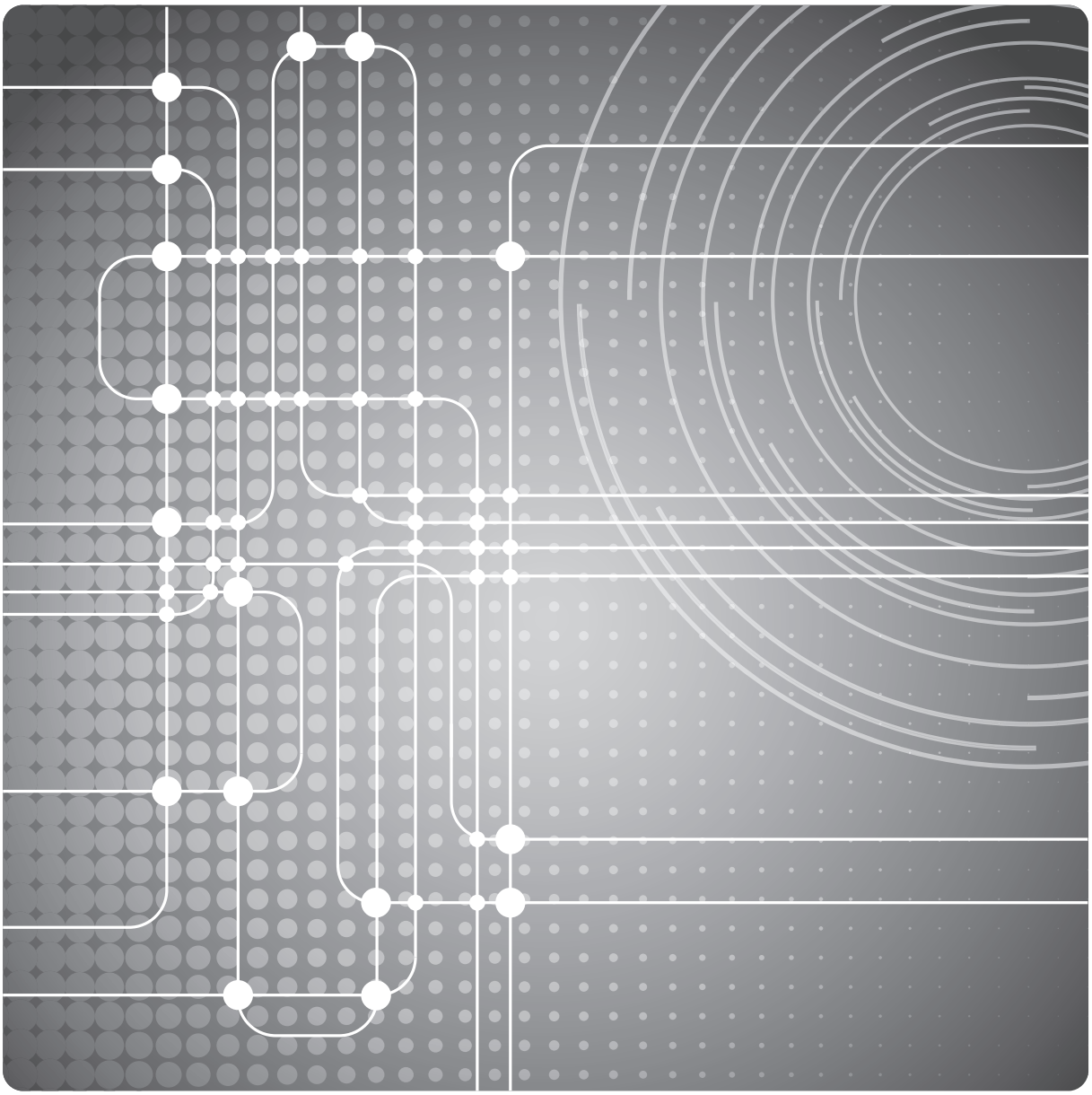
2019年度「専修学校による地域産業中核的人材養成事業」

IoT（組込み）統合システム開発教材

IoTの種



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

2019年度「専修学校による地域産業中核的人材養成事業」

IoT（組込み）統合システム開発教材

IoT の種

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

このテキストについて

このテキストは【IoT の種】です。2018 年度に作成した【IoT へのドリル】の続編になっています。今年度はマイコンのシステム開発経験者でも、実習しごたえのある内容になっています。マイコンのプログラムライタ作成から始めます。序盤は容易に行える実習ですが、中盤から複数種類のマイコンを相互接続してマイコンが他のマイコンを制御するシステム開発を行います。終盤は 3 種類のマイコン 4 個を使用して、複数名でシステム開発を行う内容としました。解説に従い実習すればシステムは完成するのですが、単独で（一人で）開発を行うのではなく、二人以上のプロジェクトチームを編成してシステム開発を分担します。実際のシステム開発と同じように、人と人とのつながりをもって、システムを完成に導いてもらいます。【手間のかかる部分】があると思いますが、最終回でシステムが完成したときには、今までと違う【達成感】が味わえるのではないでしょうか。

全 16 回の講座を通して行うことは【IoT システムに育つ種】を蒔いて、それを少しずつ育てる事を経験して【IoT 技術者の種】を育てることです。そこで、このテキストは【IoT の種】と命名しました。

皆さんの成功を祈っています。

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

【IoT の種】

目次

第1回　LED

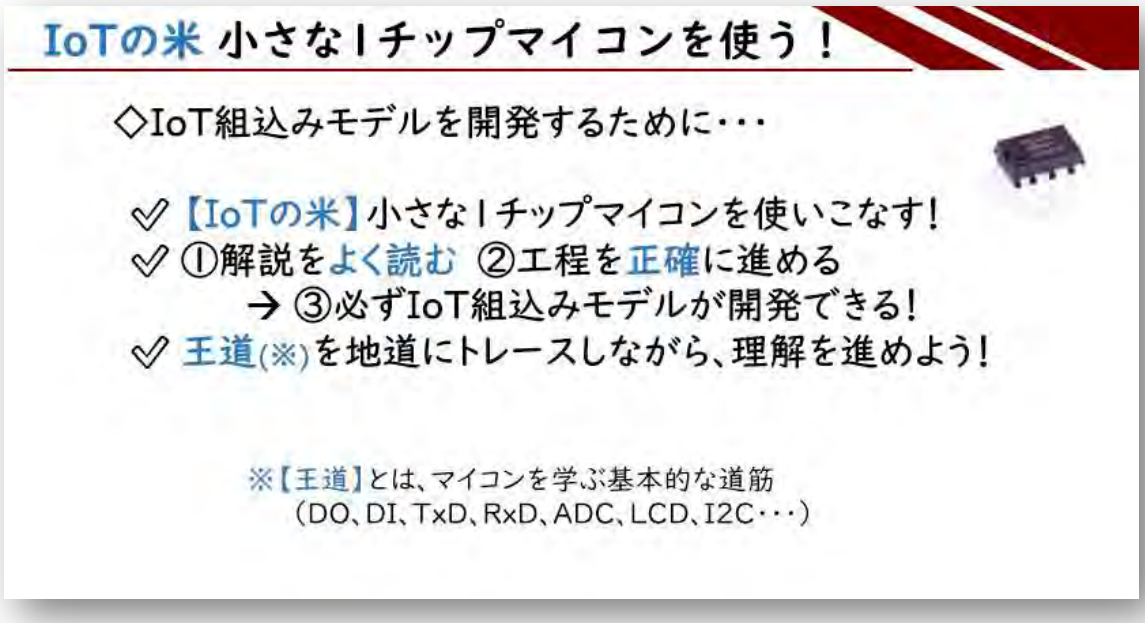


図 1

IoT 組込みモデルを開発する足掛かりとして、最小規模のマイコンを使用してみましょう。小さなマイコン【IoT の米】を使いこなすために、解説に従い工程を正確に進めることが大切です。すでにマイコン使用経験がある方も種類の異なるマイコンには、慎重に対応しましょう。地道に進むべき道をトレースすることが、成功の秘訣です。

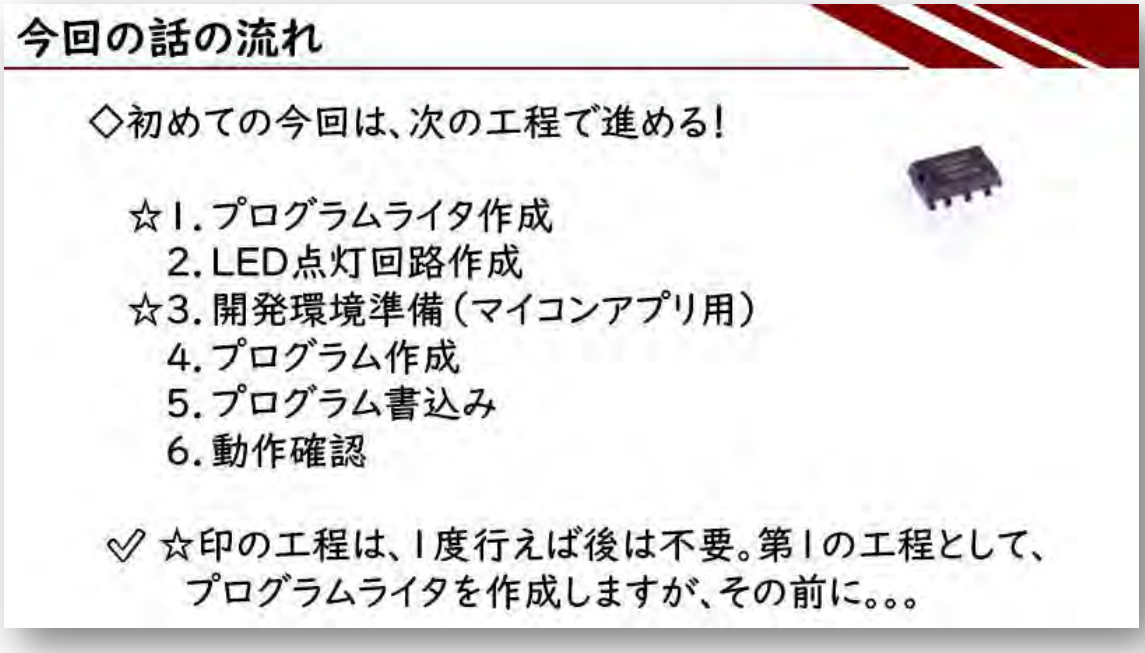


図 2

行うべき工程を図に示しました。☆マークの工程は、一度だけ行えばよい作業です。二回目以後でも行う工程は、慣れがミスを誘発するので、毎回きちんと確認を行うようにしてください。実際の作業に入る前に使用するマイコンについて説明します。

図　3

PICAXE というマイコンは入手が容易です。公開されている開発環境は、とても安定していて各種 OS 向けの IDE が整っています。また、教育向けと言うことも有り、開発環境ではフローチャートやブロックによるプログラミングもサポートしています。教育用と謳いながらも開発対象に応じてチップの種類が選択できるという自由度があります。一連の講座の序盤では図の 08M2 を、中盤では 28X2 を使用します。終盤ではそれら 2 つとは別のマイコンを使用します。3 種類のマイコンを用いて１つのシステムを構成します。

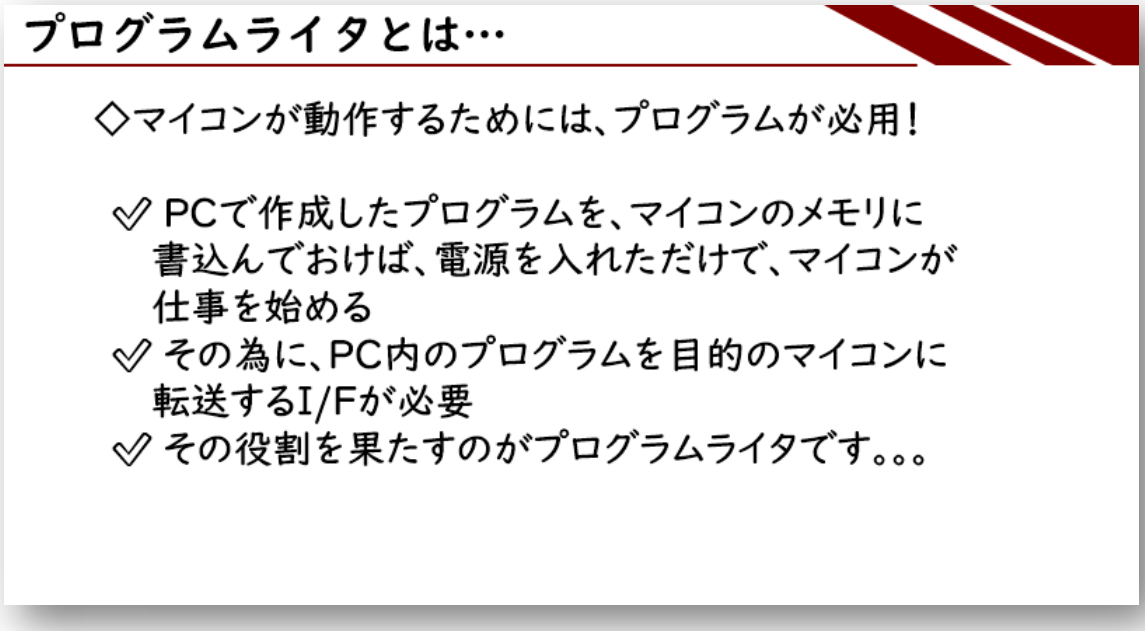


図　4

マイコンに所望の動作をさせるためには、プログラムが必要です。PCで開発したプログラムを、マイコン内部のフラッシュメモリに書き込む際に、通信インターフェースの役割を担うのはプログラムライタです。一般にプログラムライタというと、そのマイコンに特化した機器や専用のインターフェースを指しますが、ここで使用するものは、汎用的なUSB－シリアル I/F です。

図　5

このI/Fが行うのは、PICAXEとPCがシリアル通信できるようにするための、信号の【電圧レベルと論理レベルの調整・変換】です。PICAXEとPC間のシリアル通信ができれば、PC内の開発環境とPICAXEが協調してPICXEプログラムをフラッシュメモリに書込んでくれます。PICAXEの電源はUSBケーブルを通じてPCからUSBーシリアルI/Fに供給される5Vの電源を使います。

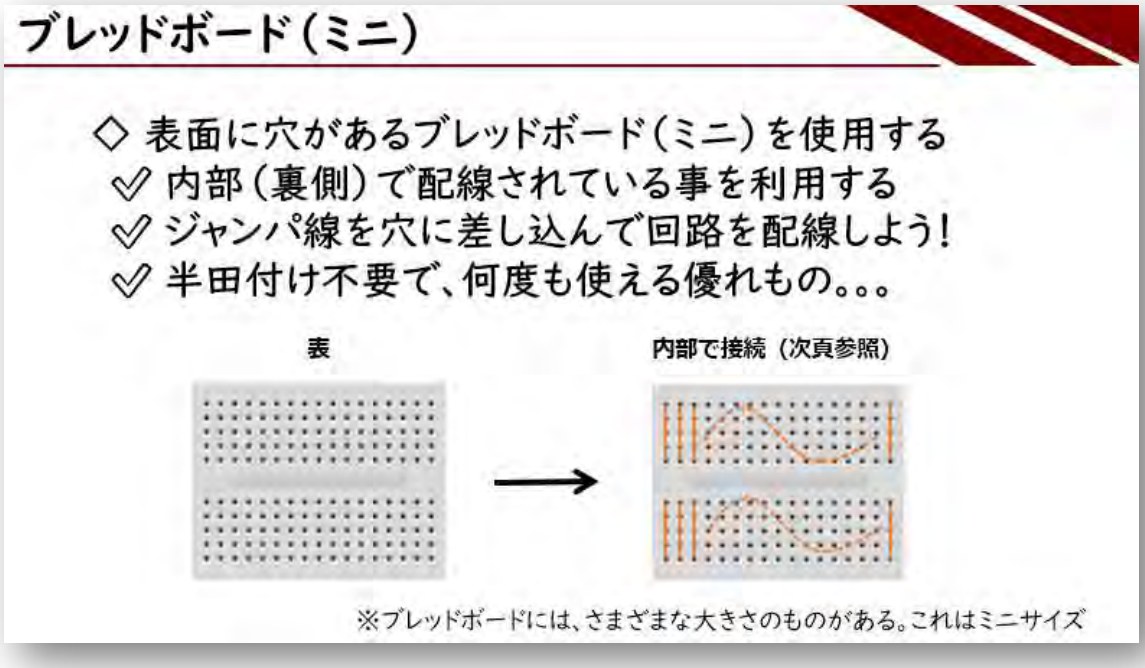


図　6

USB－シリアル I/F 用の基板は小さいので、小さいブレッドボードを使います。ブレッドボード表面に導線などを差し込む穴が沢山有り、内部で電気的に接続されています。

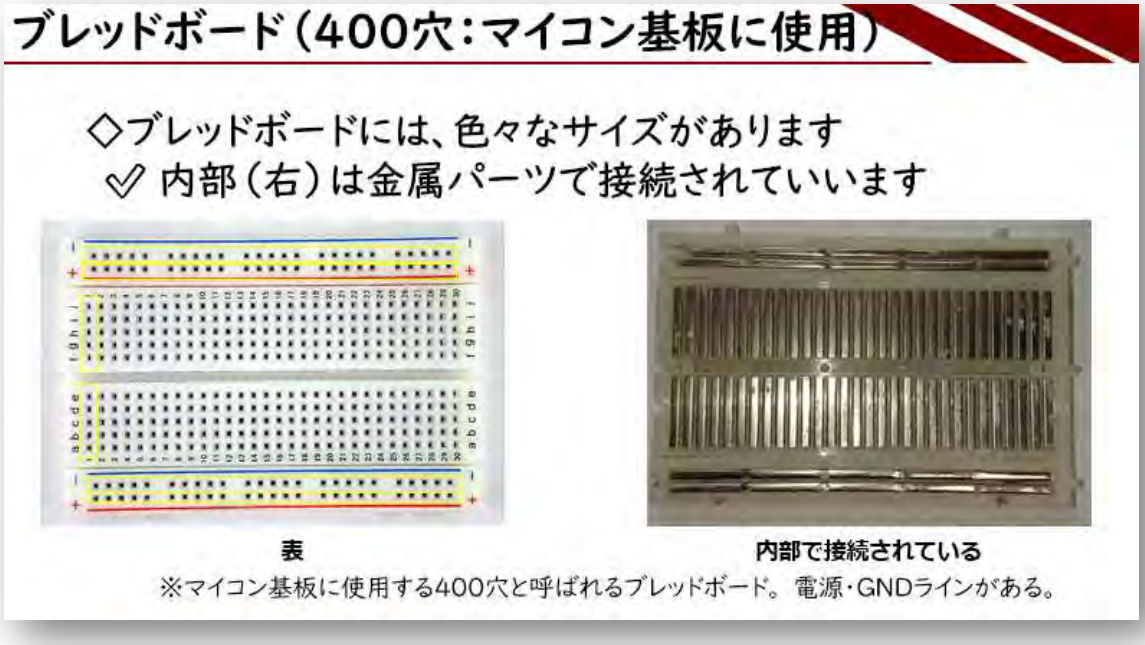


図　7

図は少し大きなブレッドボード内部を裏側から見た様子です。内部の接続が、どのようになっているかが分かります。表から見た際に、どの部分が電気的に繋がっているかを考えながら配線をしてください。また、基板の周囲には穴位置を示すためのアルファベットと数字が印刷されていますが、数字の小さい方を左に向けて使用するのが一般的です。（配線の都合やジャンパ線による引っ張りなどで基板の向きが整わないこともあります。）

図 8

ブレッドボード上に配置した、マイコンを含む様々なパーツを相互に接続するために、図のようなジャンパ線を使います。両端はブレッドボードの穴に差し込んで固定できる、金属製のピンになっています。半田付けせずに配線が行えて、回路の用途が済んだ後は取り外して別の回路の配線に何度でも使用できるので、大変便利です。

図 9

電気を用いる回路には、図のような抵抗器が多く使われています。抵抗器の役割は、電流の調整と電圧の調整が主です。一般によく使用される抵抗器では、抵抗の値を色の帯【カラーコード】で表現しています。WEB で「抵抗のカラーコード」と検索すると見やすい表が容易に見つかるはずです。カラーコードを調べられるスマートフォンアプリもあります。カラーコードを完全に記憶するのは大変ですし、誤った抵抗値のものを選んでしまうのも困ります。新しい抵抗器を使用する際には、抵抗値を書いた紙片をテープで貼りつけて使うようにしても良いでしょう。こうすれば後に再利用する際に間違えずに済み、回路作成の時間が短縮できます。図右下のようにリード線を直角に曲げて使用します。

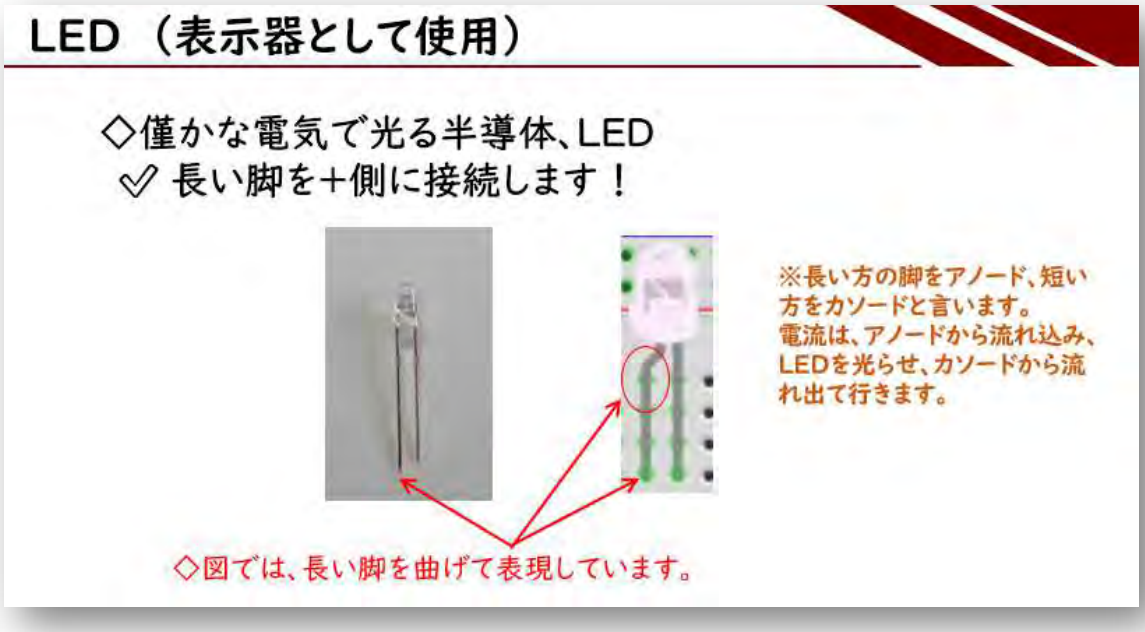


図　10

これから行う一連の実験にはLEDを使用しています。LEDはLight Emitting Diodeの頭文字を採ったもので、その名の通り発光ダイオードですが、今ではLEDの方が一般的です。LEDには、ダイオードというだけあって極性があります。図の様に2本あるリード線の長さの違いで極性を示しています。長い方をアノードと呼び＋側に接続します。短い方をカソードと呼び－側に接続します。長さの違い（＝極性）が分かるように、配線図では長い方を曲げて表現しています。

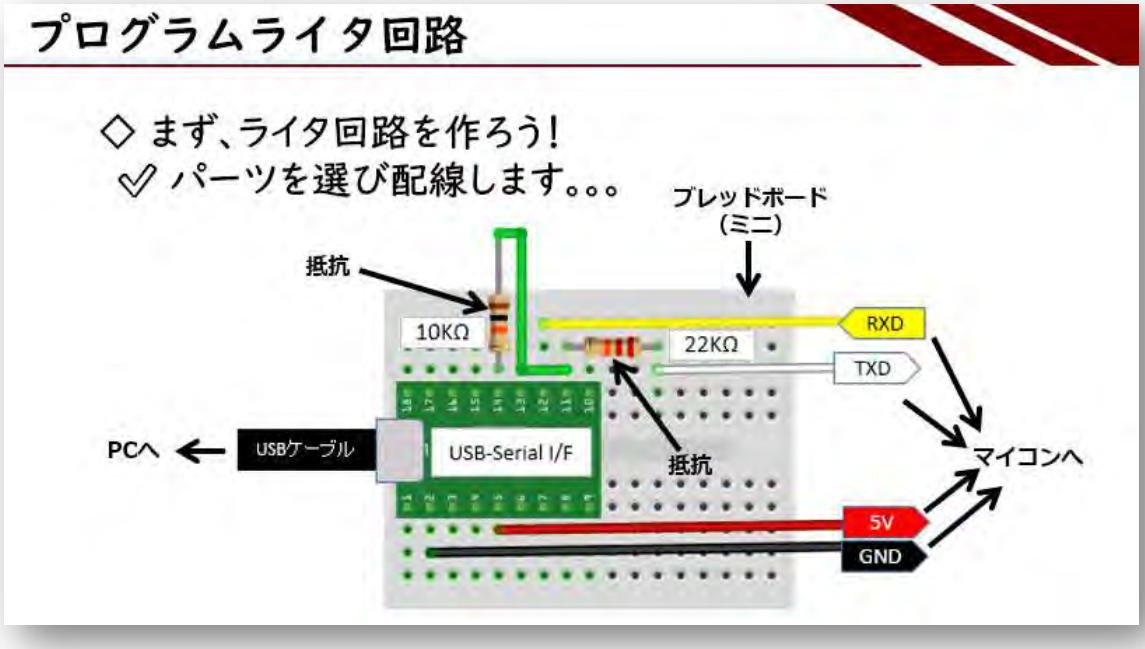


図　11

図に従い、プログラムライタ回路を作成します。10kΩの抵抗は一方のリード線を複雑に曲げて表現していますが、実際の配線はリード線を単純に曲げた配線にします。次頁の写真を参照してください。図の基板下部から、5V と GND の配線をマイコンの電源として使用します。USB ケーブルを PC に接続すると、PC から 5V の電源が供給されるので、それをマイコンの電源とします。

