GPIO14を出力に設定
②. GPIO15を入力に設定
③シリアル通信115200bpsで初期化
④WiFiアクセスポイントに接続（関数内部は後に解説）
⑤MQTT Broker 接続情報を準備

プログラムを書く　loop()

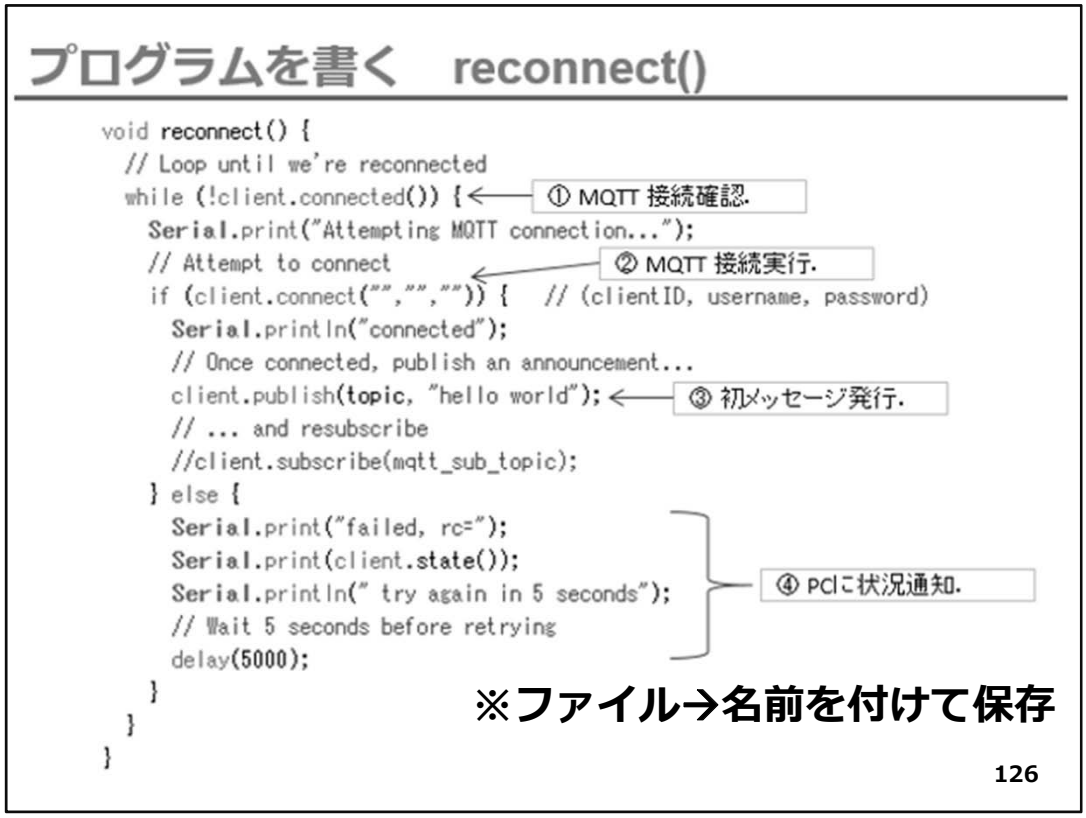
```
void loop() {
  if (!client.connected()) {          ← ① MQTT 接続確認.
    reconnect();                      ← ② MQTT 接続.
  }
  client.loop();                      ← ③ MQTT 接続情報更新.
  s = digitalRead(SW_PIN);            ← ④ 現在のSW状態読込.
  digitalWrite(LED_PIN, s);           ← ⑤ SW状態をLEDに反映.
  if(s != sw) {                       ← ⑥ SW状態変化したか？.
    ++n;                              ← ⑥ メッセージカウンタ更新.
    switch(s) {
      case 1:  // OFF--->ON           ← ⑦ SW ONした.
        snprintf(msg, sizeof(msg), "changed!! OFF--->ON #%d", n);
        break;                        ← ⑧ メッセージ編集.
      case 0:  // ON--->OFF           ← ⑨ SW OFFした.
        snprintf(msg, sizeof(msg), "changed!! ON--->OFF #%d", n);
        break;                        ← ⑩ メッセージ編集.
    }
    sw = s;                           ← ⑪ SW状態更新.
    Serial.print("Publish message: "); ← ⑫ PCに通知.
    Serial.println(msg);              ← ⑬ PCにメッセージ通知.
    client.publish(topic, msg);       ← ⑭ メッセージ発行.
  }
}
```

124

①MQTT接続状況確認。未接続であればMQTT接続
②MQTT接続実行
③MQTT接続状況とメッセージ発行状況の更新
　※この関数内部でこのプログラムとMQTT Brokerが情報交換。
loop()内で必ずこの関数を呼ぶ。
④現在のSW状態読込
⑤SW状態をLEDに出力（反映）
⑥SW状態が変化したか判断
⑥メッセージの発行回数を更新（＋１）
⑦SW　ONした
⑧メッセージ内容を編集してmsg[] 配列に格納
⑨SW　OFFした
⑩メッセージ内容を編集してmsg[] 配列に格納
⑪SW状態更新
⑫PCにシリアル通信で、メッセージ発行通知
⑬同、メッセージの内容通知
⑭MQTT Brokerに実際のメッセージを【Topic】を付けて発行【Publish】



①シリアル通信でPCにWiFi接続開始通知
②WiFi アクセスポイントに接続実行
③接続状況確認
④インジケータとして【. 】ピリオド表示
⑤PCに接続結果とIPアドレス通知



①MQTT 接続状況確認
②MQTT接続実行
③接続ができた場合、初めてのメッセージを発行
④接続が巧く行かなかったとき、PCに状況を通知

◇このreconnect()は、MQTT PuSubClientライブラリのサンプルプログラムの関数をそのまま流用しています。

◇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。

【重要】
必ず保存をしてください。

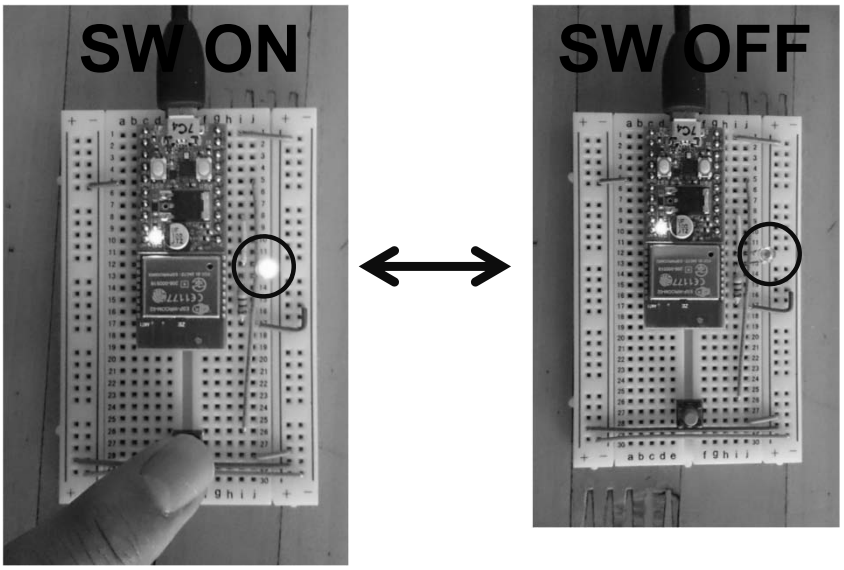
◇ここから、WiFiマイコンへの書込みまで一気に進めましょう。
以下の手順でプログラムのコンパイルからWiFiマイコンへの書き込みを行います。

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

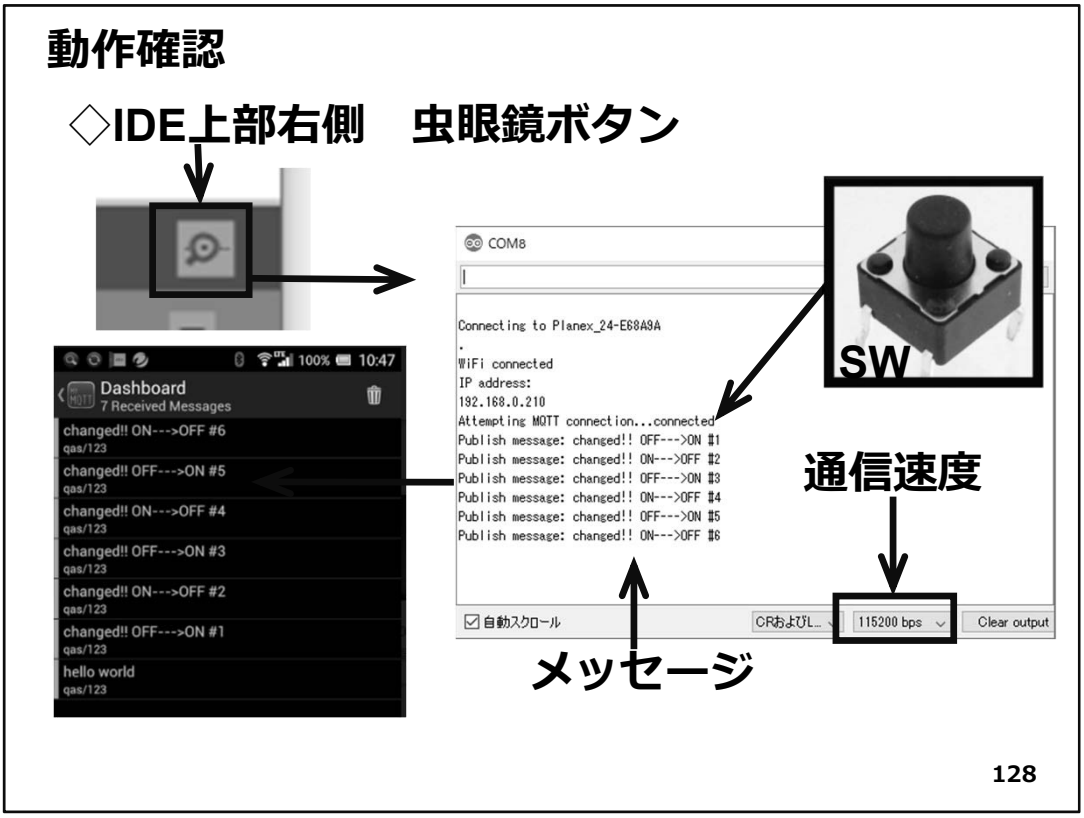
※上記の手順についてはすでに身に付いていることと思います。不安がある方は、これまでの解説を読みながら、【正確】に手順をトレースしてください。手順を誤ると巧くゆきません。私の経験では①から③は、大丈夫ですが、④を時々忘れます。IDEの下部に、マイコンへの書込みが失敗した旨のメッセージが表示されても、驚くことなくWiFiマイコンのSW操作を行い、もう一度IDEの右向き矢印ボタンをクリックしてください。
書込みが終了すると、WiFiマイコンにResetが掛かり、プログラムの実行が始まります。

動作の様子



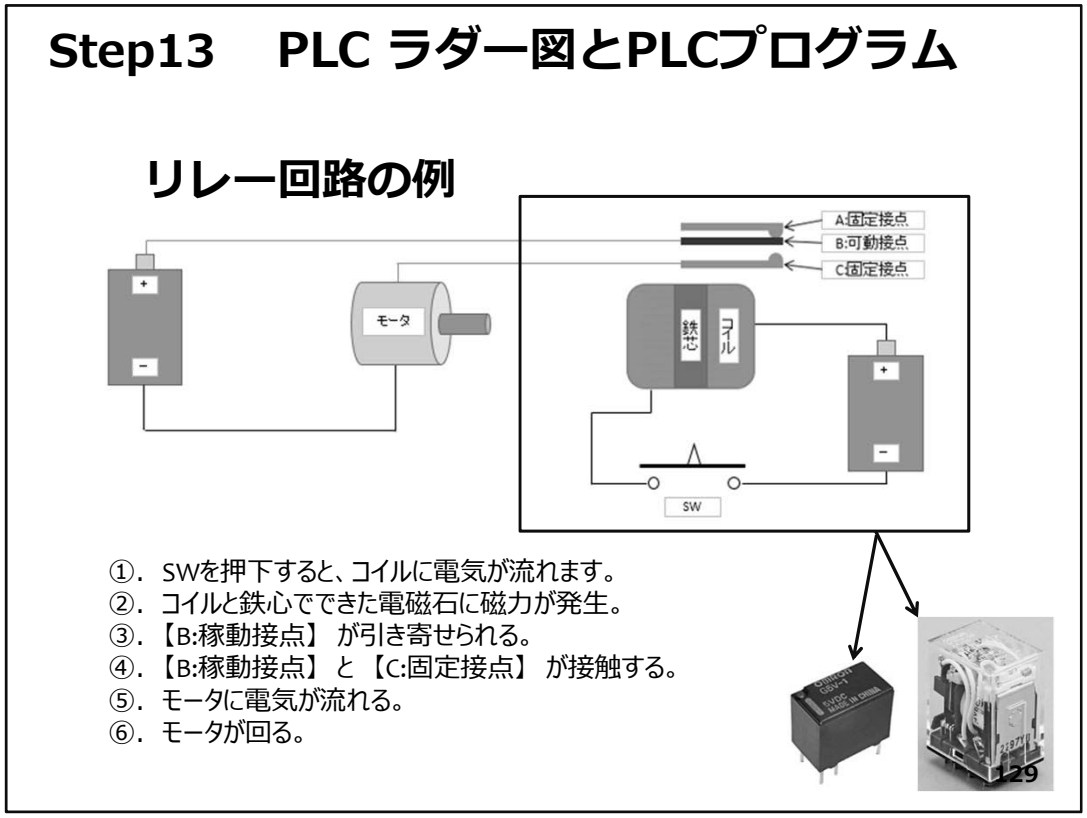
127

◇動作確認をします。まず、SW ONでLEDが点灯し、SW OFFでLEDが消灯することを確認しましょう。



◇次にシリアルモニターを起動して、通信速度を合わせます。
この状態で、SWのON/OFFに対応して、メッセージが表示されるでしょうか。また、SWをONしたまま、あるいはOFFしたままの状態では、メッセージが表示されないことも、確認しておきましょう。

◇続いて、携帯端末でメッセージ購読【Subscribe】用のアプリケーションを起動しましょう。MQTT Brokerに接続してメッセージ講読状態にしておきます。ここで、SWをON/OFFしたとき、携帯端末のアプリ画面に購読【Subscribe】したメッセージが表示されます。



Step13　PLC ラダー図とPLCプログラム

【重要】

このステップ以後【openPLC（PLCをシミュレートするシステム）】を使用します。ソフトウエアのインストールは大変時間が掛かる場合がありますので、Appendixを参照して事前に準備をして下さい。

◇リレー回路

この図は【基礎編】Step14で示した、リレーを使用した回路の例を再掲しています。

この回路の動作は、

①. 図赤枠内下のSWを押下すると、コイルに電気が流れます。
②. コイルと鉄心でできた電磁石に磁力が発生。
③.【B:稼動接点】が 引き寄せられる。
④.【B:稼動接点】と【C:固定接点】が 接触する。
⑤. モータに電気が流れる。
⑥. モータが回る。

となって、SW押下　→　モータ回転　となります。

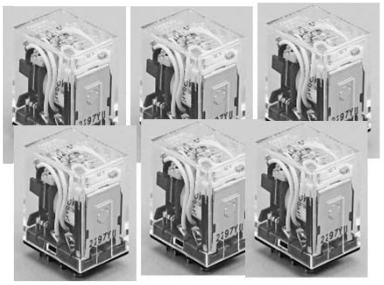
※モータ電源とリレー電源が別々になっていることは重要です。

この図の赤枠内は、前の図のSWの代役を務めていることになりますが、
このようなことを行う産業機器としてリレー（右下写真）があります。
※写真左は基板直付け用、写真右は制御盤内で用いられるものです。

このようにリレーを用いた回路を【リレー回路】と呼びます。

複雑な制御とPLC

複雑化→SW・リレーがどんどん増える！！



◎ ◎ ◎ そこでPLCの登場！！ ◎ ◎ ◎

130

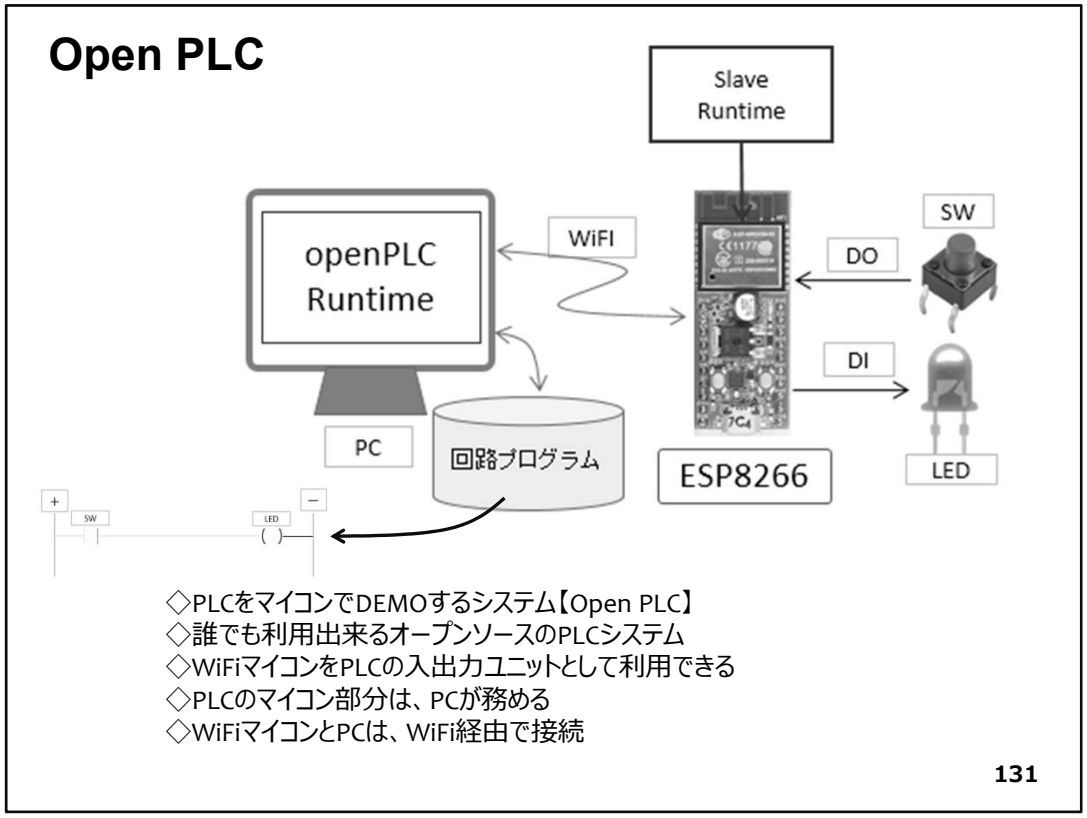
◇複雑な回路とPLC

◇条件が複数あるような複雑な制御を行うとSWやリレーなどがどんどん増えてしまい、これらを配線して納める制御盤の中も混雑してきます。そのような状態では、少しの追加・改善を行う場合でも、正確な配線が出来なかったり、リレーや配線が制御盤内に収まらないなどの問題が出てきます。制御盤を新たに作り直し回路も組み直し・・・などと言う事になると、掛かる時間も費用も工数も増加してしまいます。

◇そこで、そのような複雑な制御回路をコンパクトにまとめられる機器としてPLC（Programable Logic Controler）が使われています。

◇PLCの中身は実はマイコンです。SWやモータなど入力対象と出力対象は、そのままPLCに配線し、リレー回路の設計図と同じ回路図（ラダー図）をパソコンで描いてPLCに転送すれば、PLC内部でシーケンス回路がシミュレーションされて動作してくれます。変更したいときは、パソコンで回路図を変更すればOKです。制御対象の入力と出力の間にある中間的なリレーが、PLC内部のマイコンのメモリーで実現されるので、制御盤内のリレーの数が激減します。回路図はパソコンのファイルとして保存されるので、管理が容易です。USBメモリやSDカードなどで現場のPLCに転送することも可能になっています。配線作業も軽減されるので、工場などでは多くのPLCが使われています。

◇IoTデバイスは、この様に既に生産現場に多く存在しているPLCのシステムとも協調しながら、IoTシステムを構築する必要もあるのです。その場合PLCで稼動している回路図のどの接点から情報を取得するか検討する必要も出てきます。リレー回路の基礎を理解していると大変役立ちます。



◇openPLC

◇国際標準に従い、PLCをマイコンでDEMOするシステムで、誰でも利用出来るオープンソースのPLCシステムがあり、WindowsPCで利用できます。海外では、小規模の自動化やホームオートメーションなどに多く活用されています。

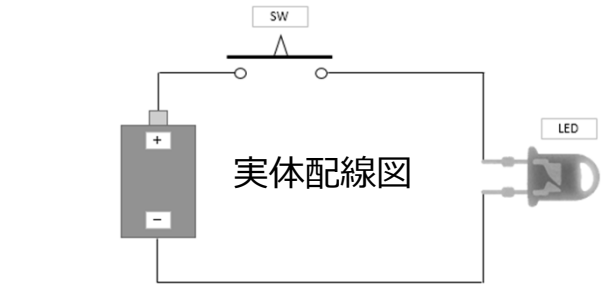
◇対象となるマイコンには、ESP8266（WiFiマイコン）が含まれているので、ソフトウエアを準備すれば、演習キットでPLCの基本的な演習ができます。

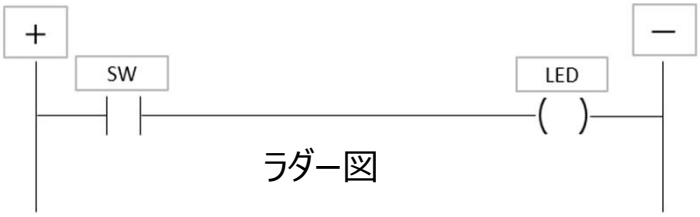
【注意】 PLCopen という名称の機関があります。
PLCプログラミングの国際標準IEC 61131-3（JIS B 3503）の普及活動と、ベンダに依存しない標準ファンクションブロックの仕様策定及び認定を行う第三者機関の名称です。 openPLCは、国際標準に沿ったシステムの名称です。混同しない様に注意しましょう。

◇PC内部でopenPLC Runtimeが稼働して、回路プログラムをシミュレートします。
RuntimeはWiFi経由の通信で、マイコンに対して形式の定まった電文でマイコンDI/DOの要求を行い、マイコンはその要求に対して同様の電文で返信します。回路プログラムはあらかじめ、PLC Open Editor というアプリケーションで作成し、システムに保存（Load）しておきます。

◇これから、基本的なリレー回路の一部をopenPLCで開発してみましょう。

実際の配線とラダー図





132

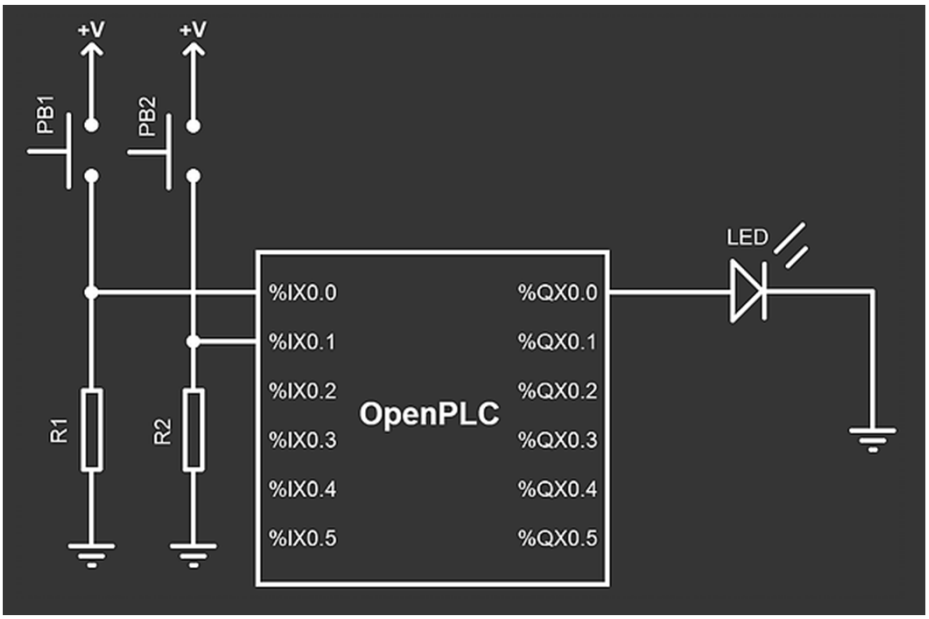
◇図にSWとLEDを1個ずつ使用した、LED点灯回路の実体配線図（上）と、PLCで使用するラダー図（下）を示します。

◇実体配線図の電源は、便宜上乾電池を描いてあります。SWを押せばSWがONして（回路が閉じて）LEDが点灯します。SWから手を離せばSWがOFFして（回路が開いて）LEDが消灯します。

◇ラダー図では、SWに該当する記号と、LEDに該当する記号があり、それらが、左右の縦線（左側縦線が＋電源、右側縦線が一電源）の間に配置されて、水平の線で接続されています。縦線は電源の線（母線と言い、一般に左側が電源＋です。）で、横線は電気が流れる導線を意味します。SWやリレーなどを多数使用した回路になると、この横線が増えて、梯子のようになるので、【ラダー図】と呼ばれています。

◇penPLCに限らず、PLCのプログラムには何種類かの入力方法がありますが、このラダー図を描くことで該当回路のPLCプログラムを作ることが多くなっています。
回路そのものを描いて分かり易いので、多くの現場で使われています。

PLCで作る初めての回路図

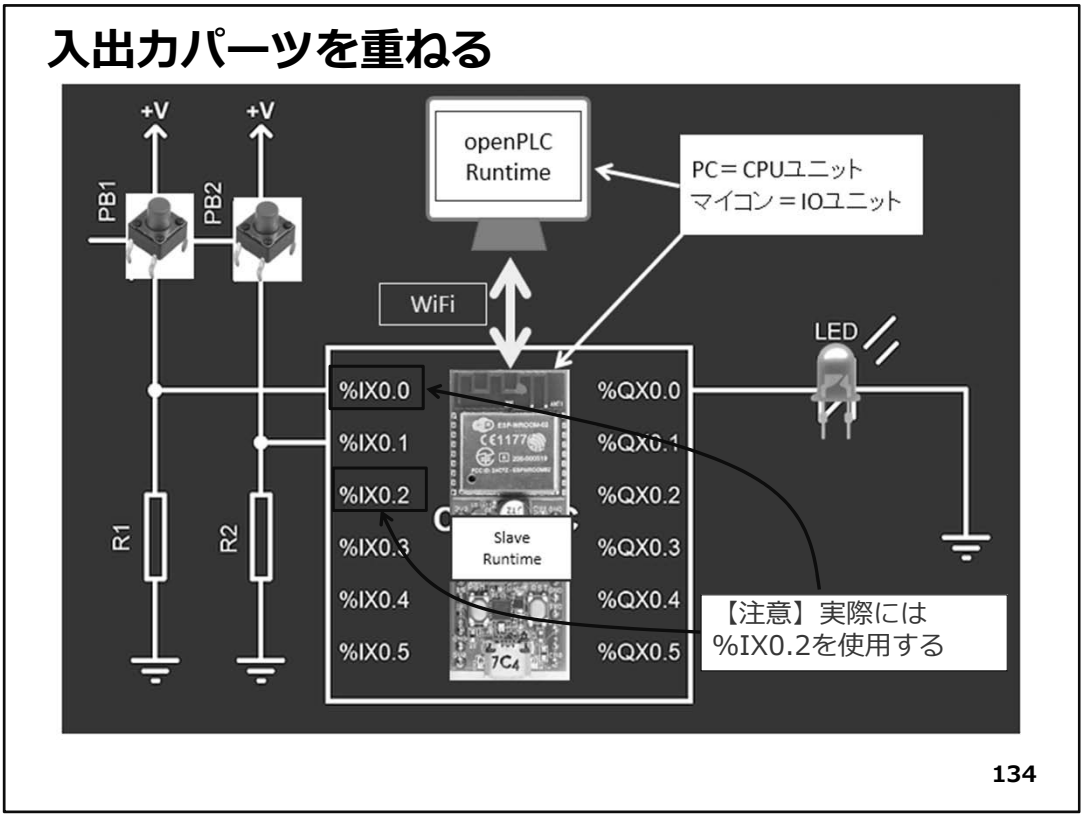


133

◇図は、openPLCプロジェクトのWebページにある、初めてのプロジェクトで作る回路図です。この回路を使い、SW1を押すとLEDが点灯し、SW2を押すとLEDが消灯するPLCのプログラムを開発する手順が、Webで紹介されています。

◇中央にOpenPLCがあり、PB1,PB2と書いてあるのがSWです。R1,R2は抵抗器。LEDが1個あります。この回路は、PLC内部が見えないので、どのような動きをするかはこの図だけでは分かりませんね。

◇回路図に実際のデバイスを重ねたものが次の図です。



◇PCでは、OpenPLC Runtime本体が稼働し、Slave用Runtimeが稼働するマイコンがWiFiで接続されています。

◇PLCのプログラムは、ラダー図として描かれたものをコンパイルして、PC用Runtimeで理解できる言語になってRuntimeの仕組みにUploadされています。
※Uploadと言っても、PC内部に存在しています。

◇PCの Runtimem は、UploadされたPLCプログラムを解釈して処理を行います。
※インタープリタ（逐次解釈して処理を進める）のようなものと考えて良いでしょう。

◇**PCは、PLCの本体（CPUユニット）**として働き、**WiFiマイコンは入出力ユニット**として働きます。これは、**本物のPLCと全く同じ構成**です。

◇PLCのプログラム処理の内容は、次のように簡単な処理の流れになります。

①. PLCのプログラムでSWの状態を調べる場合は、PCからマイコンにWiFi経由で図の【%IX0.0】と【%IX0.1】の状態を読み込む指令が送られて、マイコンは【%IX0.0】と【%IX0.1】に該当する DI の状態をWiFi経由で PC に返します。
②. PCに返されたSWの状態で、回路はどのような状態になるか、PC内部でシミュレートされて結果が出ます。
③. その結果、LEDがON（またはOFF）となるべき時には、【%QX0.0を1（または0）にせよ】と指令が送られ、マイコンはその通りに処理します。
④. マイコンは処理の結果DOがどうなっているかをPCに返信します。