図　12

完成したプログラムライタ回路は図のようになりました。ブレッドボードと USB－シリアル I/F 基板の向きに注意してください。

これからマイコンを使用した回路を作っていきます。

図 13

初めに LED を点滅させるシステムを作りましょう。マイコンから High/Low の信号を出力して、LED の点灯・消灯を行います。使用するマイコンの電圧レベルで High＝5V＝LED 点灯、Low＝0V＝LED 消灯です。

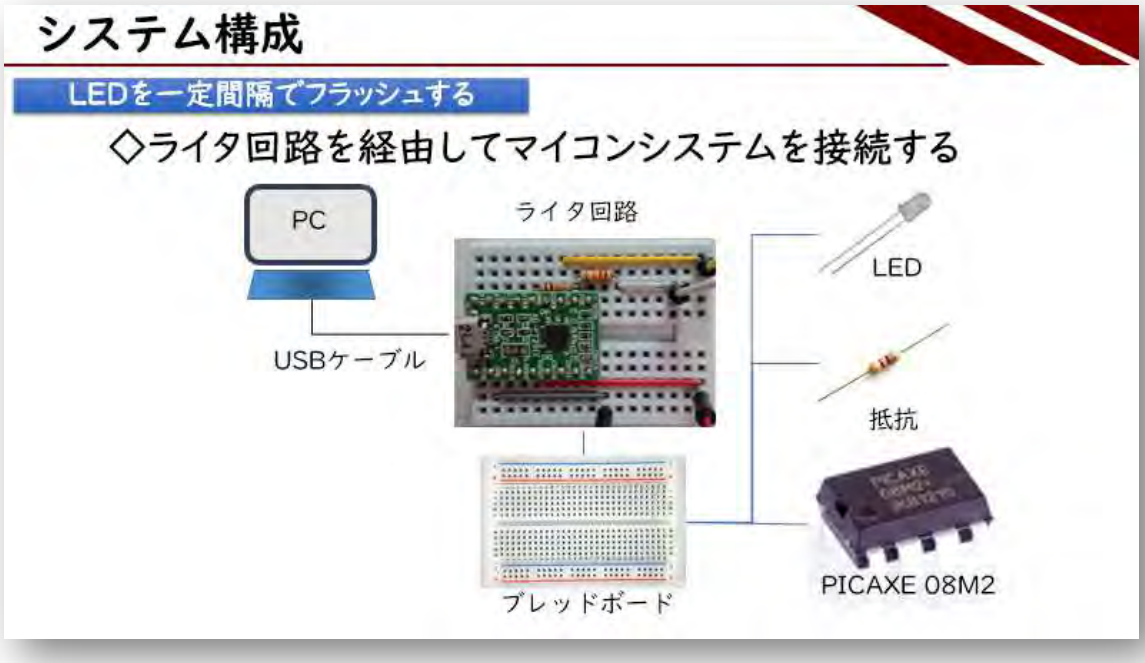


図　14

PC とプログラムライタ回路は USB－シリアルケーブルで接続し、ライタ回路とマイコン基板間は、ジャンパ線で接続します。図に示すパーツを使用しますが、詳細は回路図を見て判断し、使用するパーツをそろえてください。以後の各回も同様です。

図　15

初めに使用するのは PICAXE の中で一番小さい 8 ピンのものを使用します。型番は PICAXE08M2＋となっています。図にピン配置と役割を示します。マイコンを含めて IC のピンの番号は、マークまたは切り欠きを上に向けて、左上から左周りに 1,2,3・・・と数えます。一番下まで数えたら、右下のピンから上にかぞえまます。

使用するマイコンのピンには○を付けておきました。LED への出力には 5 番ピン：C.2 を使用します。マイコンの電源は USB ケーブルを通じて PC からライタ基板に供給されている 5V の電源を使います。

図　16

新しいマイコンやICの脚は、製造過程で脚が広がって出来上がります。図は横から見た様子です。この脚の広がりを右の様に成型して、基板に挿すようにします。この広がりを直さずにブレッドボードに挿し込むと、脚が大きく曲がってしまい、場合によっては折れてしまうことも有りますので、注意して取り扱ってください。ブレッドボードに挿し込んだ後は、横から見てチップのお腹がブレッドボードに並行で近くなるよう、押し込んでください。ブレッドボードから取る外す場合は、ピンセットなどで、少しずつ浮かして取り外します。

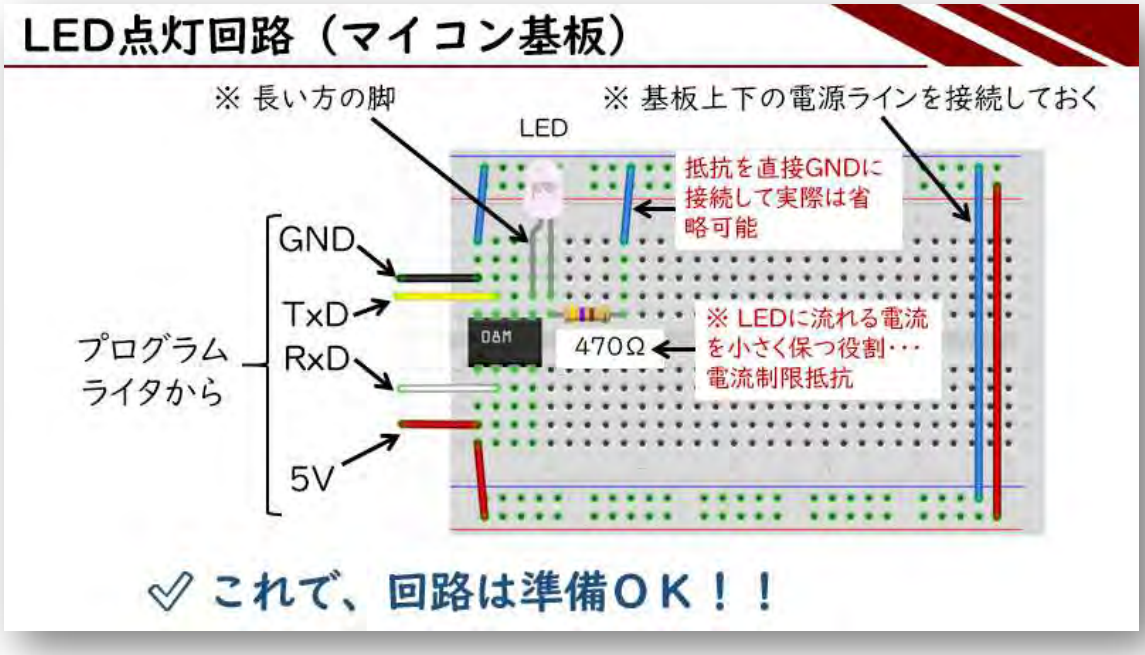


図　17

一般的な回路作成の順番は、背の低いパーツから先に配置するのが良いとされています。背の高いものを先に配置すると、残りの背の低いパーツ配置の邪魔になるからです。LED 点灯回路の配線は使用パーツも少ないので順番にこだわらなくても容易に回路が作成できますが、使用パーツが多くなってくると、上の順番を守った方が作業時間の短縮にもなり、スムースに配線を行えます。その結果綺麗な仕上がりの回路が作れます。

配線が出来たら、十分確認をして下さい。特に電源（5V と GND）を逆に配線していないかを調べてください。これを誤ると使用パーツが壊れたり、PC の USB 回路が壊れたりします。

【重要】ショート防止のために、配線を行う際は USB－シリアルケーブルを PC から外して作業してください。

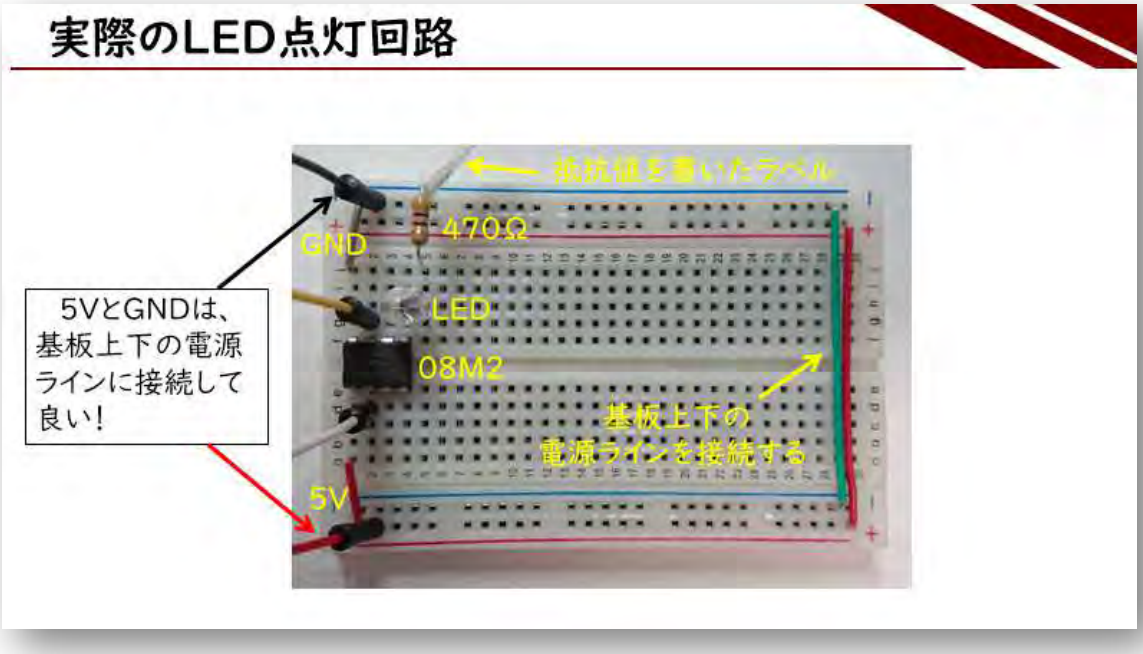


図 18

完成した LED 点灯回路を示します。ブレッドボード右側の電源配線は無くても動きますが、後に利用するので初めから配線しておきます。回路図の抵抗の片側は、ジャンパ線により GND に接続していますが、実際は抵抗を直接 GND ラインに接続して構いません。LED によって抵抗の配線が見づらくなるので、分かり易い図にしてありますが、ジャンパ線が省略できれば、作業の手間が減り、誤りの個所も減ることになります。今後も工夫して回路を作ってください。

回路が出来たら、よく確認をしましょう。

図　19

PICAXE の開発環境を整えます。PICAXE で検索すると、図の様なページがヒットします。

図 20

このページの Software をクリックします。

図　21

そのページ中に PICAXE Editor6 があります。

図　22

これをダウンロードします。ダウンロードしたファイルをダブルクリックすれば、PICAXE Editor がインストールされます。

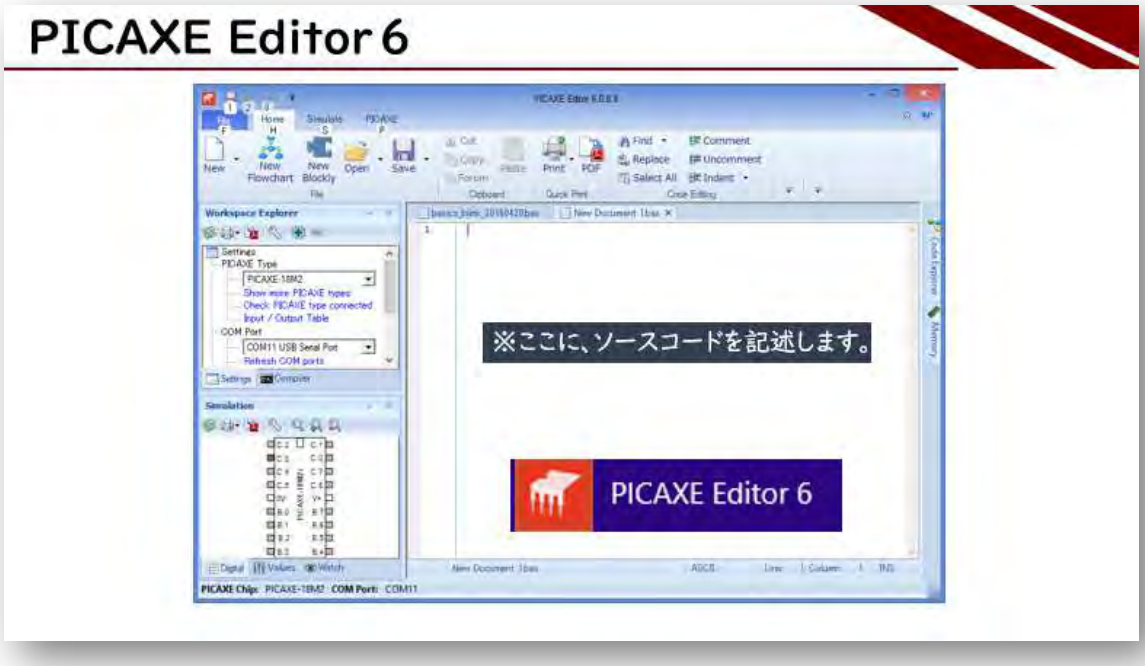


図　23

PICAXE Editor を起動すると、図に示すウインドウが開きます。これが PICAXE 専用の IDE です。右側の広いウインドウにソースコードを記述します。Home Tab の New で新しいプログラムを作成します。

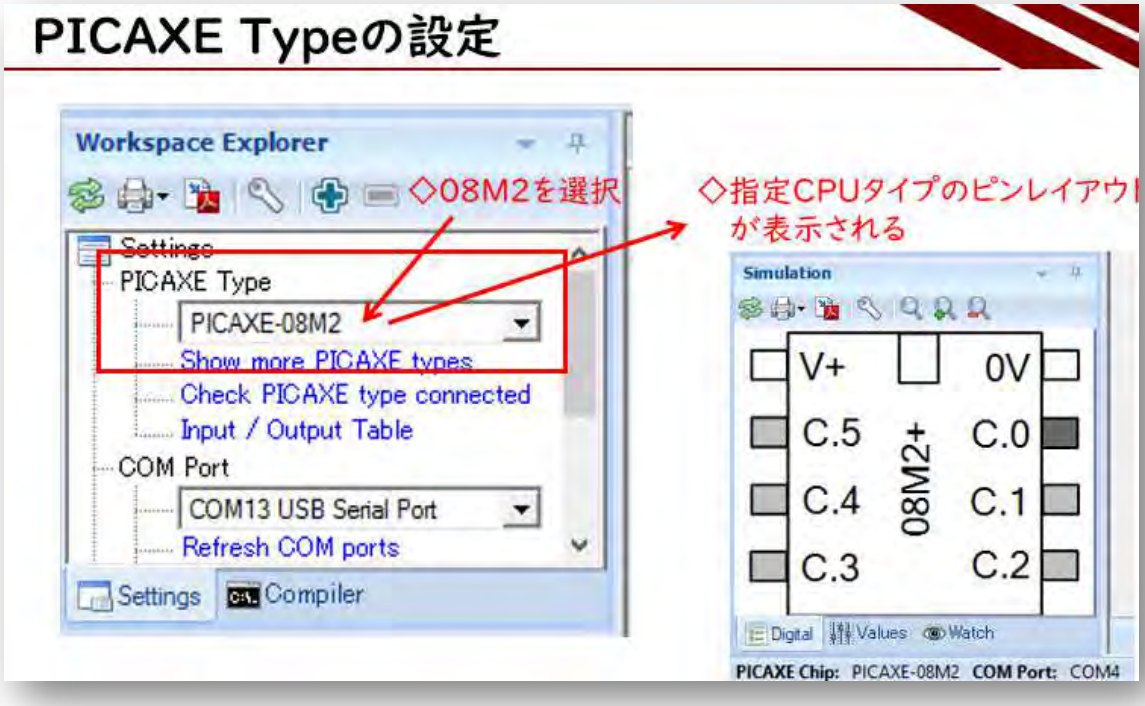


図　24

左上のウインドウで開発対象の PICAXE チップ（PICAXE08M2）を選択すると、左下ウインドウに指定 PICXE のピン配置が図示されます。

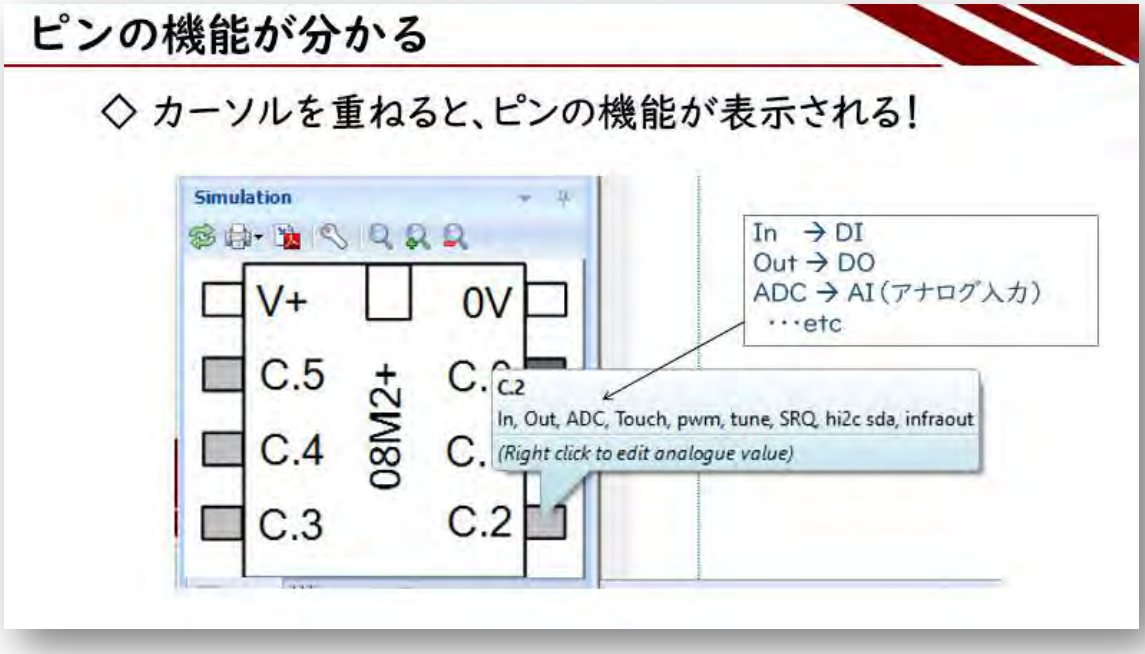


図 25

図示されたピンにカーソルを重ねると、ピンの機能が表示されます。シミュレーションを行う場合に、ピンをクリックして High/Low を切り替えることもできます。

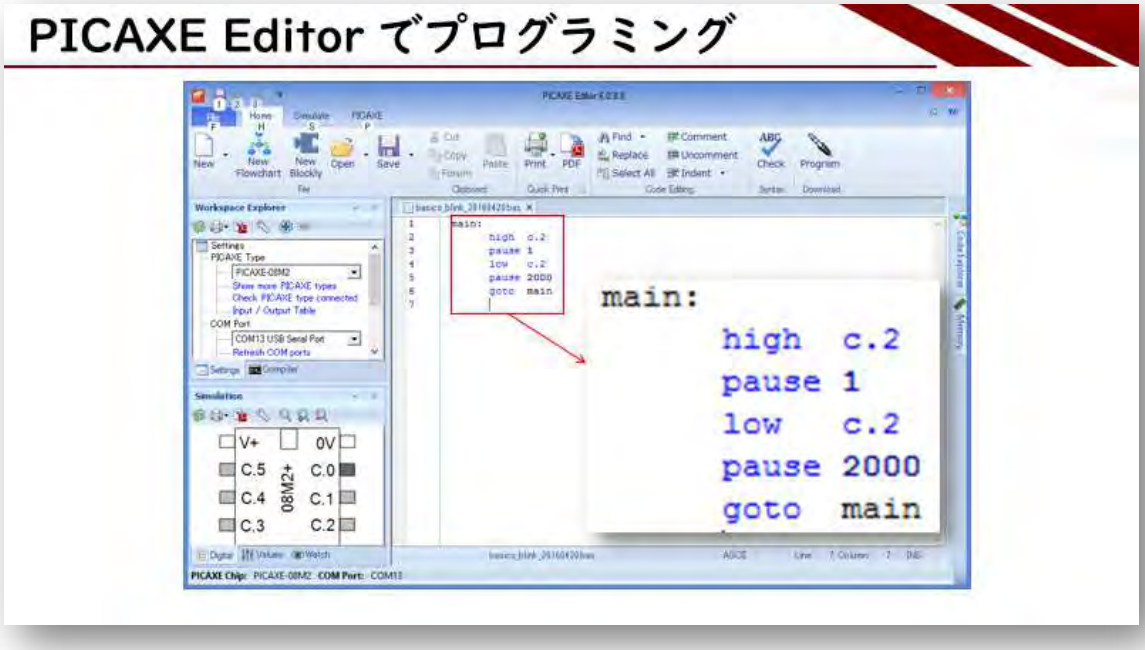


図 26

初回の LED 点滅プログラムを図に倣って記述してください。ここで用いる言語は PICAXE BASIC です。とても充実したマニュアルが準備されているので、参照しながらプログラムを記述すると、初めてでも容易にプログラムが作れます。他のコンピュータ言語と同様に、インデントを整えながら記述すれば、ソースコードは見やすくなります。

図 27

ソースコード各行の内容を図に示します。図で解説している部分は、コメントではありません。コメントの記述は後に説明します。ソースコード中の C.2 という記述は、PICAXE のピン配置のウインドウと対比すれば使用しているピンが分かります。ソースコードの入力が終りましたら、適当な名前を付けて保存しましょう。

※ソースコードは、実習キット付属の CD に含まれています。

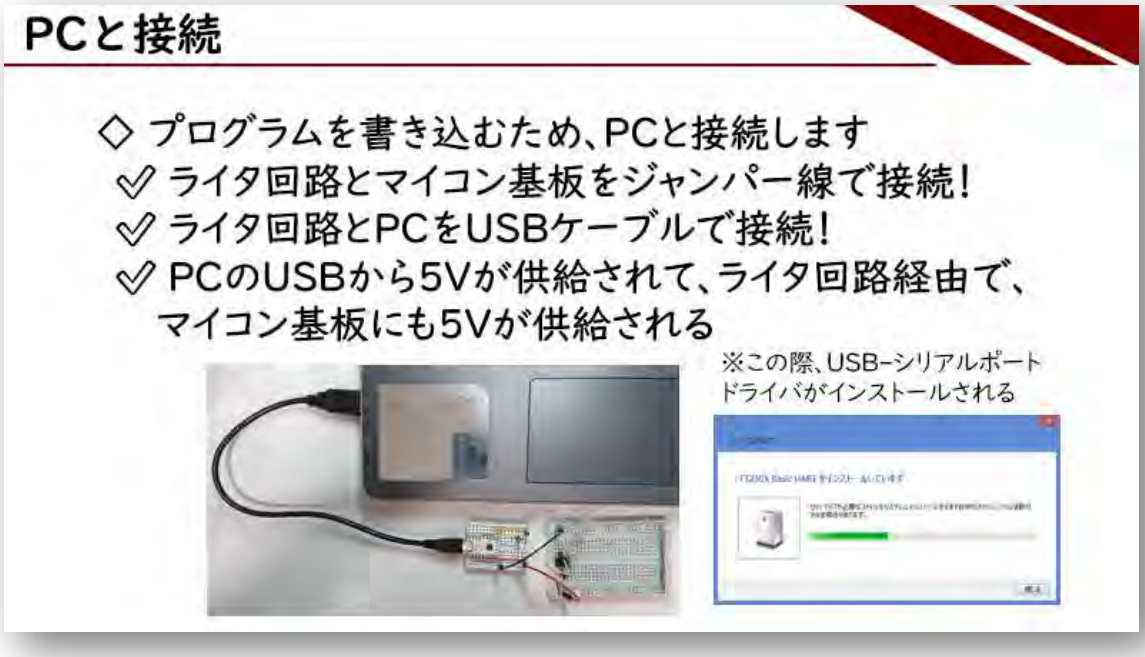


図 28

プログラムが完成したら、コンパイルと書込みを行います。図の手順で PC、ライタ回路、LED 点灯回路を接続して下さい。初めて PC とライタ回路を接続すると図右下の様にドライバがインストールされます。