PC内のRuntimeは上記を繰り返します。

※【%～】の表記は、スレーブデバイス（マイコン）の端子を示す記号と考えて下さい。
後で説明します。
※ここで重要なことは、DI（SW）の状態もDO（LED)の状態もPC側で、記憶されている と言う事です。
PC Runtimeが処理すべき次のPLCプログラムでは、DO（LED）の状態もSWと同じ接点として利用できることで実際の回路では必要な配線を大幅に省略できます。
物理的な電磁リレー1個が持っている接点の数は決まっています。その数を超えて、LEDに出力したことを他の回路の条件として使う場合は、一度別の電磁リレーをONして、その接点でLEDを点灯し、残りの接点を次の回路の条件で使います。

◇通常PLCはCPUユニット（本体）と入出力ユニット（IOユニット）で構成されますが、
OpnePLCでは、

★CPUユニット　→ PC
★入出力ユニット → WiFiマイコン

になります。

◇もう少し解説の必要な部分があります。図の中央のPB1、PB2、LEDがそれぞれ

PB1 → %IX0.0
【重要】
プロジェクト指定の%IX0.0では、Reset時マイコンの動作が不安定なので、以後%IX0.2とします。
PB2 → %IX0.1
LED → %QX0.0

となっていて、具体的なマイコンのDI/DOとどの様に対応しているのかが不明です。

◇ WiFiマイコン（ESP8266）では、次の様に決まっています。

DI側（図の左側）		Editorでの表記		GPIO番号		WiFiマイコンPIN番号
%IX0.0	→	%IX100.0	→	GPIO2	→	PIN7
%IX0.1	→	%IX100.1	→	GPIO14	→	PIN3
%IX0.2	→	%IX100.2	→	GPIO12	→	PIN4
%IX0.3	→	%IX100.3	→	GPIO13	→	PIN5
%IX0.4	→	割り当て無し				
%IX0.5	→	割り当て無し				

DO側（図の右側）		Editorでの表記		GPIO番号		WiFiマイコンPIN番号
%QX0.0	→	%Q100.0	→	GPIO16	→	PIN19
%QX0.1	→	%Q100.1	→	GPIO5	→	PIN16
%QX0.2	→	%Q100.2	→	GPIO4	→	PIN12
%QX0.3	→	%Q100.3	→	GPIO0	→	PIN8

%QX0.4　　→　割り当て無し
%QX0.5　　→　割り当て無し

※分かりにくい面倒な割り当てと感じる方も多いでしょう。Openな環境で高い汎用性を持たせた仕組みで、あらゆるデバイスに対応できるようにするために、この様になっています。

【重要】
上記GPI番号とWiFiマイコンPIN番号の割り当て表は、スライド番号39、67を参照してください。

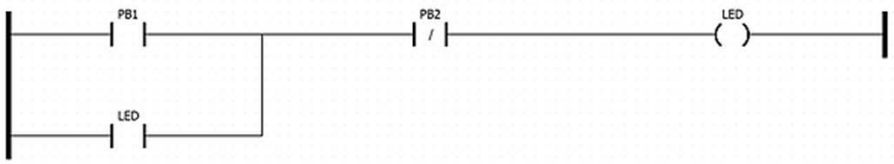
◇実際の配線では、

PB1 → %IX0.2 → %IX100.2 → GPIO12 → PIN4 【注意】
PB2 → %IX0.1
LED → %QX0.0

【注意】
Webで公開されている回路図（上図）で指定のSW1（PB1）の端子%IX0.0→%IX100.0→GPIO2→PIN7は、この基板内部で、10KΩでPullUPされていて、通常HighとなるDIの入力用になっています。ですから、それを上図の様にPullDownで通常LowとなるDIに使うのには向かないと言う事で、%IX100.2 のPIN4に変更しました。

【Appendix J：PLCOpenの使い方】で、作成したラダー図を次に示します。

回路図は、Editorで作成する【ラダー図】



135

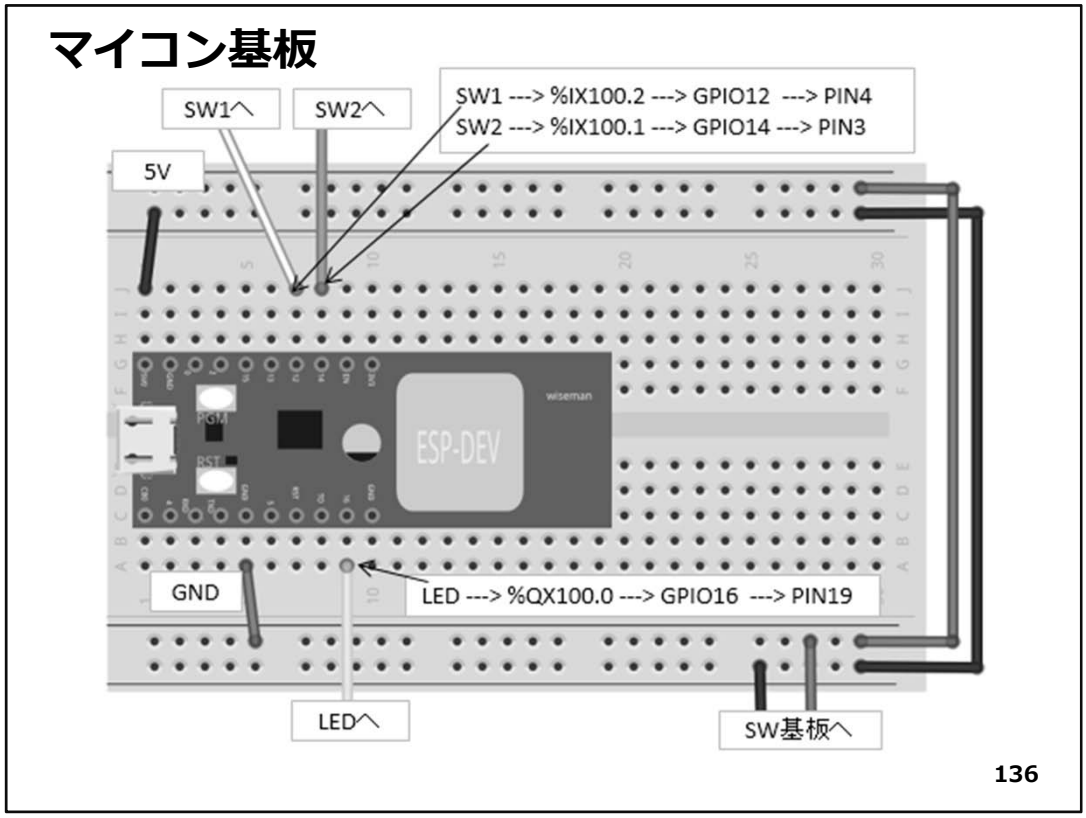
◇前頁の図も回路図ですが、この図もまた回路図です。前頁の回路図は単なる接続図ですが、ラダー図は、回路の動作が見えるというのが大きな違いです。

◇この回路の動作は、
①. SW1を押すとLEDが点灯する
②. SW2を押すとLEDが消灯する
となります。

◇SW1を押し続けなくても、LEDの点灯が**保持**される。
このような回路を **【自己保持回路】** と呼びます。

◇PLCは入出力ユニットから外側に実際の回路が必要です。それを作りましょう。

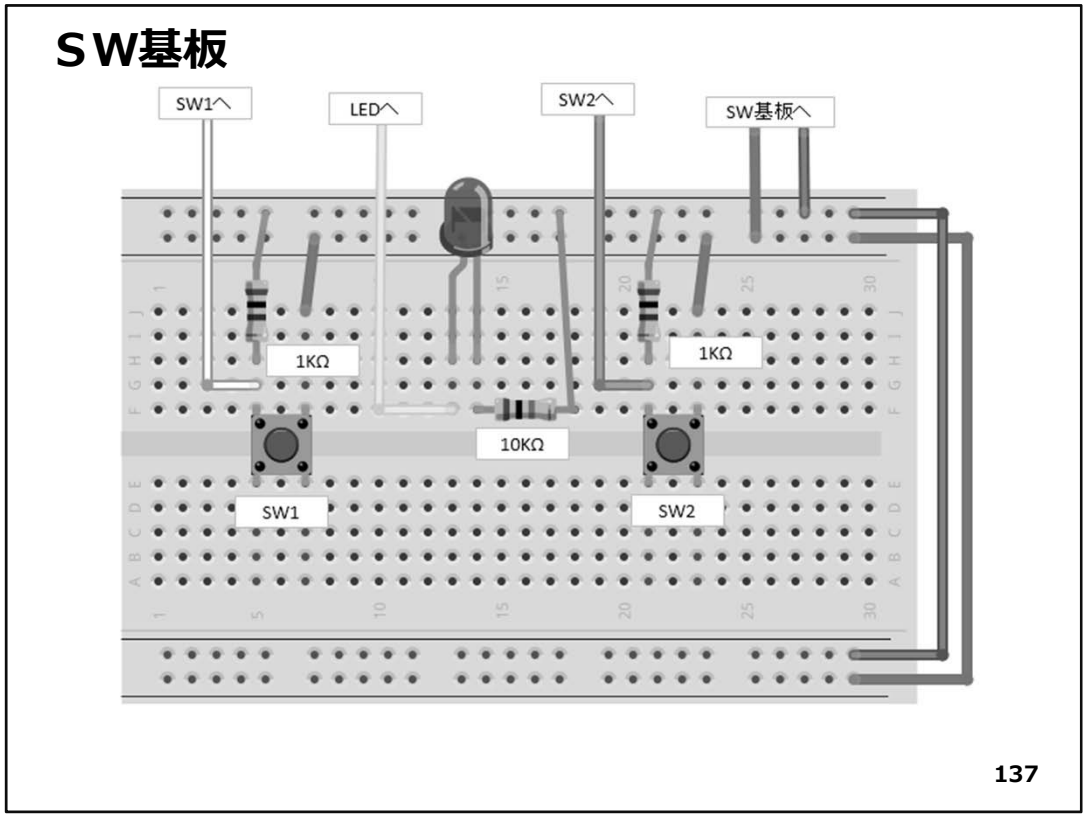
◇次頁に実際の回路図を示します。



◇PLCの配線は、2枚のブレッドボードを使用して、マイコン基板とSW基板に分けて作ります。

◇この基板は、WiFiマイコンと数本のジャンパー線だけですので、短時間でできると思います。

◇もう一枚のSW基板への配線が5本あります。



◇SW基板の配線は、SW2個とLED、抵抗が3本必要です。

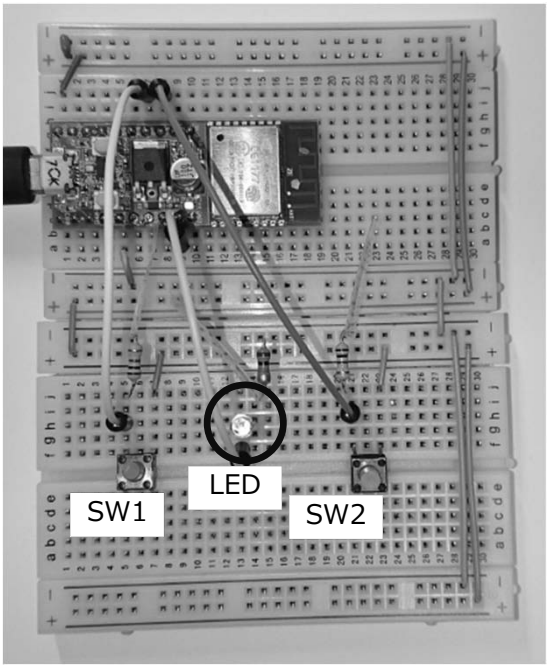
①. LED　×　1個
②. SW　×　2個
③. 抵抗器
　　10KΩ　×　1本（LEDの電流制限用）
　　1KΩ　　×　2本（Pull Down用）

◇ SWに接続している抵抗器（1KΩ）は、マイコンへの入力（DI）が、SWオープン時に、確実にLowになる様にするために、GNDに接続しています。このようにしておけば、通常は確実にLowで、SW押下時には、ショートすることなく、Highになります。 **この抵抗はPull Down抵抗と呼ばれます。**

◇LEDに接続している抵抗は、図134には無いものです。LEDは、電流が沢山流れると明るく点灯しますが、だからと言ってやたらに大きい電流流すと寿命が短くなってしまいます。また、LEDを駆動しているマイコンの回路もあまり大きな電流が流れることは、良いことではありません。ですから抵抗を直列接続して、流れる電流を制限しています。これを電流制限抵抗と呼びます。

◇ SW基板の配線が済んで確認ができたら、マイコン基板とSW基板を5本のジャンパー線で配線・接続しましょう。

実際に配線して、動作確認



138

◇実際に配線が済んだ様子を写真に示します。

◇ここでは、2枚のブレッドボードをつないで使っています。つなぎ目のGNDと＋電源の配線を忘れない様にしてください。

◇回路を作成したら、【Appendix J：PLCOpenの使い方】 を参照してPLCプログラムを作り、Uploadして動作確認をしましょう。

◇SWを1度押下すると、LEDが点灯し続けます。 これを自己保持というのでした。
SW2を押下するとLEDは消灯します。

◇初めてOpenPLCを使うときは、工程が多く大変に感じますが、モノつくり企業の中では、このようなことが日常行われているのです。

◇次回からは、今回作製したプロジェクトを異なる名前で保存して、回路設計、回路作成（上図作成）、Uploadして動作確認という流れを繰り返します。

◇では、次に基本的な回路をPLCで動かしてみましょう。

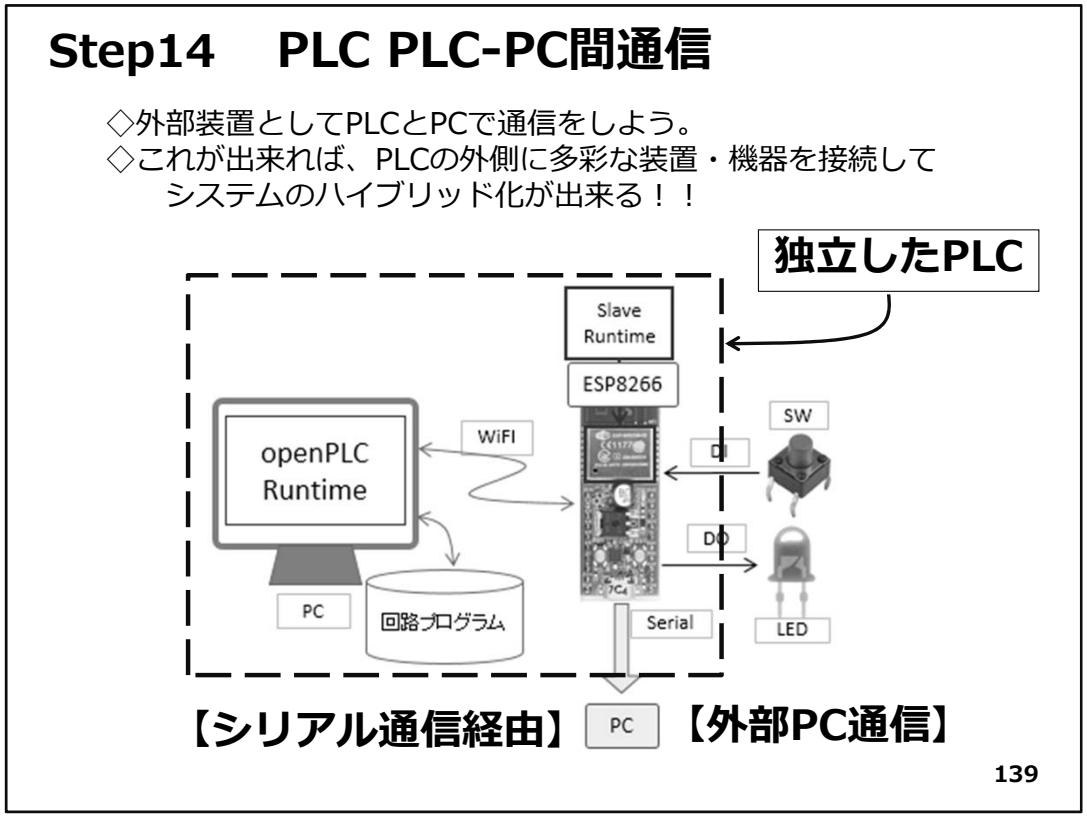
◇まず、WiFiマイコンをUSBケーブルでPCと接続して下さい。
そして、次の手順でPLCプログラムを稼動しましょう。

①. Windows メニューから OpenPLC→OpenPLC Runtimeを起動する。

②. ブラウザを起動し、URLに localhost:8080 と指定する。
③. ID、Password共に　openplcでログインする。
→OpenPLC Dashboard が開く。
④. Programで、前Stepで開発したPLCプログラムの【.st】ファイルをUploadし、Dashboard で Start PLC ボタンをクリックする。
→これで、WiFi経由でPCとWiFiマイコンが接続されてPLCプログラムが動いています。

◇動作確認を行います。
まず、WiFiマイコンをUSBケーブルでPCと接続して下さい。
そして、次の手順で動作を確認しましょう。

①SW1押下　→　LED点灯
②SW2押下　→　LED消灯
③SW1押下後、SW1を繰返し押下
④SW2を繰返し押下
⑤SW1押下後、SW1押下したまま、SW2を繰り返し押下
・・・
などの動作確認をして下さい。
意図したように(ラダー図で描いた回路の通りに)動作していますか？



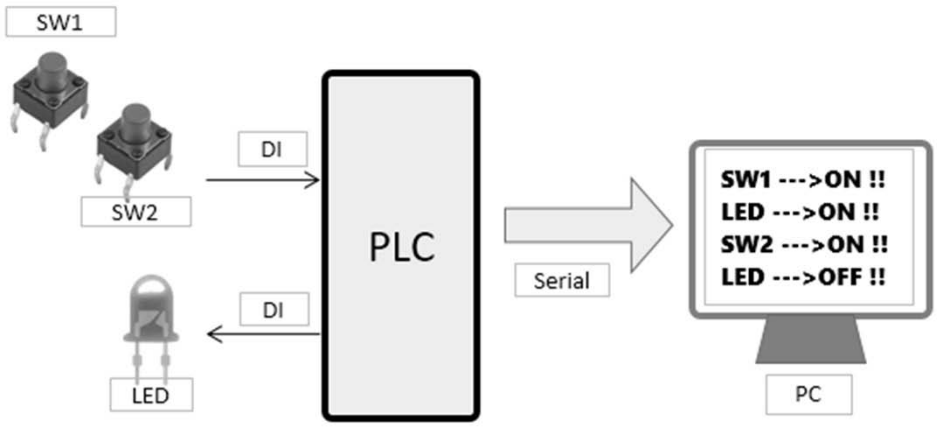
Step14　PLC PLC-PC間通信

◇システムのハイブリッド化

次に行うのは、図の破線（赤色）で囲んだ部分を、独立したPLCと考え、外部装置（PC）とシリアル通信を行います。これによって、PLCの外部にPCだけでなく、マイコンや、シリアルインターフェースを持つ様々な装置や機器の接続が可能になり、システムの拡張性が広がります。演習の初期に使用した無線マイコン（TWE-Lite）の応用で、この部分の無線化も可能になります。

システム概要

◇SWやLEDの状態変化をシリアル通信でPCに通知する！！



140

◇システム概要

前図を書き直し、仕様を整理すると、

①. SW1　を押下すると、LEDが点灯する

②. SW2　を押下すると、LEDが消灯する。

と、これは、前Stepで開発したPLCプログラムと同じ内容です。

【重要】

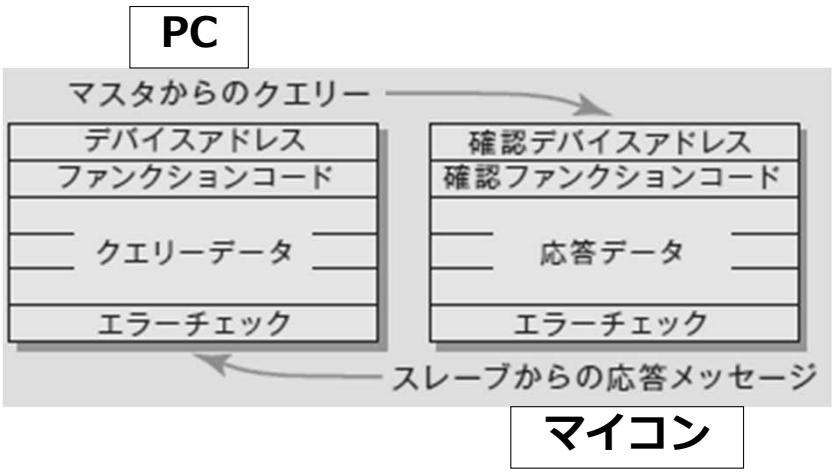
PLCプログラムと回路はそのまま利用します。

◇今回は、次の仕様を追加します。

【スレーブデバイス（ESP8266)内部でSW1、SW2、LEDの状態を検出して、上の動作を行いながら、PCにモニター用電文をシリアル通信で通知する】ことにします。

Modbus プロトコル

◇マスタ＝PC　：　スレーブ＝マイコン



141

◇Modbus （モドバス）プロトコル
通信プロトコルとしてModbus Protocol を実装したネットワークを、Modbus と呼んでいます。Modbus Protocolは、Modicon Inc（. AEG Schneider AutomationInternational S.A.S.）がPLC 用に開発した通信プロトコルで、その仕様が公開されています。
http://www.modicon.com/ で、そのプロトコル仕様書を閲覧できます。

◇OpenPLCは、本体がPCでI/OユニットがWiFiマイコンと説明してきましたが、PCをマスタ、WiFiマイコンをスレーブとして、Modbus Protocolを Ethernet で接続した Modbus/TCP で通信を行っています。そのためにOpenPLCの設定でスレーブデバイスを追加する際に、マイコンのIPアドレスを入力するのです。

◇Modbus Protocolではマスタだけが通信の開始（クエリと呼んでいます）を発行出来て、スレーブデバイスはそれに対する応答メッセージを返す、という通信を繰り返します。マスタはPCで、スレーブデバイスはWiFiマイコンです。

◇具体的な通信内容がどうなっているかは、マイコン側Runtime のソースコードとプロトコル仕様書を比較しながら見ると、良く分かります。

◇IDEでRuntimeのソースを見ると、unsigned char modbus_buffer[100]; と宣言している変数があります。これが上図のマスターからのクエリ―とスレーブからの応答をやり取りするバッファ配列で、この中身が電文そのものになっています。

◇ loop()関数内にある下記記述部分のコメント記号（/* */）をはずせば、PCから送られてきた電文を、シリアルモニタで見ることができます。

```
//DEBUG
/*
Serial.print("Received MB frame: ");
PrintHex(modbus_buffer, i);
*/
```

◇さらに、ソースコードには、

```
extern bool mb_discrete_input[MAX_DISCRETE_INPUT];
extern bool mb_coils[MAX_COILS];
```

という宣言箇所があります。実体は、modbus.h で宣言されていますが、この配列変数は、それぞれDIとDOの状態、すなわちSWとLEDの状態を bool型【True/False（真/偽）】で表しています。この配列は、先頭から順に%IX100.0、%IX100.1・・・、%QX100.0、%QX100.1・・・に対応しているので、

SW1→mb_discrete_input[2]
SW2→mb_discrete_input[0]
LED→mb_coils[0]

で調べることができます。これを利用してソースコードを書いたものを次頁以後に示します。

WiFiマイコン内 Runtime の改造①

◇ESP8266 内部の Runtime プログラムを改造します。

ソースコードの初めの方に変数宣言を追加します。

```
unsigned char modbus_buffer[100];
bool sw[4];   //<--- for Step14: SW condition
bool led[4];  //<--- for Step14: LED condition

int processModbusMessage(unsigned char *buffer, int bufferSize);

#include "modbus.h"
```

// 変数定義の最後に入力と出力の状態を保存する配列を宣言

142

◇変数宣言の最後に以下の2行を追加

bool sw[4]; //<--- for Step14: SW condition　　SW（DI）の状態を保存する配列
bool led[4]; //<--- for Step14: LED condition　　LED（DO)の状態を保存する配列

WiFiマイコン内 Runtime の改造②

◇loop()関数内の client.write() の下に、下記を追加する。

```
//DEBUG
/*
Serial.print("Received MB frame: ");
PrintHex(modbus_buffer, i);
*/

updateIO();
unsigned int return_length = processModbusMessage(modbus_buffer, i);
client.write((const uint8_t *)modbus_buffer, return_length);

//--- for Step14 : Check DI(SW1, SW2)
//--- for Step14 : Check DI(SW1, SW2)
//--- for SW1
if ( (sw[2]!=mb_discrete_input[2]) && (mb_discrete_input[2]==true) )
{
  Serial.println("1: SW1--->ON!!");
  sw[2] = true;
}
else if ( (sw[2]!=mb_discrete_input[2]) && (mb_discrete_input[2]==false) )
{
  Serial.println("2: SW1--->OFF!!");
  sw[2] = false;
}
```

// sw1の状態が変化した場合該当するメッセージを
PCにシリアル通信で出力
※メッセージの出力箇所が判断できるよう、番号を付けておく

◇以下同様に、SW2とLEDい対応する処理を追加する。
ソースコードが出来たら、名前を付けて保存し
WiFiマイコンに書き込む。書込み手順はStep6参照。

143

◇SW1の状態を調べて、変化したら該当するメッセージを、シリアル通信経由でPCに送信します。

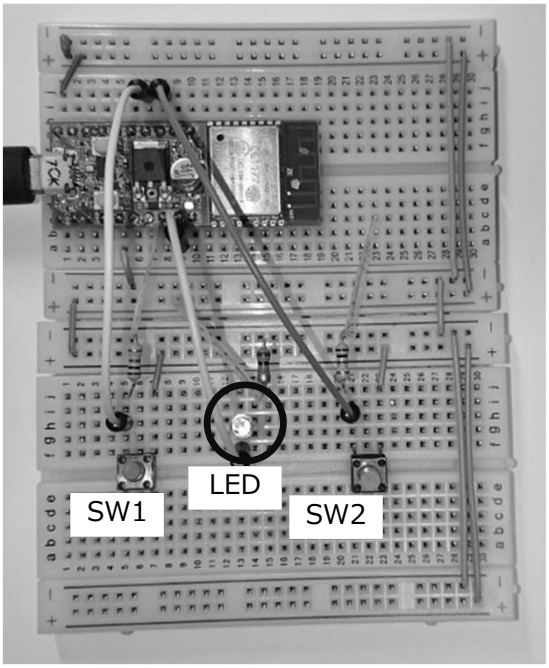
◇loop()関数内の、client.write(・・・・・); の下に以下を追加

```
//--- for Step14 : Check DI(SW1, SW2)
//--- for Step14 : Check DI(SW1, SW2)
//--- for SW1  SW1の状態が以前と変化したら該当メッセージを、シリアル
  通信経由でPC送信し最新の状態を配列に保存する

    if ( (sw[2]!=mb_discrete_input[2]) && (mb_discrete_input[2]==true) )
       //---SW2はDIの3番目を使用に変更したので、配列の[2]を使う
    {
      Serial.println("1: SW1--->ON!!");
      sw[2] = true;
    }
    else if ( (sw[2]!=mb_discrete_input[2]) && (mb_discrete_input[2]==false) )
    {
      Serial.println("2: SW1--->OFF!!");
      sw[2] = false;
    }

    //--- for SW2  SW2の状態が以前と変化したら該当メッセージを、シリアル
```

```
    通信経由でPC送信し最新の状態を配列に保存する
 if ( (sw[1]!=mb_discrete_input[1]) && (mb_discrete_input[1]==true) )
 {
   Serial.println("3:              SW2--->ON!!");
   sw[1] = true;
 }
 else if ( (sw[1]!=mb_discrete_input[1]) && (mb_discrete_input[1]==false) )
 {
   Serial.println("4:              SW2--->OFF!!");
   sw[1] = false;
 }

   //--- for LED   LEDの状態が以前と変化したら該当メッセージを、
      シリアル通信経由でPC送信し最新の状態を配列に保存する
  if ( (led[0]!=mb_coils[0]) && (mb_coils[0]==true) )
 {
   Serial.println("5:                        LED--->ON!!");
   led[0] = true;
 }
 else if ( (led[0]!=mb_coils[0]) && (mb_coils[0]==false) )
 {
   Serial.println("6:                        LED--->OFF!!");
   led[0] = false;
 }
 //--- for Step14 : Check DI(SW1, SW2)
 //--- for Step14 : Check DI(SW1, SW2)
```

◇ソースコードが完成したら、名前を付けて保存しWiFiマイコンに書き込みます。書込み手順は、Step6以降で繰り返していますが、不安がある様でしたらStep6を参照してください。

動作確認



144

◇実際に配線が済んだ様子を写真に示します。(再掲)

◇動作確認を行います。WiFiマイコンは、改造Runtime書込みの為に、USBケーブルで PCと接続されていると思います。その状態で、次の手順でPLCプログラムを稼動しましょう。

①. Windows メニューから OpenPLC→OpenPLC Runtimeを起動する
②. ブラウザを起動し、URLに localhost:8080 と指定する
③. ID、Password共に openplcでログイン
→OpenPLC Dashboard が開く
④. Programで、前Stepで開発したPLCプログラムの【.st】ファイルをUploadし、Dashboard で Start PLC ボタンをクリックする。
--これで、WiFi経由でPCとWiFiマイコンが接続されます

◇次に、Arduino IDEを起動します。
(改造Runtimeを書込んだので、既に起動されていると思います。)

◇そして、シリアルモニター(IDEの右上:虫眼鏡マーク)を起動します。

シリアルモニターに注意しながら、

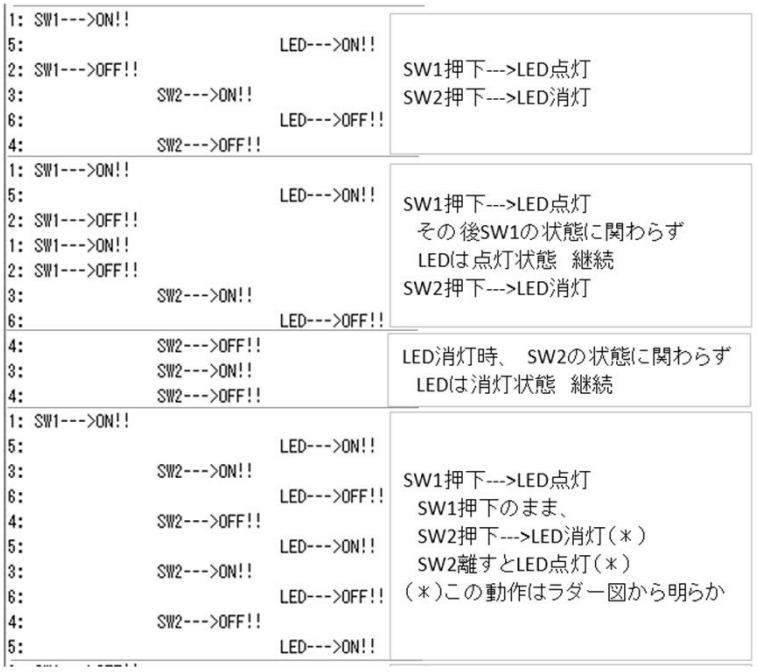
①SW1押下
②SW2押下
③SW1押下後、SW1を繰返し押下

④SW2を繰返し押下
⑤SW1押下後、SW1押下したまま、SW2を繰り返し押下
・・・
などの動作確認をして下さい。
意図したように(ラダー図で描いた回路の通りに)動作するはずです。
※前Stepで確認済みですから。

◇ではシリアルモニターの表示はどうなっているでしょうか？

シリアルモニターの様子

◇シリアルモニター表示は、どにょうに稼動するかを観察できる！！



145

◇シリアルモニタに表示される、メッセージの一部を図に示します。

◇動作の様子が分かりますね。

【本題】

忘れないうちに本題に戻っておきましょう！！

このStepのテーマは、PLCとPC間通信でした。 OpenPLCは、PCとWiFiマイコンが協調してPLCを構成しているのでした。

そこで、PCを別に準備せずに、PC内の別タスクとしてArduinoIDEのシリアルモニターを稼動させ、**それを外部PCと位置付けて**シリアル通信を行いました。

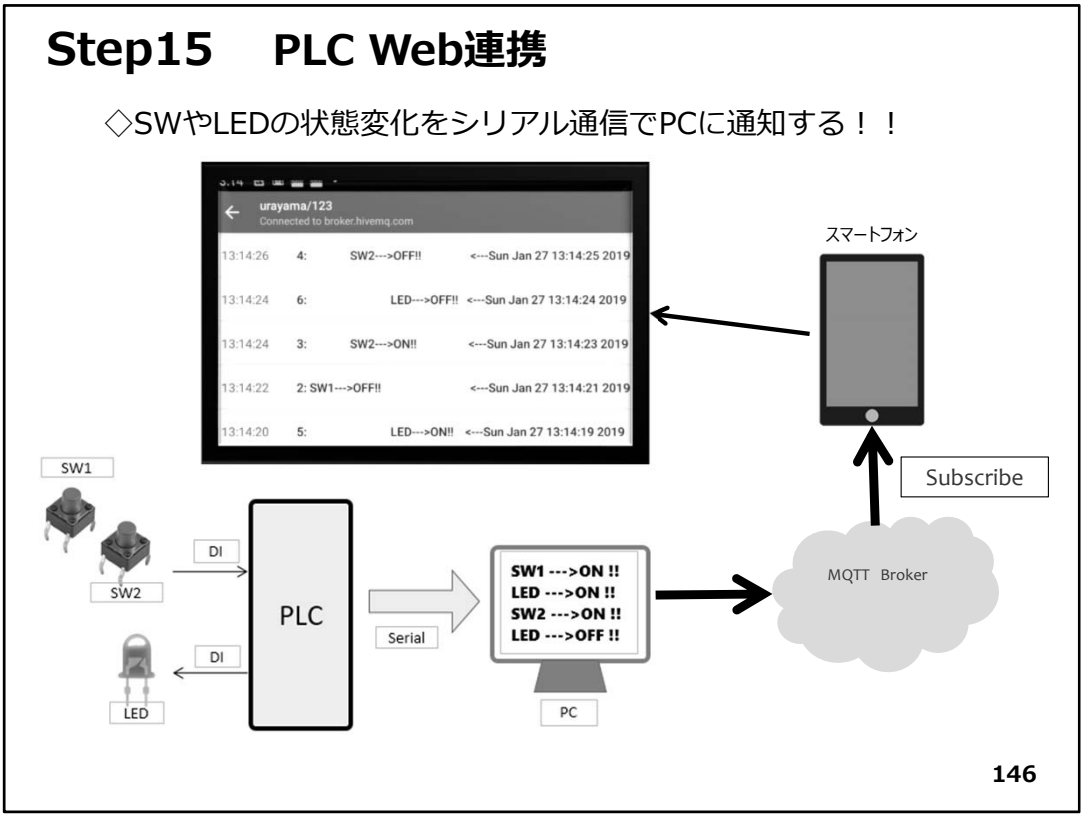
★PCに接続しているUSBコネクタを外し、例えば、隣の方のPCに接続し、そのPCでシリアルモニターを開けば、本当の外部PC接続ができていることが分かるでしょう。

◇実際のPLCは、ラダー図だけで外部通信ができるように、【色々な命令をラダー図内に描ける※】ようになっています。内蔵シリアル通信ポート以外にも、拡張通信ユニットが追加接続できるような設計になっていて、複数チャンネルの通信ができます。

※メーカーごとに仕様は異なるものの、外部とのシリアル通信やLAN通信ができるような機能ブロック(ファンクションブロックという)や命令が、ラダー図で描けるようになっています。

【重要】

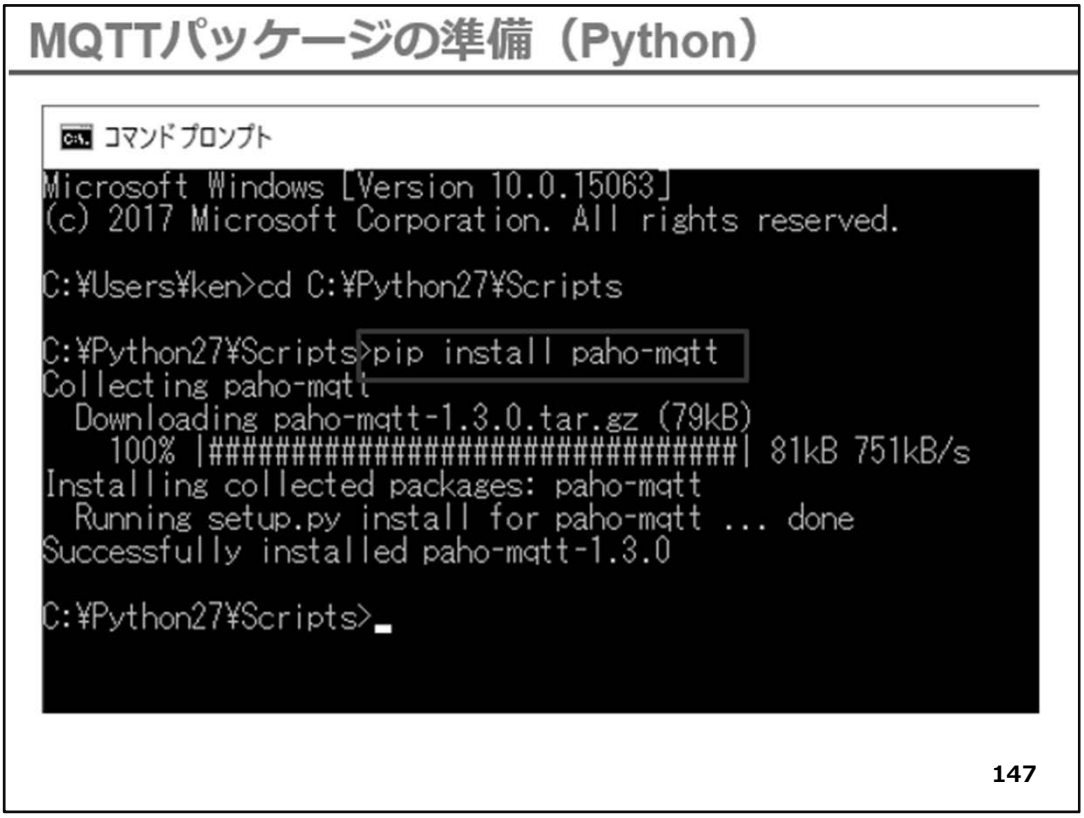
ここで改造したWiFiマイコンRuntimeは、次のStepで使用しますので、そのまま進めてください。



Step15　PLC Web連携

◇演習編最後のテーマは、前Stepの成果をさらに拡張して、PCで受信したメッセージをWebサービス（MQTTブローカー）に送り、スマートフォンやタブレットで受信してみましょう。

◇最終ステップは既に現場に導入されているPLCのプログラムに少し機能を追加することで、Web連携システムを容易に構築できるサンプルとなります。



◇Pythonプログラムで【MQTTパッケージ】を使用しますので、あらかじめインストールしておきます。

◇コマンドプロンプトを起動して、図の様に

pip install paho-mqtt

と入力してEnterキーを押下します。

◇ダウンロード、インストールが行われて、Successfully～の文字が表示されます。この時、Internetに接続できる状態にしておいてください。

ソースコード　※拡張子【.py】を付けて保存！

```
import paho.mqtt.client as mqtt   # MQTT パッケージ
import serial     # シリアル通信パッケージ
import time       # 日付・時刻を取り扱う


s = serial.Serial('COM10', 115200)  # COM10を開く

host = 'broker.hivemq.com'      # 利用するブローカー
port = 1883                     # 接続するポート番号
topic = 'urayama/123'           # MQTT Topic name

client = mqtt.Client()
client.connect(host, port=port, keepalive=60)   # mqtt broker 接続

n = 100    #繰返し数設定

while 1:   # ずっと繰り返し
    buff = s.readline()         # PLCから1行のメッセージ受信

    # mqtt 用メッセージを作製（topicの内容）
    #受信したメッセージに空白を追加
    msg = buff.replace('¥n','') + "                                                    "
    msg = msg[0:45] + " <---" + time.ctime()   #メッセージ先頭から45文字切取る
    print msg                                  #PCコンソールにメッセージ内容を表示
    client.publish(topic, msg)   # mqtt broker にtopic を 発行

    n -= 1     #残り回数更新
    if n < 0:  #終わりか？
      break    #終わりなので、ループらか抜け出す

s.close()     # COMを閉じる
```

148

◇ソースコードは、全体で20行ほどしかありませんが、これでWebサービスを利用できてしまいます。各行の内容は、コメントを参照していただければ理解できると思いますが、上から順に説明します。

・importで始まる3行は、それぞれがこのプログラムで使用するライブラリを取り込んでいます。

・シリアルポートは、USBケーブルでWiFiマイコンを接続をした際に設定されるCOMポート番号を記述します。リストはCOM10となっていますが、各自のPCで確認した番号に変更します。

・今回は　broker.hivemq.com を使用しています。　※他にも多くのMQTTブローカーがあります。調べてみてください。

・MQTTサービスと接続するポート番号は、1883固定です。

・topic は、トピック名文字列です。あらかじめメッセージの発行者と購読者間で取り決めをしておきます。トピック名は、【/】スラッシュで階層を設定できます。

・あらかじめ client オブジェクトを作成してからMQTTブローカーに接続します。このプログラムでは、MQTTブローカーに接続したまま連続使用していますが、メッセージ発行の都度、接続・切断を繰返し行う方法もあります。

・無限ループで処理を繰り返す場合、プログラムを強制終了させると、COMポート

のCloseが行われず、USBケーブルの抜差しによって、仮想COMポートの解放を行う必要があり、煩雑なので、一定の回数（ここでは100回にしました。）繰り返したら、繰返しを抜けてCOMポートをCloseするようにしました。
この部分も、毎回、COMポートを開く処理と閉じる処理を、ペアで行うようにした方が、使い易くなるかもしれません。

・whileループの中では、
PLCから1行のメッセージを受信して、その最後にある改行コードを取り除き空白文字列を付加します。
そこから45文字取り出して、日付時刻を付加したメッセージにします。
メッセージを、PCのPythonシェルコンソールに表示して、同じ内容をMQTTブローカーに送ります。（メッセージ発行）
指定の回数（ここでは100回）終わった場合は、ループを抜けます。

・ループから抜け出したら、COMポートを閉じます。

ソースコードを入力したら、拡張子に【.py】として保存してください。

◇ソースコード全体を示します。
※ソースファイルは、演習キット付属のCDに収録してあります。

```
#---------------------------------
#--- 2019.01.23
#--- PLCから受信した電文を表示する
#--- 同時に MQTT でメッセージを送る
#---------------------------------
#--- 演習 Step15
#---------------------------------


import paho.mqtt.client as mqtt   # MQTT パッケージ
import serial     # シリアル通信パッケージ
import time       # 日付・時刻を取り扱う


s = serial.Serial('COM10', 115200)  # COM10を開く

host = 'broker.hivemq.com'     # 利用するブローカー
port = 1883                    # 接続するポート番号
topic = 'urayama/123'          # MQTT Topic name

client = mqtt.Client()
client.connect(host, port=port, keepalive=60)   # mqtt broker 接続

n = 100    #繰返し数設定
```

```
while 1:   # ずっと繰り返し
    buff = s.readline()        # PLCから1行のメッセージ受信

    # mqtt 用メッセージを作製（topicの内容）
    #受信したメッセージに空白を追加
    msg = buff.replace('¥n','') + "                                    "
    msg = msg[0:45] + " <---" + time.ctime()   #メッセージ先頭から45文字切取る
    print msg                                  #PCコンソールにメッセージ内容を表示
    client.publish(topic, msg)   # mqtt broker にtopic を 発行

    n -= 1     #残り回数更新
    if n < 0:  #終わりか？
      break    #終わりなので、ループから抜け出す

s.close()     # COMを閉じる
```

【.py】と拡張子を付けて、保存するのを忘れないでください。

動作確認

◇次の手順で動作確認をします。

【PLC開始】

①．PLCの回路（前Stepのもの）をUSBケーブルでPCと接続。
②．OpenPLC Runtimeを起動。
③．ブラウザでhttp://localhost:8080をアクセスしてログイン。
④．Start PLCボタンをクリック→PLC Run

【Python実行】

⑤．IDLEを起動（Python）し、ソースファイルを開く。
⑥．Run →Run Module でPythonプログラムを実行。

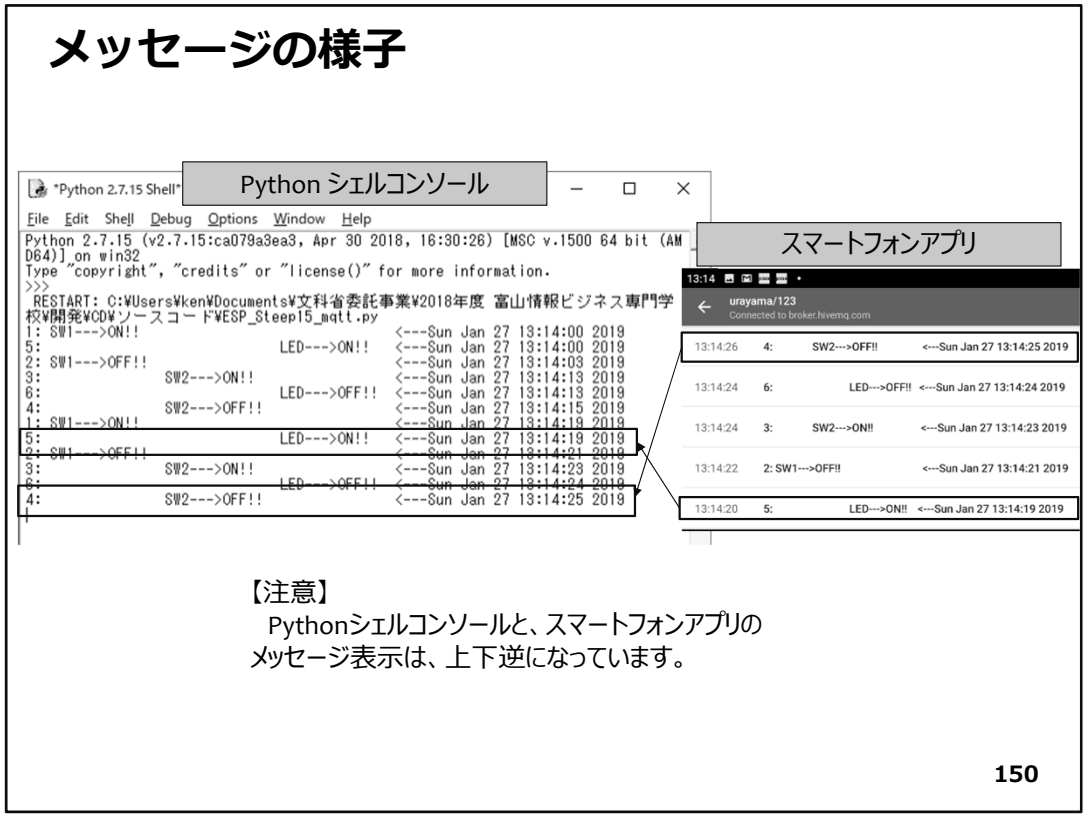
【スマートフォンアプリ起動】

⑦．アプリを起動して、MQTTブローカーに接続
⑧．ソースコードで指定したトピック名でSubsribe（購読）する。

【動作確認】

⑨SW1,2を操作し、Pythonシェルとスマートフォンアプリで
メッセージを観察します。

149



◇メッセージの様子

Pythonシェルコンソールは、新しいメッセージが下に追加されていきますが、スマートフォンアプリのMQTT Dashboardは、新しいメッセージが最上段に追加されて、古いメッセージは下の方に下がっていきます。付き合わせる際は、上下逆さまなので注意して下さい。

PLCのシステムが複数ある場合は、メッセージが多数表示されていきます。それを観察していると、現場でデータを取得すると、直ぐに Big Data化するのが理解できます。混在するメッセージの中から特定のシステムのデータだけ見たい場合は、トピック名を分けるか、または、発行するメッセージに、発行元IDなどを付加しておけば良いでしょう。

◇今後の為に

発行元IDが付加されたメッセージを発行すれば、この演習で初めのころに経験したAccessなどが使えます。データベースアプリケーションを開発して、受信したメッセージを記録しておきます。IDを指定すれば容易にデータの選別ができます。

ここでは、PLCとしてWiFiマイコンを利用しましたが、実際にWebサービスに接続しているのはPCでした。しかしWiFiマイコン（ESP8266)は、直接Webサービスにアクセスできることを演習したので、Step12辺りをもう一度振り返り、現場で使用出来る簡単なIoTデバイスを作ってみて下さい。また、マイコンでメッセージを発行するだけでなく、メッセージを受信することもできるので、装置の遠隔制御などにも活用できます。さらに、ここで使用したMQTTは、災害などの際に利用される伝言サービスなどとよく似ていますので利用方法を巧く考えると、グループや家族間の緊急連絡などに活用できます。

◇最終ステップは、とても難しい内容だったと思います。既にあるPLCの為のRuntimeを解読して、それを改造して動かすなど、かなり高級な事を行いましたが、仕事としてはよくある内容です。大変お疲れさまでした。

☆これで【演習編】は終了です。

お疲れさまでした！！

終わり

151

Appendix【基礎演習】

Appendix

A:生産管理データベースの作り方

B:パーツの使い方・注意点

C:抵抗器のカラーコード

D:A/D 変換と温度センサー

E:Python 開発環境の準備

F:ソースコード(Step5)

G:MQTT アプリの利用

H:MQTT ライブラリの準備

I:PLCOpen の準備

J:PLCOpen の使い方

A:生産管理データベースの作り方

Access2013 を使用して生産管理データベースを作ります。以下の**手順をトレース**してデータベースを構築して下さい。

１．空のデスクトップデータベースを作る

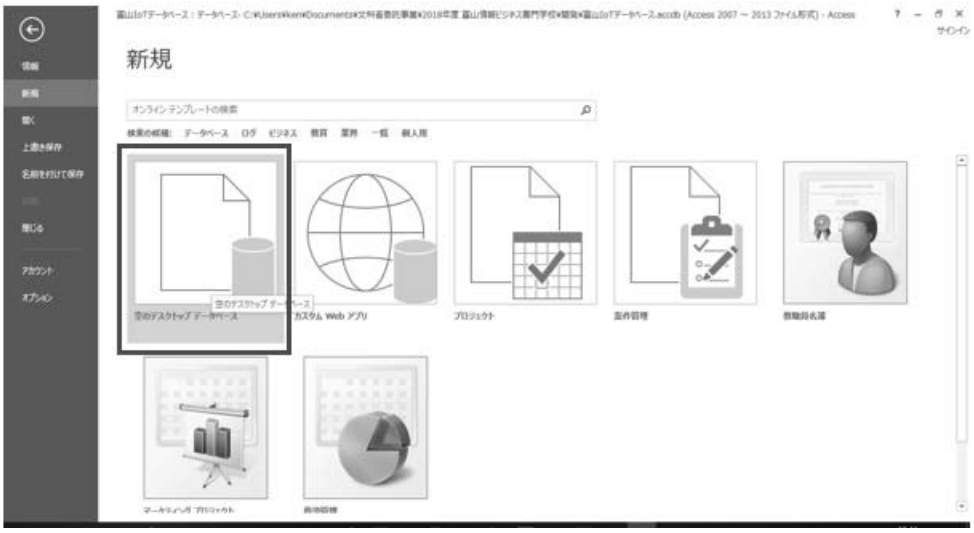


図　1

Access を起動すると上図が表示されます。左上の**【空のデスクトップデータベース】**（赤枠）を選択します。

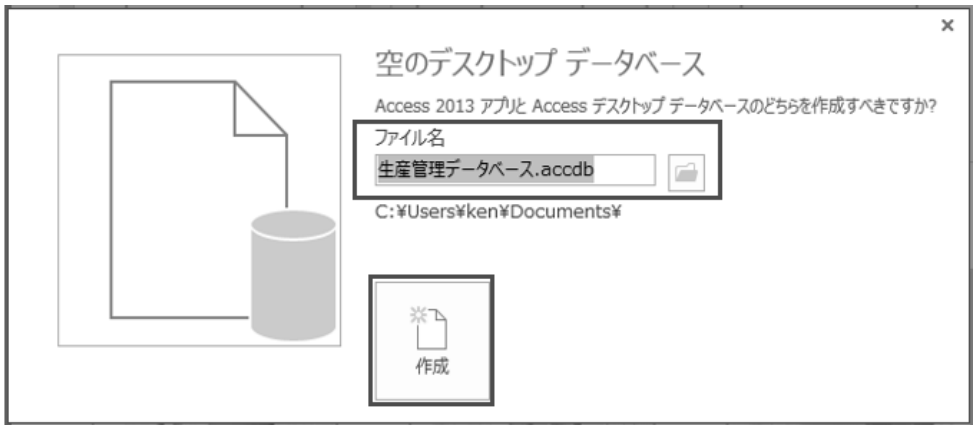


図　2

保存に適当なフォルダを選び、ファイル名（データベースファイル名）を入力して、**【作成】を押下**します。データベースファイルが保存されて、**テーブルデザインウインドウが表示されます。**

２．生産計画テーブルとフィールドの作成

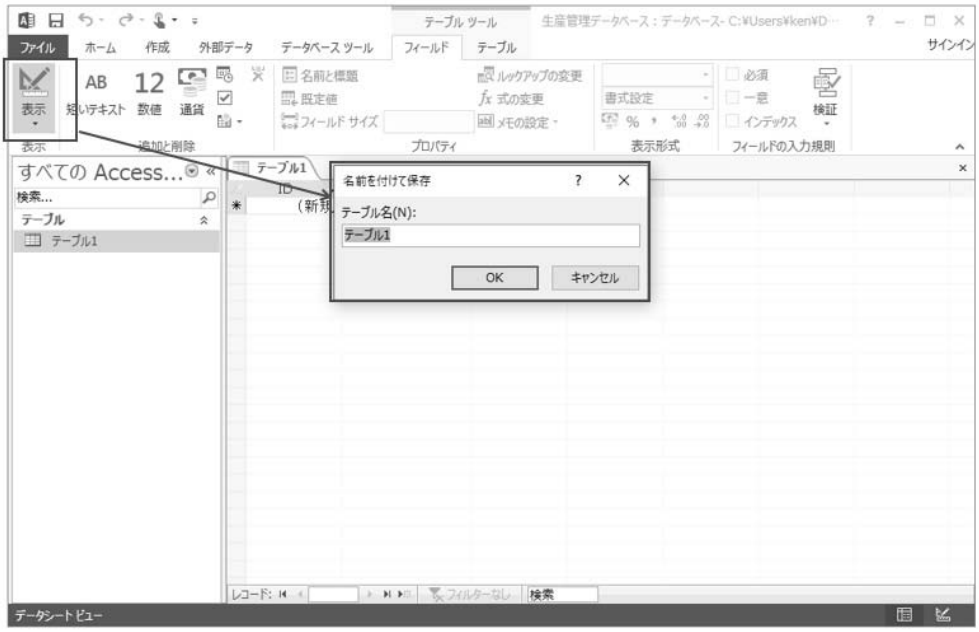


図　3

上図左上の表示にある**定規と鉛筆マークのボタン**を押すか、左側の【テーブル】の下に表示されている【テーブル 1】を**右クリックして【デザインビュー】を選択**すると名前を付けて保存ウインドウが表示されるので、**【生産計画】と入力して OK ボタン**を押します。これで生産計画テーブルが出来ました。続けて、テーブル内のフィールドを作成します。フィールド名 1 行目の ID と表示された場所を**【計画番号】**に変更して、データ型を**【短いテキスト】**に設定し、フィールドの内容が分かるように【説明】を入力します。**（説明は空欄でもかまいません。）**

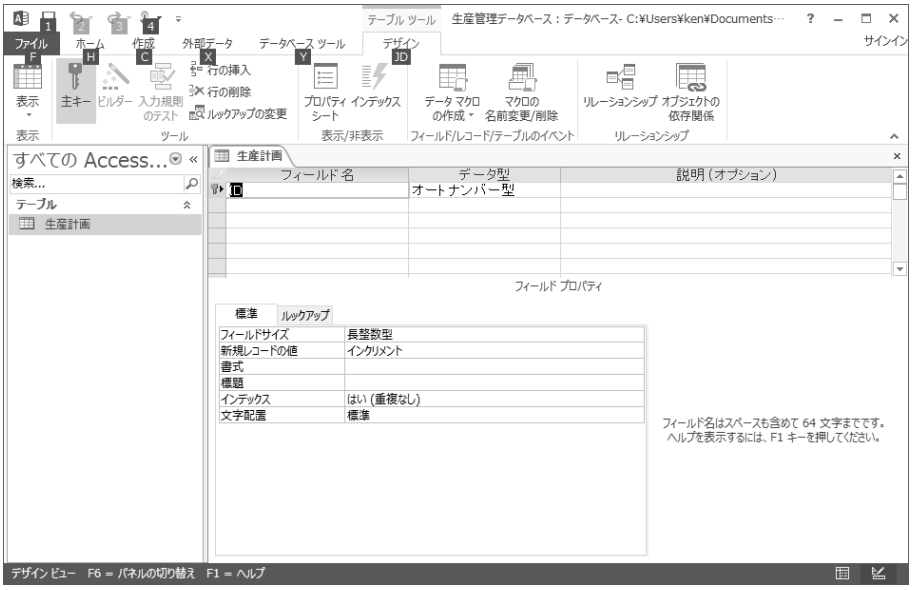


図 4

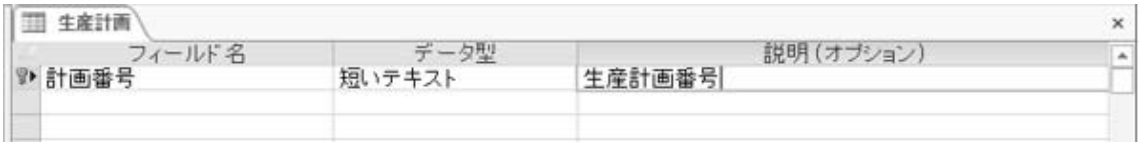


図 5

図 5 の様に**計画番号フィールド**が設定できました。このフィールドの先頭にある**鍵のマークは【主キー】**と言い、このテーブルの中の**1 レコードを計画番号だけで特定できる（これを一意に選択できると言う）**フィールドであることを示しています。主キーの設定は、図 4 左上の**主キーボタンで設定・解除できます**。