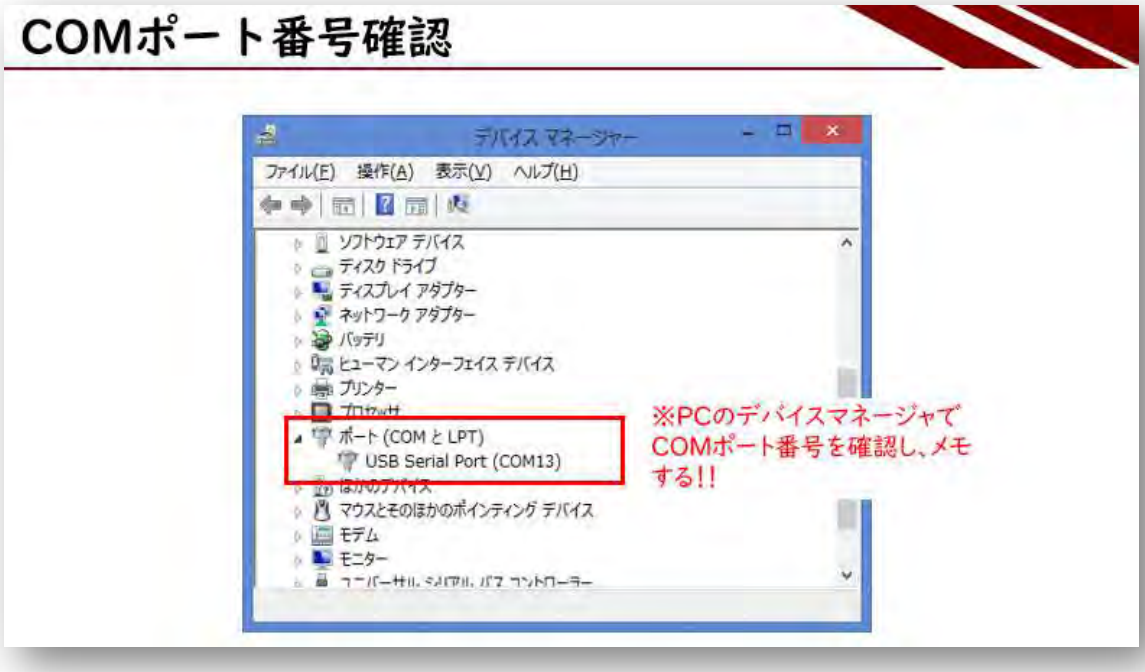


図 29

PC との接続後、Windows のデバイスマネージャで、図の COM ポート番号を確認してメモしてください。COM ポート番号は同じ PC で同じライタ回路を用いる場合は、以後も変わりませんが、同じ PC でも別のライタ回路を接続すると番号が変わりますので、注意してください。

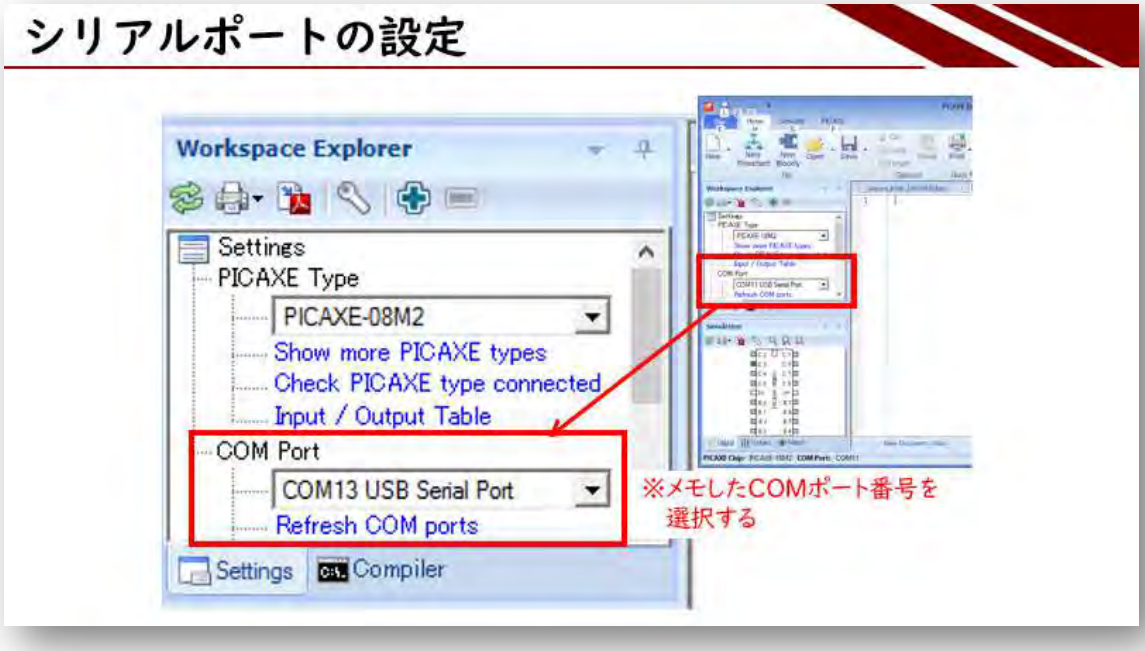


図　30

メモした COM ポート番号を PICAXE Editor 左上の窓のプルダウンで選択します。この COM ポートを通して PC とマイコンがシリアル通信を行います。

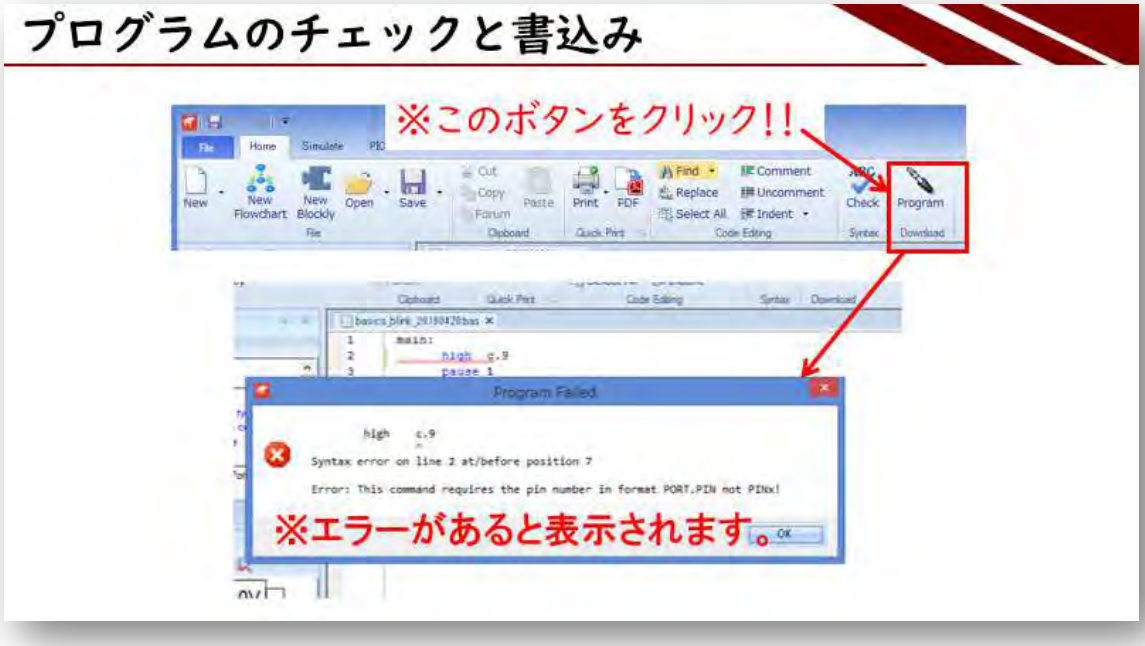


図 31

PICXE Editor の Home タブ右側にある Program ボタンを押下すると、ソースコードのコンパイル等が行われます。エラーがあると、図の様なメッセージでエラー箇所が示されます。その際は修正して再び Program ボタンを押下します。

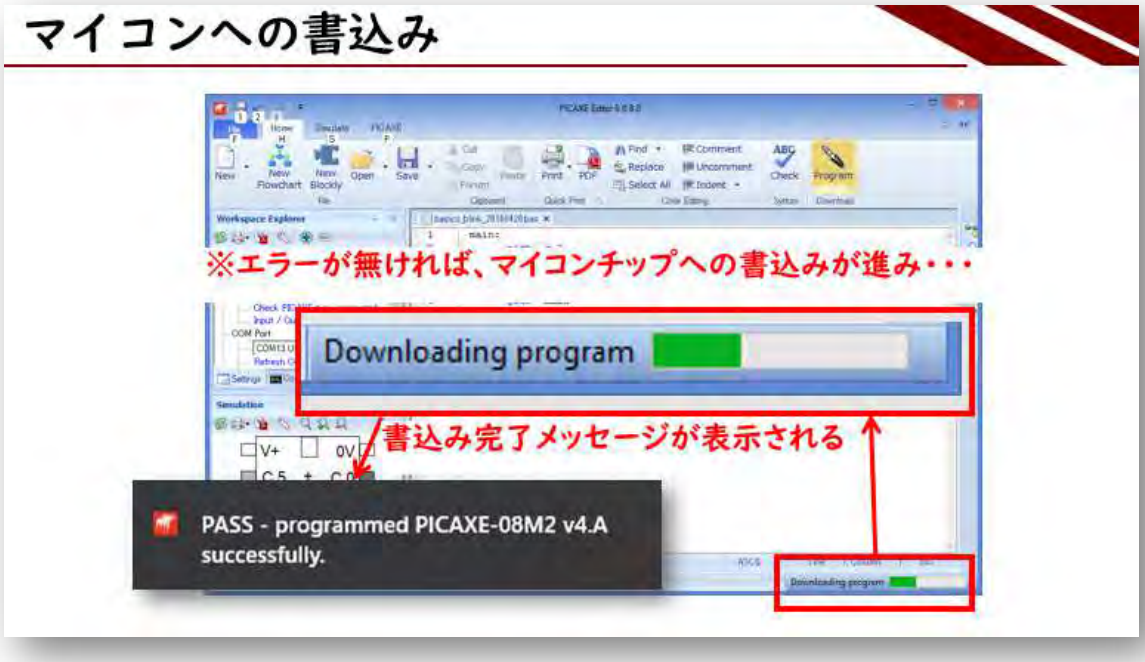


図　32

エラーが無ければ、PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

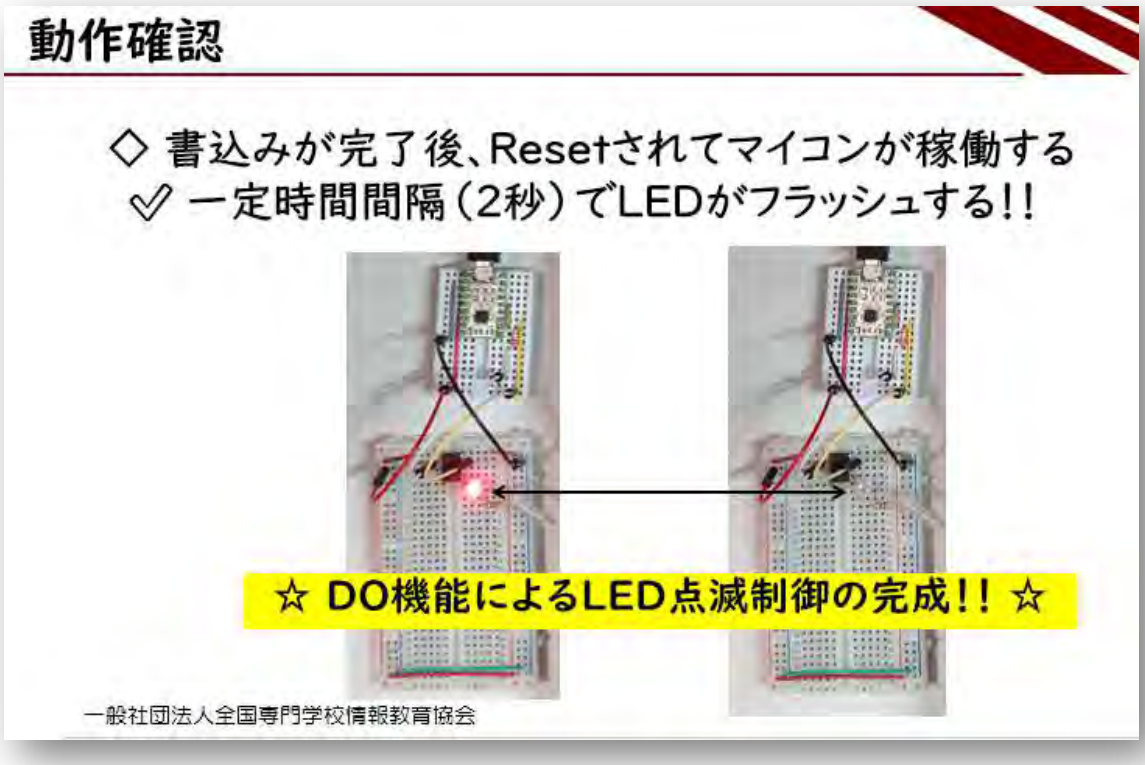


図　33

稼動の様子を図に示します。LED を見ていると、概ね 2 秒ごとに LED が点灯を繰り返します。これで初めての PICAXE システムは完成です。

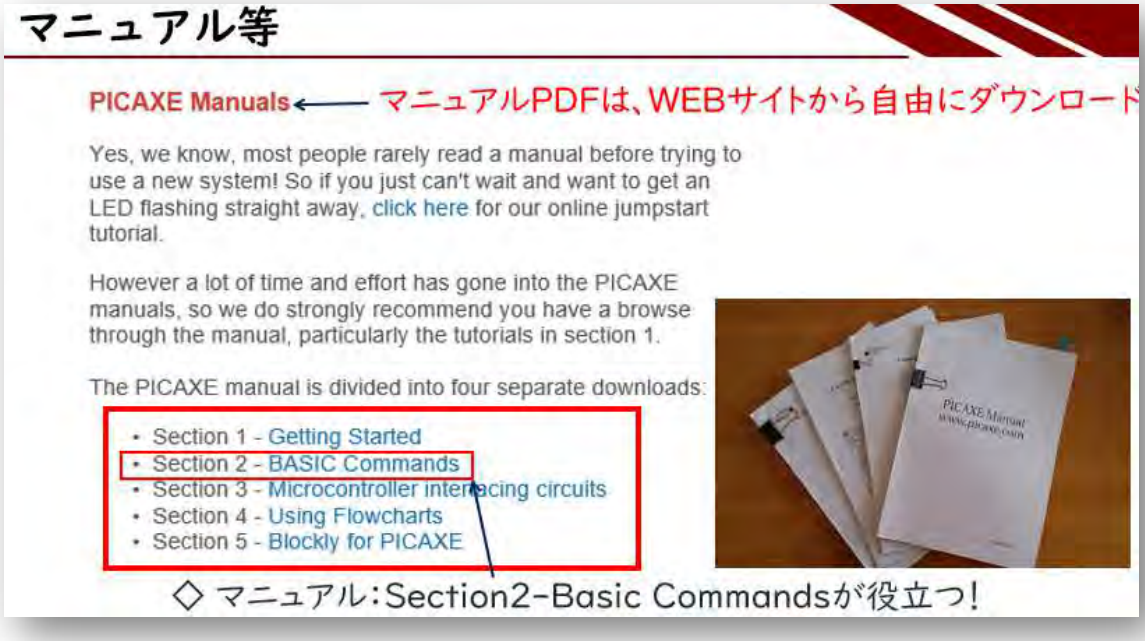


図 34

数百ページの充実したマニュアルが準備されています。WEB で提供されています。

※マニュアル PDF ファイルは、実習キット CD に含まれています。

第2回　SW

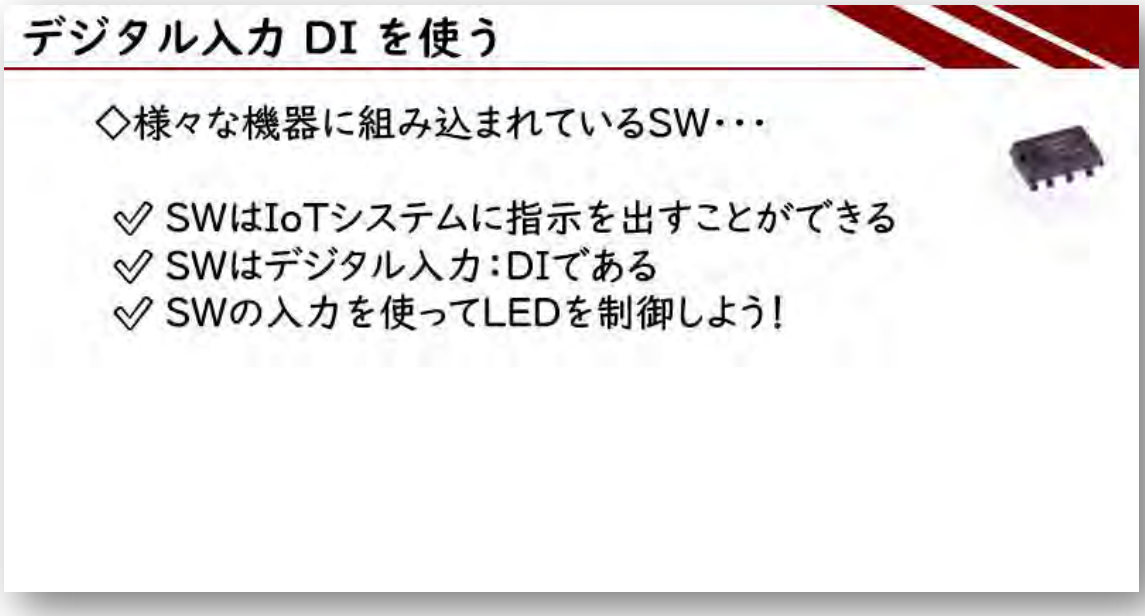


図 35

次はデジタル入力(DI)を使いましょう。多くの機器に使われているSWは、マイコン外部から情報を伝える最も基本的な機能です。これを用いて先に行ったLEDの点灯を制御してみましょう。

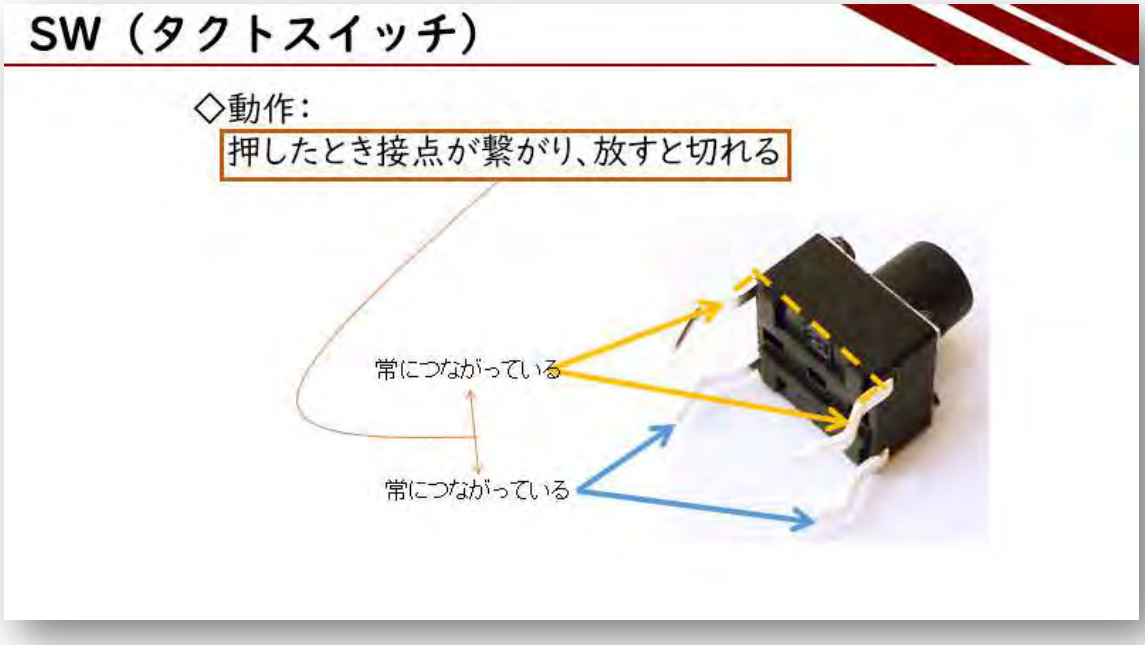


図　36

図はタクト SW と呼ばれているパーツです。裏から見ると、4 本の脚があります。2 本の脚は内部で繋がっていて、内部に 2 対の脚を接続する接点があります。ボタン部分を押下すると 2 対の脚が電気的に繋がり、離すと切れるようになっています。このようにボタンを押下すると接続する接点を A 接点と言います。反対に通常は繋がっている接点が、ボタンを押下すると切れる接点を B 接点と言います。

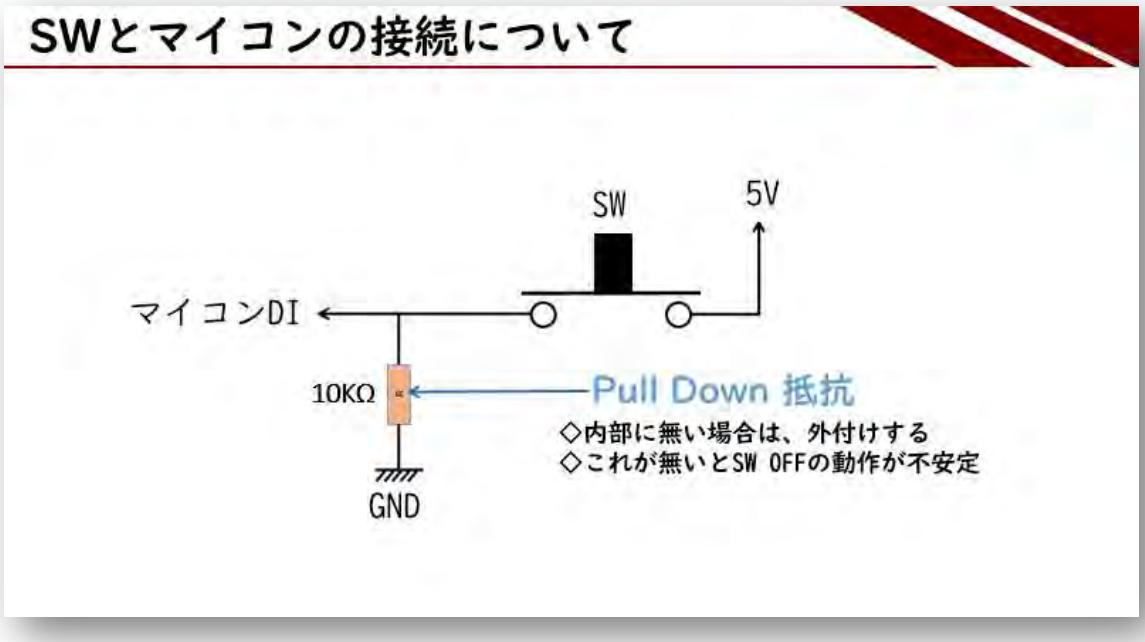


図 37

図はマイコンと SW の接続方法です。SW の片側接点は 5V に、他方はマイコン DI に接続されています。DI 側は 10kΩの抵抗を介して GND にも接続されています。通常状態ではマイコン DI が抵抗を介して GND に接続されているので、DI の電圧は 0V、プログラムで DI の状態を読むと 0 になります。SW を押下して接点が閉じると DI の電圧は 5V になり、これをプログラムで読むと 1 になります。抵抗が無かった場合を考えてみると、接点が開いていれば DI はどこにも接続されないので、0 でも 1 でもない状態になっています。この時、DI を読み込んで 0 である保証はありません。抵抗を用いず直接 GND に接続すると、接点が開いている際は良いですが、接点を閉じると 5V が直接 GND に接続されるので、ショートしてしまいます。これは困るので、接点が開いているときは必ず 0 になる様にするため、抵抗を介して GND に接続します。この抵抗を PULL DOWN 抵抗と呼びます。