同様に他のフィールドも以下の様に設定します。

図　6

設定が済んだら右上の×マークをクリックして保存します。

これで、**【生産計画テーブル】が完成**しました。

次に生産実績テーブルを作ります。

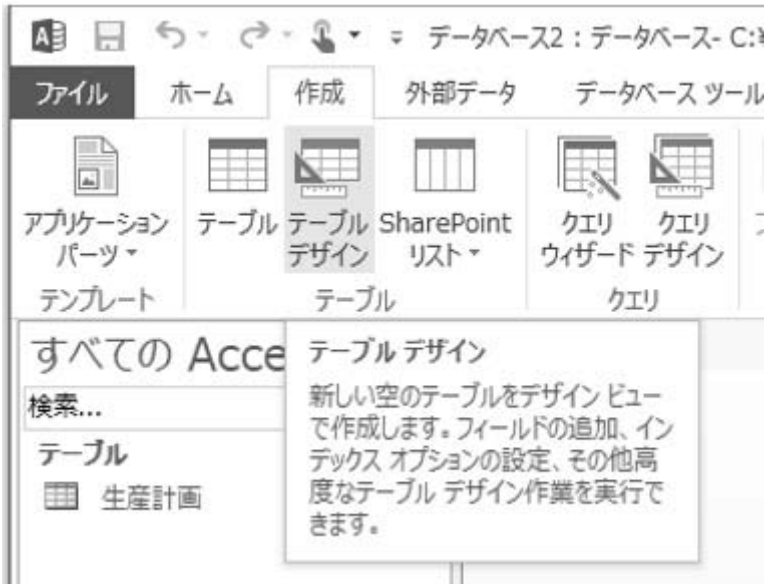


図　7

上部の**【作成タブ】→【テーブルデザイン】**と選択すると、新しいテーブルが作られるので、上と同様にフィールドを設定します。

図 8

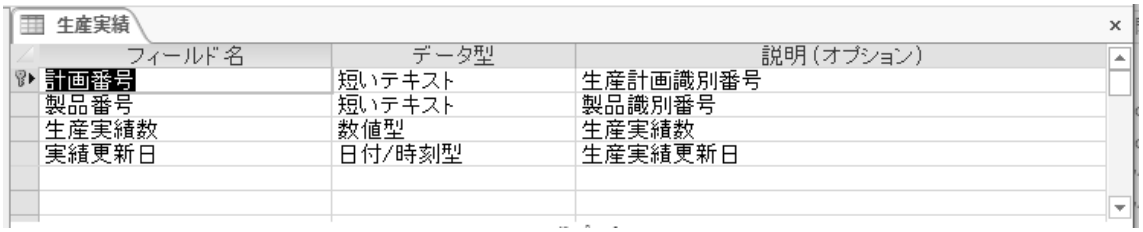


図 9

4つのフィールドを設定し、右上の×マークをクリックして、**【生産実績】**という名称で保存します。**計画番号の主キーは、ウインドウ左上の主キーボタンで設定します。**

※**【生産計画】**テーブルと**【生産実績】**テーブルを作成する手順が少し違いますが、どちらの手順で作ってもかまいません。

図 10

テーブル名を右クリック（上図）して表示されるメニューで、

①.【開く】でレコードデータの入力・編集※

②.【デザインビュー】で、テーブルのフィールド設定変更

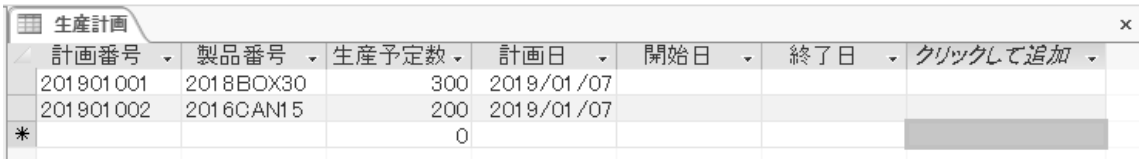
③.【名前の変更】でテーブル名の変更

④.【削除】でテーブル削除ができます。

※テーブル名をダブルクリックすれば①と同様にテーブルが開きます。

3．データ入力・編集

(1).【生産計画】データを次の様に 2 レコード作成します。



生産計画

計画番号	製品番号	生産予定数	計画日	開始日	終了日	クリックして追加
201901001	2018BOX30	300	2019/01/07			
201901002	2016CAN15	200	2019/01/07			
*		0				

図 11

※**【全て半角】で入力**します。各フィールド入力後 Enter キー押下。

生産計画テーブルを開き、1 行目の計画番号から順に、

①. 計画番号 ＝ 201901001 …2019 年 1 月の 1 番目の計画

②. 製品番号 ＝ 2018BOX30 …2018 年設計、金属箱 30 番

③. 生産予定数 ＝ 10 …10 個生産

④. 計画日 ＝ 2019/1/7 …計画作成日

※日付は西暦で記録する。

⑤. 開始日 ＝ 空（Null）

⑥. 終了日 ＝ 空（Null）

と入力します。

続けて 2 レコード目にも、

①. 計画番号 ＝ 201901002 …2019 年 1 月の 2 番目の計画

②. 製品番号 ＝ 2016CAN15 …2016 年設計、金属缶 15 番

③. 生産予定数 ＝ 20 …20 個生産

④. 計画日 ＝ 2019/1/7 …計画作成日

※日付は西暦で記録する。

⑤. 開始日 ＝ 空（Null）

⑥. 終了日 ＝ 空（Null）

と入力します。

2 件の生産計画ができました。テーブルを閉じておきます。

(2).**【生産実績】は、まだ入力せず、空の状態**にしておきます。

４．画面（フォーム）作成

Access では、画面のことを**フォームと呼びます**。フォームを作成するには、まず左側ペイン（ウインドウの左区画）で生産計画テーブルを選択します（下図）。次に作成タブのフォームボタンをクリックします。

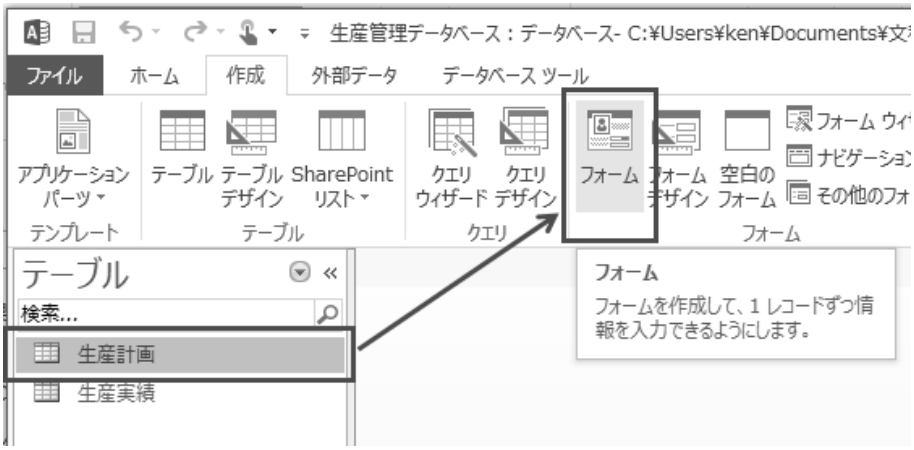


図　12

テーブルと同名のフォームが作成されます。

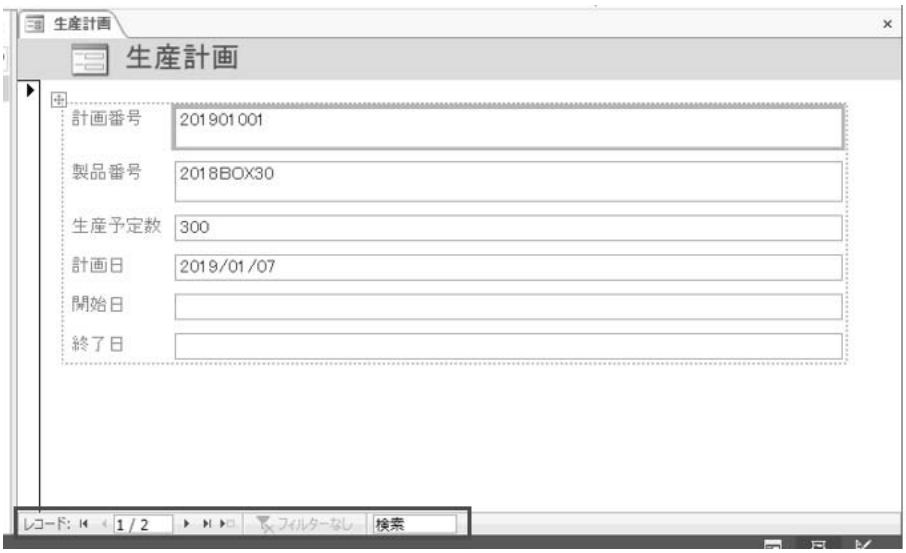


図　13

上図下部の**【レコードと書いてある部分】**で、テーブル内を自由に移動でき、そのレコード内容がフォームに表示されています。また、同じ個所の右側には、**【検索フィールド】**もあり、データの内容を入力して特定レコードを検索する事も可能です…が、今はまだ2レコードしかないので、検索の必要は有りませんね。このフォームは、あたかも出来上がっているかのように見えますが、**まだ保存されていないので、フォームの名前を付けて保存します。**

生産計画フォームの右上の×マークをクリックして、メッセージ（下図）の【はい】と答えます。

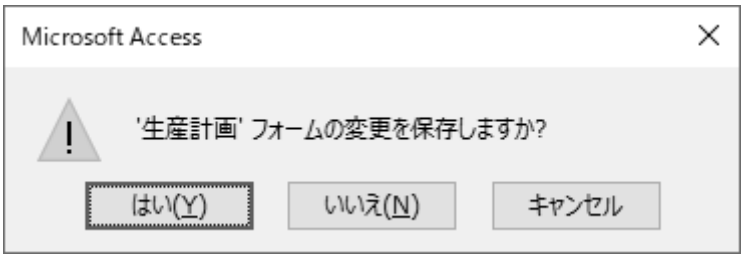


図　14

フォームの名称を変更することもできますが、ここでは、そのままOKボタンをクリックします。

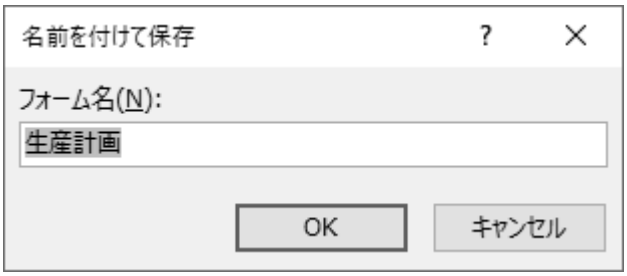


図　15

今後の為に、生産実績フォームも同じようにして作っておきます。**生産実績テーブルを選択して、作成タブのフォームをクリックすると、生産実績フォームが作られます。**

図　16

右上の×マークをクリックして、生産実績フォームを保存します。

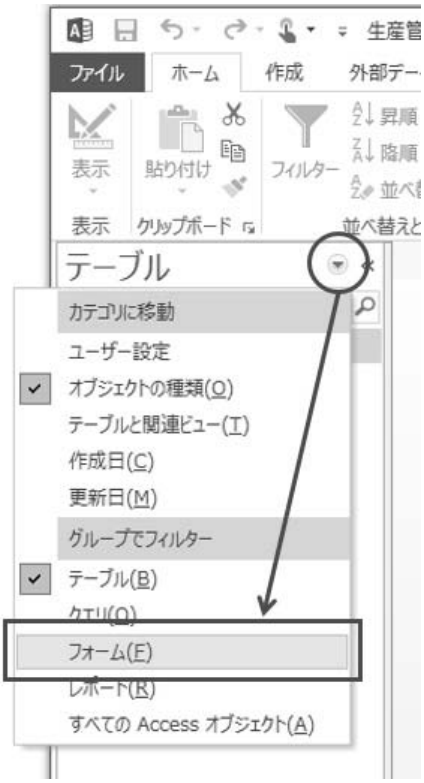


図　17

テーブルの右側の下向き▼をクリックしてプルダウンからフォームを選択すると、2つのフォームが出来ています。

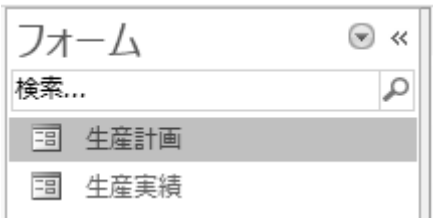


図　18

５．ボタンの追加

フォームには、ボタンを作ることができます。そのボタンには、Excel VBA で説明したように**【プログラムを割り当て】**て、ボタンを押すと、手作業では面倒な仕事を行わせることができます。

ここでは、次のようなボタンを作ります。

①．**生産開始ボタン**：現在、生産計画は有りますが、生産実績は空のままです。生産開始時に生産実績レコードが生産実績数 0 で作成されます。その仕事を行うボタンを作ります。

②．**実績更新ボタン**：生産が進むにつれて、徐々に生産実績数が増えて行くので、ボタンを押せば生産実績数が 1 ずつ加算されるようなボタンを作ります。

図　19

図 20

ここで、ボタンを作る**生産計画フォームを右クリックして、デザインビューを選択**します。右下部分が生産計画フォームのデザインビューに変わります。ウインドウ右側の垂直のスクロールバーを下にスライドして、フォームフッター部分が広く見えるようにします。

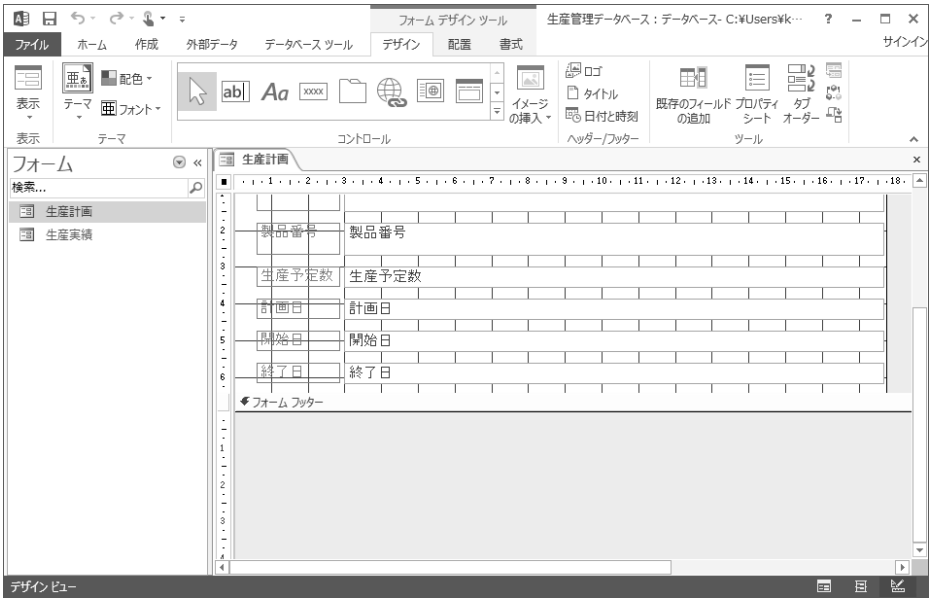


図　21

図　22

フォームデザインツールにあるボタン（上図赤枠）をクリックして、フォームフッター部に適当な大きさの四角形を描くと、【コマンド XX】ボタンができます（XX は数字）。ウインドウ上部、ツールにあるプロパティシート（下図赤枠）をクリックします。


フォーム デザイン ツール
生産管理データベース : データベース- C:¥Users¥k…
部データ
データベース ツール
デザイン
配置
書式
サインイン
ab
Aa
xxxx
イメージの挿入
ロゴ
タイトル
日付と時刻
ヘッダー/フッター
既存のフィールドの追加
プロパティシート
タブオーダー
コントロール
ツール
生産計画
製品番号
製品番号
生産予定数
生産予定数
計画日
計画日
開始日
開始日
終了日
終了日
フォーム フッター
コマンド20

図　23

図 24

今、**コマンドボタン（以後は、ボタンと言う）**が選択されているので、ボタンのプロパティが表示されます。このプロパティシートの【すべて】タブを選択して、以下の内容を変更します。

①．名前：　生産開始

②．標題：　生産開始

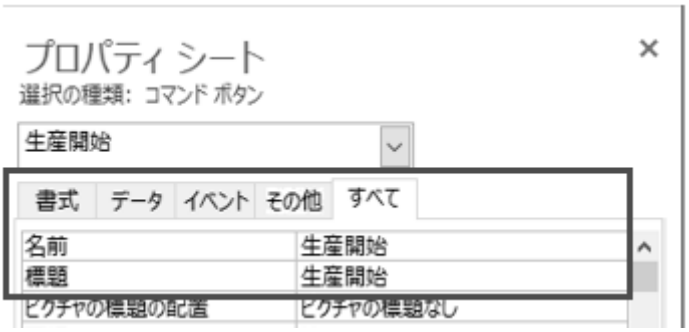


図 25

次に、**イベントタブを選択**して、**【クリック時】**のプルダウンから**【イベントプロシージャ】を選択**します。（下図）

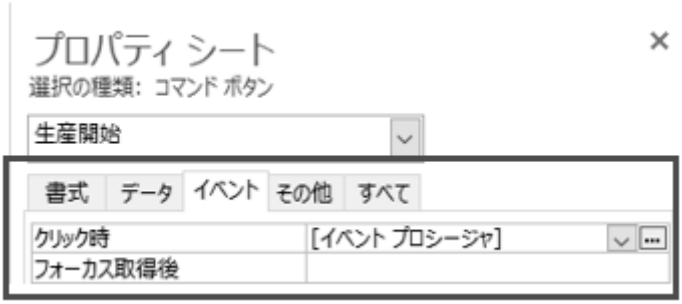


図 26

イベントプロシージャを設定したら、**プルダウン右側の…ボタン【ビルドボタン】**をクリックします。すると、Access 用の VBA ウインドウが開き、そこにはすでに生産開始ボタンのクリック時イベントを処理する関数【生産開始_Click()】の宣言だけの関数フレームがあり、その中に VBA のソースコードを書けば、ボタンが機能するようになっています。

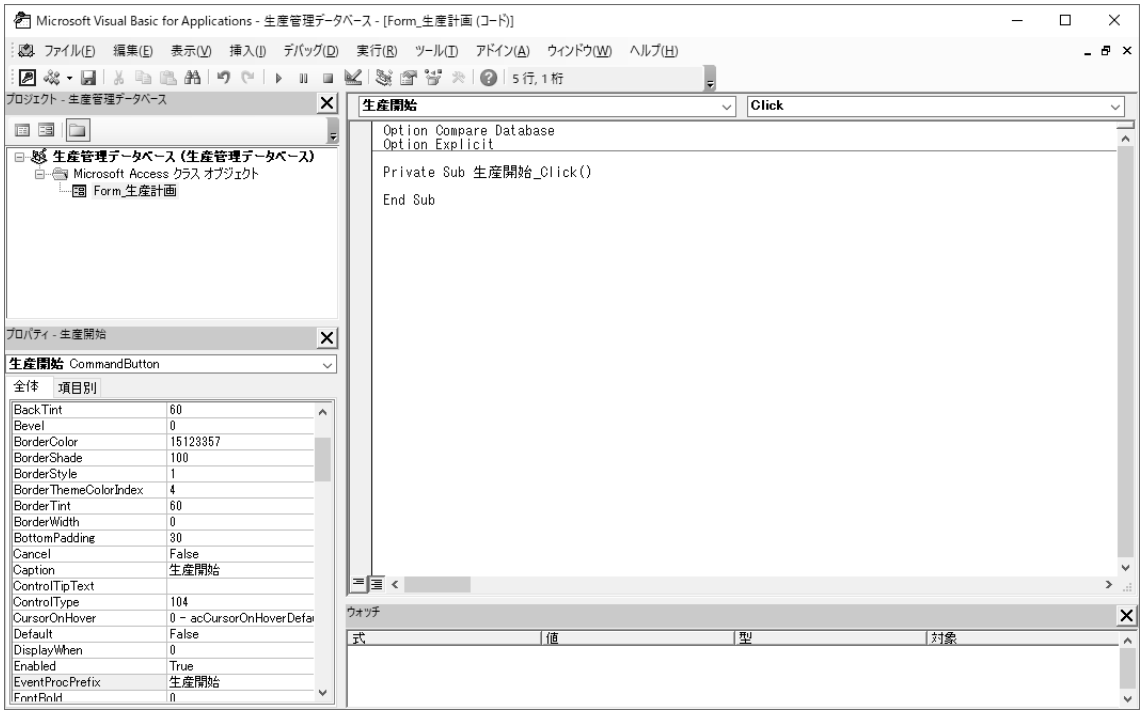


図 27

ここでは、ソースコードはまだ作らず、もう一つ**【生産実績更新】ボタンを作っておきます。**Access のフォームデザインウインドウに切り替えます。VBA ウインドウはそのままで構いません。

図 28

フォームフッター部に、先ほどと同じように適当な大きさのボタンを作ります。プロパティシートは表示されているので、そのまま（今作ったボタンが選択されている状態で）まず初めに**【その他タブ】**を選択して、ボタンの名前と標題を次の様に変更します。

①．**名前：　実績更新**

②．**標題：　実績更新**

その後に【イベント】タブを選択して、【クリック時】イベントのプルダウンから【イベントプロシージャ】を設定します。

先にイベントプロシージャの設定を行うと、コマンドXXというイベントが作られてしまうことがあるので、**この順番は間違えない様にしてください。**実際はこの後の…ボタン【ビルドボタン】押下時にイベントプロシージャ（プログラム）のフレームが作られます。

フォーム上に配置したボタンの名称と同じイベントプロシージャ（またはプログラム、または関数）が存在すると考えて下さい。

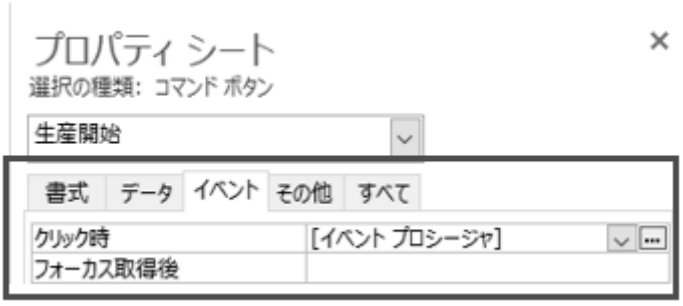


図 29

プルダウン右側の…ボタン【ビルドボタン】をクリックすると、先ほどの様に、今度は【実績更新】イベントの関数フレーム【実績更新_Click()】が出来上がります。

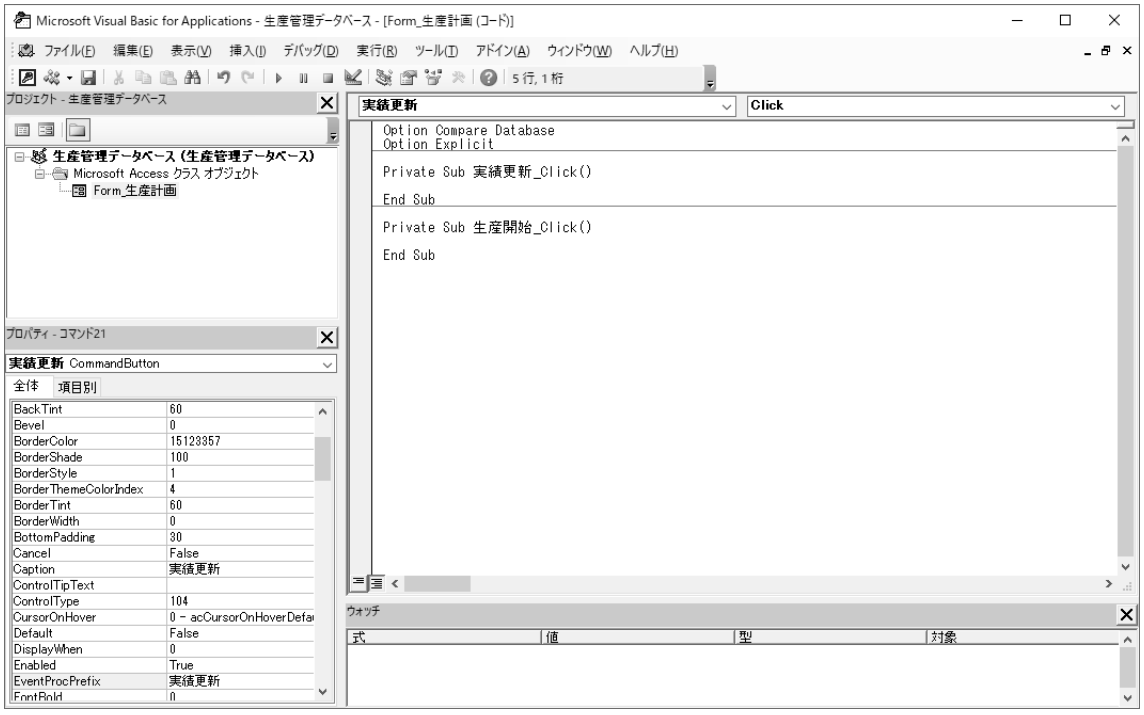


図 30

6．ボタンの処理イメージ

ここで、2つのボタンの処理イメージを考えてみましょう。

まず、Access フォームデザインウインドウ左上の【表示】（下図赤枠）からフォームビューを選択してください。

図　31

図　32

フォームには、**2 つの【生産開始】【実績更新】ボタン**が追加されています。

(1)【生産開始】ボタンの処理イメージ

※以下、**【テーブル名：フィールド名】**の様に記述する。

【ボタンを押す前】に生産開始する生産計画を**【選択する】**。

※【選択する】とは、生産計画フォームで該当の計画を表示しておくということ。

以下、ボタン押下後の処理。

①. 選択（画面に表示）されている【生産計画：開始日】が入っているか確認。【生産計画：開始日】が入力済み（≠Null）であれば**【開始済みと判断】**し、【エラーメッセージ】を表示して、処理終了。

上記以外は、以下を継続。

②.【生産計画：開始日】に現在の日付を入力して更新する。

③.【生産実績】に新規レコードを作成する。

□【生産計画：計画番号】--->【生産実績：計画番号】

□【生産計画：製品番号】--->【生産実績：製品番号】

□【現在日付】--->【生産実績：実績更新日】

※【生産実績：生産実績数】は、既定で0が入る。

④. 生産開始した旨のメッセージを表示して終了。

(2)【実績更新】ボタンの処理イメージ

【ボタンを押す前】に実績を更新する【生産計画】を**【選択する】**。

以下、ボタン押下後の処理。

①.選択（画面に表示）されている**【生産計画：終了日】**が入っているか確認。**【生産計画：終了日】が入力済（≠Null）**であれば**【生産終了と判断】**し、エラーメッセージを表示して、処理終了。

上記以外は、以下継続。

②.**選択中（表示中）**【生産計画：開始日】＝Nullか確認。

Ａ.【生産計画：開始日】が未入力（＝Null）であれば

【生産未開始と判断】--->(1)②以下を実施

Ｂ.【生産計画：開始日】が入力済（≠Null）であれば

【実績更新と判断】--->③へ進む

③.【生産計画：計画番号】＝【生産実績：計画番号】でヒットする【生産実績】レコードを対象に以下で更新

□【生産実績：生産実績数】+1--->【生産実績：生産実績数】

□現在日付--->【生産実績：実績更新日】

④.【生産実績：生産実績数】≧【生産計画：生産予定数】であるか確認。

Ａ.**≧の場合：【生産終了】**と判断--->⑥へ進む

Ｂ.**＜の場合：【生産継続】**と判断--->⑤へ進む

⑤.**【実績更新済みメッセージ】を表示して終了。**

⑥.**選択（表示中）**【生産計画】レコードを以下で更新

□現在日付　---> 【終了日】

⑦.【生産終了メッセージ】を表示して終了。

以上、処理イメージを文章で記述してみました。単純な処理でも処理分岐が含まれると、全体を把握しづらくなります。

問題：文章で全体が把握しづらい場合、誤解を生み易くなり、結果としてシステムに不具合の原因を作ってしまいかねません。上記の処理をフローチャートで記述してください。

7．プログラムとソースコードの入力

2つのボタンの処理プログラムを以下に示します。6の処理イメージと、以下のコメントの番号を照合しながら、内容を追ってください。

※プログラム（コメントも含む）が1行に収まらない場合は、行末に【↓】を付けて、次の行に続いていることを示している箇所があります。

※編集の都合で、ソースコードの下の行にコメントが記述されている箇所があります。

※リスト中 枠で囲んだ部分 はSQL（Structured Query Language）と言って、リレーショナル・データベース・システム（RDBMS）でデータの操作や定義を行うための問い合わせ言語の構文作成と、実行を行っている部分です。

```
‘=================================================
Private Sub 生産開始_Click()     '---サブプロシージャの始まり
    '---(1)①
    If IsNull(Me.開始日) = False Then   '生産計画：開始日≠ Null
        MsgBox ″この計画は、既に生産開始されています。″,  ↓
vbExclamation + vbOKOnly, ″生産管理システム″
        Exit Sub
    End If
    '---(1)②以下　生産開始手続き　を実行
    Call 生産開始手続き      'call は無くても可
End Sub '---サブプロシージャの終わり
‘=================================================

‘=================================================
'---(1)②以下　生産開始手続き（サブプロシージャ）
Public Sub 生産開始手続き() '---サブプロシージャの始まり
    Dim strSQL As String
        '---文字列変数：SQL(データベースを操作する言語)用
        '---文字型なので先頭に【str】を付けると分かり易い
    Me.開始日 = Format(Now, ″yyyy/mm/dd″)
        '---(1)②【生産計画：開始日】に現在の日付を入力して更新
    Me.Refresh
        '---開始日更新の確定（これで、レコードが保存される）

    '---(1)③【生産実績】に新規レコードを作成（ここから）
```

```
'---(1)③【生産実績】に新規レコードを作成（ここから）
DoCmd.Echo False
  '---画面の書換えを禁止（項目が沢山あると画面がチラつくから！）
DoCmd.Hourglass True
  '---マウスカーソルを砂時計クルクルにする(処理中の表示)
DoCmd.SetWarnings False '---確認ダイアログを非表示

'----レコード追加 SQL 作成（ここから）strSQL に文字列を↓
    つないで SQL を作っている。半角スペースに注意！
strSQL = "insert into 生産実績 "
  '---生産実績テーブルへのレコード追加
strSQL = strSQL & "(計画番号, 製品番号, 実績更新日) "
  '---対象フィールド列挙

'---追加する値の列挙　↓
     values ('計画番号','',#YYYY/MM/DD#)＋最後に【;】
'---文字列は【'】(アポストロフィー)で挟む↓
     (SQL の中だけ。VBA では【"】で挟む)
'---日付は【#】で挟む（SQL の中だけ）
strSQL = strSQL & "values ('" & Me.計画番号 & "','" & ↓
  Me.製品番号 & "',#" & Format(Now, "yyyy/mm/dd") & "#)"
strSQL = strSQL & ";"   '---最後に【;】セミコロンを付ける
'----レコード追加 SQL 作成（ここまで）

'--- for test SQL の内容を見るには、下記を実行
'MsgBox "SQL の内容：" & strSQL, vbInformation + vbOKOnly, "For
```

```
Test"
    '--- for test
    DoCmd.RunSQL strSQL      '---SQL実行・・・ここでレコードが追加される
    DoCmd.SetWarnings True   '---確認ダイアログを表示
    DoCmd.Hourglass False    '---マウスカーソルを元に戻す
    DoCmd.Echo True          '---画面の書換えを許可
    '---(1)③【生産実績】に新規レコードを作成（ここまで）
    '---(1)③【生産実績】に新規レコードを作成（ここまで）

    '---(1)④生産開始した旨のメッセージを表示
    MsgBox "生産開始しました。", vbInformation + vbOKOnly, ↓
      "生産管理システム"
End Sub '---サブプロシージャの終わり
'================================================


'================================================
Private Sub 実績更新_Click()     '---サブプロシージャの始まり
    Dim strSQL As String
      '---文字列変数：SQL(データベースを操作する言語)用
      '---文字型なので先頭に【str】を付けると分かり易い

    Dim intN As Integer
      '---整数型変数：【生産実績：生産実績数】用
```

```
'---(2)①【生産計画：終了日】が入力済（≠Null）か判断
If IsNull(Me.終了日) = False Then
    '---生産終了と判断、メッセージを表示して終了。
    MsgBox "この計画は生産完了しています。", vbInformation + ↓
           vbOKOnly, "生産管理システム"
    Exit Sub
End If

'---(2)②
If IsNull(Me.開始日) = True Then    '生産計画：開始日 ＝ Null
    Call 生産開始手続き      'call は無くても可
    Exit Sub
End If

'---実績更新と判断
'---(2)③（ここから）
'---(2)③（ここから）
DoCmd.Echo False
  '---画面の書換えを禁止（項目が沢山あると画面がチラつくから！）
DoCmd.Hourglass True
  '---マウスカーソルを砂時計クルクルにする(処理中の表示)
DoCmd.SetWarnings False '---確認ダイアログを非表示

'---現在の生産実績数を検索・取得--->intN
'---データベース関数：Dlookup(フィールド名, テーブル名, 検索条件)
```

```
    intN = DLookup("生産実績数", "生産実績", "計画番号 = '" & ↓
                    Me.計画番号 & "'")

    '----レコード更新 SQL 作成（ここから）↓
        strSQL に文字列をつないで SQL を作っている。半角スペースに注意！
    strSQL = "update 生産実績 "
    strSQL = strSQL & "set 生産実績数 = " & (intN + 1) & ", "
    strSQL = strSQL & "実績更新日 = #" & Format(Now, "yyyy/mm/dd") & "#
"
    strSQL = strSQL & "where 計画番号 = '" & Me.計画番号 & "'"
    strSQL = strSQL & ";"   '---最後に【;】セミコロンを付ける
    '----レコード追加 SQL 作成（ここまで）

    '--- for test SQL の内容を見るには、下記を実行
    'MsgBox "SQL の内容：" & strSQL, vbInformation + vbOKOnly, ↓
        "For Test"
    '--- for test

    DoCmd.RunSQL strSQL     '---SQL 実行・・・ここでレコードが更新される
    DoCmd.SetWarnings True  '---確認ダイアログを表示
    DoCmd.Hourglass False   '---マウスカーソルを元に戻す
    DoCmd.Echo True         '---画面の書換えを許可
    '---(2)③（ここまで）
    '---(2)③（ここまで）
```

```
    '---(2)④【生産実績：生産実績数】≧【生産計画：生産予定数】か？
    If (intN + 1) < Me.生産予定数 Then
        '---(2)⑤をIfブロック内で実施するため、逆の条件で確認。
        '---(2)⑤【実績更新済みメッセージ】を表示して終了。
        MsgBox "実績を更新しました。", vbInformation + vbOKOnly, ↓
              "生産管理システム"
        Exit Sub    '---終了
    End If
    '---(2)⑥選択（表示中）【生産計画：終了日】を現在日付で更新
    Me.終了日 = Format(Now, "yyyy/mm/dd")
        '---(1)②【生産計画：終了日】に現在の日付を入力して更新
    Me.Refresh  '---終了日更新の確定（これで、レコードが保存される）

    '---(2)⑦【生産終了メッセージ】を表示して終了。
    MsgBox "生産を終了しました。", vbInformation + vbOKOnly, ↓
          "生産管理システム"
End Sub '---サブプロシージャの終わり
'================================================
```

以上のソースコードを、VBAで記述するには、下図の様に生産計画フォームを選択して、右クリックで現れるメニューから、**デザインビューを選択**します。

図 33

図 34

生産計画フォームのデザインビューで縦方向にスクロールして、ボタンが見えるようにして（上図）、ボタンをクリック（選択）します。

図　35

プロパティシートを表示して、イベントタブのクリック時イベント右端の…　ボタンをクリックすると、VBA ウインドウが開きます。

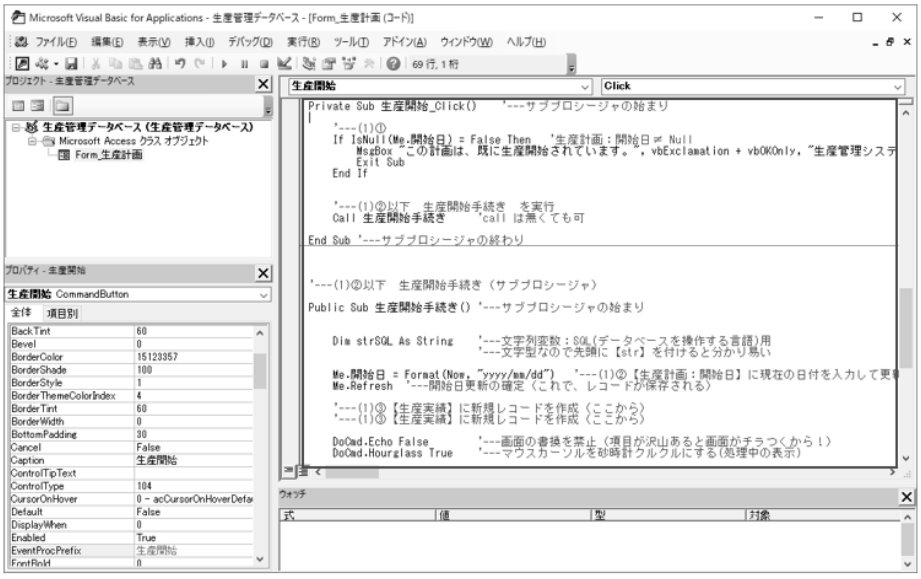


図　36

既に、ボタンをクリックした際のイベントを処理する関数フレームは出来ているので、そこにソースコードを記述します。

入力が終わりましたら、上書き保存をしてください。【ソースコードは、入力の途中で、こまめに上書き保存をしておくことがお勧めです。】

８．実行

ソースコードが完成したら、実行してみましょう。

図 37

ウインドウ左上の表示→フォームビューを選択（上図）すると、【生産計画フォーム】が開き**1件目の計画が表示されています**。（下図）

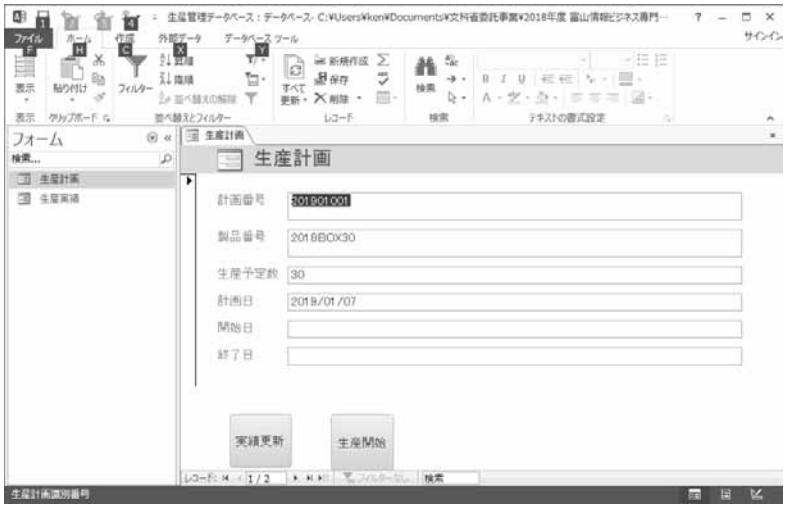


図　38

次に**生産開始ボタンをクリック**すると、生産開始日が自動入力されて、メッセージが表示されます。（下図）OK でメッセージは閉じます。

図　39

もう一度、生産開始ボタンをクリックすると、次のメッセージが表示され、重複処理を避けていることが分かります。

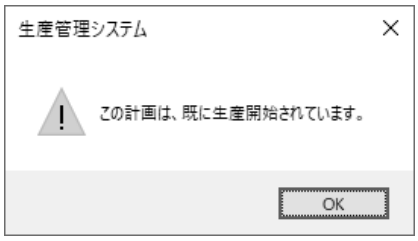


図　40

メッセージを閉じて、生産実績フォームを開きましょう。

図　41

1 件の生産実績レコードが作られていて、生産実績数＝0 でまだ、生産開始したばかりである事が分かります。フォームを生産計画に切り替えて（**フォーム上部のタブで切替られます**）今度は実績更新ボタンをクリックすると、実績更新のメッセージが表示されます。

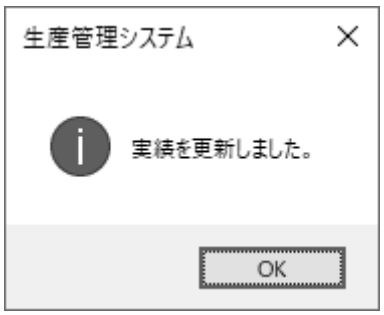


図　42

メッセージを閉じて、生産実績フォームに切り替えます。

図 43

生産実績数と日付（ここでは同じ日なので、分かりにくいですが、）が更新されています。もう一度生産計画フォームの実績更新ボタンをクリックすれば、**生産実績数が+1**されていることが分かるでしょう。

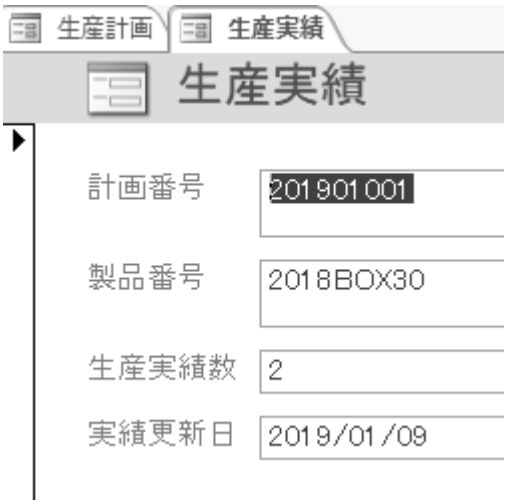


図 44

その調子で、何度も実績更新ボタンをクリックし続けると、やがて生産終了のメッセージが表示されます。

図　45

生産計画レコードには、**終了日も入力されている**ことが分かります。

このとき、**生産実績は生産予定数の 30 に達しています**。

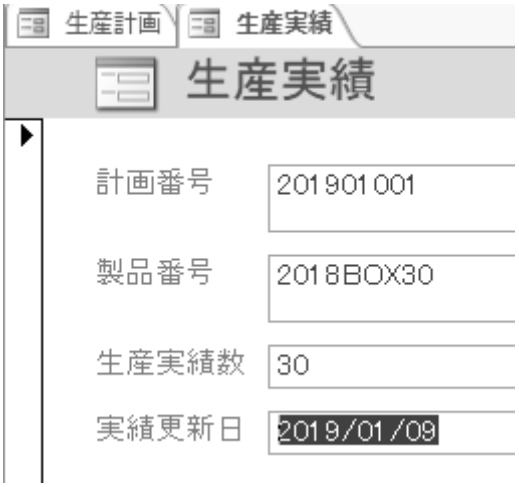


図　46

生産計画も終了となり、実績も予定数となりました。この状態でもう一度、実績更新ボタンをクリックしてみると、下記メッセージが表示されて、この計画が生産完了していることを、判断できています。

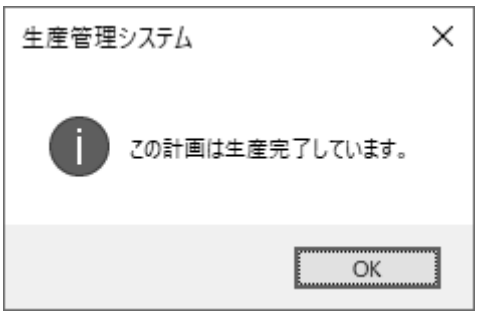


図 47

次に、生産開始ボタンをクリックすると、次のメッセージが表示されます。

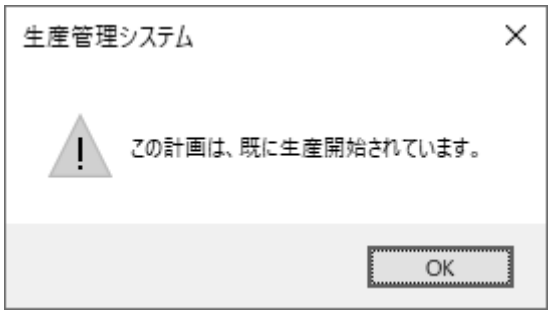


図 48

確かに、この生産計画は生産開始していますが、既に生産終了しています。これは、少し不自然ですね。この場合なら前のメッセージ【既に生産完了しています】の方が適切ですね。ではメッセージの内容を変えれば良いのかというと、もう少し複雑で【生産開始ボタンの処理イメージ(1)①】の内容を少し考え直さなくてはいけません。そこで、問題です。

問題：【生産開始ボタンの処理イメージ(1)①】の内容を検討して、【生産計画：開始日】と【生産計画：終了日】の入力状況に応じた判断をして適切なメッセージが表示されるように、このシステムを修正して下さい。

８．まとめ

今回作成した生産管理データベースは、簡易的に作成したので、まだまだ改善すべき部分が多いのですが、生産計画と生産実績の関係は、理解できると思います。実際には、製品が出来る都度、人が画面を開いて【実績更新ボタン】を押すのではなく、**実績収集用 IoT デバイス**からデータを受信したシステムが、実績更新_Click()関数を Call するようにすれば良いわけです。

※今回作製した生産管理データベースは、生産実績フォームのアクティブ時イベントに次の VBA のプログラムを追加した物が、実習キット付属 CD に収録されています。

最後に少しだけプログラムを追加しておきます（下図、アクティブ時）。

図 49

```
Private Sub Form_Activate()
|    me.Refresh
End Sub
```

図 50

これは、生産実績フォームを開いたまま、ボタンの処理を行い、生産実績テーブルの内容が更新されても、画面に反映されないことを防ぐためのコードです。

B:パーツの使い方・注意点

1．ブレッドボード

ブレッドボードは、半田付けの要らない基板なので電子回路を実験するのに大変便利です。

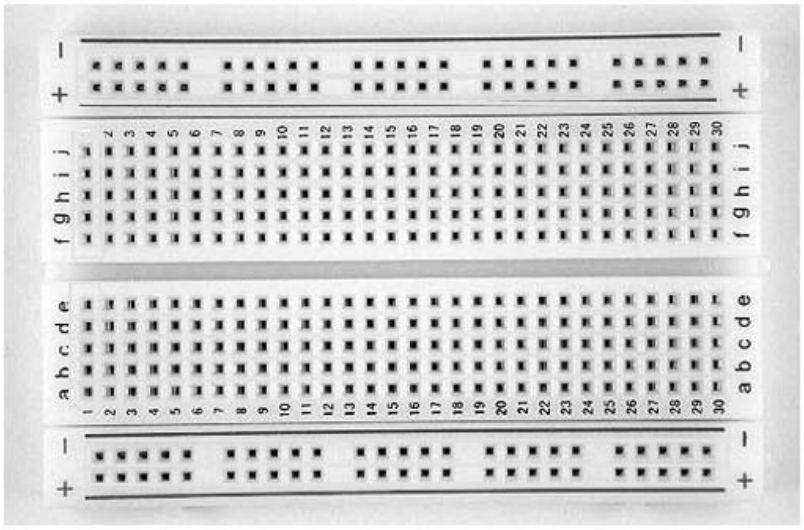


図 51

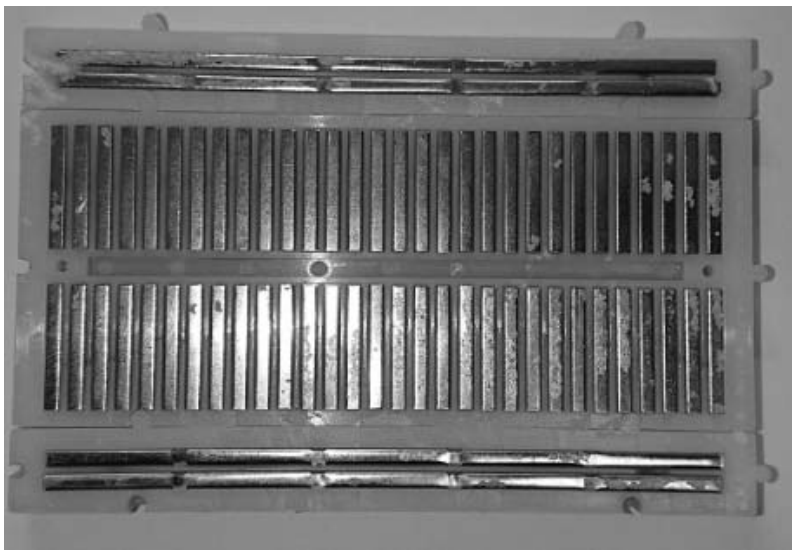
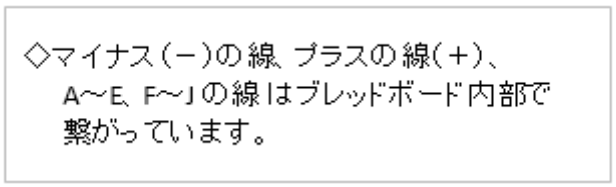


図 52(ブレッドボード内部：裏側から)



上の図のように、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、裏側写真では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。

２．電池ケース

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 本使用するものと 3 本使用するものがあります。この実習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。モーターを駆動する場合は、乾電池 3 本で 4.5V にして使います。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 53

図 54

３．ジャンパー線

図 55

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。

４．ＬＥＤ

LEDは、次の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。

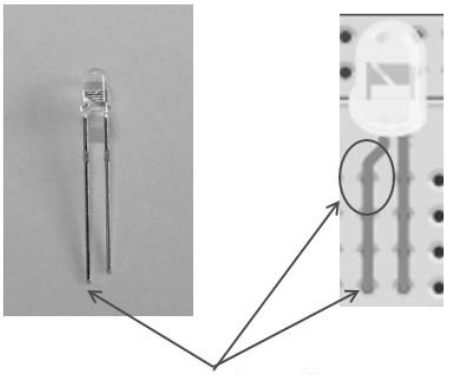


図 56

４．抵抗器

抵抗器は、カラーバーで抵抗値を示しています。LEDに接続する抵抗器は470Ω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。

【Appendix　C:抵抗器のカラーコード】にカラーコード表を掲載しました。スマートフォン用アプリには、抵抗カラーコード表もあって便利です。

◇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。

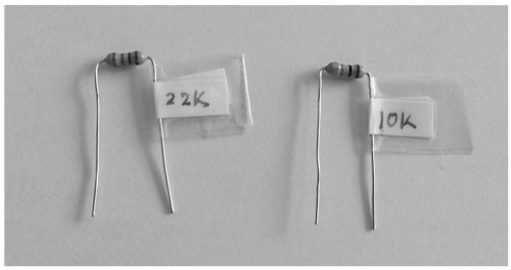


図 57

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。

5．無線マイコンモジュール（TWE-Lite）

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。

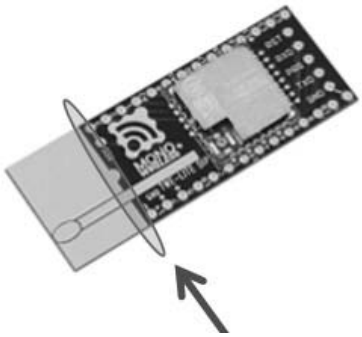


図 58

C：抵抗器のカラーコード

抵抗器のカラーコードを下図に示します。抵抗の表面に印刷されているカラーの帯で抵抗値を示しています。写真と実物とよく見比べて、間違えない様に抵抗を選んでください。図には、一部の抵抗器しか掲載されていないので、注意して下さい。図上部のカラー抵抗早見表を見ると、掲載されていない抵抗値も読めると思います。

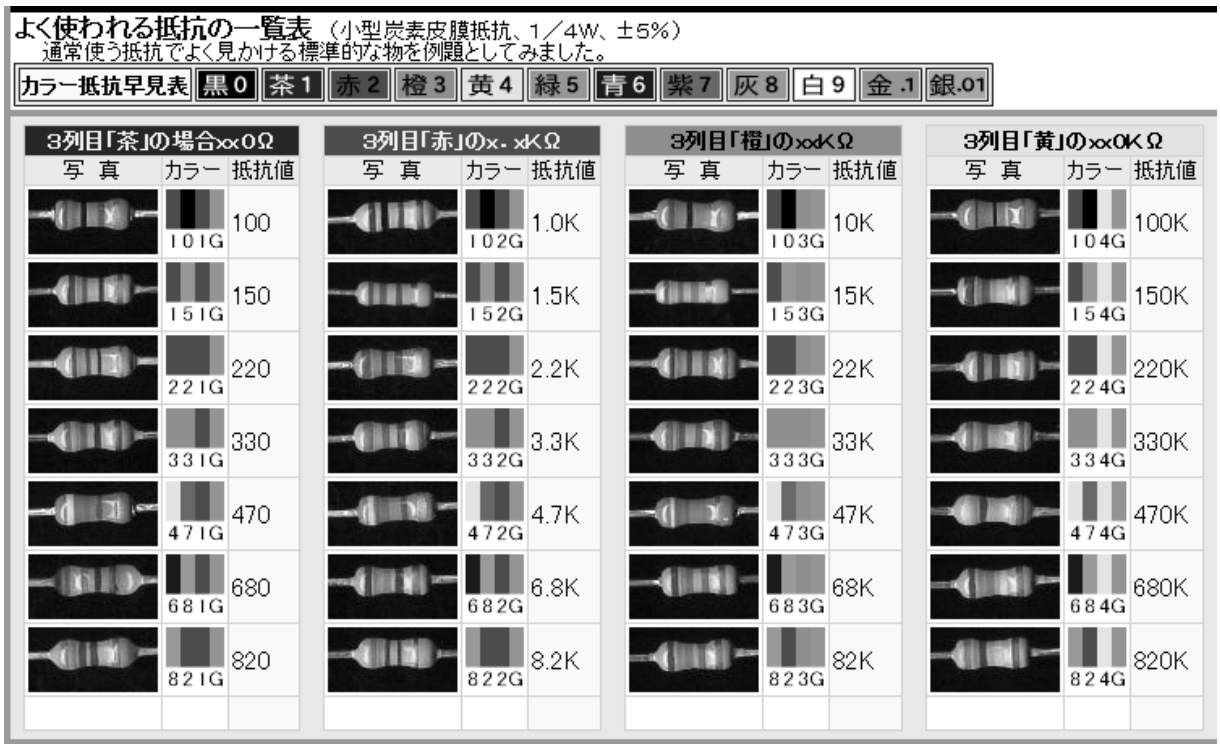


図 59

D:A/D 変換と温度センサー

◇温度センサーはアナログ電圧出力

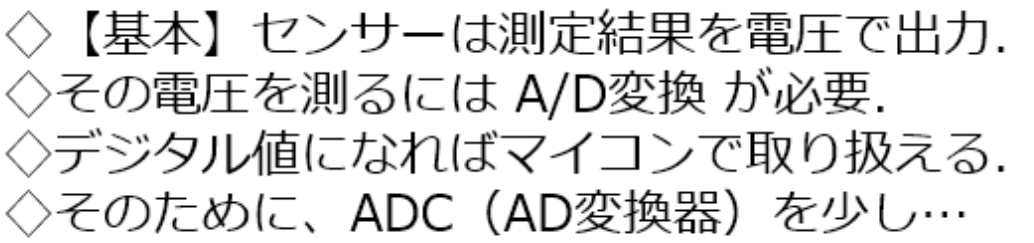


図 60

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。

無線マイコンモジュールブロック図 ADC

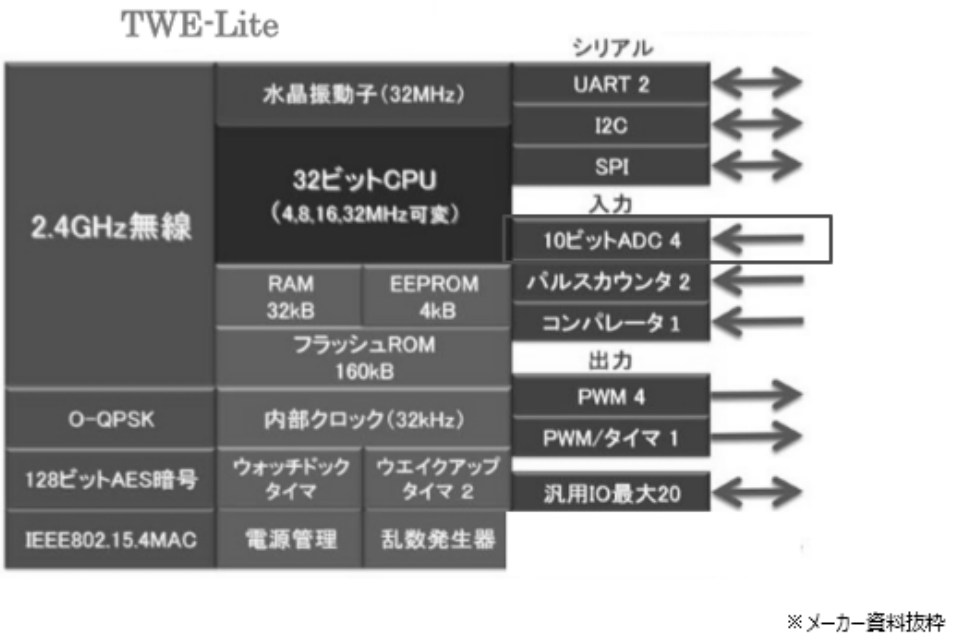
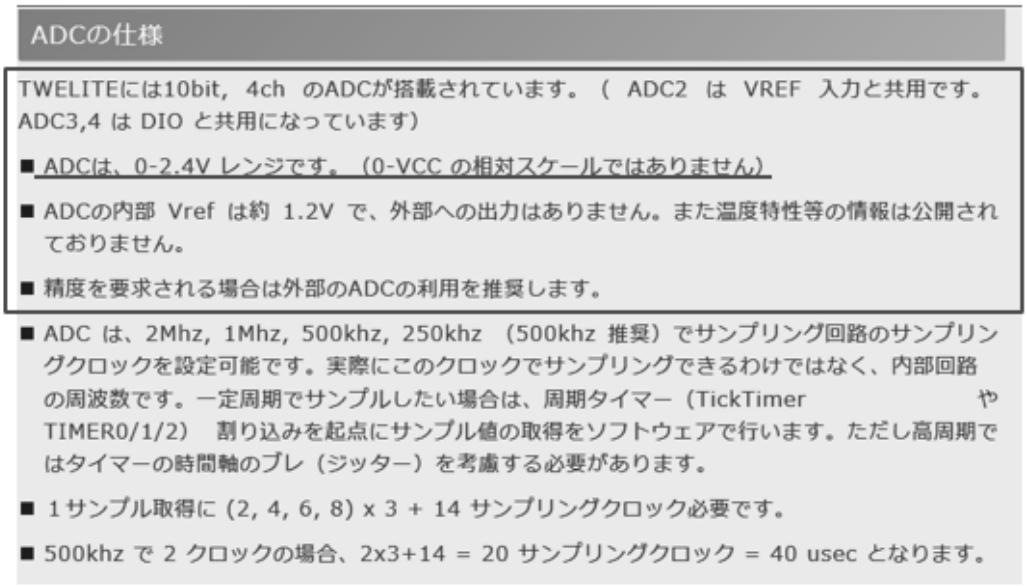


図 61

上図は、無線マイコンモジュール（TWE-Lite）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。



ADCの仕様

TWELITEには10bit, 4ch のADCが搭載されています。(ADC2 は VREF 入力と共用です。ADC3,4 は DIO と共用になっています)

- ADCは、0-2.4V レンジです。(0-VCC の相対スケールではありません)
- ADCの内部 Vref は約 1.2V で、外部への出力はありません。また温度特性等の情報は公開されておりません。
- 精度を要求される場合は外部のADCの利用を推奨します。
- ADC は、2Mhz, 1Mhz, 500khz, 250khz (500khz 推奨) でサンプリング回路のサンプリングクロックを設定可能です。実際にこのクロックでサンプリングできるわけではなく、内部回路の周波数です。一定周期でサンプルしたい場合は、周期タイマー (TickTimer や TIMER0/1/2) 割り込みを起点にサンプル値の取得をソフトウェアで行います。ただし高周期ではタイマーの時間軸のブレ (ジッター) を考慮する必要があります。
- 1サンプル取得に (2, 4, 6, 8) x 3 + 14 サンプリングクロック必要です。
- 500khz で 2 クロックの場合、2x3+14 = 20 サンプリングクロック = 40 usec となります。