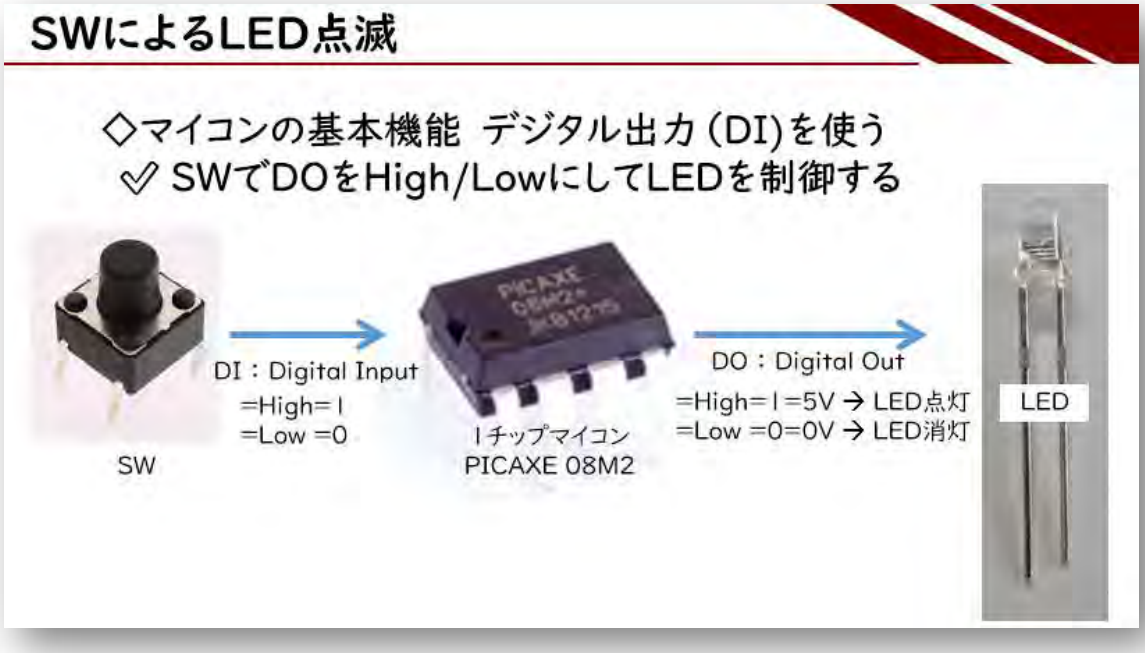


図 38

前述のように SW の状態は High=1 または Low＝0。これをマイコンで読取り LED に出力します。

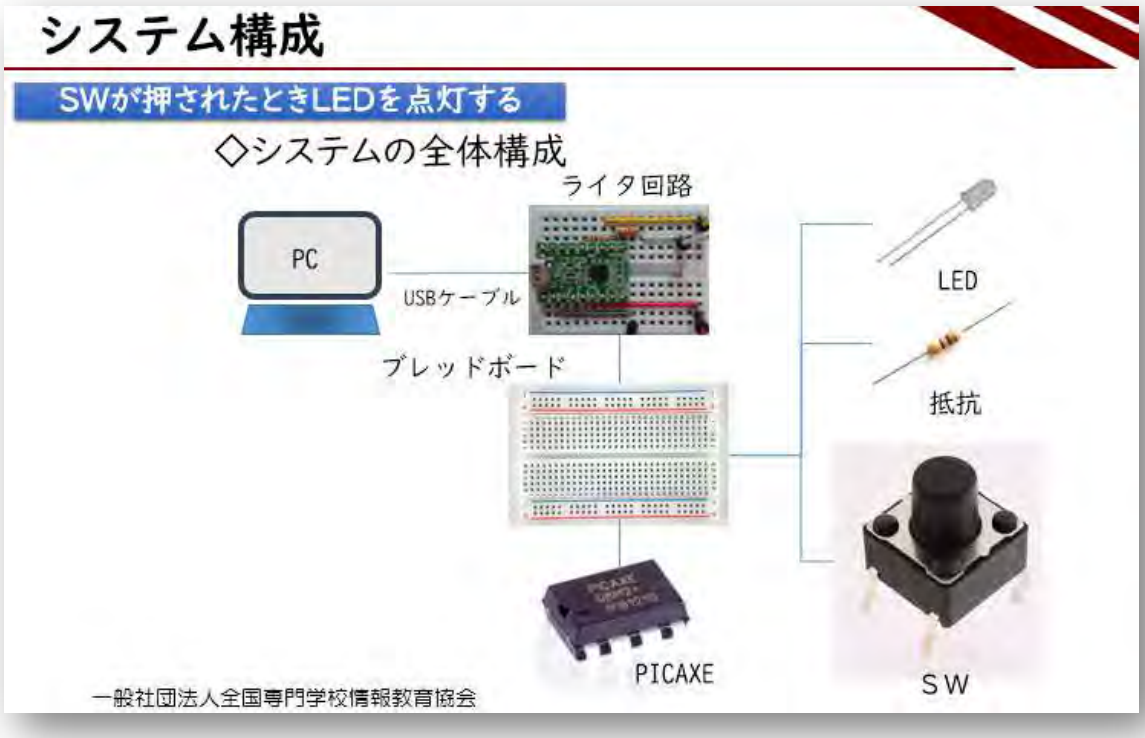


図　39

前回作成した回路に SW を追加しましょう。

図 40

SW からの入力には 4 番ピン：C.3 を使用します。このピンに In という表記があるのは DI 専用のピンであるということです。In/Out と記されているピンは DI/DO どちらにも使用できます。

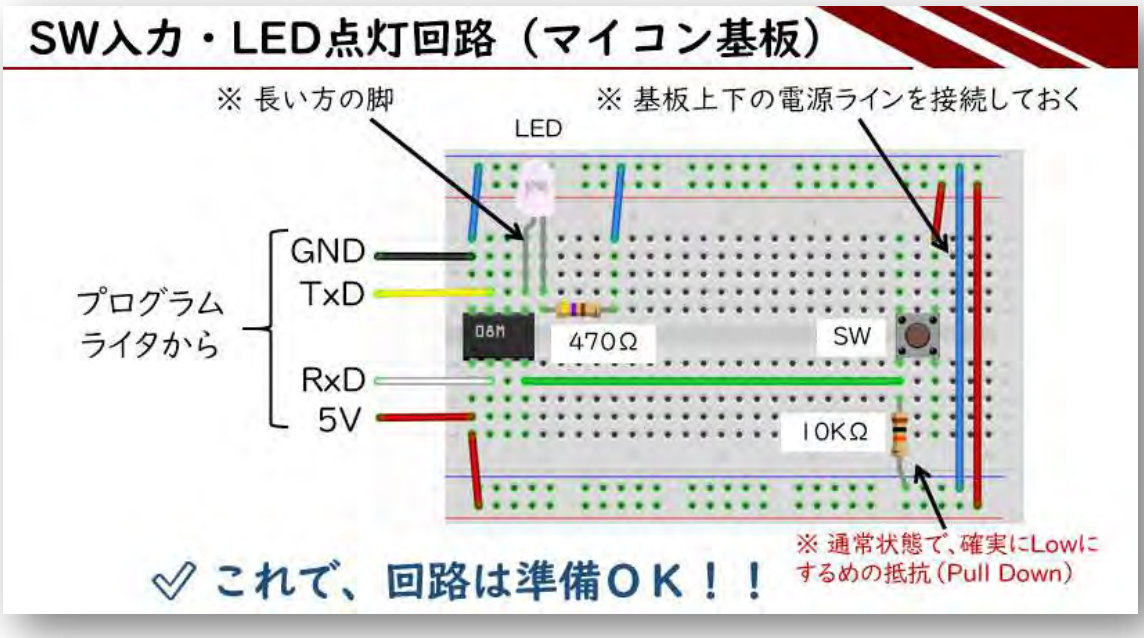


図　41

ブレッドボードは広いので、何処でも自由に使用して良いよう思えますが、回路図に従いSWは指定の場所に正しく配置します。複雑な回路の場合、配線確認を他の方が行うこともあります。その際、回路図と御暗示であれば確認も行い易く、その作業時間を短縮することもできます。回路が出来たら、電気の流れを考えながら確認してみて下さい。

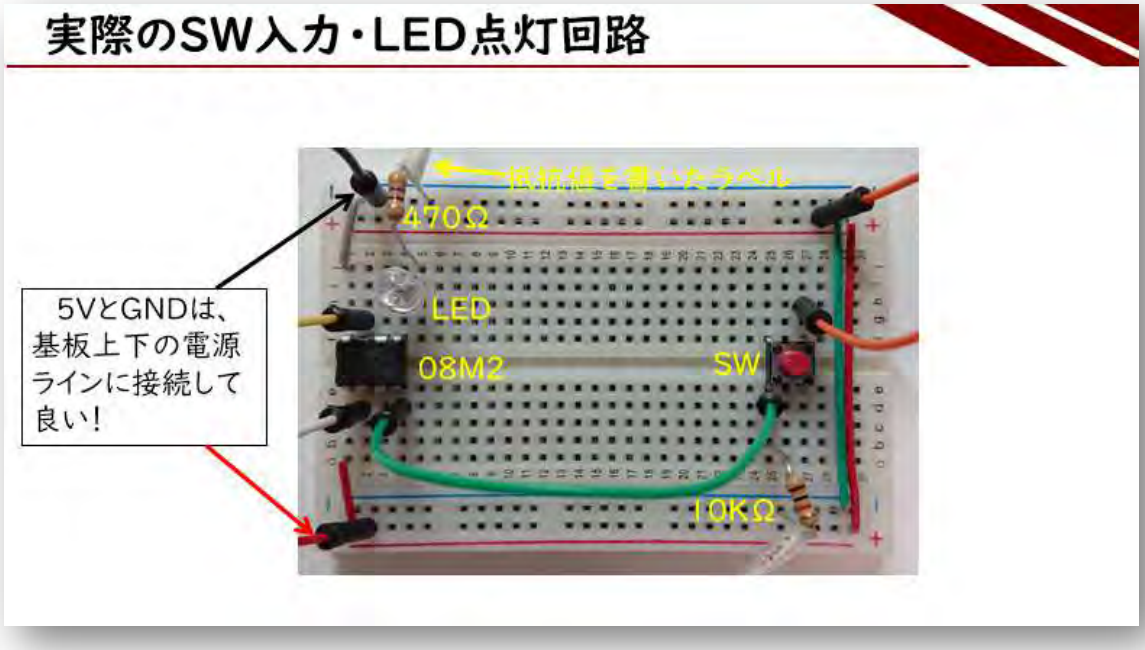


図　42

実際に作成した回路を示します。

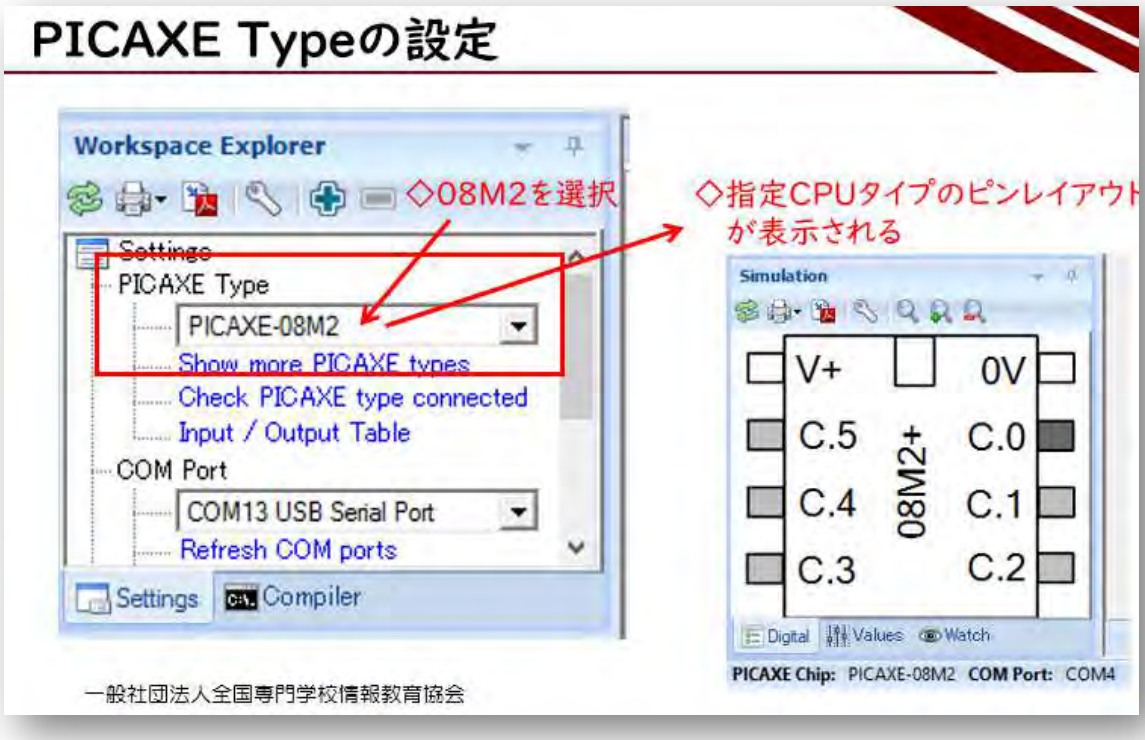


図　43

PICAXE Editor 左上のウインドウで対象の PICAXE（PICXE08M2）チップを選択すると、左下ウインドウに指定 PICXE のピン配置が図示されます。

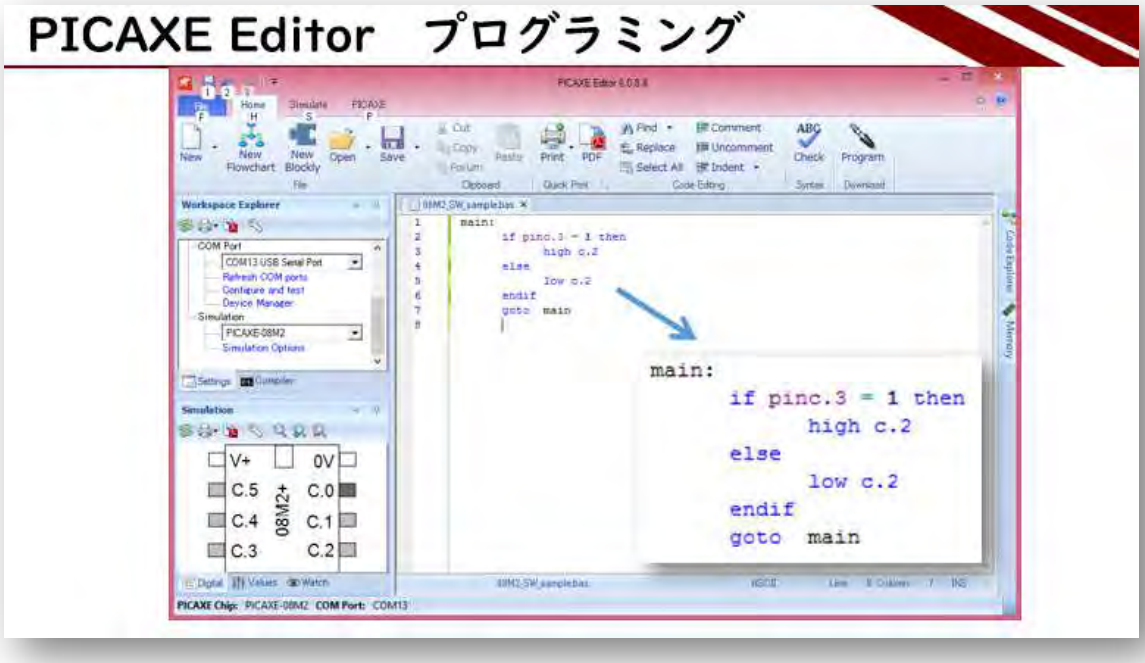


図　44

Home Tab の New で新しいプログラムを作成します。図に倣ってソースコードを記述しましょう。

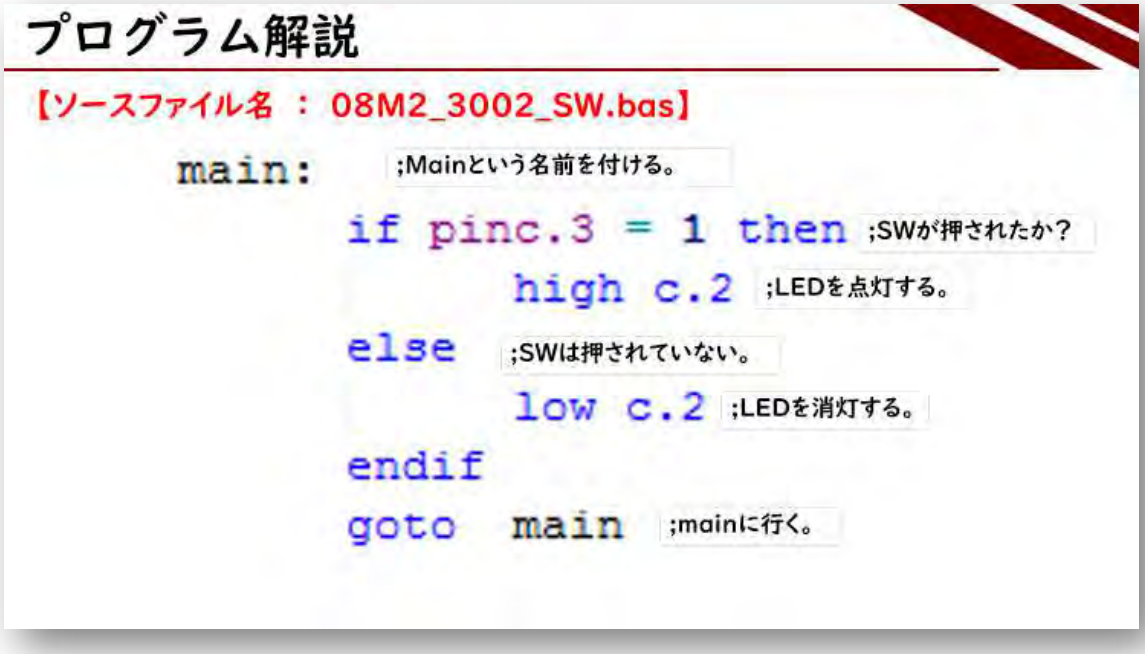


図 45

ソースコード各行の内容を図に示します。図で示すように；（セミコロン）を使えばその右側は行末迄コメントと解釈されます。ソースコード中の C.2、C.3 という記述は、PICAXE のピン配置のウインドウと対比すれば使用しているピンが分かります。ソースコードの入力が終りましたら、適当な名前を付けて保存しましょう。

※ソースコードは、実習キット付属の CD に含まれています。

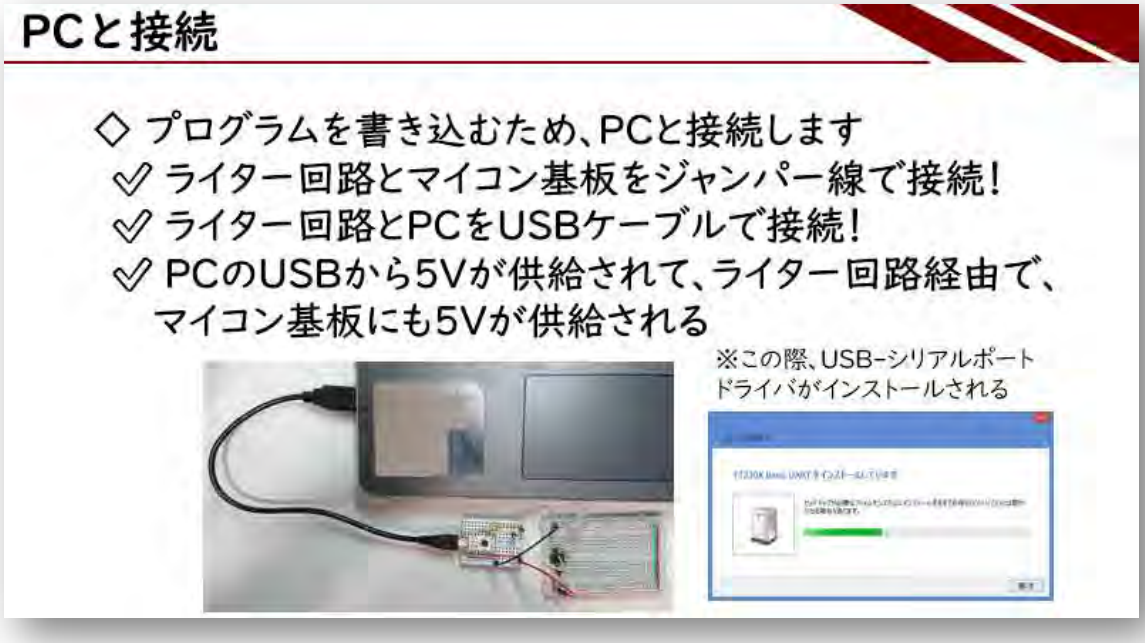


図　46

プログラムが完成したら、コンパイルと書込みを行います。図の手順で PC、ライタ回路、マイコン回路を接続して下さい。初めて PC とライタ回路を接続すると図右下の様にドライバがインストールされます。

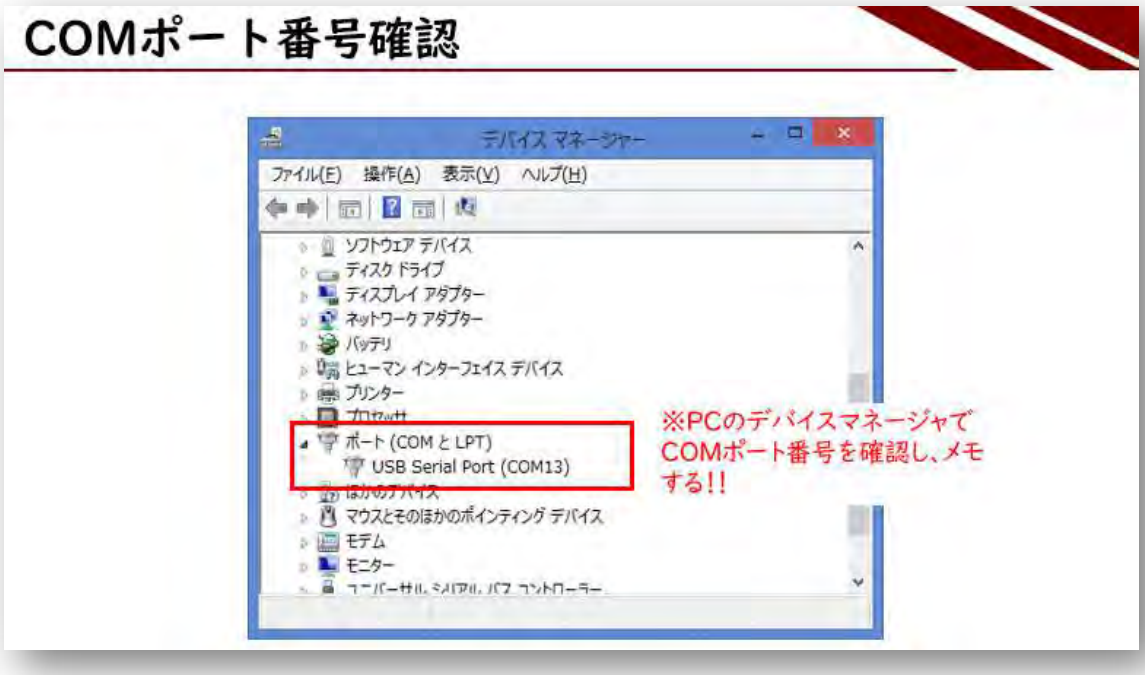


図 47

PC との接続後、Windows のデバイスマネージャで、図の COM ポート番号を確認してメモしてください。COM ポート番号は同じ PC で同じライタ回路を用いる場合は、以後も変わりませんが、同じ PC でも別のライタ回路を接続すると番号が変わりますので、注意してください。

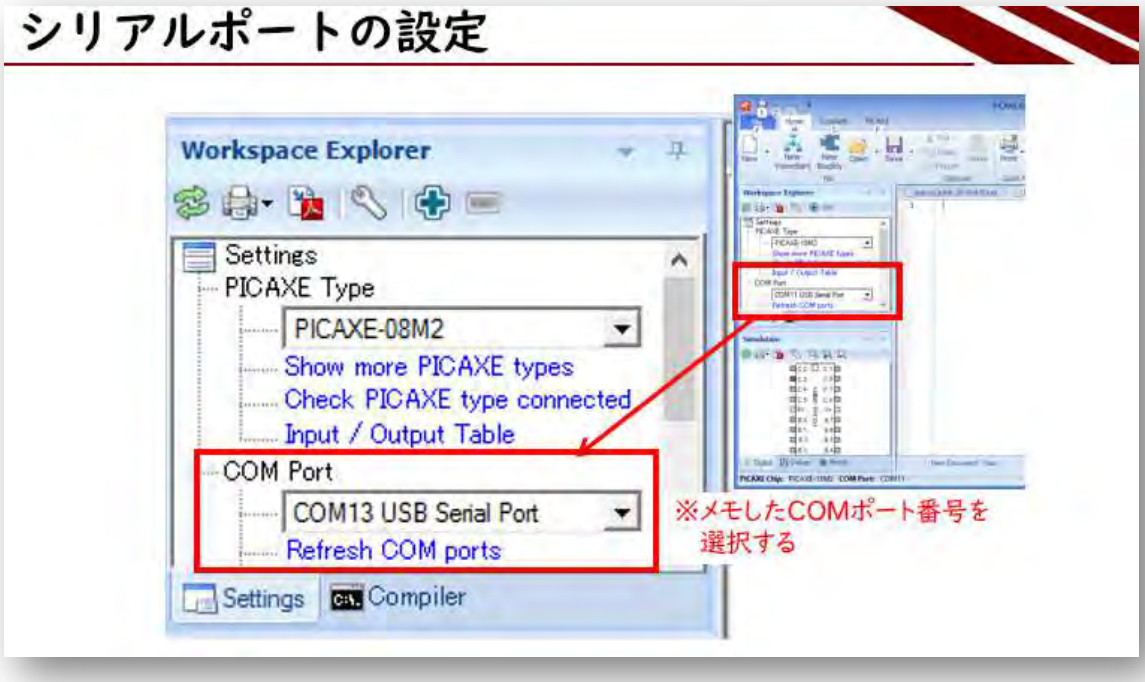


図 48

メモした COM ポート番号を PICAXE Editor 左上の窓のプルダウンで選択します。この COM ポートを通して PC とマイコンがシリアル通信を行います。

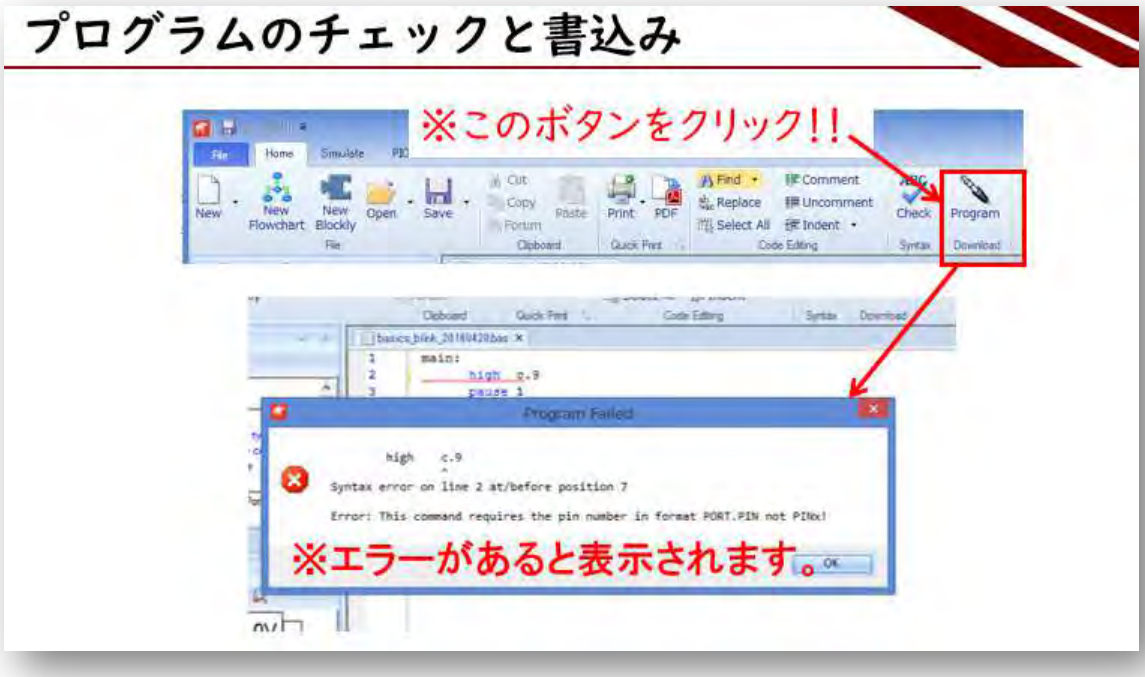


図 49

PICXE Editor の Home タブ右側にある Program ボタンを押下すると、ソースコードのコンパイル等が行われます。エラーがあると、図の様なメッセージでエラー箇所が示されます。その際は修正後、再び Program ボタンを押下します。

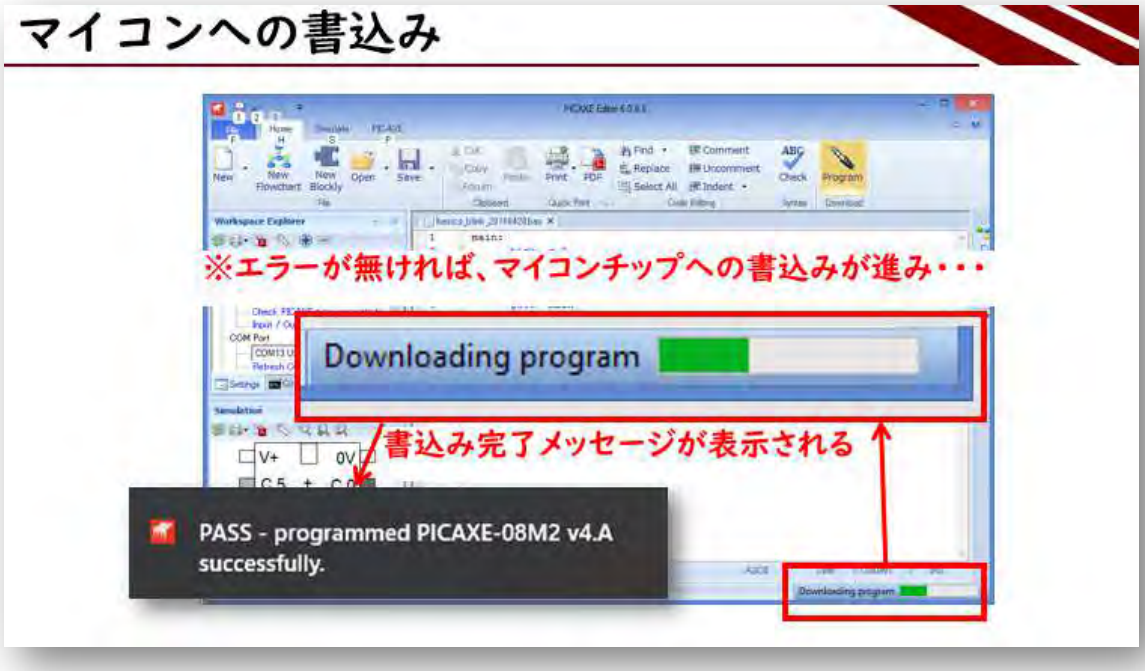


図　50

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

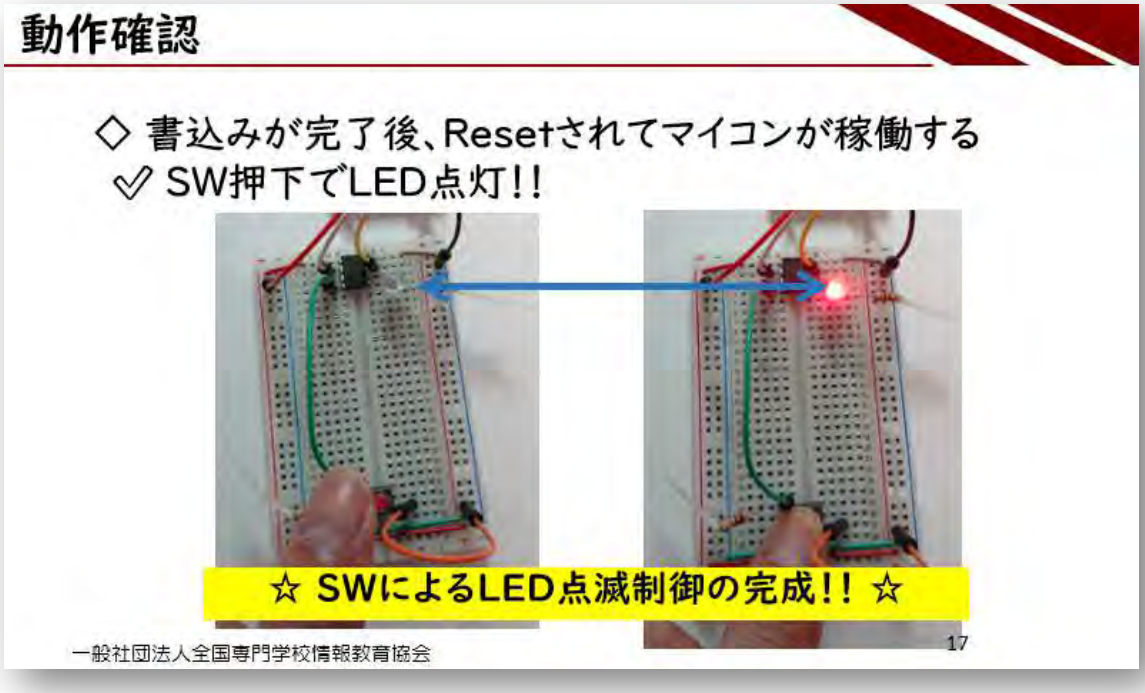


図　51

動作の様子を図に示します。書込み完了後にマイコンは動作を始めています。LED は消灯したままになっています。SW を押下すると LED が点灯します。指を離すと LED は消灯します。

第3回　シリアル通信【送信】

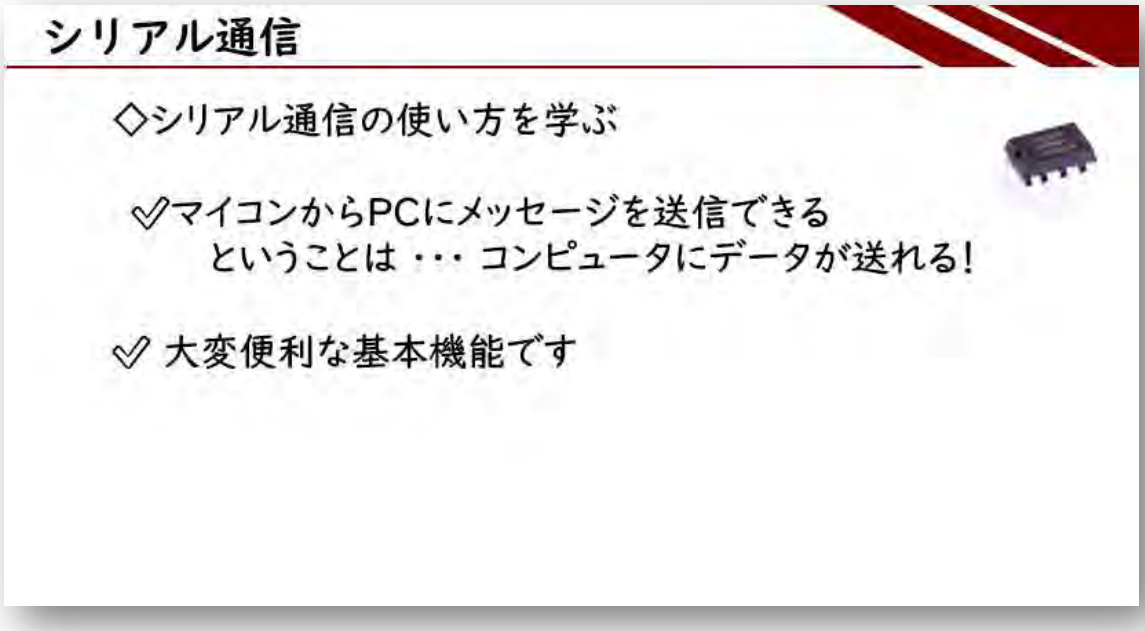


図 52

シリアル通信が出来るとマイコン内部の様子やデータを出力する手段が増えます。また外部からマイコンに対してデータを入力したり、命令を与えたりできます。

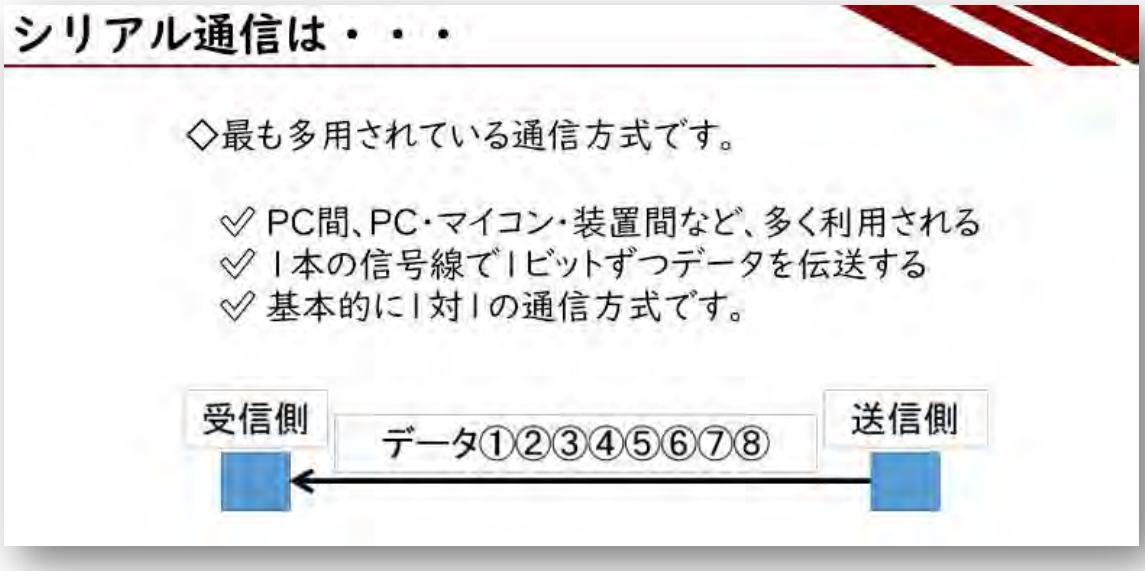


図 53

シリアル通信はコンピュータ同士やコンピュータと装置などが通信をする手段として多く利用されています。1本の信号線を使い 1bit ずつデータを送ります。信号の単位が bit なのでシリアル通信の速度は bps（bit par second）という単位で表します。

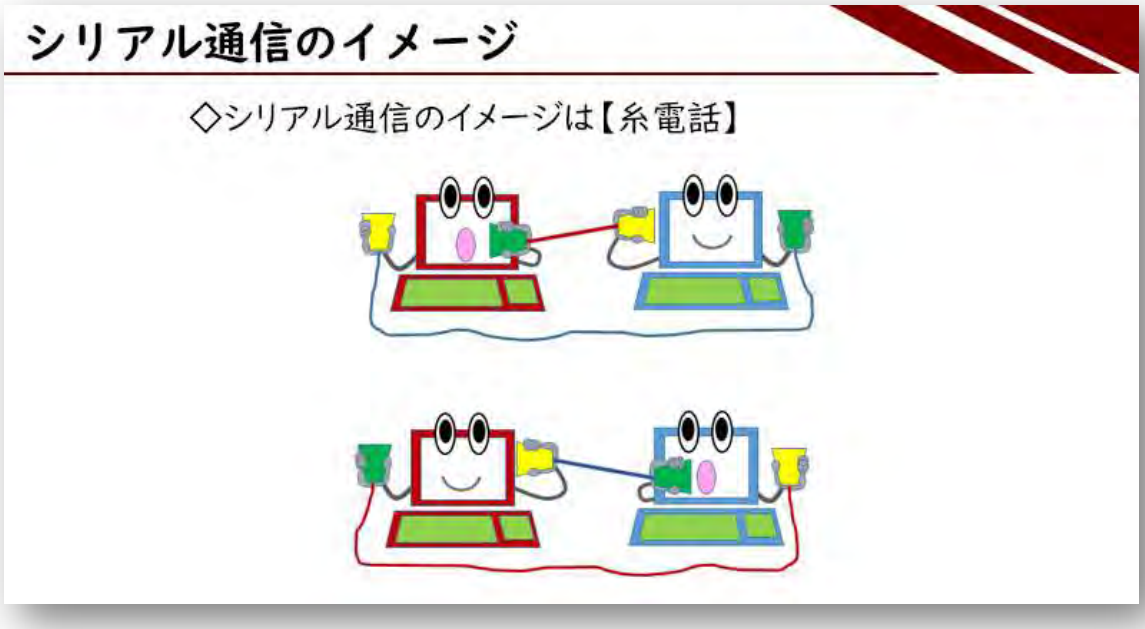


図　54

シリアル通信のイメージは、糸電話に例えることができます。一方は喋る側、もう一方は聴く側です。お互いに話したいときは二組の糸電話が必要です。一方から「お元気ですか？」と送ると「元気です！」と返事が返ってきます。一本の糸電話で「お元気ですか」の文字列が送られます。この通信方法がシリアル通信です。コンピュータでは１文字（1バイト）ずつ送ります。相手に受け取り準備をしてもらうため、「送りますよ」「1文字目」「終わりました」「送りますよ」「2文字目」「終わりました」…と１文字ごと、送り初めと終わりを知らせます。

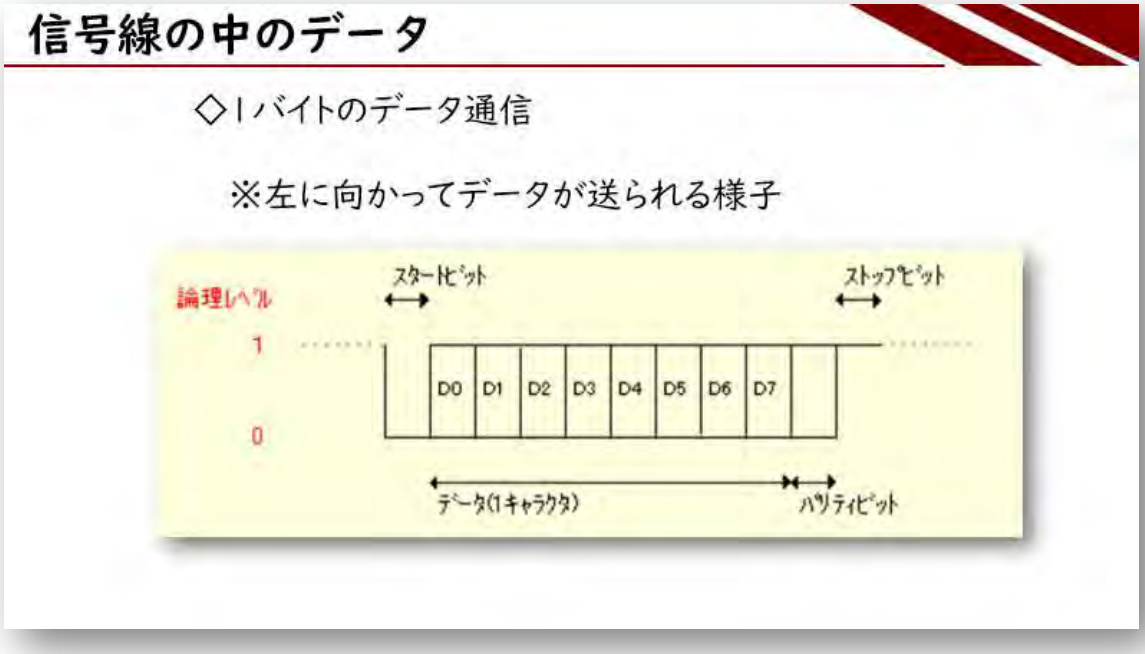


図　55

図は１バイトのデータを左向きに送る様子です。シリアルインターフェースに１バイトのデータを書き込むと、最初にスタートビット（送り始め）が送られて、相手に「これから通信データが来る」ことを知らせます。続いて 8bit のデータが送信され、後に誤り検出のためのパリティビットが続きます。最後にストップビット（送り終わり）が送られて１バイトのデータの送信が終わります。スタートビットは、１ビットですが、ストップビットはパリティビットの有無などにより、1~2 ビットになります。パリティビットは偶数パリティ・奇数パリティ・パリティ無しが設定できます。パリティ無しに設定した場合は、ストップビットを 2bit にします。