※メーカー資料抜粋

図 62

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。

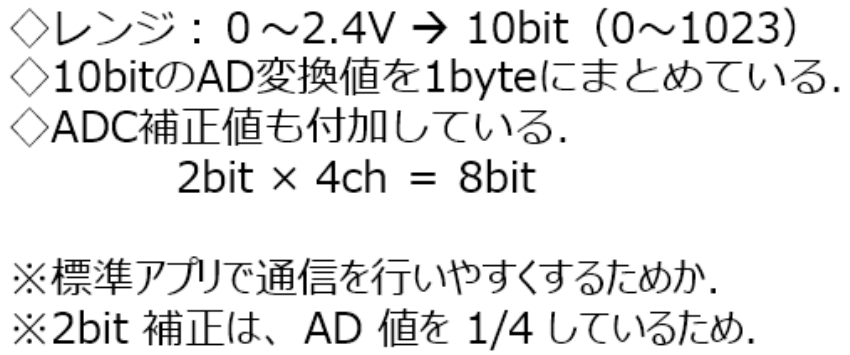


図 63

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。

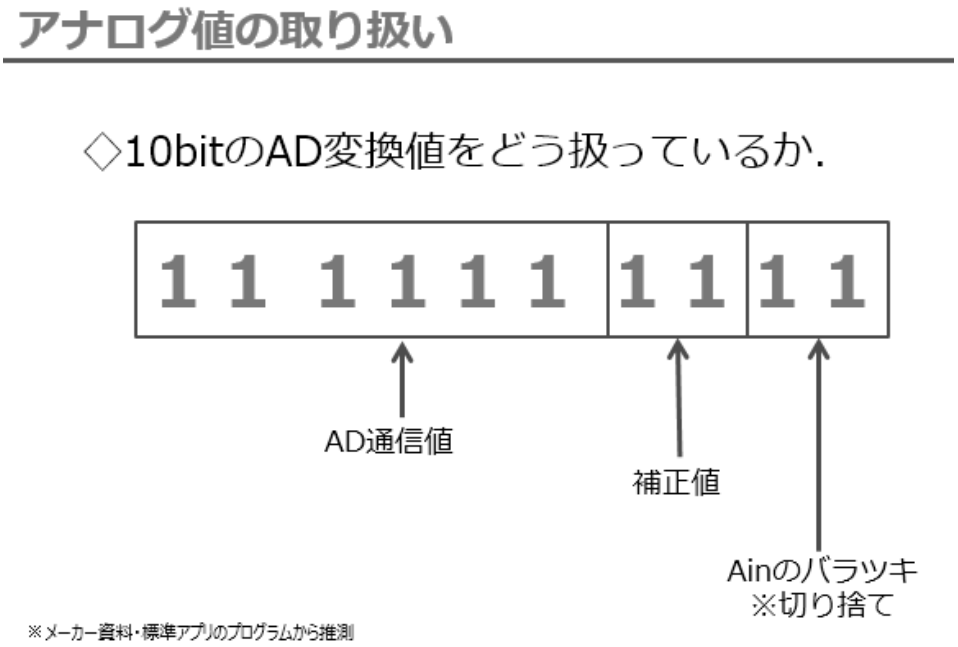


図 64

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　65

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、上の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

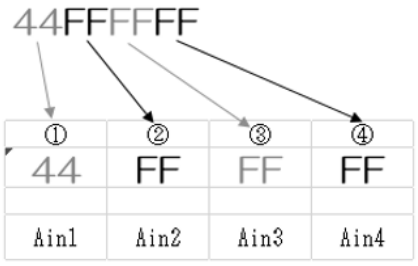


図　66

アナログ入力値の部分は上の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。

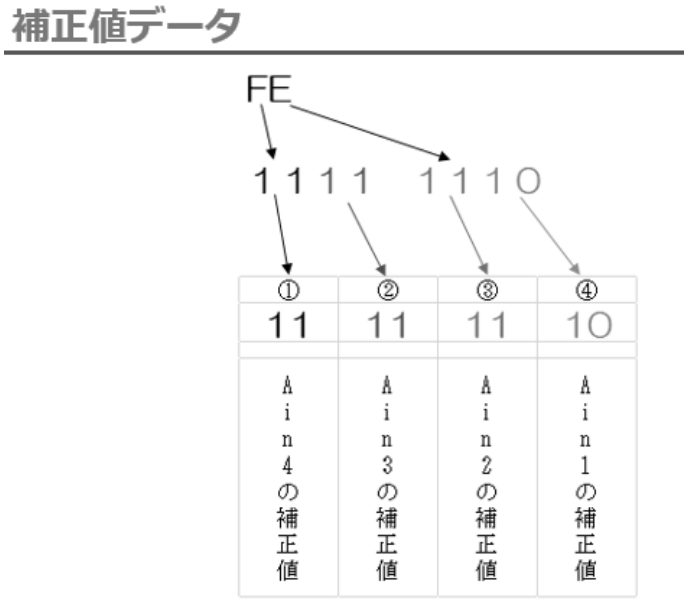


図 67

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。

これらの電文中の情報から、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 68

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。

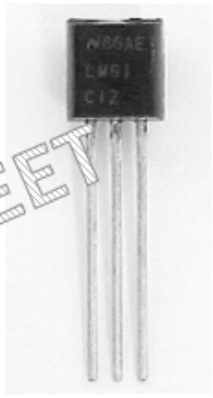
温度センサー

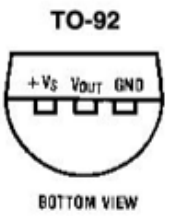
◇LM61CIZ　リニアな特性

◇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)

DATA SHEET

温度＝（センサー出力電圧－600mV）÷１０mV

図 69

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。

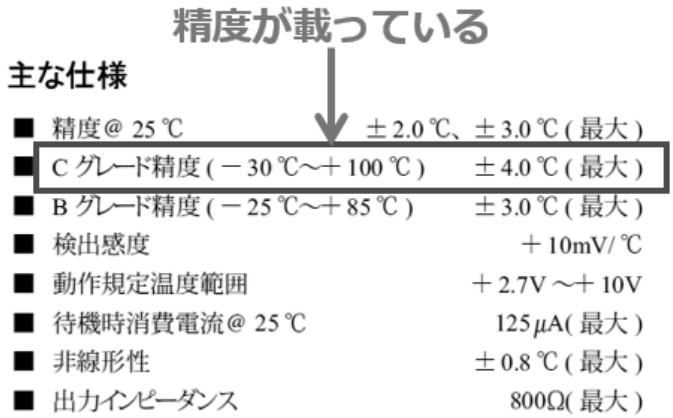
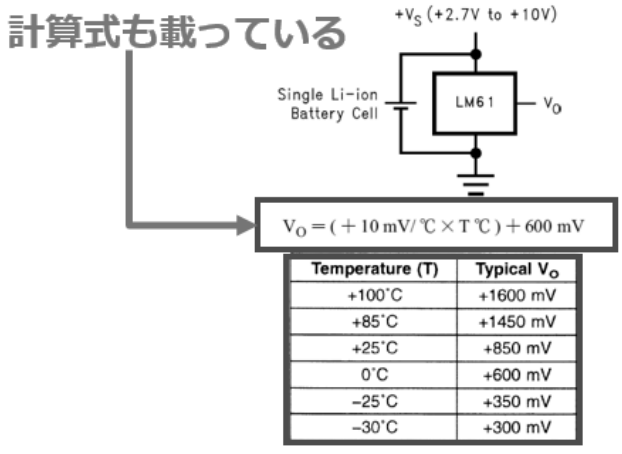


図 70

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 71

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。

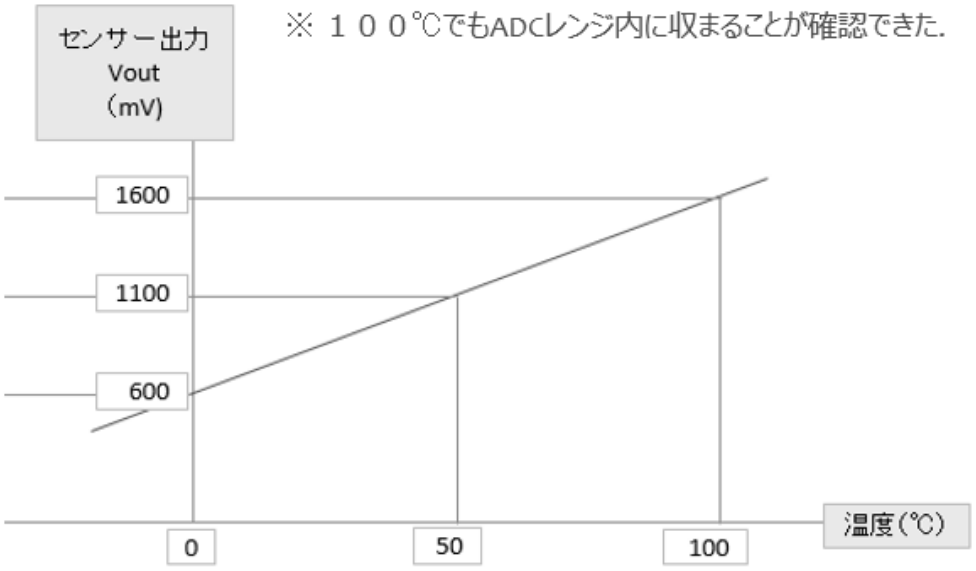


図 72

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の図です。**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 73

ピンの誤りは高熱破壊！！

ピン配置を十分確認.

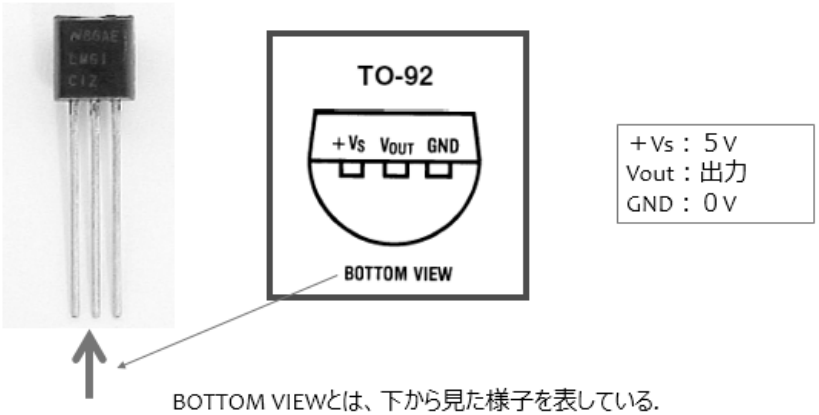


図 74

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。

E:Python 開発環境の準備

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るようになってきました。

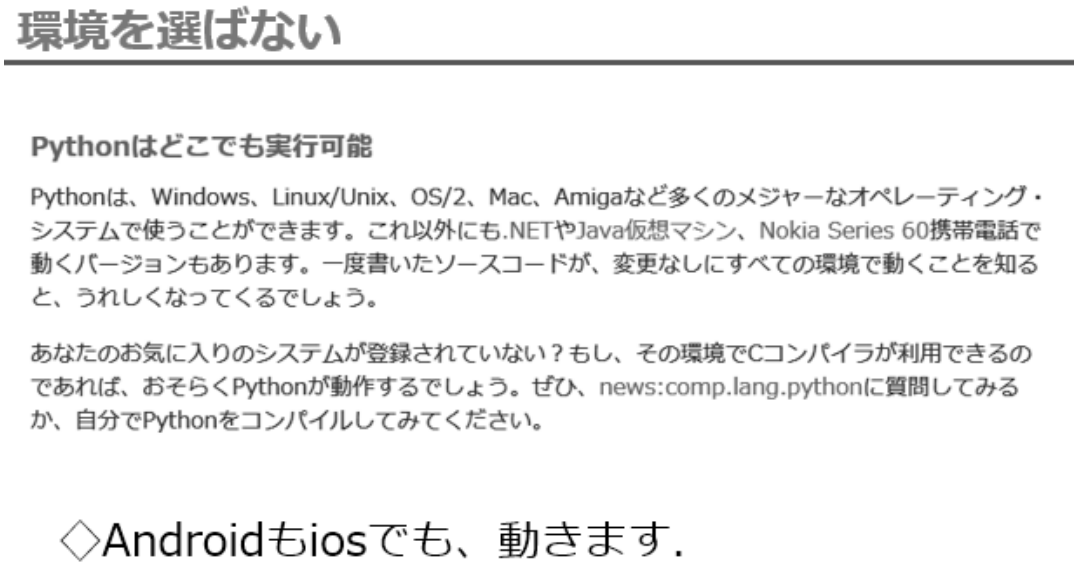


図 75

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。ここでは Python 環境の準備を行います。

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　76

この演習では、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

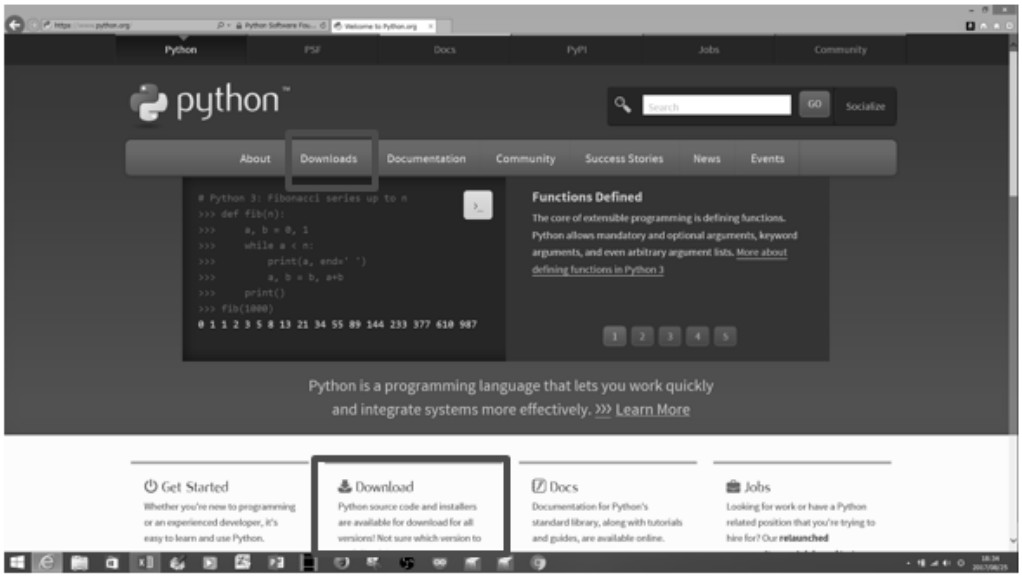


図　77

Python2.7系

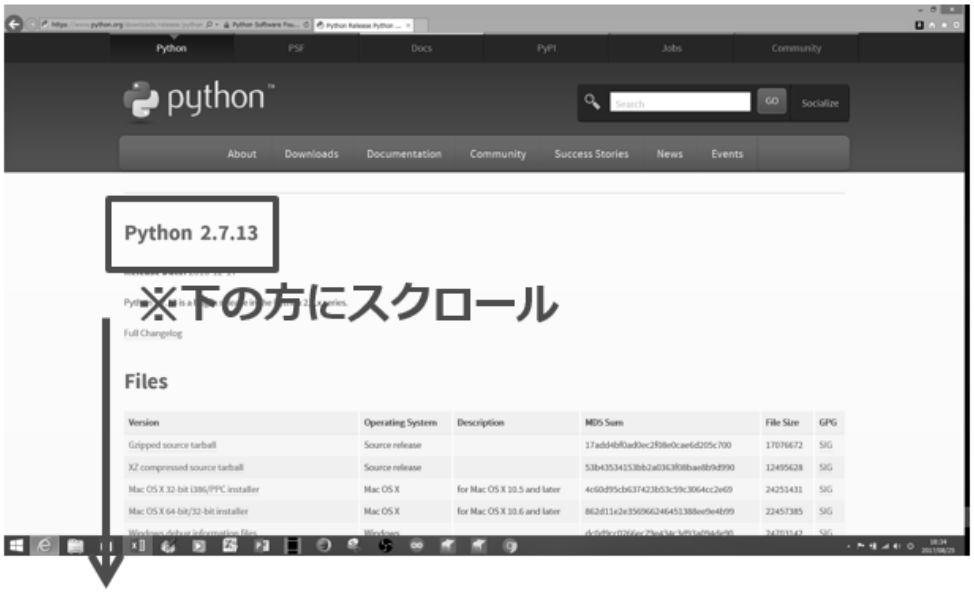


図　78

Windows x86 MSI Installer

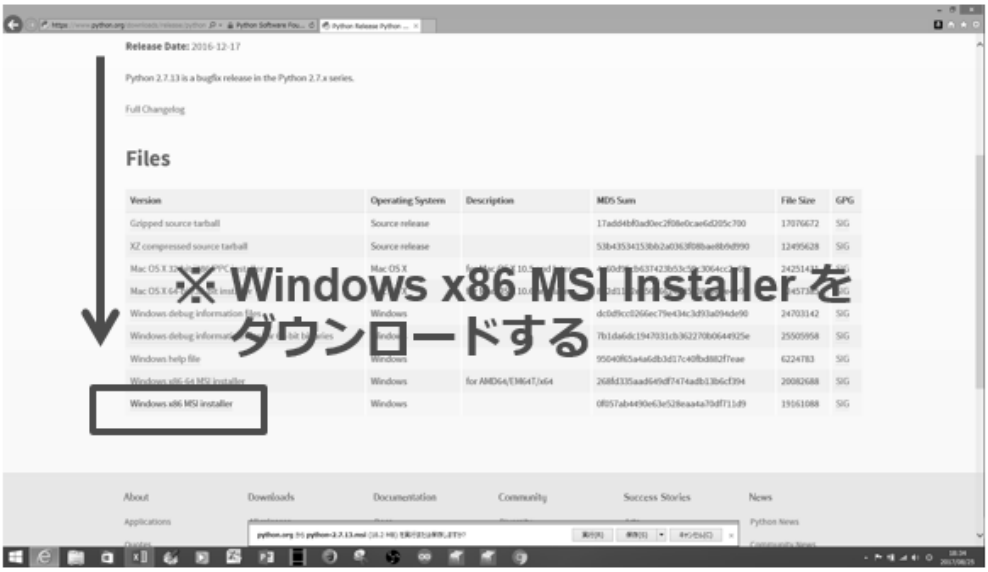


図　79

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 80

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.

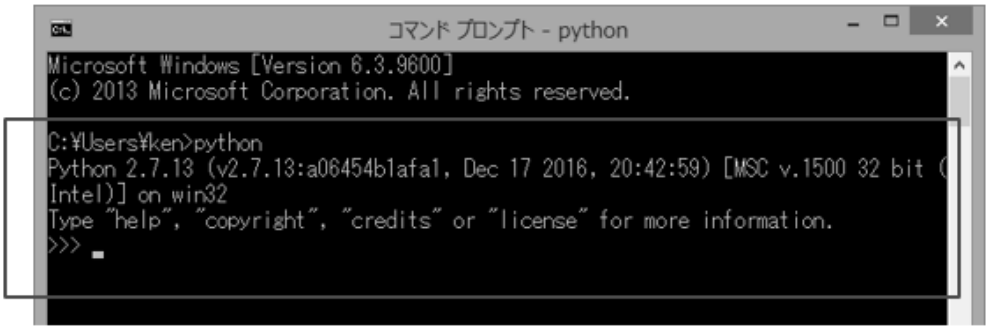


図 81

上の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 82

上の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リも準備完了です。

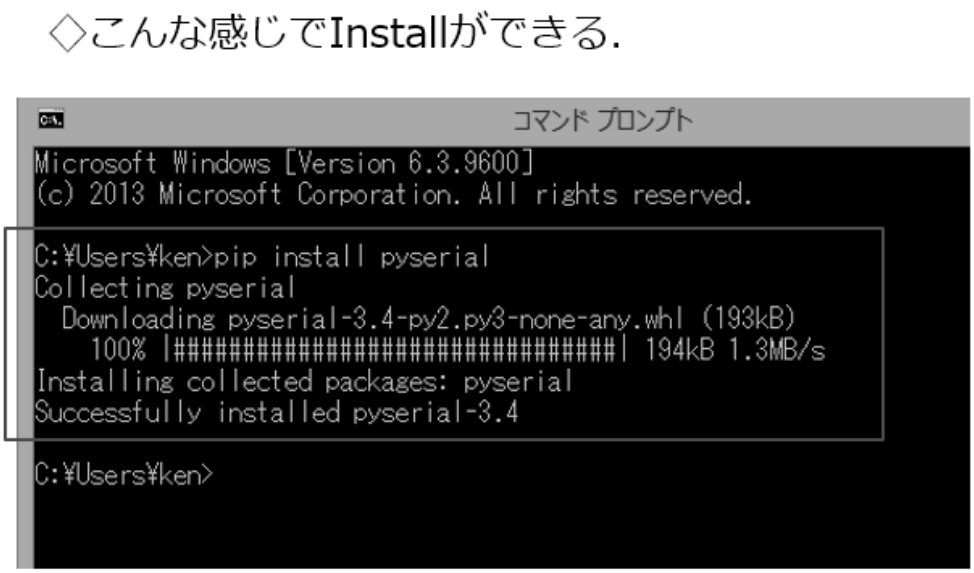


図 83

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 84

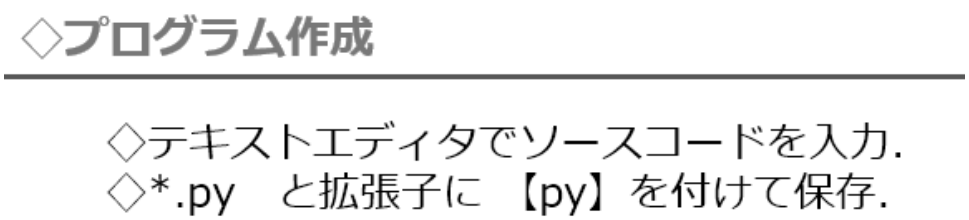


図 85

後は 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。Windows 付属のメモ帳でも OK です。）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　86

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。

F：ソースコード（Step5）

以下に Step5 の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。

G:MQTT アプリの利用

Web サービスとして公開されている MQTT Broker を使用した、メッセージ交換アプリが公開されています。この実習では、MyMQTT（Android 版）というアプリの使用方法を説明します。他にもたくさんのアプリが公開されています。この解説は他のアプリを利用する際の参考にもなるでしょう。

図 87

MQTT サービスに発行したメッセージを読むためには、スマートフォン側でアプリケーションを準備します。沢山のアプリケーションが公

開されていますが、その中でMyMQTT（Android版）を使用します。

（※Android以外のものも沢山公開されていますので、皆さんの環境にあったものを見つけてください。）

MyMQTTをインストールすると、上図右のようなショートカットができます。このアプリの使いかたを説明します。

図 88

MyMQTTを起動すると図左の画面になります。

1．Dashboardをタップすると、右側の画面になります。

2．Settingsをタップします。次の画面に変わります。

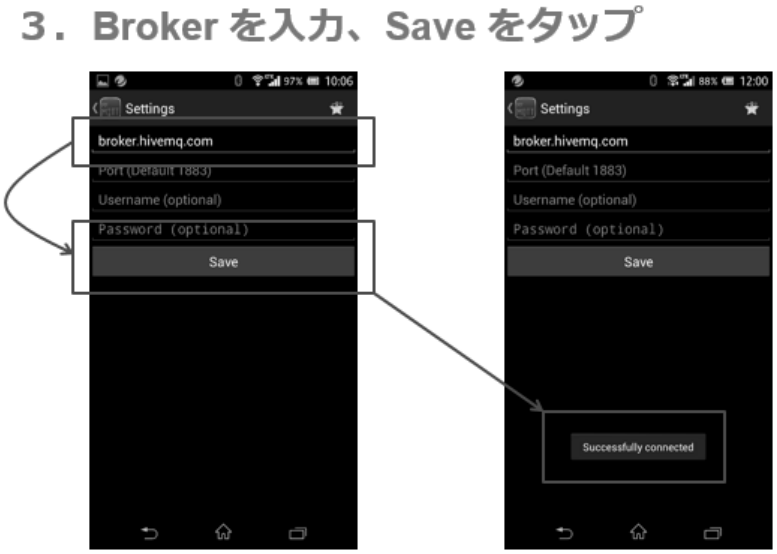


図 89

３．Broker 名 【broker.hivem.com】を入力して、save をタップします。

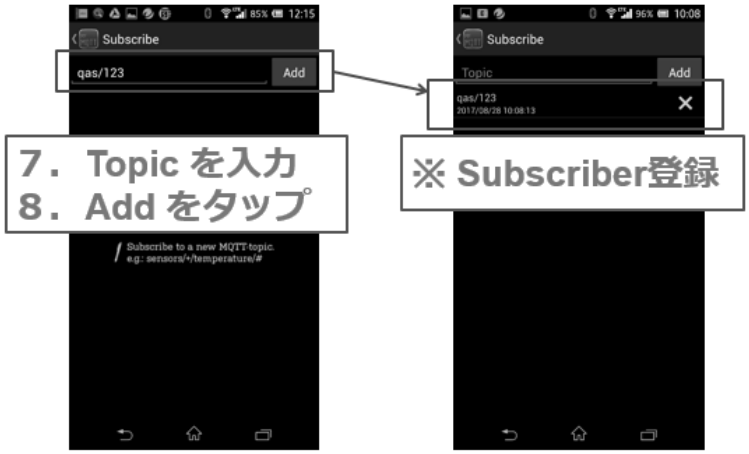
※この時、指定した MQTT Broker に接続が行われます。

Successfully conneted とメッセージが表示されない場合は、入力した MQTT Broker を確認して下さい。このアプリケーションは、起動するたびに設定した MQTT Broker への接続を行って、接続の状況をメッセージで知らせてくれるので、それを確認してから使用するようにしてください。

図 90

4．Settings をタップして、前の画面にもどります。

5．Subscribe をタップします。



【重要】

ここで登録する Topic 名はどのような名称でも構いませんが、後で作成するプログラムに記述しますので、記録しておいてください。

図 91

6．Topic を入力して Add をタップして登録します。（上図の７，８）

図 92

９．Subcribe をタップして元に戻ります。

１０．Dashboard で Subscribe します。

※通常、アプリを起動して MQTT Broker に接続確認したら、Dashboard を使用して発行されたメッセージを購読する、という流れで使用します。

使い方の細かな点は省略していますが、使用しながら操作に慣れて下さい。また、他のアプリも試してみて、皆さんの使い易いものを発見していただくのも面白いと思います。

◇Subscribeの様子

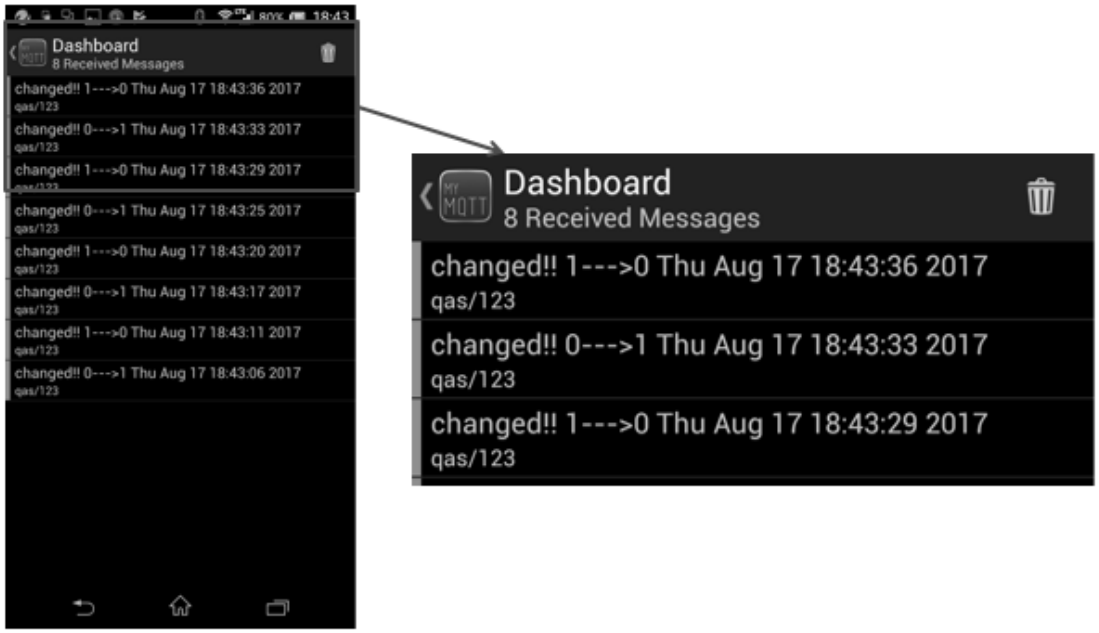


図 93

上図に Subscribe の様子を示します。Dashboard を見ていると、メッセージが発行される度に Dashboard にその内容が表示されていきます。

購読したメッセージは、新しいものが上に積上げられ、古いものは下に下がっていきますので、動作確認を行う際は、Dashboard の最上部に注目していて下さい。

H:MQTT ライブラリの準備

WiFi マイコンのソフトウエア開発に必要な MQTT ライブラリを準備します。

MQTTライブラリの準備

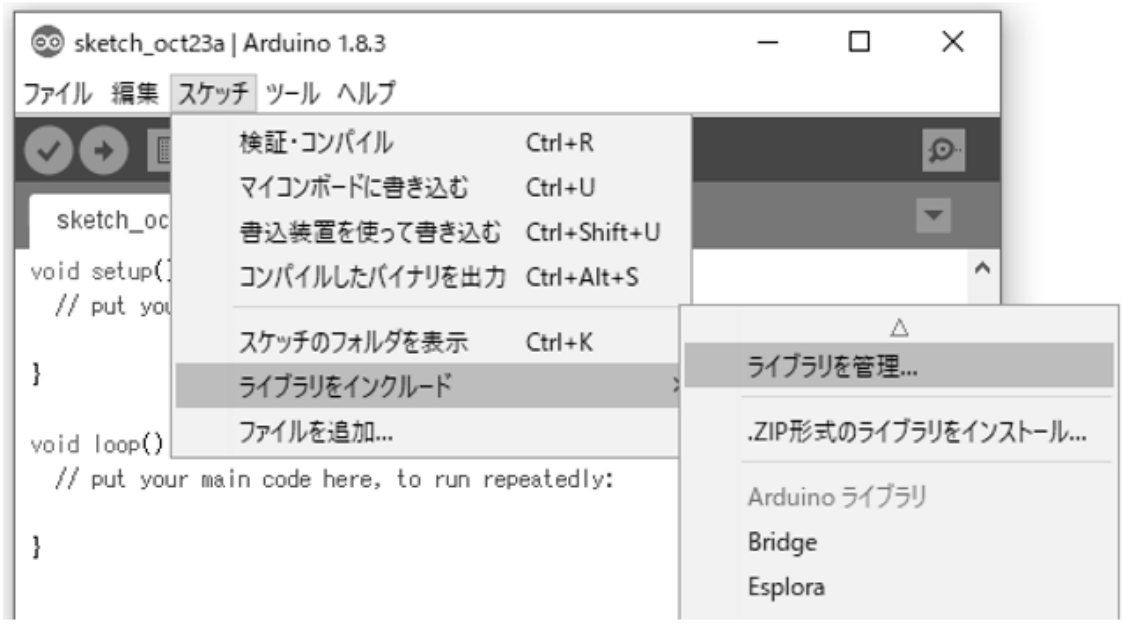


図 94

IDE で上図の手順、【スケッチ→ライブラリをインクルード→ライブラリを管理】を辿りライブラリマネージャを開いてください。

【注意】

この時 PC は Internet に接続された状態であることが必要です。ネットへの PC 接続を確認してください。

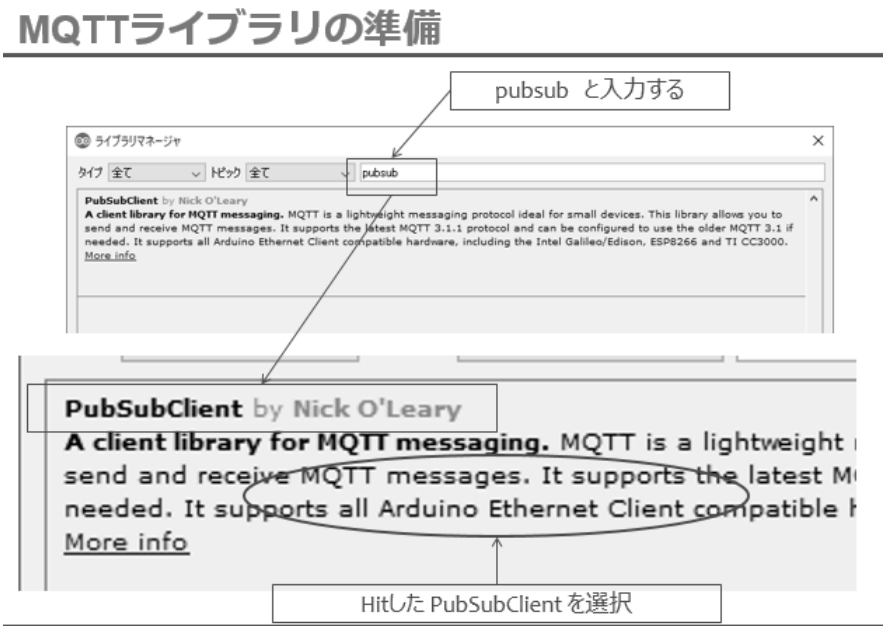


図 95

ライブラリマネージャの上部にある検索テキストに【pubsub】(半角小文字で可)と入力します。すると、該当のライブラリがネット上で検索されて【PubSubClient】がヒットします。この中央部分をクリックして選択すると、右下にインストールボタンが現れます。(下図) これをクリックして、インストールして下さい。