図　56

図は半角の A（16 進数で 0x41）を左向きに偶数パリティで送信した場合の信号線上に現れるビットデータです。

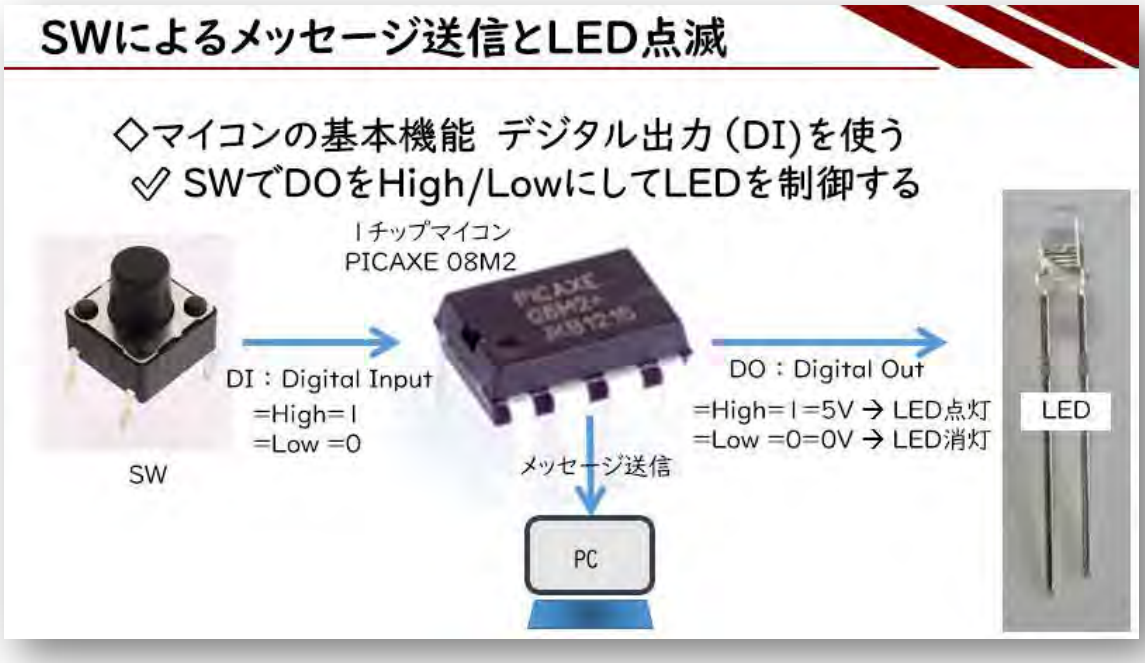


図 57

今回の実験では、SW の ON/OFF をシリアル通信で PC に向けて送信しながら、ON/OFF 状態を LED に反映させます。

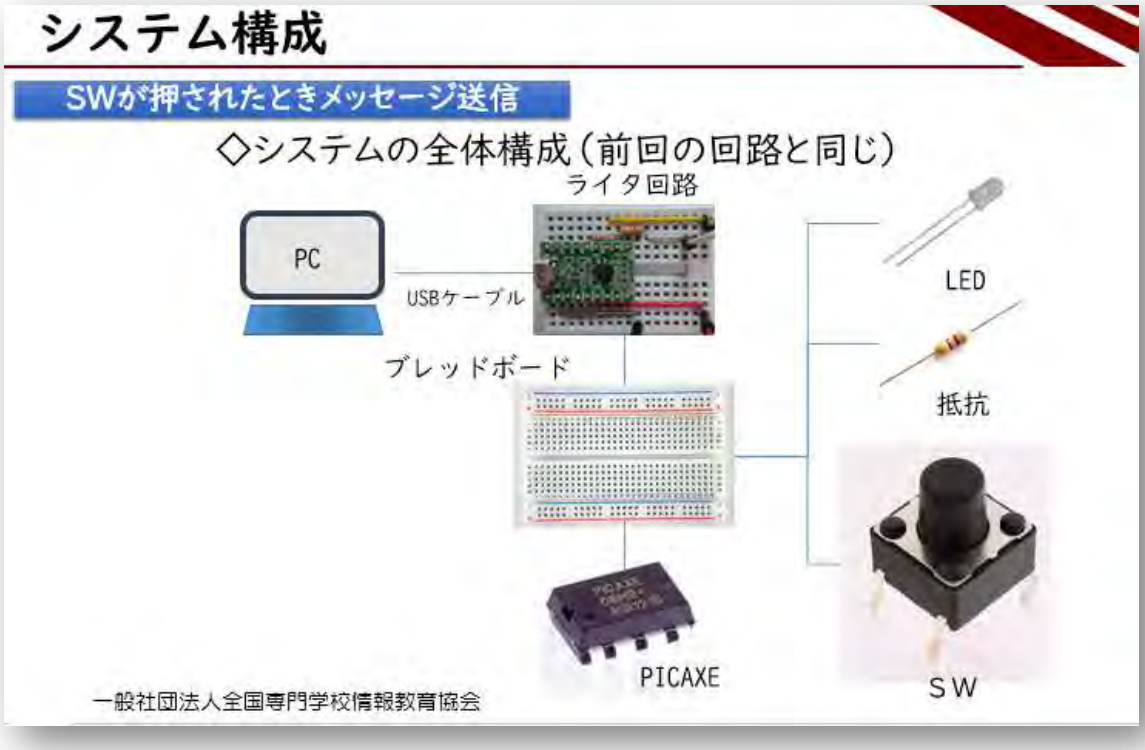


図　58

使用するパーツは前回と同じです。USB ケーブルがマイコンと PC 間の糸電話の【糸】の役目をします。

図 59

PICAXE の 7 番ピン（Serial Out の表記）を使用して送信を、2 番ピンで受信を行いす。

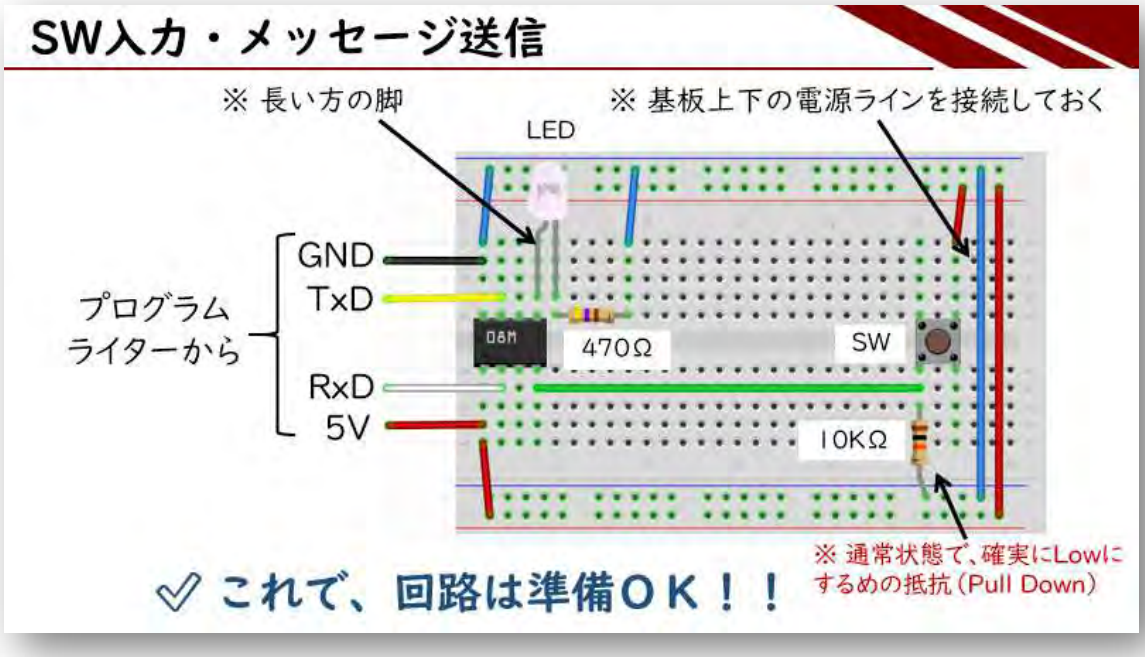


図　60

前回作成した回路をそのまま使います。配線に緩みなどがないか、よく確認してください。

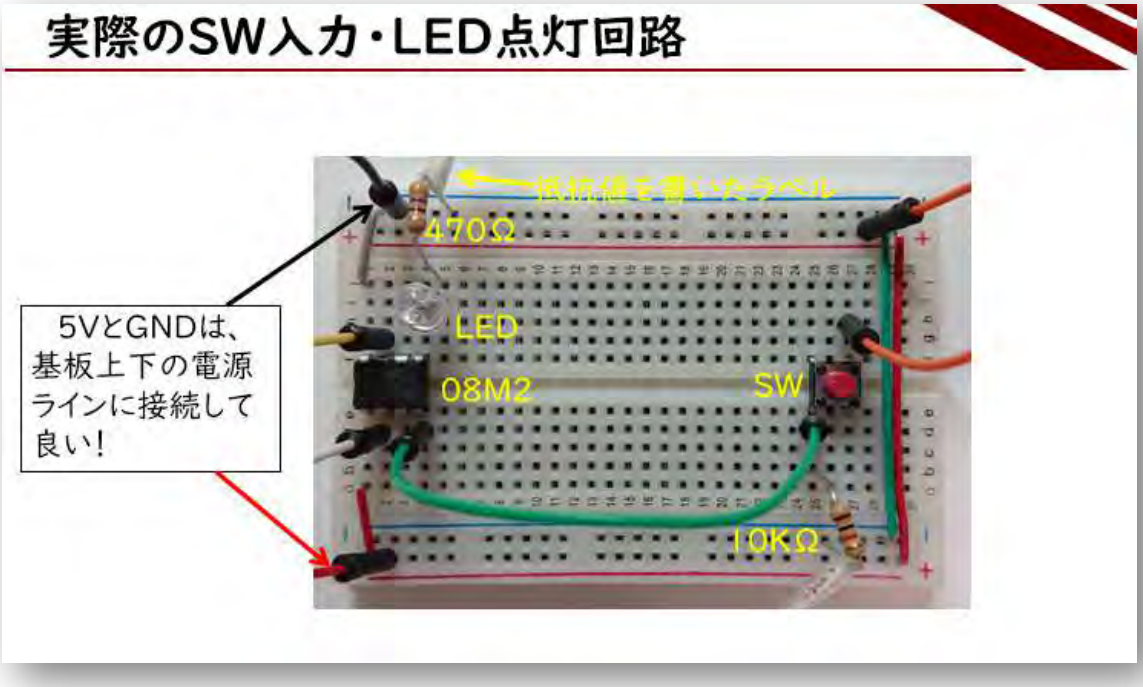


図　61

実際の回路を図に示します。

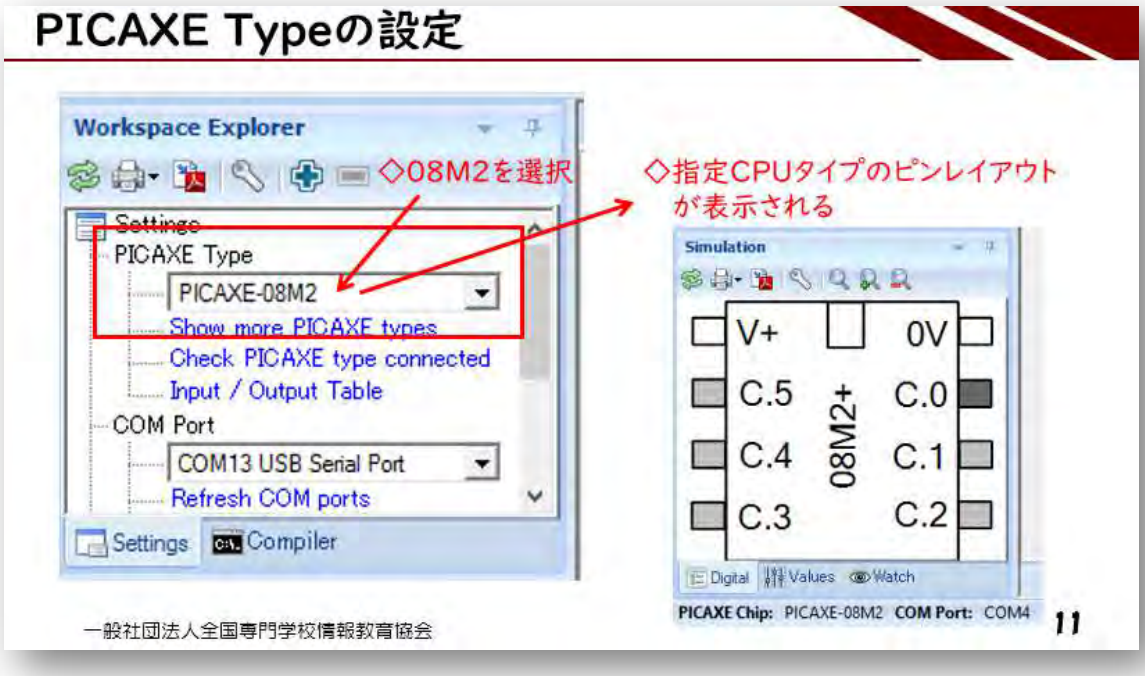


図　62

左上のウインドウで対象の PICAXE チップ（PICAXE08M2）を選択すると、左下ウインドウに指定 PICXE のピン配置が図示されます。

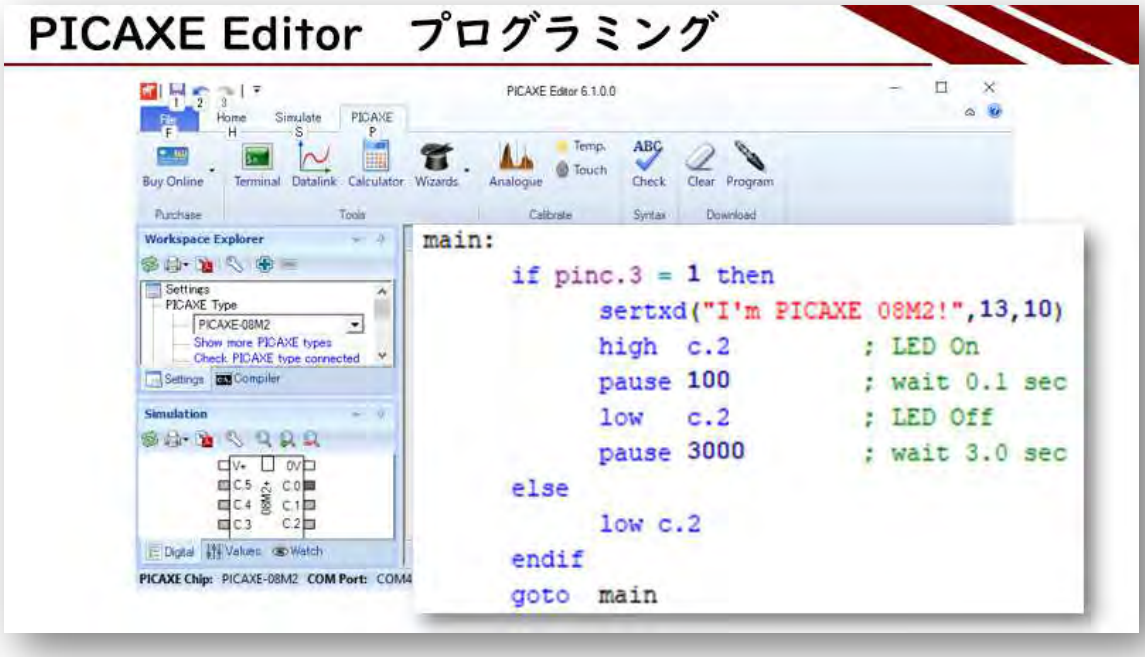


図　63

Editor の Home 左上の New ボタンを押下して新しいプログラムを作成し、図に倣ってソースコードを記述しましょう。

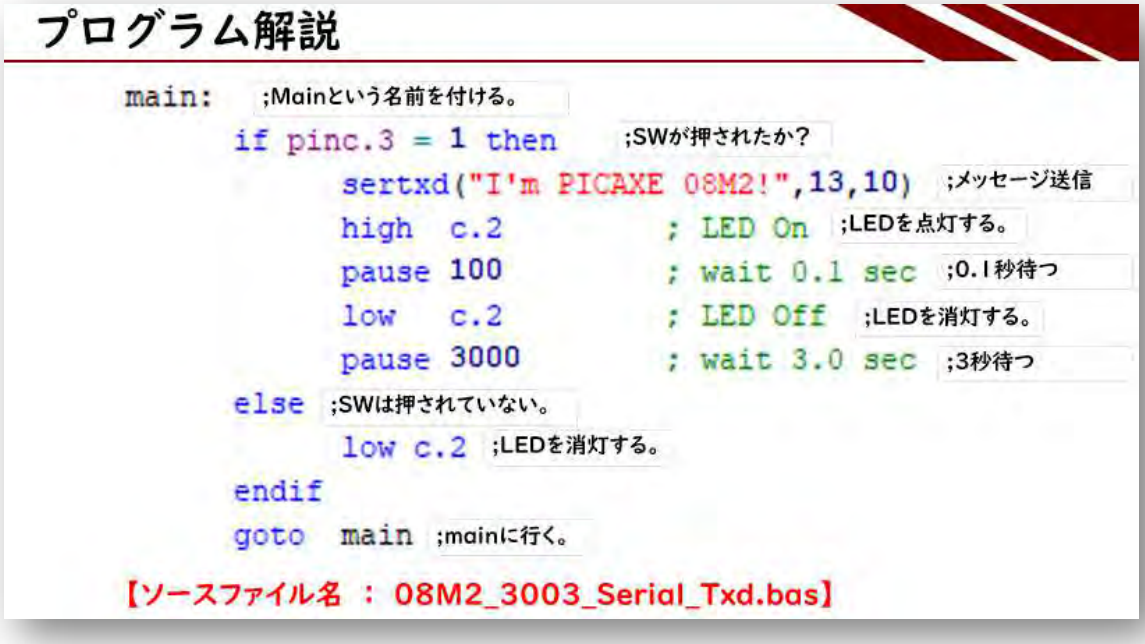


図 64

ソースコード各行の内容を図に示します。図で示すように;(セミコロン)を使えばその右側は行末までコメントと解釈されます。ソースコード中の sertxd()という箇所でメッセージ送信を行っています。() 内の文字列の後にある 13 は 0x0D、10 は 0x0A でそれぞれ CR、LF を表す ASCII コードです。文字列の後にこれらのコードが付加されて送信されるので、受信側では改行付きメッセージになります。ここでは固定のメッセージを送信していますが、状況に応じた内容を送信できれば、自由に外部と通信ができるシステムになります。ソースコードの入力が終ったら、適当な名前を付けて保存しましょう。

※ソースコードは、実習キット付属の CD に含まれています。

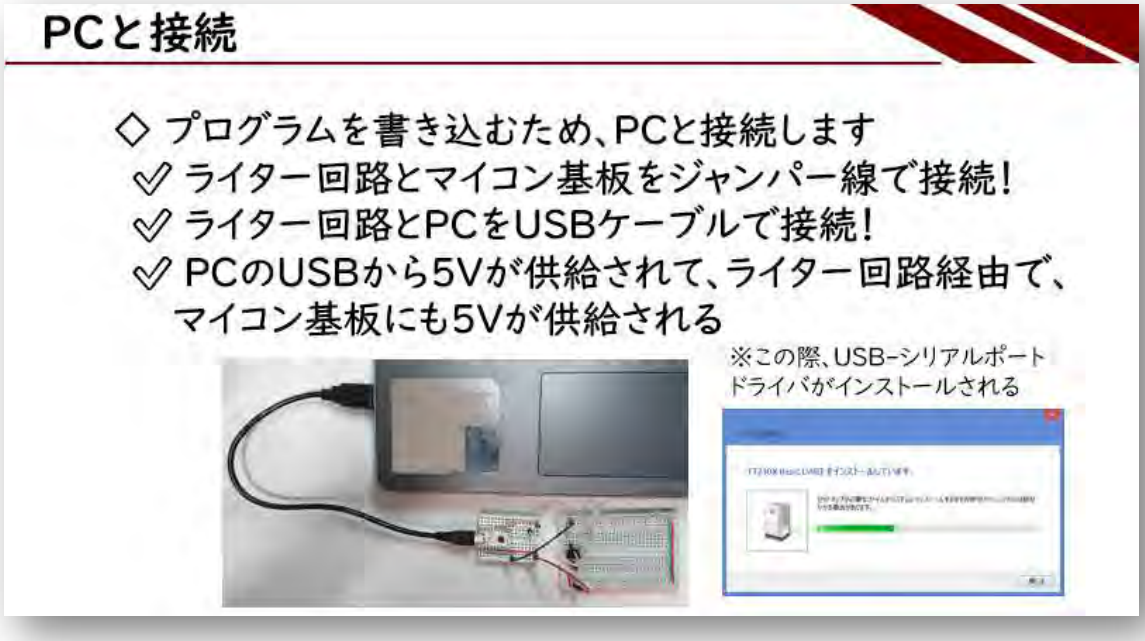


図 65

プログラムが完成したら、コンパイルと書込みを行います。図の手順で PC、ライタ回路、マイコン回路を接続して下さい。初めて PC とライタ回路を接続すると図右の様にドライバがインストールされます。

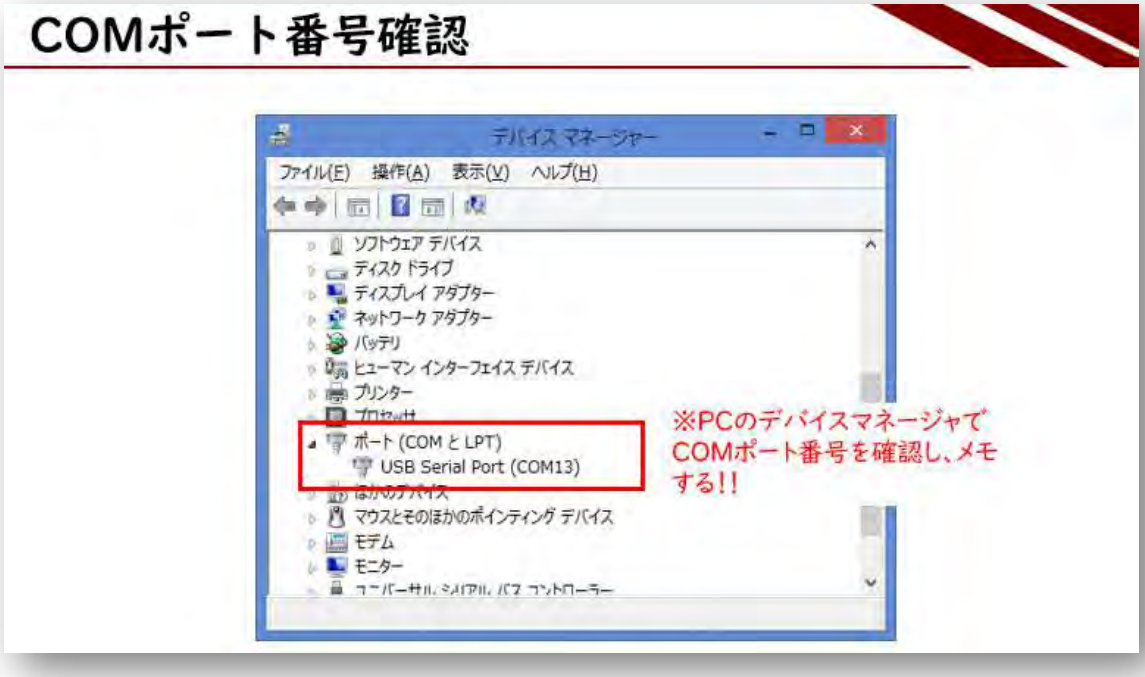


図 66

PC との接続後、Windows のデバイスマネージャで、図の COM ポート番号を確認してメモしてください。COM ポート番号は同じ PC で同じライタ回路を用いる場合は以後も変わりませんが、同じ PC に別のライタ回路を接続すると番号が変わりますので、注意してください。

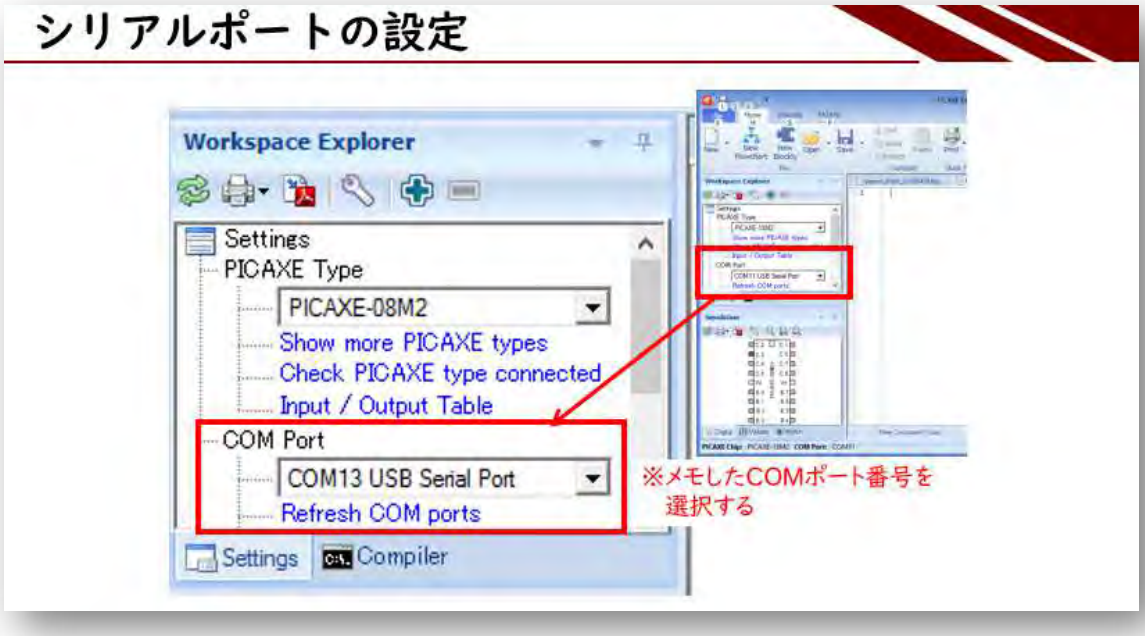


図　67

デバイスドライバで確認した COM ポート番号を設定します。

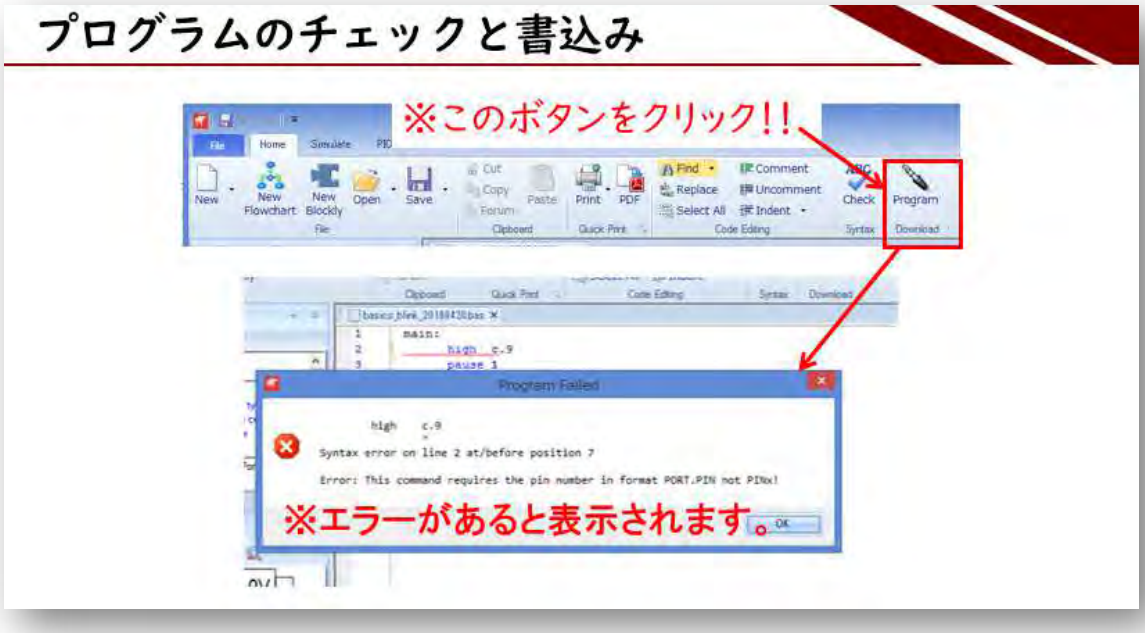


図 68

PICXE Editor の Home タブ右側にある Program ボタンを押下すると、ソースコードのコンパイル等が行われます。エラーがあると図の様なメッセージが表示され、エラー箇所が示されます。その際は修正して再び Program ボタンを押下します。

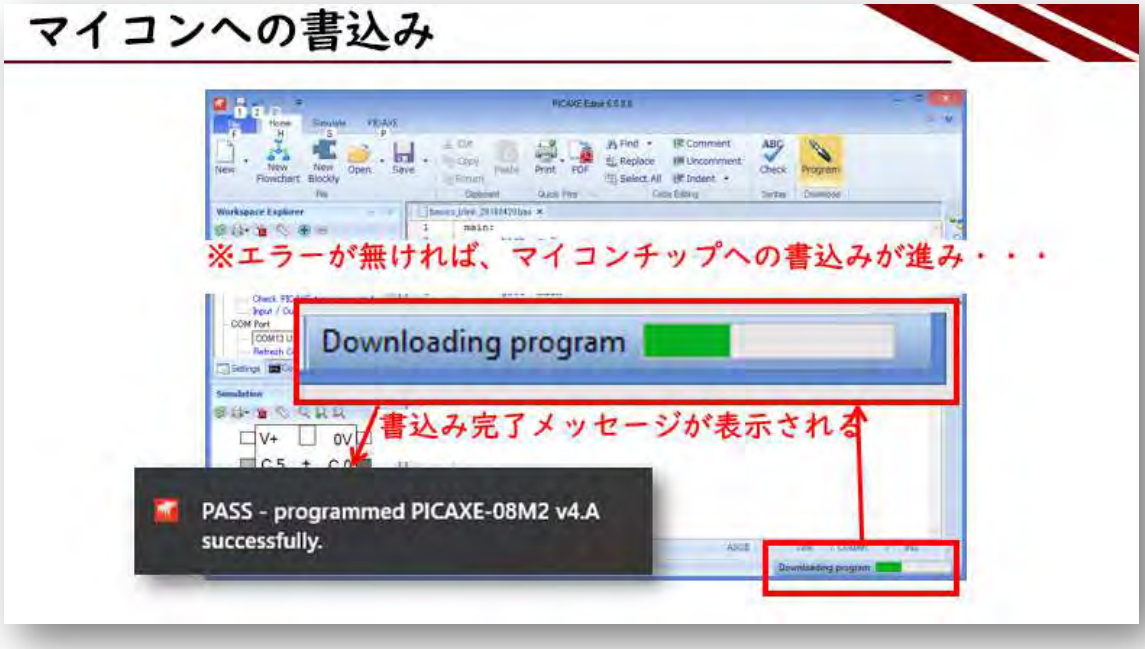


図 69

エラーが無ければ、PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムが書込まれていきます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了するとマイコンが Reset されてプログラムの実行が始まります。

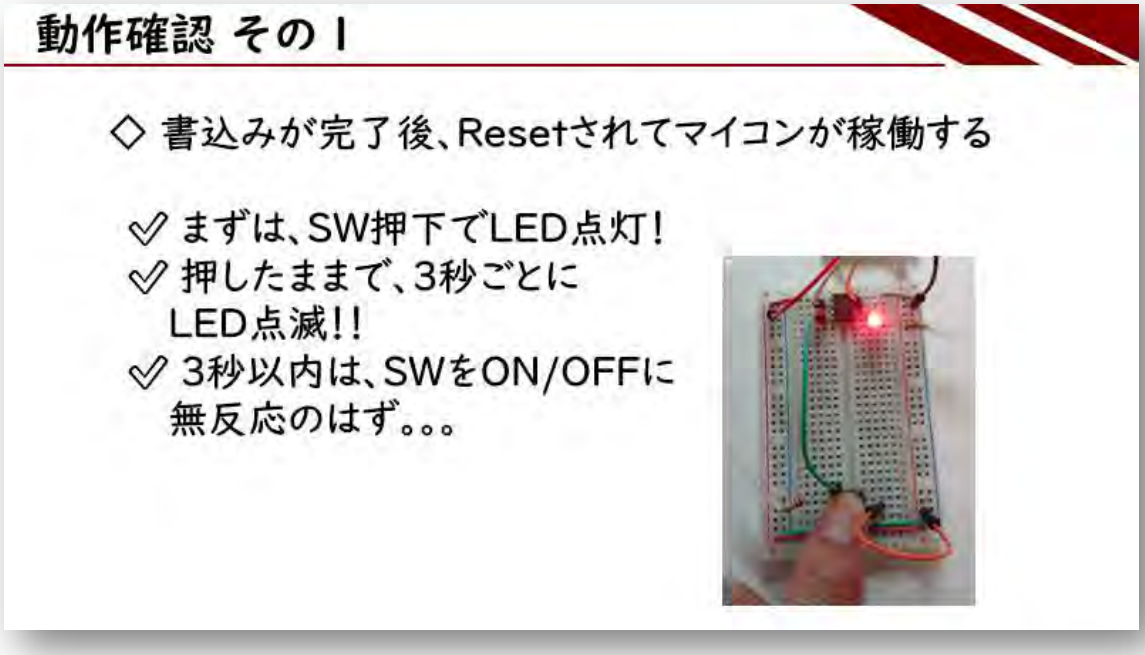


図　70

動作確認をしましょう。通常の状態では LED は消灯しています。SW を押下すると LED が点灯します。SW を押し続けると 3 秒ごとに点滅します。3 秒以内の SW 連打は無効で、3 秒間の pause(3000)が働いていることが分かります。

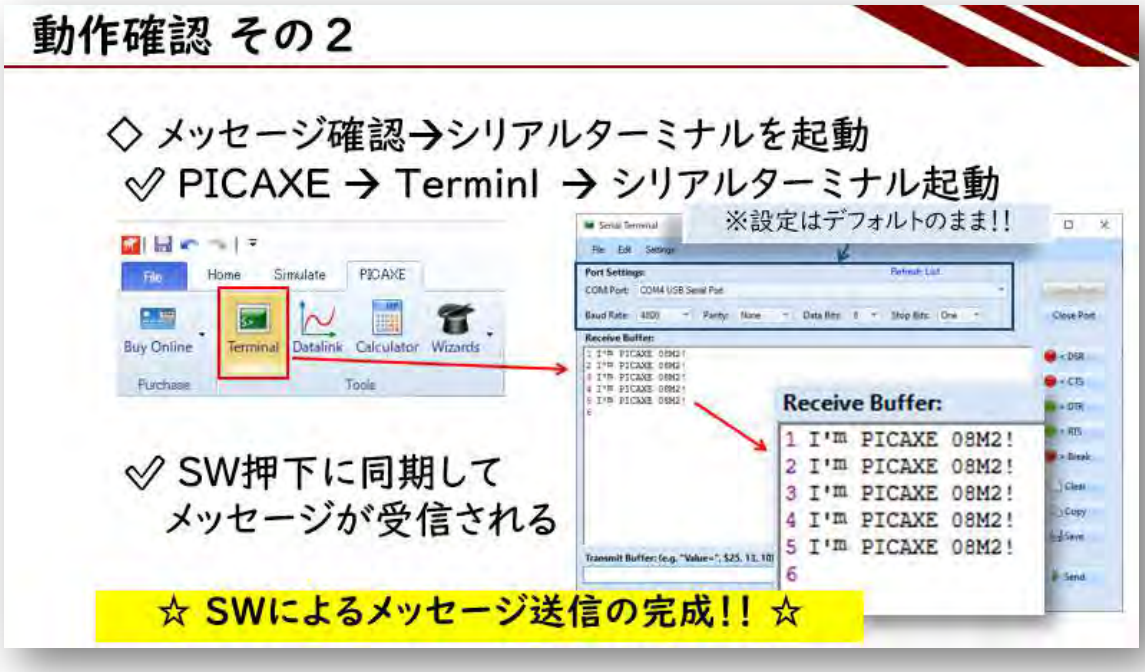


図　71

次にメッセージの送信が巧くできているか確認しましょう。PICXE Editor の PICAXE タブにある Terminal をクリックしてシリアルターミナルを起動します。回路上の SW を押下するたびに、メッセージが表示されます。sertxd()で送信したメッセージが、改行して表示されていればシリアル通信の送信は成功です。

第4回　シリアル通信【受信】

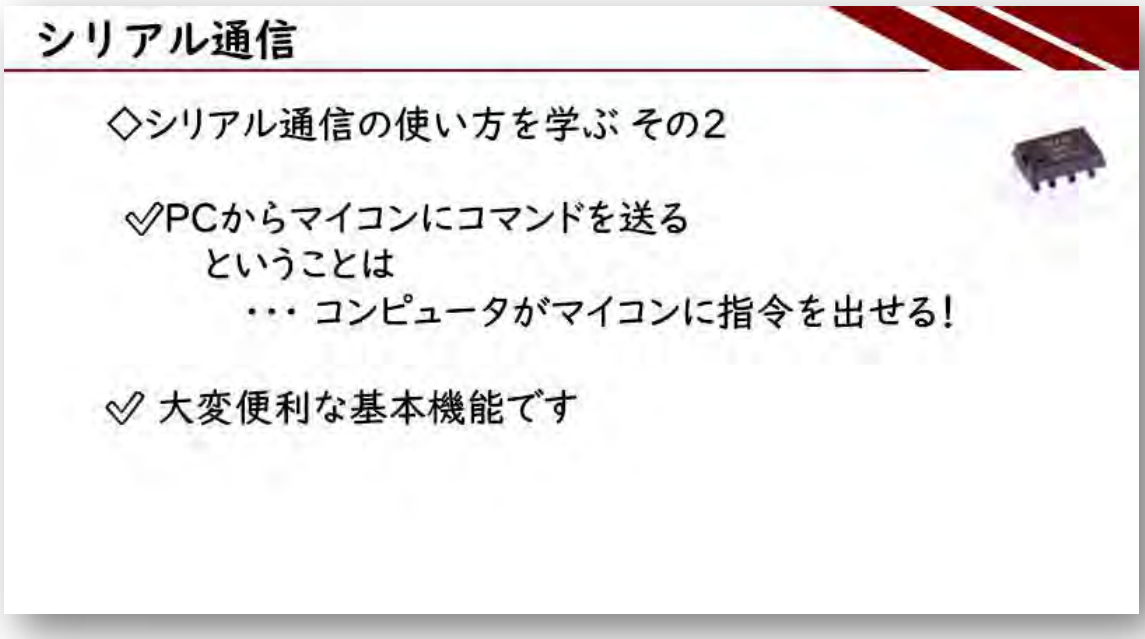


図 72

次は、受信機能を試してみましょう。シリアル通信の送受信ができれば、PC や他のマイコンからリモートコントロールできるシステムが開発できます。

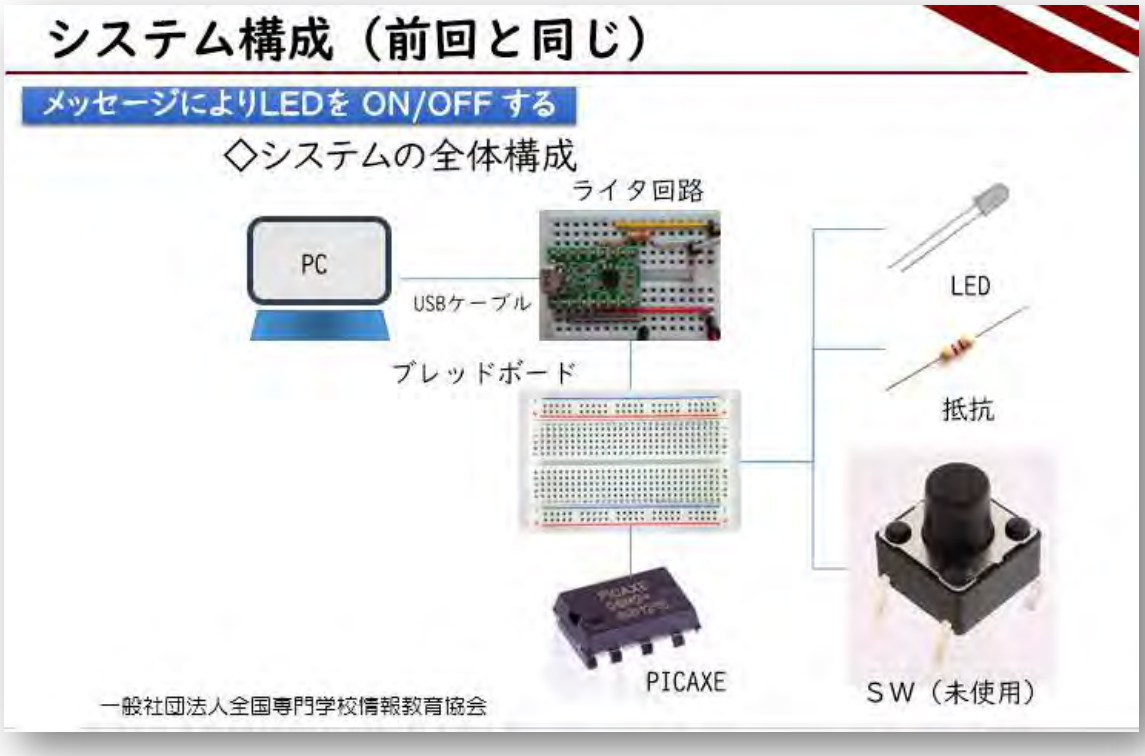


図　73

今回の回路も、これまでの物を使用します。但し SW は実験では使用しません。パーツやジャンパ線のはずれ、緩みをチェックしてください。

図 74

回路が前回と同じなので、PICAXE の使用ピンも同じです。空いているピンが少なくなってきました。PICAXE の電源は USB ケーブルを通じて PC からライタ基板に供給されている 5V の電源を使います。

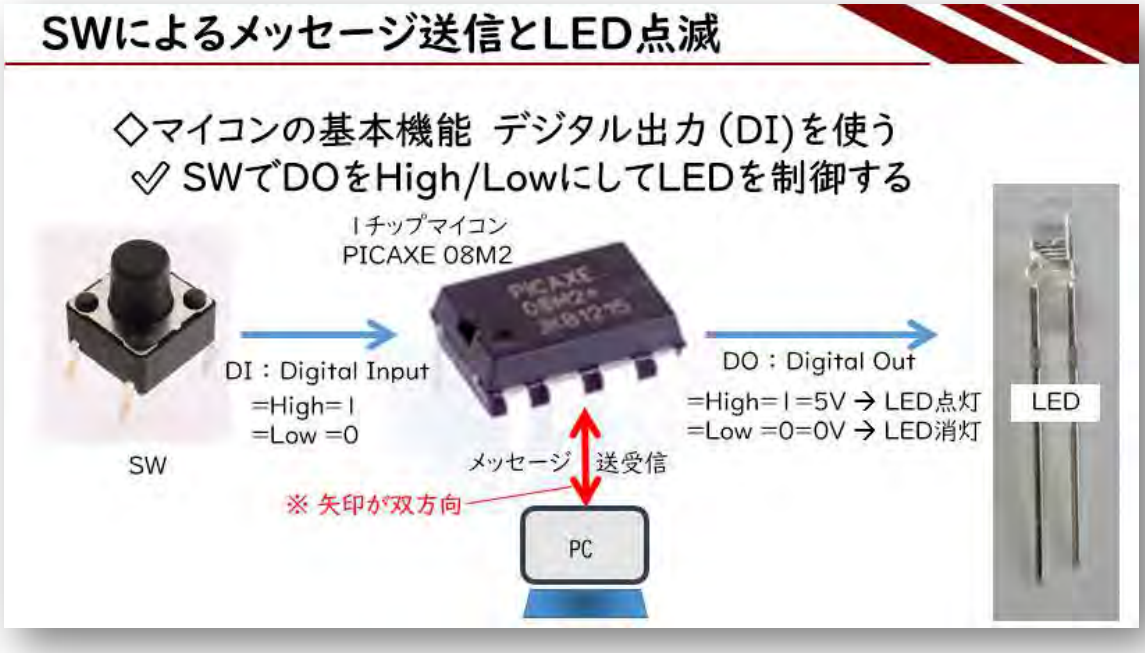


図 75

前回は PC 間の矢印が一方向でしたが、今回は双方向になっています。

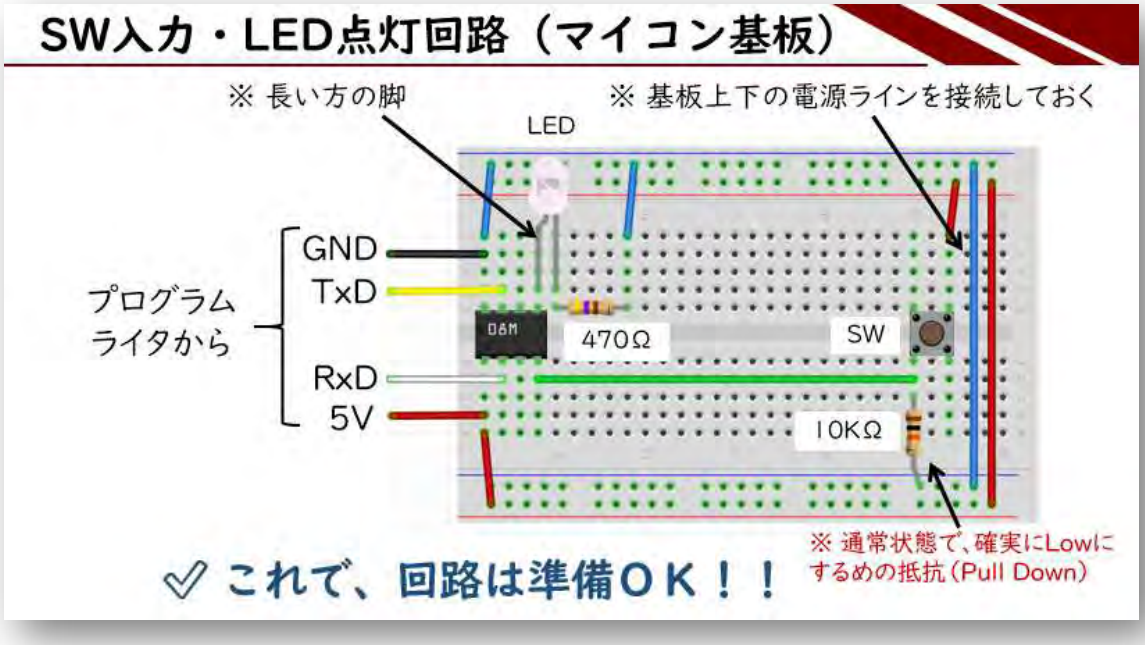

SW入力・LED点灯回路（マイコン基板）
※ 長い方の脚
※ 基板上下の電源ラインを接続しておく
LED
GND
TxD
プログラム
ライタから
RxD
5V
470Ω
SW
10KΩ
※ 通常状態で、確実にLowに
するめの抵抗（Pull Down）
これで、回路は準備ＯＫ！！

図　76

回路も同じです。

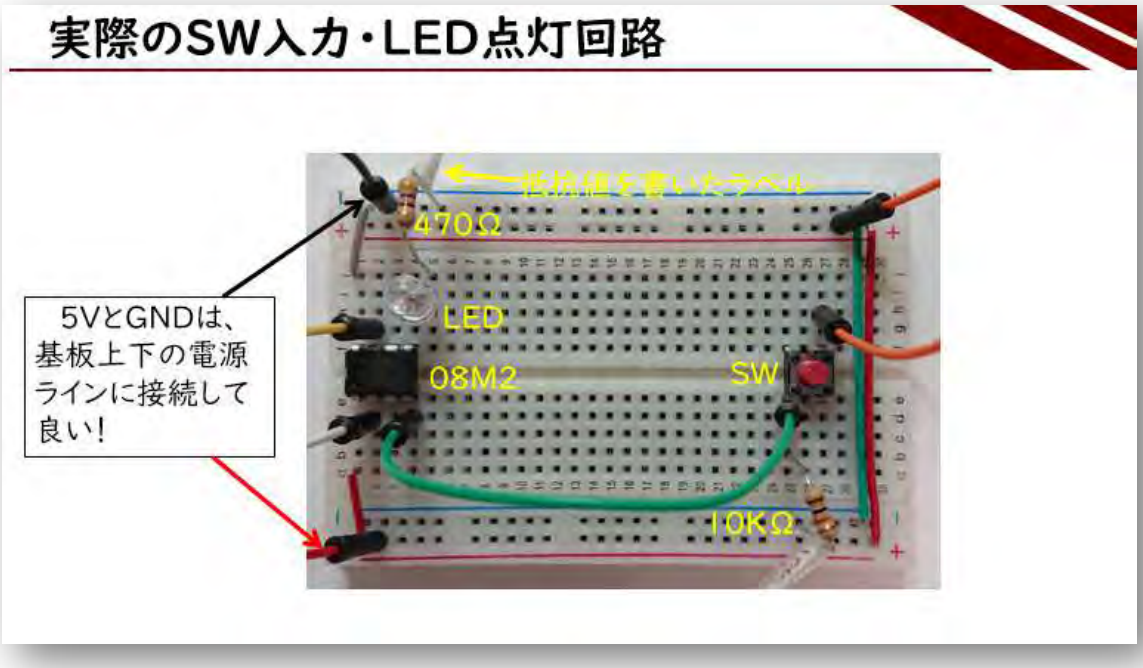


図　77

実際の回路を図に示します。

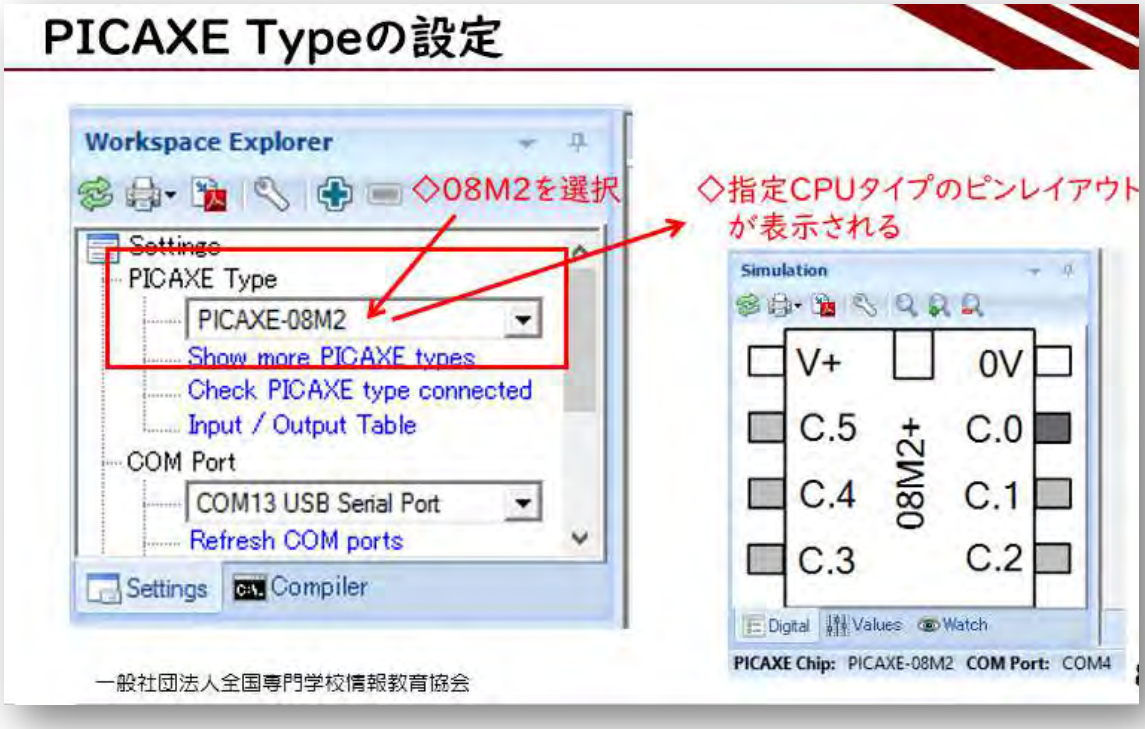


図　78

PICAXE Editor 左上のウインドウで対象の PICAXE（PICXE08M2）チップを選択すると、左下ウインドウに指定 PICXE のピン配置が図示されます。CPU タイプが間違っているとプログラムのコンパイルや書込みができません。