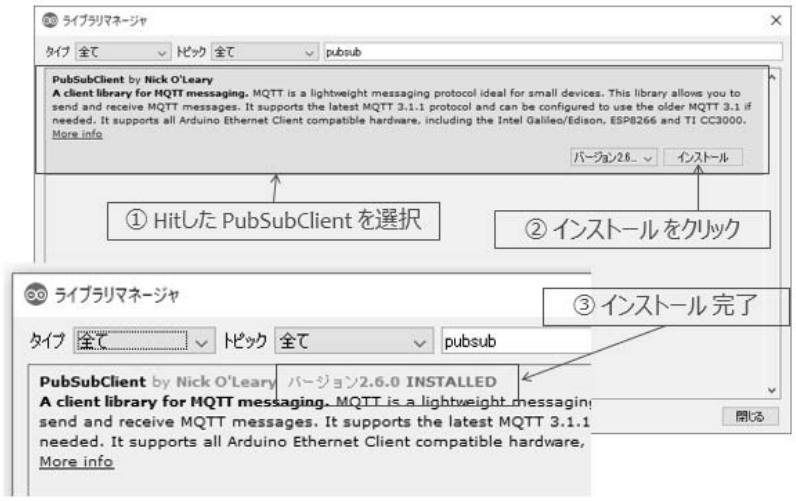


図 96

MQTTライブラリの準備

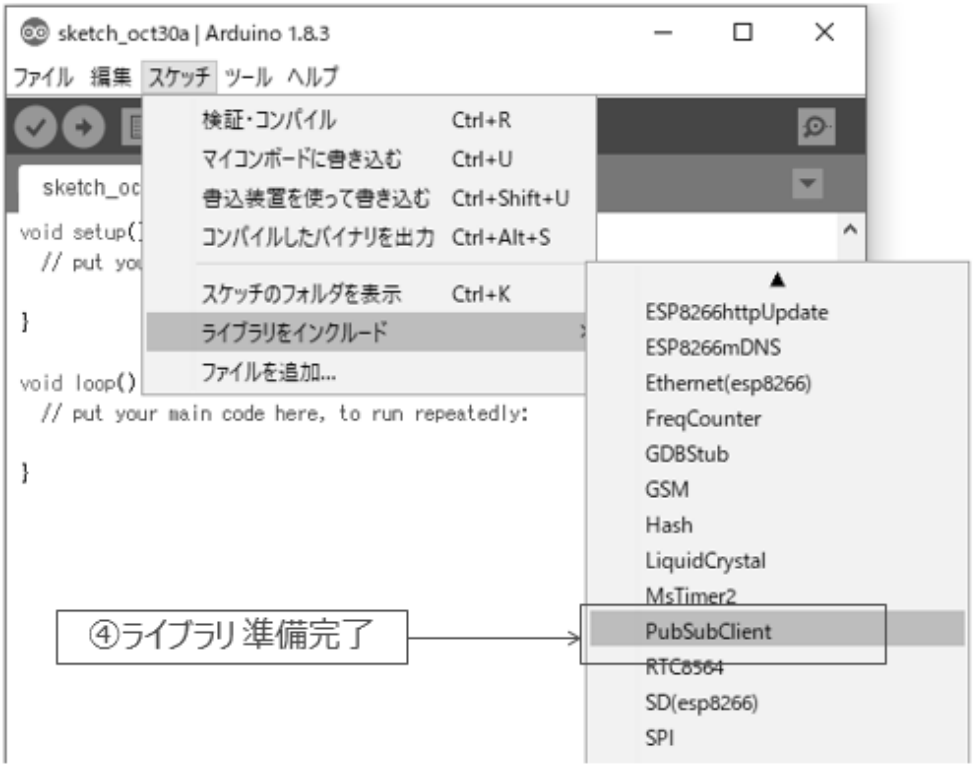


図 97

ライブラリが正しく準備できたことを確認するには【スケッチ→ライブラリをインクルード】と辿って表示される一覧の中に【PubSubClient】が表示されていれば、準備完了です。

I:PLCOpen 環境の準備

openPLC 環境を準備します。準備は次の 2 つのソフトウエアのインストールです。

◇Soft PLC（Windows 版） Runtime

◇Slave Device（ESP8266）用 Runtime

１．必要ファイルのダウンロード

ブラウザで

http://www.openplcproject.com/

にアクセスすると、下記 HP が開きます。

図 98

GETTING START を選択して、次の画面に進みます。

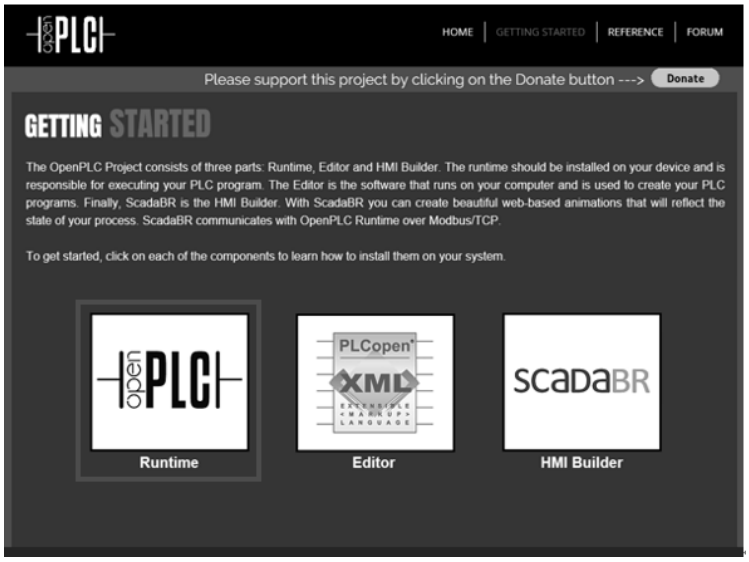


図 99

左側の Runtime を選択します。

図 100

SOFT PLC と ESP8266 から該当のファイルをダウンロードします。

まず、Windows 版 SOFT PLC を選択します。

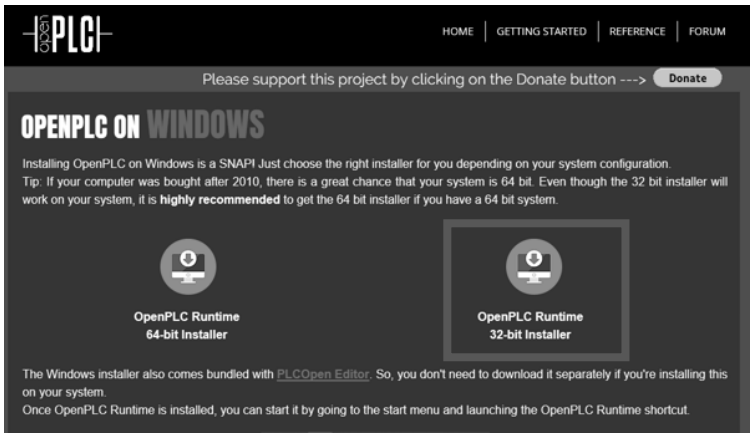


図 101

32bit 版をダウンロードします。（**64bit 版は動作が不安定**になることがありました。）

前画面に戻り、ESP8266 を選択します。

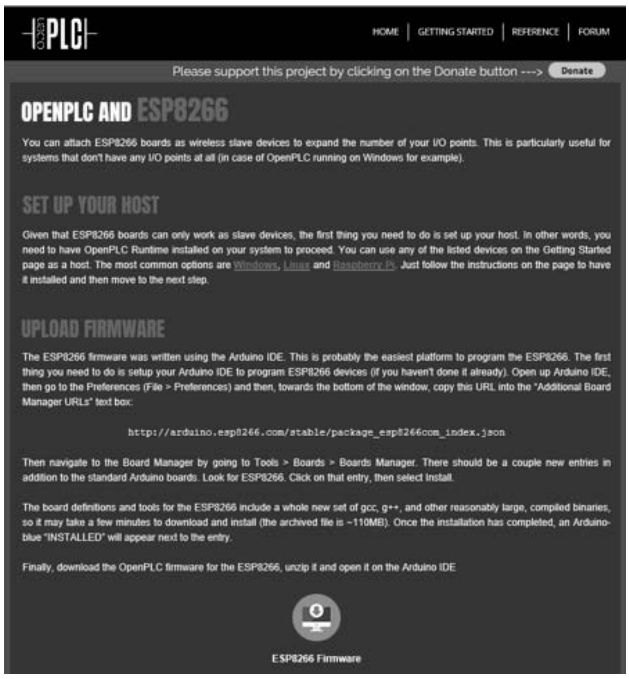


図 102

ESP8266 Firmware をダウンロードします。

図 103

上記の OpenPLC Installer と OpenPLC_esp8266 のファイルがダウンロードされました。

2．Windows 版 Runtime のインストール

該当のファイルをダブルクリックしてインストールして下さい。

オプションは、特にありません。

【注意】このプログラムのインストールは、通常のアプリケーションのインストールと異なり、PC の既存環境に対応した必要ファイルをダウンロードしながら環境を新たに作りますので、状況によっては長い時間が掛かることがあります。(新規 PC：Core i7 ではおよそ 15 分程度で終了しました。いろいろな環境が入っている PC：Core i5 では 1 時間以上かかりました。)

3．ESP8266（WiFi マイコン）用 Runtime の書込み

図 104

OpenPLC_esp8266 を解凍すると、次の Arduino スケッチが含まれています。

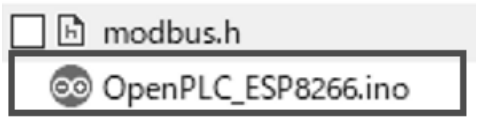


図　105

OpenPLC_ESP2866 を ArduinoIDE で開きます。ソースコードの冒頭に WiFi アクセスポイントの SSID と Password を記述する部分があります（下図）ので、そこを使用する環境に合わせて変更した後、適当な名前を付けて保存してから、**WiFi マイコンアプリケーションの書込みと同様**の手順で、WiFi マイコンに書き込みます。

```
#include <ESP8266WiFi.h>

/**********NETWORK CONFIGURATION**********/

const char *ssid = "WiFi Name";
const char *password = "WiFi Password";

/****************************************/
```

図　106

４．書き込みが完了したら、シリアルモニタ（IDE の右上にある虫眼鏡マーク）を起動して、通信速度を 115200bps に合わせて、マイコンの Reset ボタンを押します。

図　107

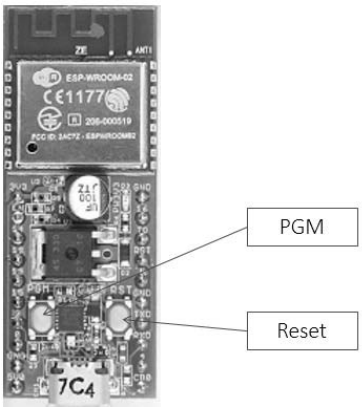


図 108

シリアルモニタを見ていると、WiFi Access Point に接続して WiFi マイコンが取得した **IP Address が表示されますので、これを控えておきます**。この IP アドレスは、本編で使用します。

```
Connecting to Planex_24-E68A9A
........
WiFi connected
Server started
My IP: 192.168.0.10
```

図 109

以上で openPLC の環境準備は終了です。

J:PLCOpen の使い方

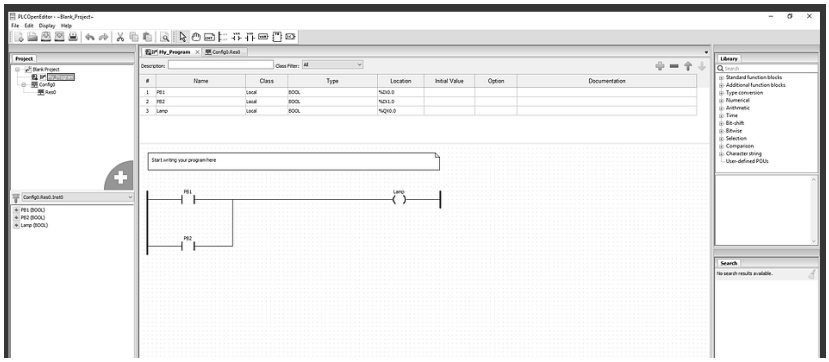


図 110

PLCOpen を利用するというのは、ラダー図を PC 上で描き、それをファイルに保存し、PLCOpen が理解できる言語に変換（Compile）して動かすことです。既に環境は準備されているので、OpenPLC プロジェクト Web ページ掲載の例を用いて、使用手順を解説します。

１．初めてのプロジェクト作成

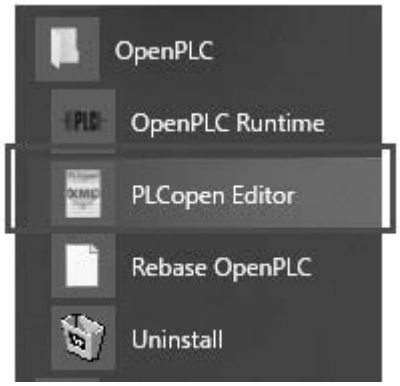


図 111

PLCOpen Editor を開いた後、新しいプロジェクトを作る必要があり

ます。新しいプロジェクトを作るためには、File--->New とクリックします。

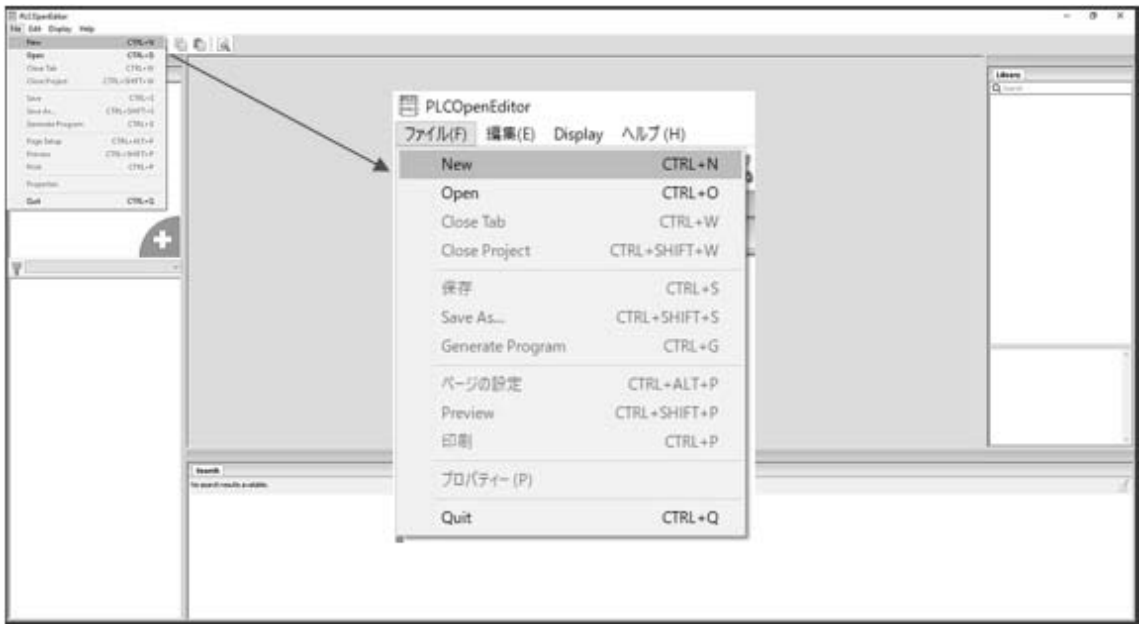


図 112

プロジェクト情報を入力するためにダイアログが表示されます。

Project タブと Author タブに必要な情報を入力します。

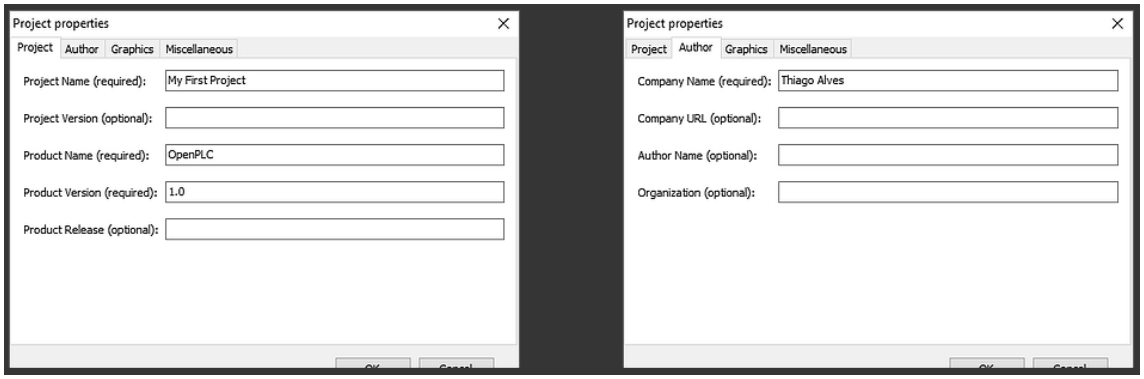


図 113

Graphics タブでは、FBD、LD と SFC タブの Grid Resolution（Horizontal と Vertical 両方）を 10 に変更します。これで、画面上でグラフィック要素が適当な位置に配置されるようになります。

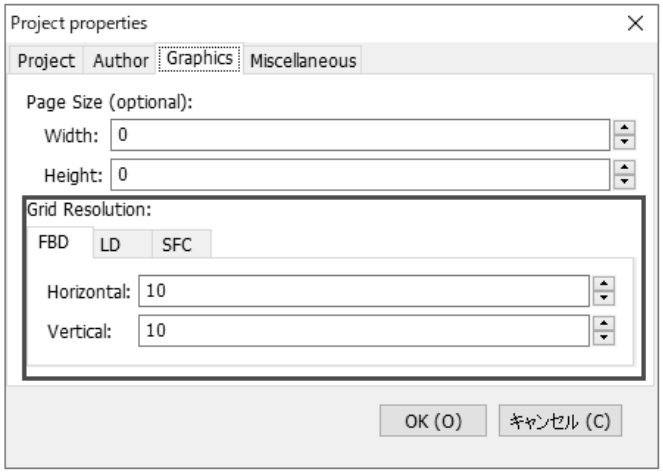


図 114

プロジェクト作成の最後に、OK をクリックします。直ちにプロジェクトが作られ、左側パネルにはプロジェクト名が見えます。今はまだ、中にプログラムの無い、空のプロジェクトです。プログラムを作成するためには、左パネルの角にある＋マークをクリックし、“program”を選択します。プログラム名を入力して、POU タイプが”program“、Language が”LD“であることを確認します。プログラム名にはスペースや特殊文字は使えません。

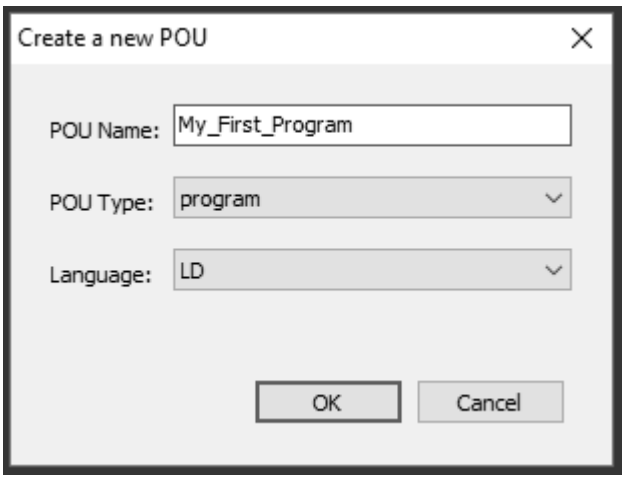


図 115

プログラムを書く前に、configuration と resource を作る必要があります。これらの項目は OpenPLC に、これから作るプログラムが何をするのかを教えます。configuration を作るためには、再び＋マークをクリックして“configuration”を選択します。**左パネルに“configuration0”の項目**が現れるので、それを**”Config0”**に変更します。左パネルの“configuration0”を 2 回クリック（ゆっくりとダブルクリック）すると、名称変更するため指定した項目の周りに箱が現れます。大文字の**“C”**を使って**”Config0”**とタイプし Enter キーを押します。再度＋マークをクリックして今度は、“Resource”を選択します。左パネルの **Config0 の下に“resource0”という名前の項目が現れます**。同じようにして、この名前を“resource0”から大文字“R”を使い**“Res0”に変更**します。この結果左パネルは次の様になります。

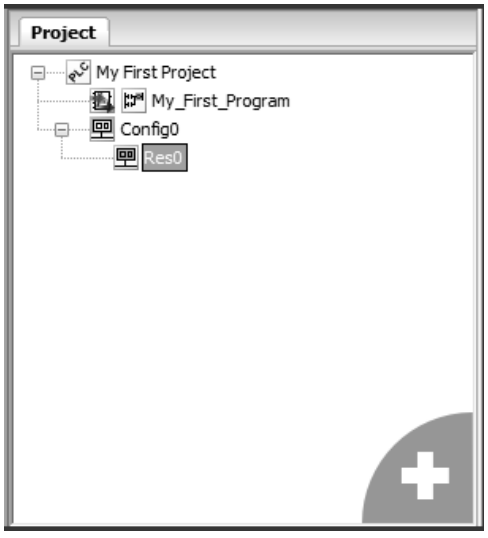


図 116

次にタスクとプログラムを動かすためのインスタンスを作ります。**左パネルの Res0 をクリック**します。上部にグローバル変数入力フィールド（そこでグローバル変数を作る）を表示しているメイン・ウインドウ、タスク・ウインドウ、インスタンス・ウインドウが表示されます。

新しいタスクを作るために、**タスク・ウインドウ内の緑の＋マーク**をクリックします。Name の下にタスクの名前（どんな名前でも OK）を入力します。**Trigging は“Cyclic”**を選択します。**Single は空白**のまま。**Interval では、“...”をクリックして、duration（継続期間）には 50msec と入力**します。これは、タスクが 50msec ごとに呼び出されることを意味します。もっと多く繰返し実行する必要のあるプログラムなら、希望の時間に調整できます。しかし、ゆっくりとした繰返し時間（1msec のように）を選んだ場合、スレーブデバイス（WiFi マイコン）CPU の能力 100%をプログラムが消費してしまい、結果としてプログラムが適切に稼働できなくなります（余裕がないとうこと）。**あらゆるスレーブデバイスで安全な数値は、通常 50msec です。**

一度タスクが作られたら、続けてインスタンスを作ります。**インスタンス・ウインドウ内の緑＋マークをクリックします。**Name の下にインスタンスの名前を入力します（どのような名前でも OK）。**Type はプログラムを選択**します。**Task は今作ったタスクを選択**します。これでプロジェクトを作る準備が出来ました。Resource ウインドウは次の様になっています。

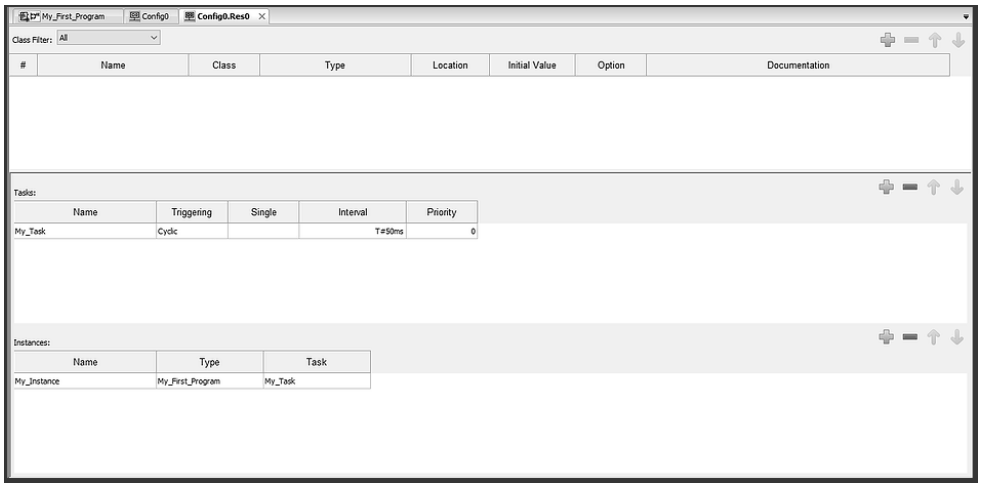


図 117

今が、空のテンプレートとしてプロジェクトを保存するのに良いタイミングです。新しいプロジェクトを作成する際、毎回このように全ての情報を入力しないで済ますためには、ここで保存してテンプレートにすると良いでしょう。新しいプロジェクトを作りたいときには、空のテンプレートを開いて、それを異なる名前で保存し、プログラムを上書きコーディングします。空のテンプレートとしてプロジェクトを現状のまま保存するには、File--->Save、異なる名前で保存するには、File--->Save Asとクリックし、異なる名前でプロジェクトを保存します。

プロジェクトが作成できたら、これからプロジェクトのラダー図を描きます。**ラダー回路エディタを開くために左パネルのプログラム名をダブルクリックします（図 116 の My_First_Program となっているアイコン)。**画面上部では、変数を予約します。中央部は回路図に使います。変数を追加する事から始めます。緑＋マークをクリックして、次の３つの変数を追加します。

【注意】下図は、Web で公開されている汎用的な図ですが、スレーブデバイスとして WiFi マイコン（ESP8266）を使用した実習では、Location は次の様に入力します。

PB1→％IX100.2（※）、　PB2→％IX100.1、LED→％QX100.0

（※）PB1→％IX100.2 を指定するのは、Web 掲載の PB1→％IX100.0 の DI ポートを使用すると、Reset 時のマイコンの動作が不安定になるからです。これは該当の信号が基板内部で電源電圧にプルアップされていることに起因します。

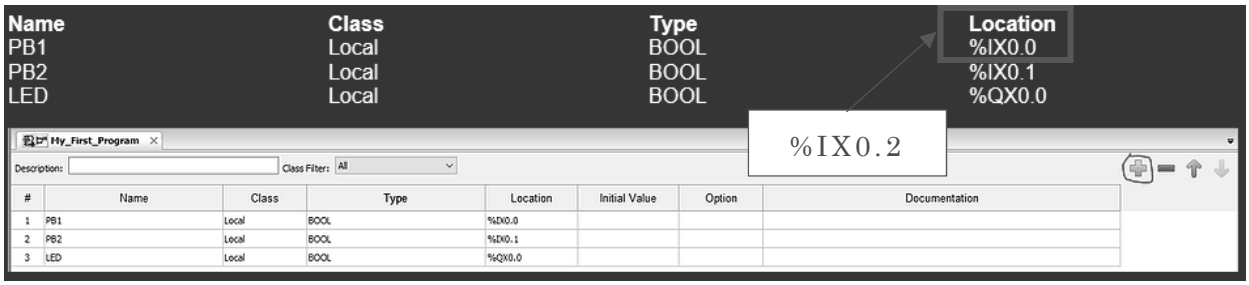


図 118

このプログラムで行いたいことは、PB1 が押されたとき LED が点灯し、PB2 が押されるまで点灯していることです。これを行うには、次のようなラダー図を描きます。

【注意】この回路の配線は【本編】で図解しています。

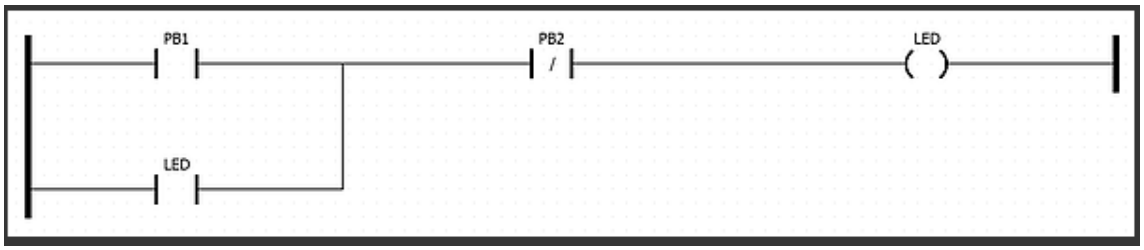


図 119

エディタ上でこの回路を作るため、ツールバーの**電源レールアイコンをクリック**して、まず左電源レールを追加します。**（または、右クリックで追加--->Power Rail）**

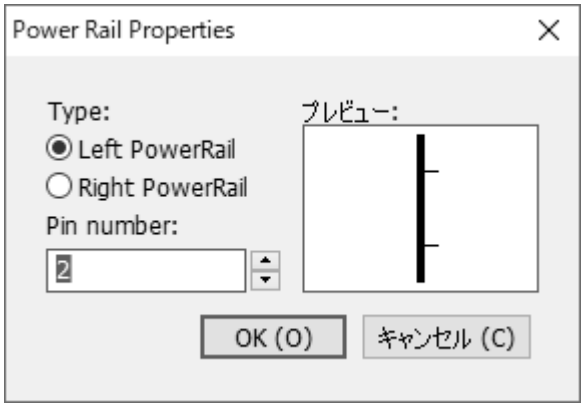


図 120

左電源レールの横木用の枝を 2（**Pin number を 2**）**に設定**します。ツールバーの**接点ボタンをクリックするか、エディターウインドウの空白部分を右クリックして、Add--->Contact と選択して接点を追加します。**ウインドウにそれが表示され**“Variable”引数のプルダウンで PB1 を選択**します。

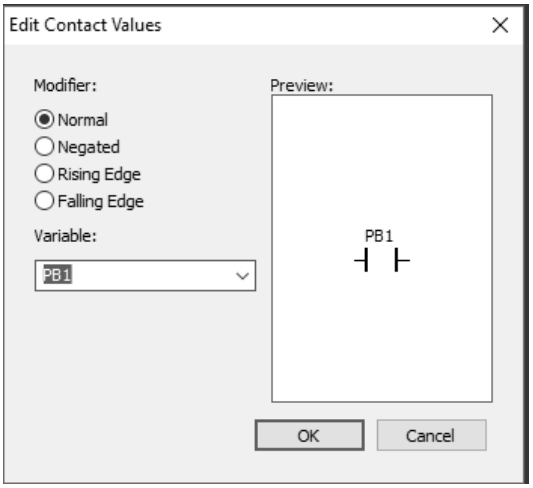


図 121

他の 2 つの接点について、同じ手順で、1 つは PB2 と関連付け、他は LED と関連付けます。**PB2 接点については、Modifier で Negated（ブレーク接点：押したときに回路が開く）を選択します。**最後は、ツールバーの**コイルボタンをクリックするか、エディターウインドウの空白部分を右クリックして、Add--->coil と選択し、コイルを追加します。**新しい**コイルと LED 変数を関連付け**たら、右電源レールを追加します。ラダー図エディタ上で部品を綺麗に配置して次の図の様になります。

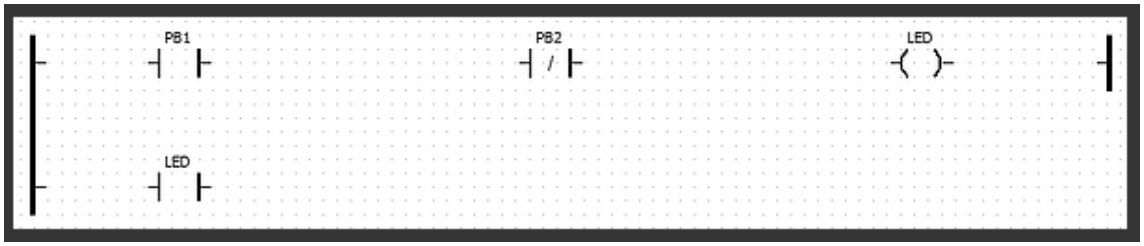


図 122

最後にドラッグ操作で部品の両端に線を描いて全部品を接続します。PB1 の左側を接続して、LED 接点を左電源レールと接続します。PB1 の右側を PB2 と接続し、PB2 の右側を LED コイルと接続し、LED コイルの右側を右電源レールに接続します。LED 接点の右側と PB2 を接続して並列 LED 接点回路を描きます。プロジェクトは最後に次の様になっているはずです。

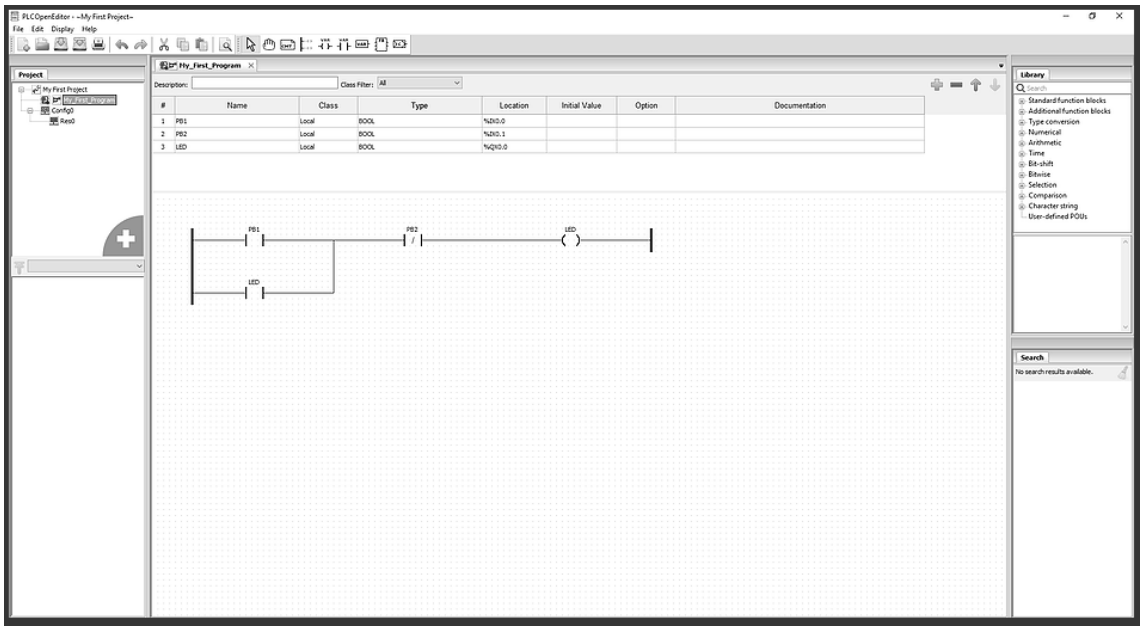


図 123

この回路は、初め LED は消灯しています。PB1 を押した瞬間 LED が点灯します。一度 LED が点灯すると、回路上の PB1 ボタンをバイパスして LED に電流が流れ PB1 を押し続けなくても LED は点灯を続け

ます。これが**【ラダー回路の巧い仕掛け】**で、**【出力を接点として巧妙に使うことができる】**のです！！PB2 を押せば、LED は消灯します。PB2 は通常閉じていて、押されたときに開く**【ブレーク接点】**なので、PB2 が押されると回路は開いて LED が消灯します。

最後のステップは、OpenPLC が理解できるフォーマットでプログラムを生成する事です。その手順は、File--->GenerateProgram とクリックして【 .st】（拡張子） ファイルを保存します。このファイルは、OpenPLC が理解できる言語で書かれたラダー回路のプログラムです。**後でこの【 .st】 ファイルを Web インターフェースでアップロードし OpenPLC で稼動させます。**

２．OpenPLC Webserve へのアクセス

OpenPLC runtime は、OpenPLC の設定や新しいプログラムをアップロードするために、**【ビルトイン Webserver】**を持っています。この Webserver はブラウザから【PC の IP アドレス：8080】でアクセスできます。

OpenPLC runtime for Windows では、OpenPLC Rutime を起動した後、ブラウザを開き localhost:80080 とタイプすれば OpenPLC Webserver にアクセスできます。

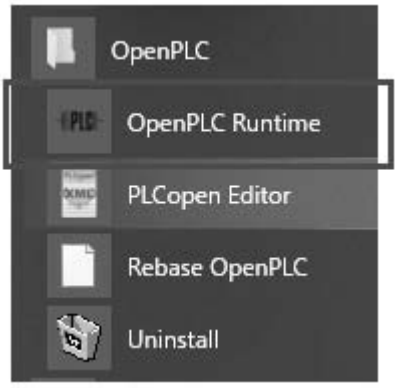


図 124

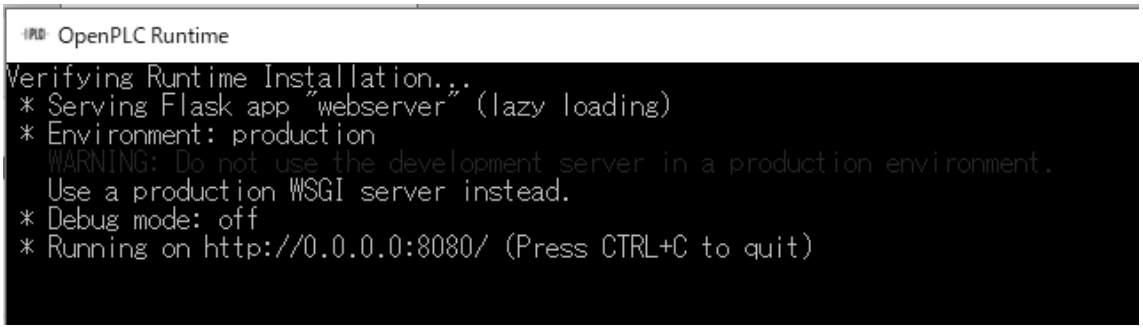


図 125

初めて OpenPLC Webserver にアクセスすると、次の様にブラウザにログインページが表示されます。

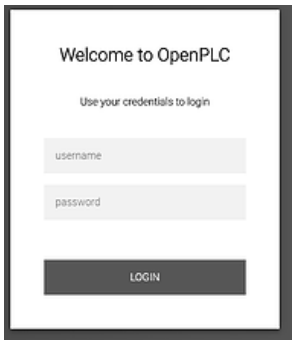


図 126

OpenPLC v3 をインストールした後、設定した ID とパスワードでログインします（変更していなければ ID と Password は共に【openplc】になっています）。

３．スレーブデバイスの追加

WiFiマイコンをスレーブデバイスとして登録します。左側メニューで“**Slave Device**”を選択して“**Add new device**”ボタンをクリックします。

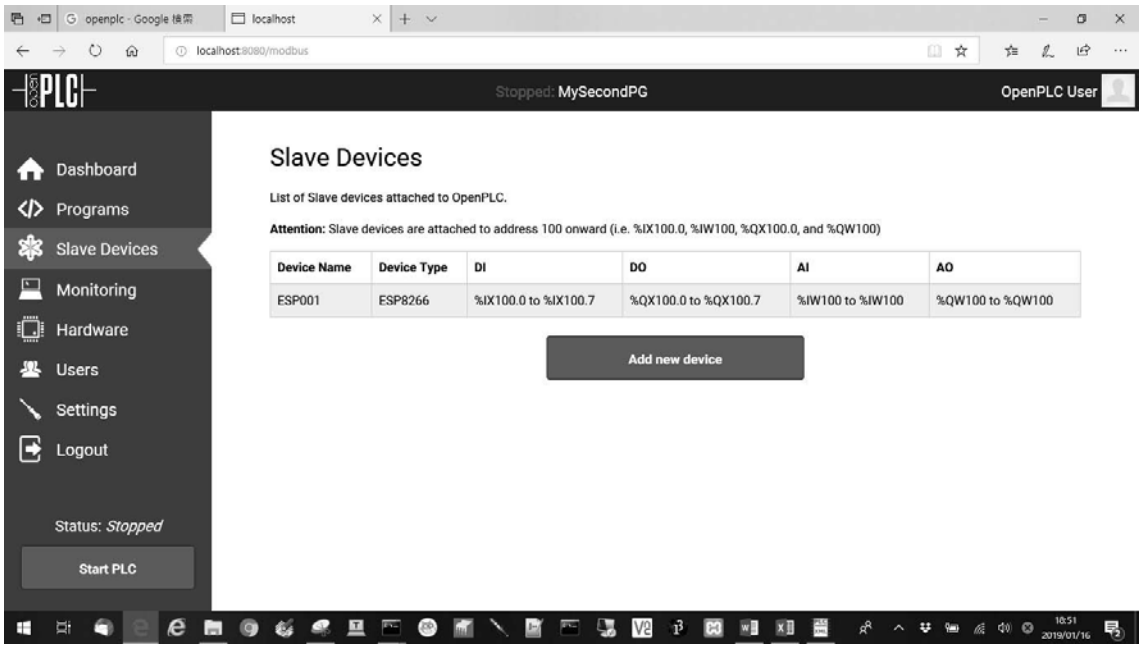


図　127

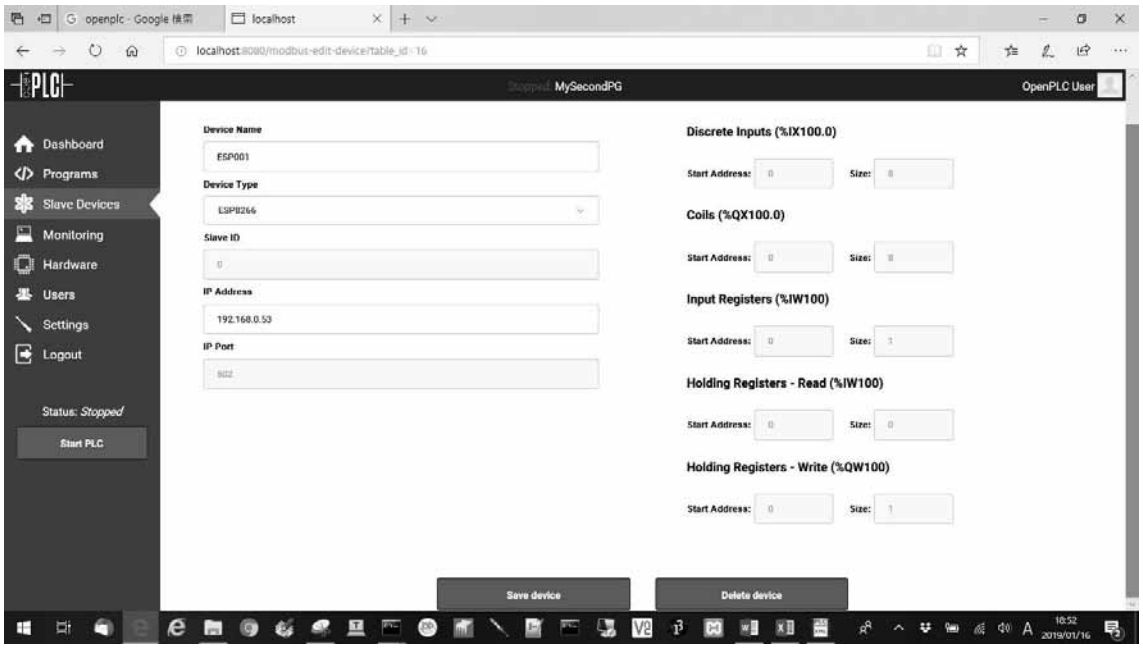


図 128

そこに次の様に入力します。

Device Name ---->適当な名称

Device Type ---->ESP8266 を選択

IP Address ---->準備でメモした WiFi マイコン（スレーブデバイス）の IP アドレス（準備の際にメモしたもの）

画面下の“Save Device”ボタンをクリックすれば、WiFi マイコンがスレーブデバイスとして登録されます。IP アドレスが異なれば、複数のスレーブデバイスを登録することができます。

４．プロジェクトのアップロード

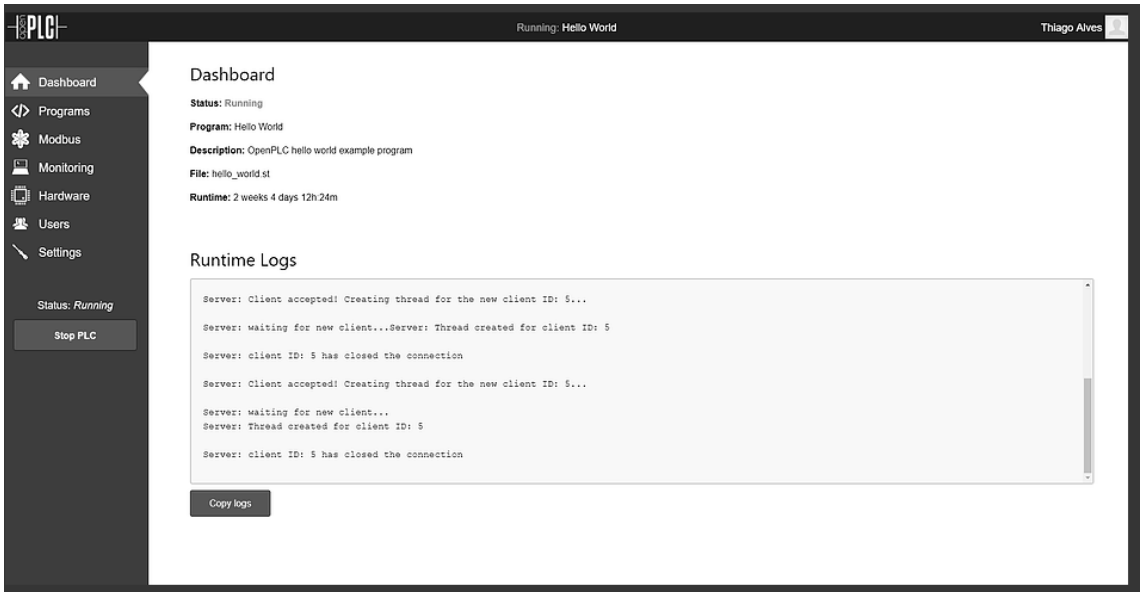


図 129

既に一度 upload したプログラムを再び upload するためには左メニューで Program に移動し、最近 upload したプログラムを reload すれば以前 upload したプログラムに戻せます。新しいプログラムを upload するには、“参照”をクリックし、ラダー図エディタで保存した【.st】ファイルを選択し“**Upload Program**”をクリックします。

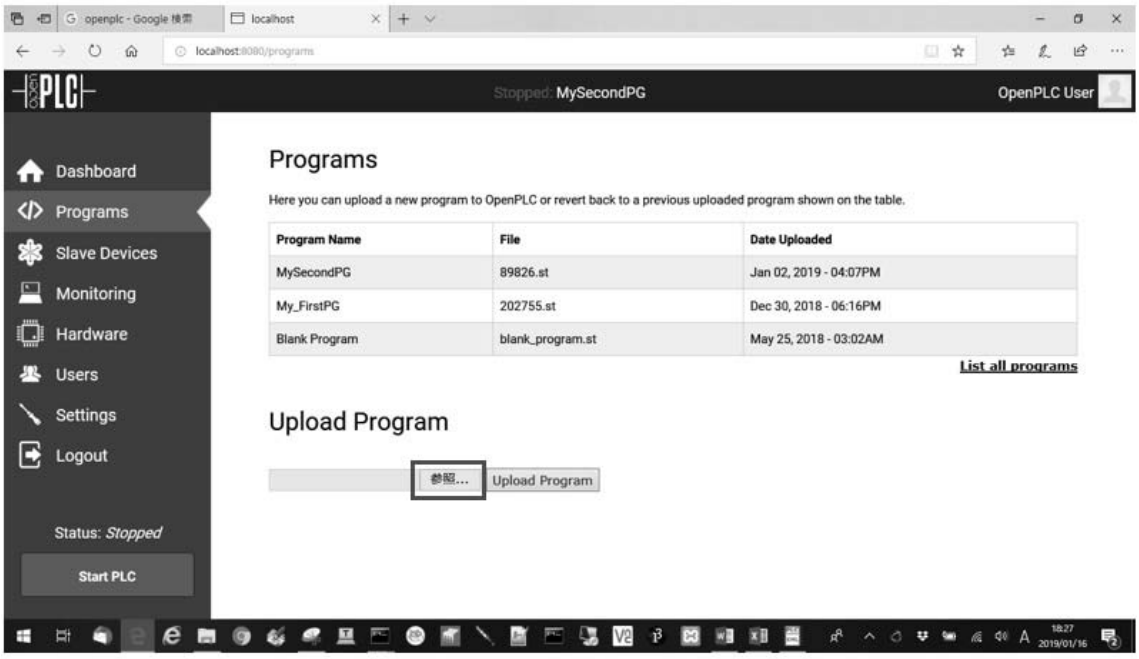


図 130

表示されるウインドウに、プログラムの内容が分かる様に、プログラムの情報を書込みます。

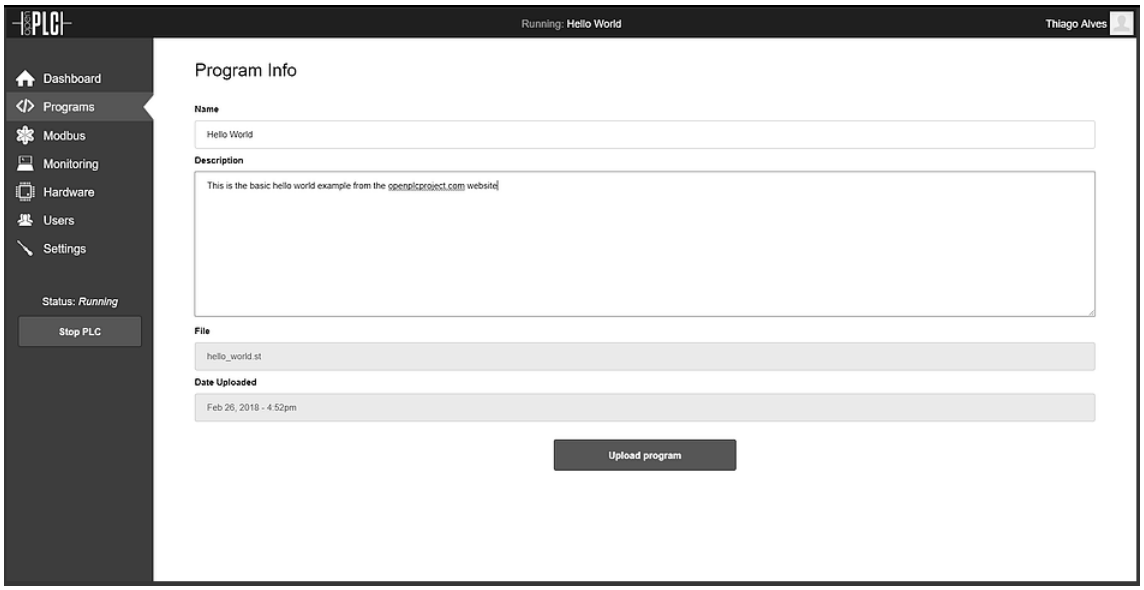


図 131

upload すると直ぐにコンパイルが始まります。

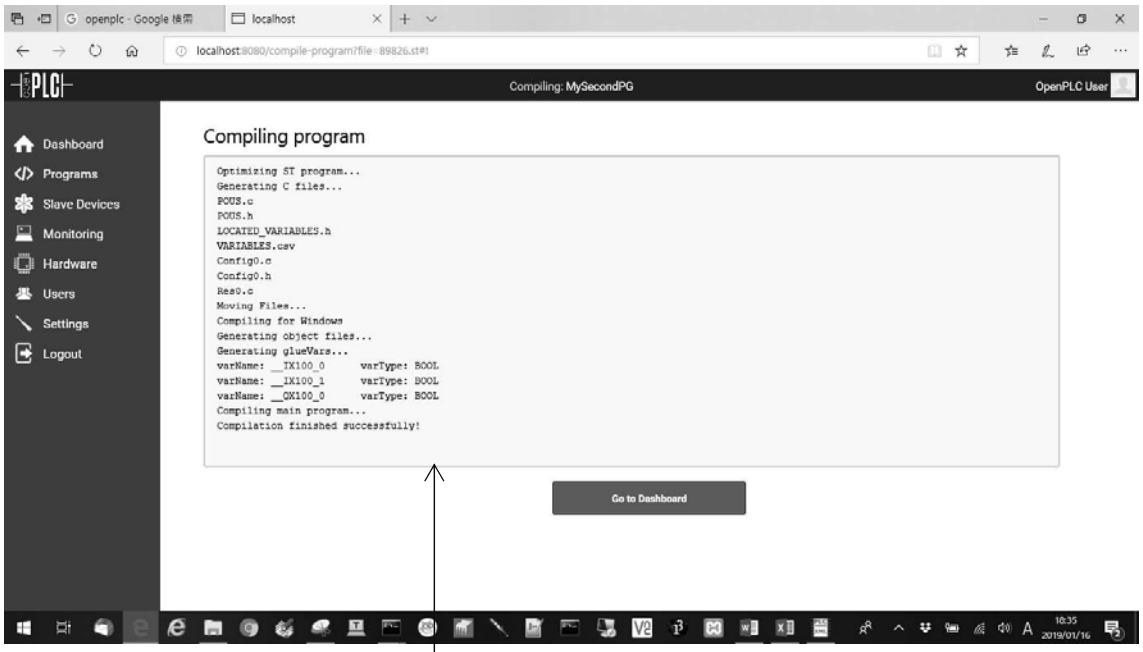


図 132

コンパイル時のログは log box に表示されます。エラー無しで終了したら、ウインドウ中央下の“**Goto Dashboard**”をクリックして Dashboard に移動します。ウインドウ左下の“**Start PLC**”をクリックすると PLC プログラムが動き始めます。

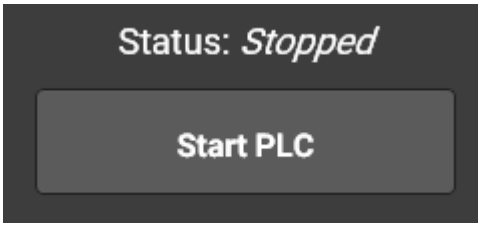


図 133

この時、スレーブデバイス（WiFi マイコン）は、USB ケーブルで PC と接続しておく必要があります。

【まとめ】

OpenPLC を使い、WiFi マイコンをスレーブデバイスとして利用する手順について、整理しておきます。マイコンには専用 Runtime アプリが書込まれていて、WiFi 接続した際に割り当てられた IP アドレスが既に分かっているものとします。

１．マイコン周辺の回路を作成する。

※ESP8266 の場合 DI・DO 共に 4 点が利用できる。

２．PLCOpen Editor でプロジェクトを作り、その中に回路（ラダー図）を描き保存する。

--->プロジェクトファイル【.xml】ファイルとして保存される。

３．PLCOpen Editor でファイル--->Generate Program と辿り、ラダー図を【.st】ファイルに変換する。

４．マイコンを USB ケーブルで PC に接続する（通電する）。

※この時、WiFi でアクセスポイントに接続されます。

５．PC 内のアプリ【OpenPLC Runtime】を起動します。下図のウインドウが開きます。

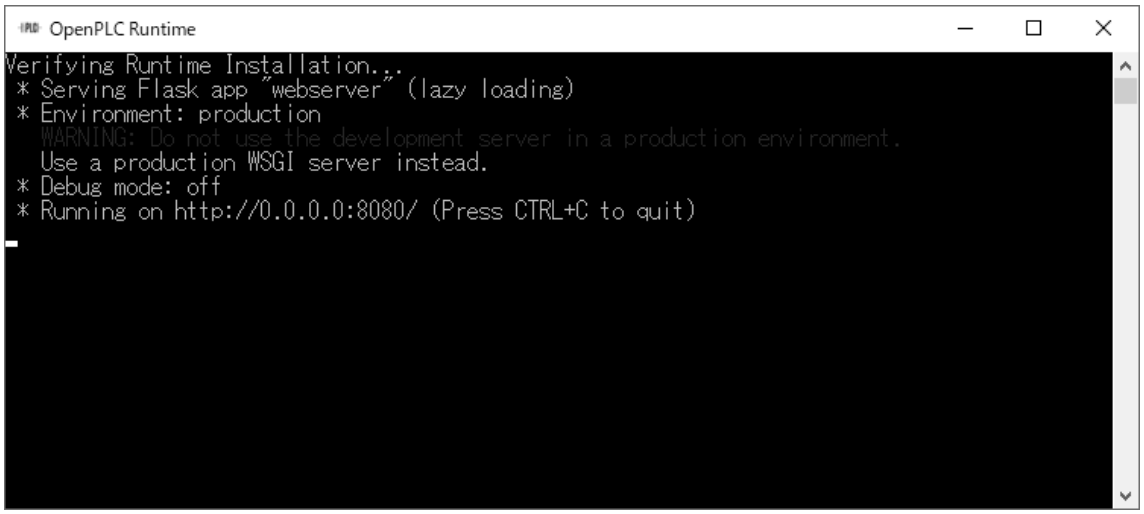


図 134

6．PCのブラウザで【http://localhost:8080/】にアクセスし、ログインします。

7．左メニューのSlave Devicesに移動し、スレーブデバイス(WiFiマイコン：ESP8266)を追加登録する。

※この時、マイコンのIPアドレスも登録します。

この登録は、最初に1度行うだけです。

8．OpenPLC Dashboardが表示されるので、左メニューでProgramを選択し、表示される画面で、【.st】ファイルをUploadする。

--->コンパイルが行われる。

9．Dashboardに移動し、START PLCボタンを押して、ラダープログラムを開始スタートする。

以上でPC+WiFiマイコンで構成するPLCが稼働します。後は、SWを押すなどして、回路の動作を確認します。

PLCラダー図の修正を行う場合は、Stop PLCで停止して2以後のステップを繰り返します。

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」

(Society5.0 等対応カリキュラムの開発・実証)

富山県をモデルとした「モノづくり」現場に

IoT を導入する中核的人材育成

製造 IoT 基礎演習　テキスト教材

平成３１年　３月発行

学校法人浦山学園　富山情報ビジネス専門学校

〒939-0341 富山県射水市三ヶ５７６
TEL：　0766-55-1420　Fax　：0766-55-0757