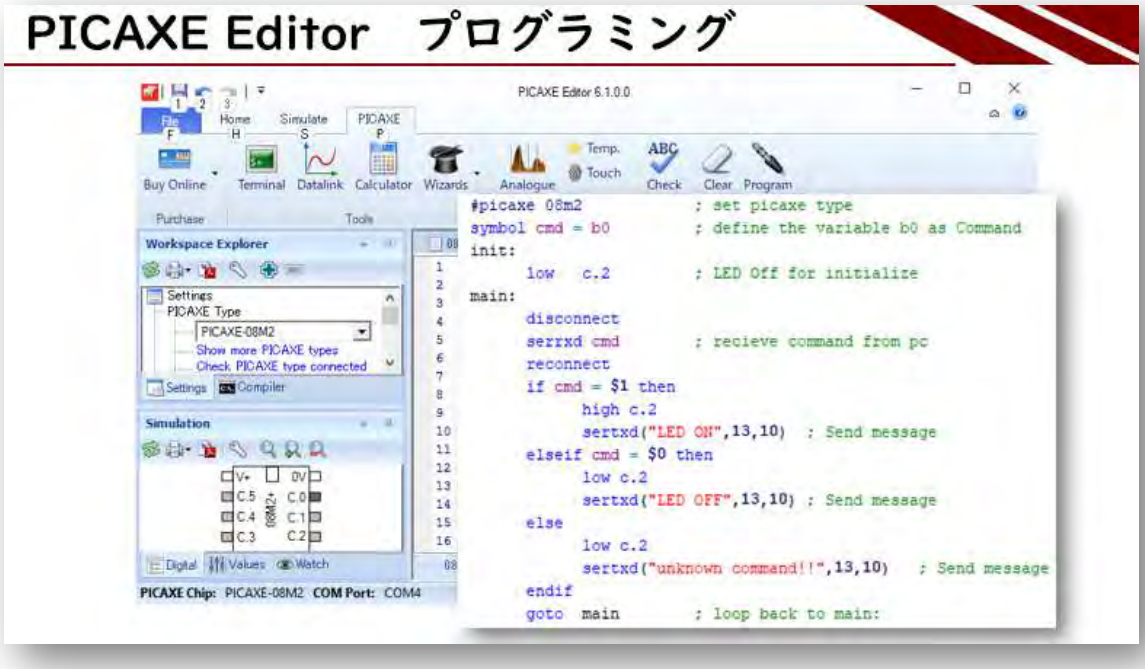


図　79

新しいプログラムを作成します。

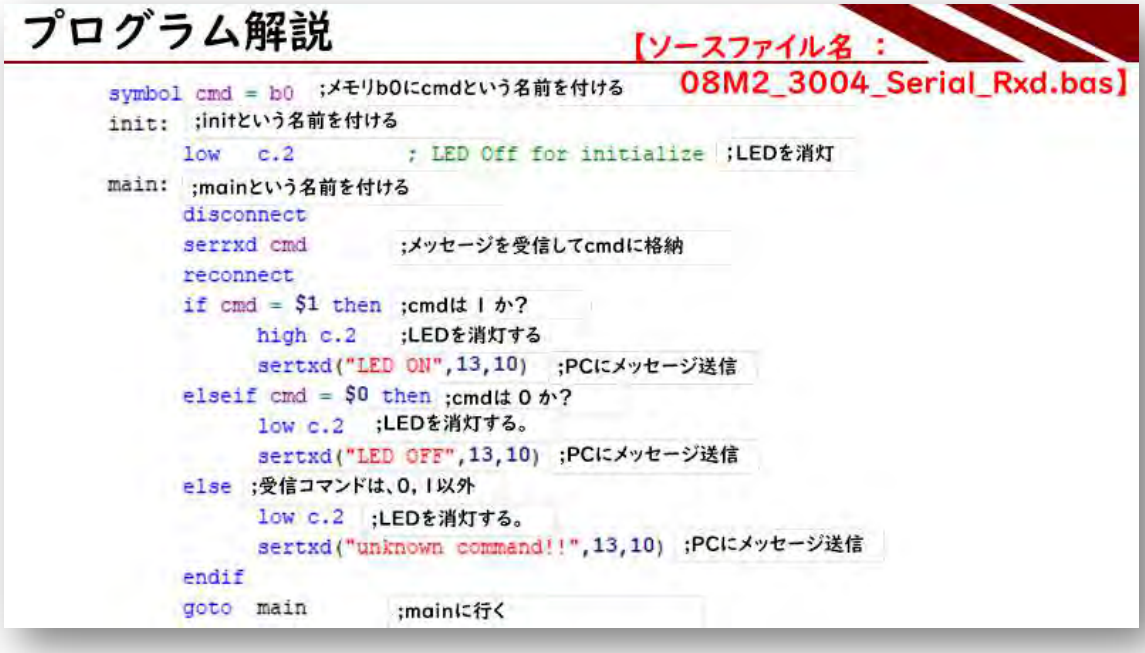


図 80

回路を作成しない分、送受信機能のプログラムは少し長めになりました。disconnect と reconnect という記述があります。シリアル通信はプログラムの書込みを行うためにも使われます。マイコン側ではプログラムの書き込み要求がいつ来るか分かりません。書き込みを行う際にマイコンのアプリケーションがシリアル通信を行っていると、PICAXE が PC からの書き込み要求を判断できなくなってしまうので、disconnect で新しいプログラムの書き込み要求のスキャンを停止してアプリケーションの通信に専念し、reconnect で再びプログラムの書き込みスキャンを有効にしています。この部分のプログラムを誤って設計すると、二度と新しいプログラムが書込めなくなってしまうので、注意が必要です。

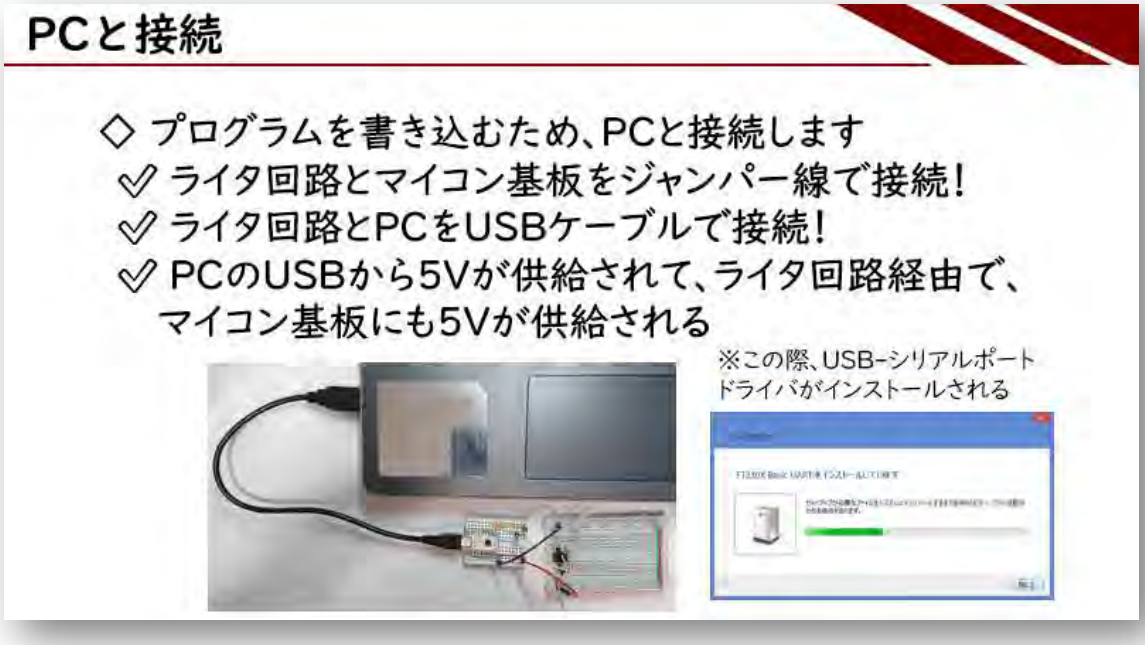


図　81

プログラムが完成したらコンパイルと書込みを行います。図のようにPC、ライタ回路、マイコン回路を接続して下さい。初めてPCとライタ回路を接続すると図右の様にドライバがインストールされます。

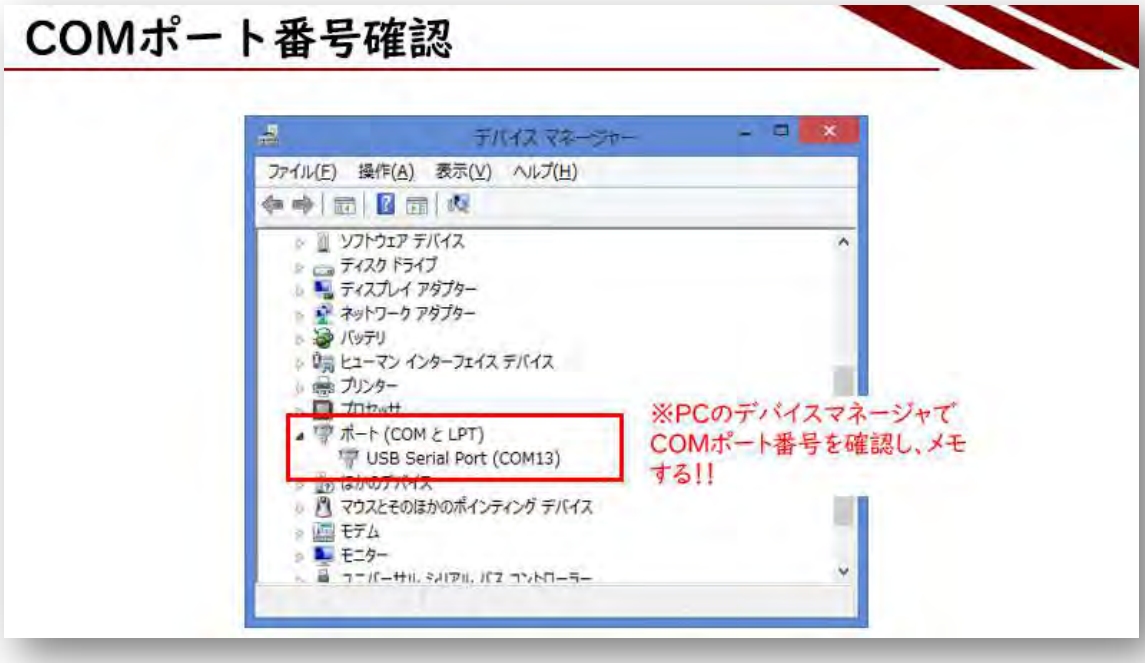


図 82

PC との接続後、Windows のデバイスマネージャで、図の COM ポート番号を確認してメモしてください。COM ポート番号は同じ PC で同じライタ回路を用いる場合は以後も変わりませんが、PC に別のライタ回路を接続すると番号が変わりますので、注意してください。

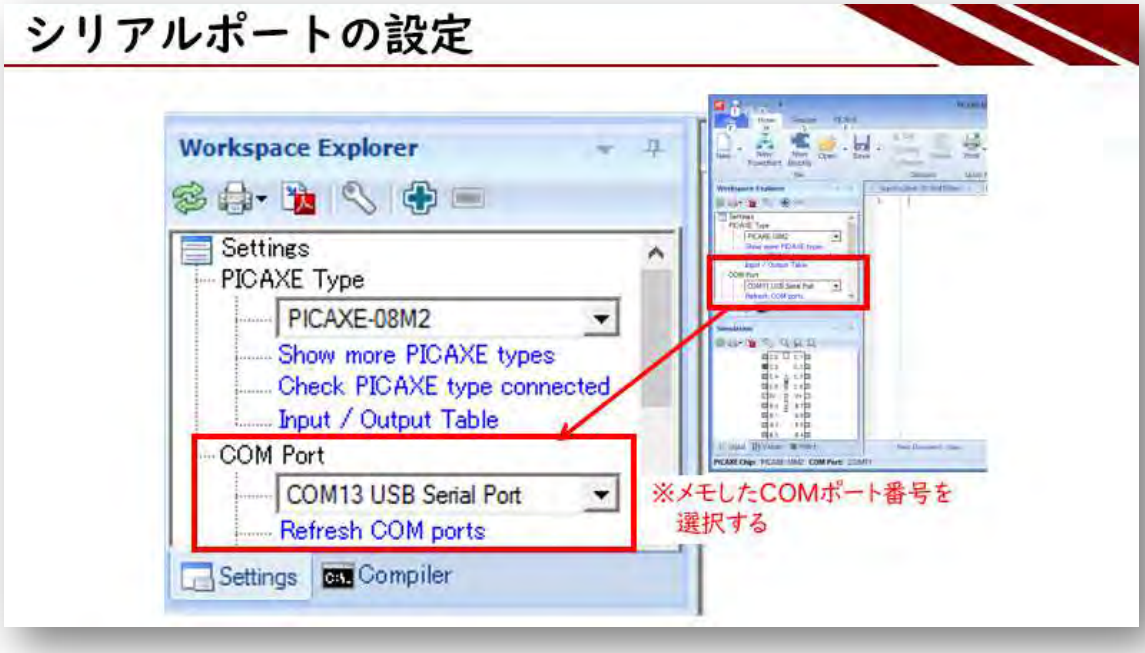


図 83

確認した COM ポート番号を設定します。

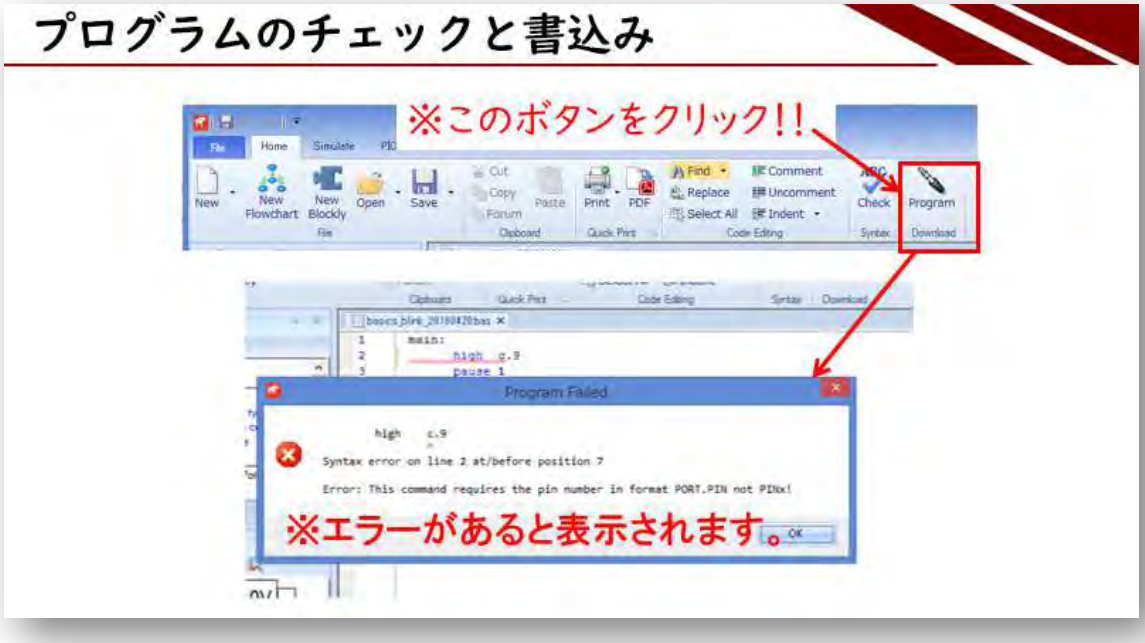


図 84

PICXE Editor の Home タブ右側にある Program ボタンを押下すると、ソースコードのコンパイル等が行われます。エラーがあると図の様なメッセージが表示され、エラー箇所が示されます。その際は修正して再び Program ボタンを押下します。

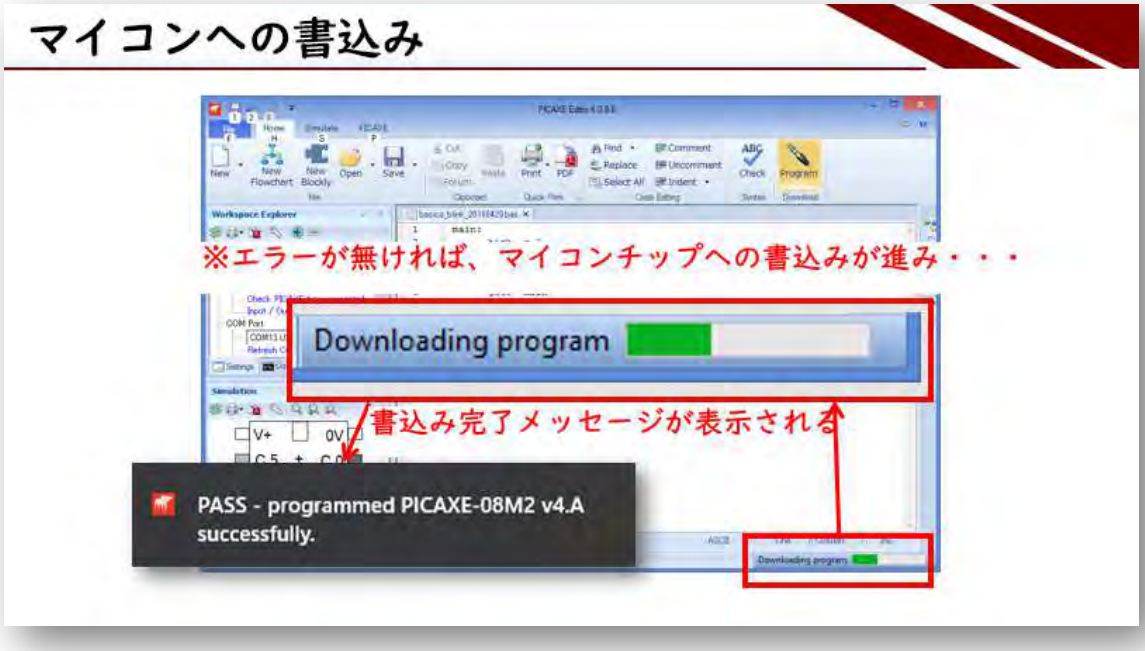


図　85

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

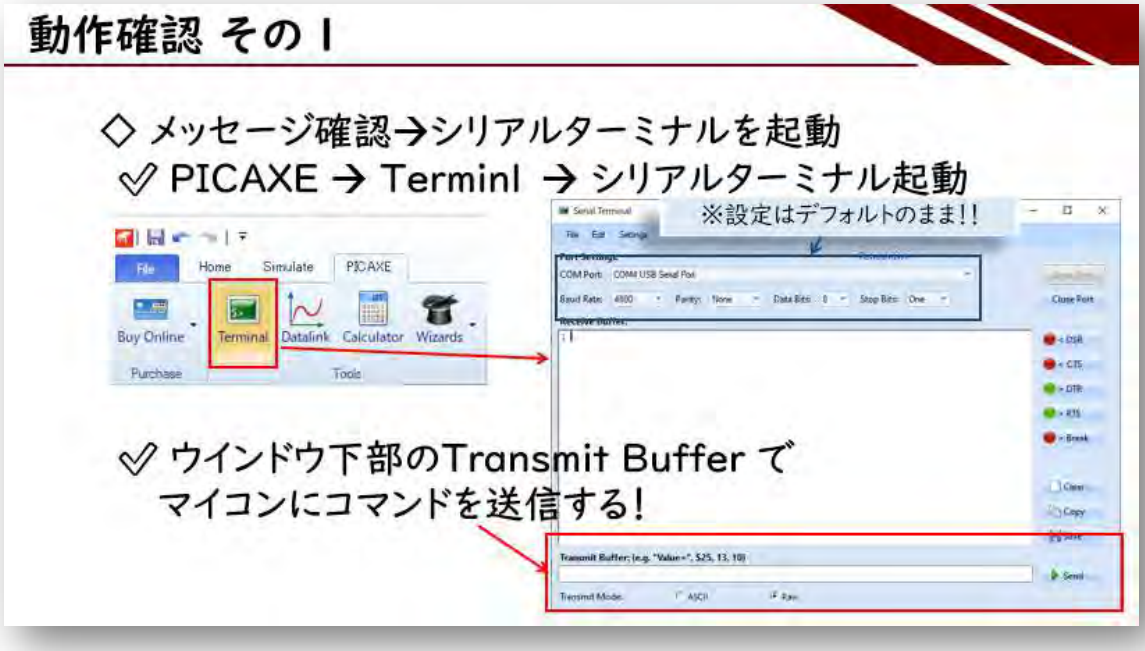


図　86

動作確認はシリアルターミナルを起動して行います。シリアルターミナルウインドウの下部に Transmit Buffer と書かれている窓があります。この窓にカーソルを入れて、送信文字列を書込みます。

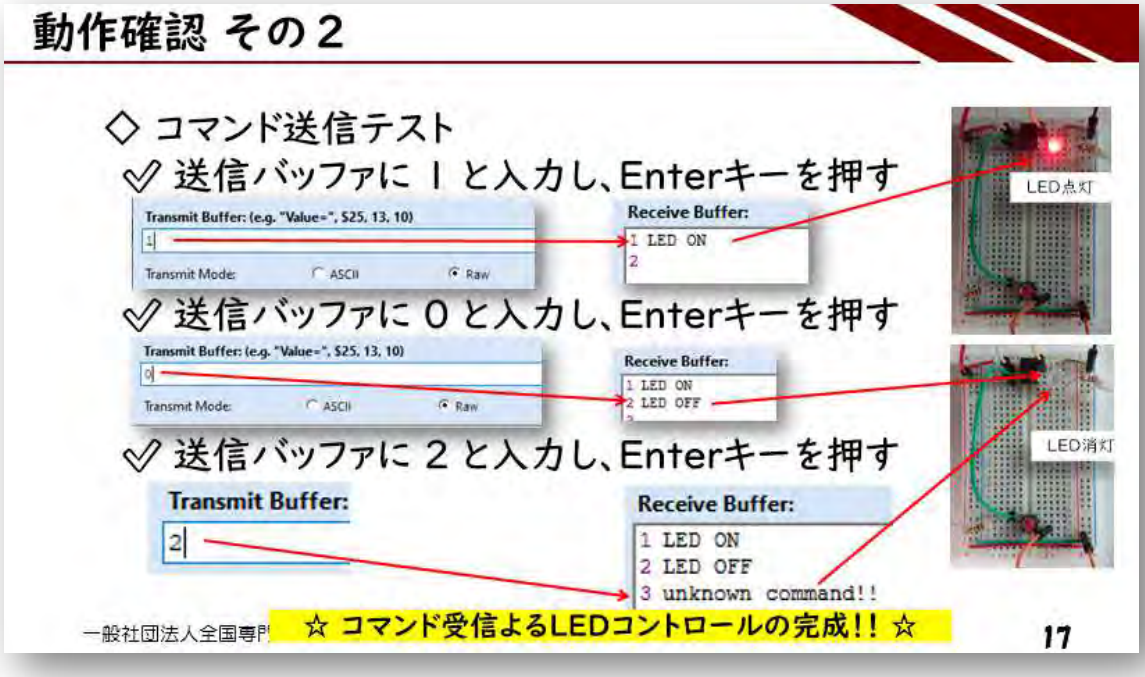


図　87

送信バッファに 1 と入力して、Enter キーを押下します。すると LED が点灯して、シリアルターミナルの Receive Buffer に LED ON とメッセージが返ってきます。次に 0 と入力して Enter キーを押下すると LED は消灯して、Receive Buffer に LED OFF と返信があります。さらに 0,1 以外の文字を送信すると LED は消灯して、返信は unknown command!! となります。

これがシリアル通信の送受信です。上手くできたら、他の数字や文字で別の LED 点灯パターンを実現するプログラムを作ってみて下さい。

第5回　VR（ADC）

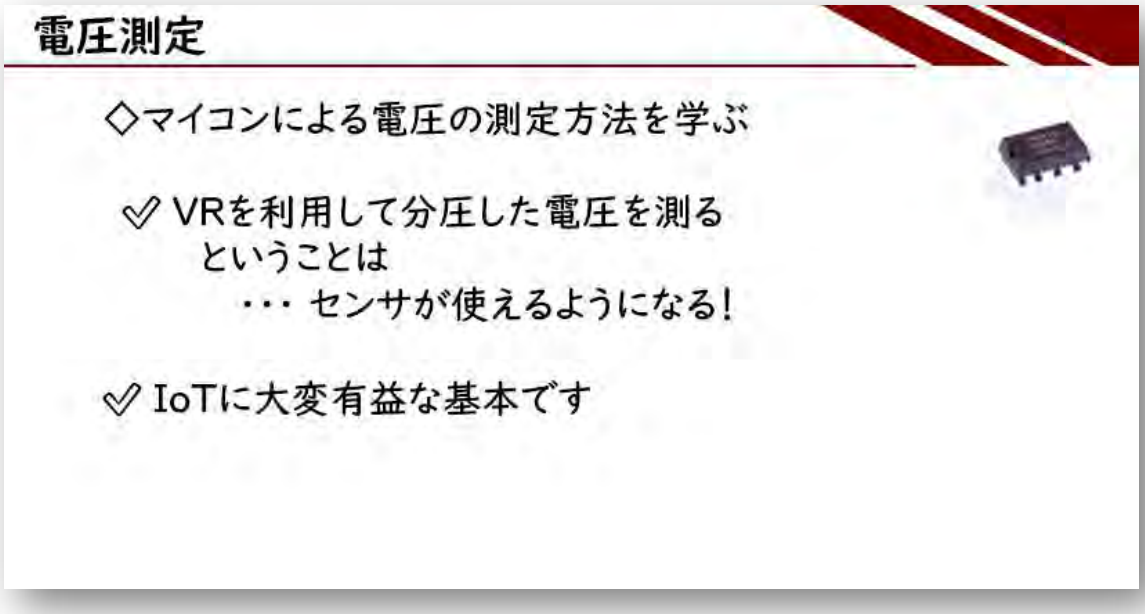


図 88

IoT では様々なセンサを利用して環境測定を行います。これまでに最も多く利用されているセンサは、測定結果を電圧で出力するアナログセンサです。今回は電圧測定を行います。タイトルにある VR と電圧は密接な関係が有って、VR は電圧を変化させる機能を持っています。VR によって変化する電圧の測定方法を知れば、ほとんどのアナログセンサを取り扱える基礎が整います。

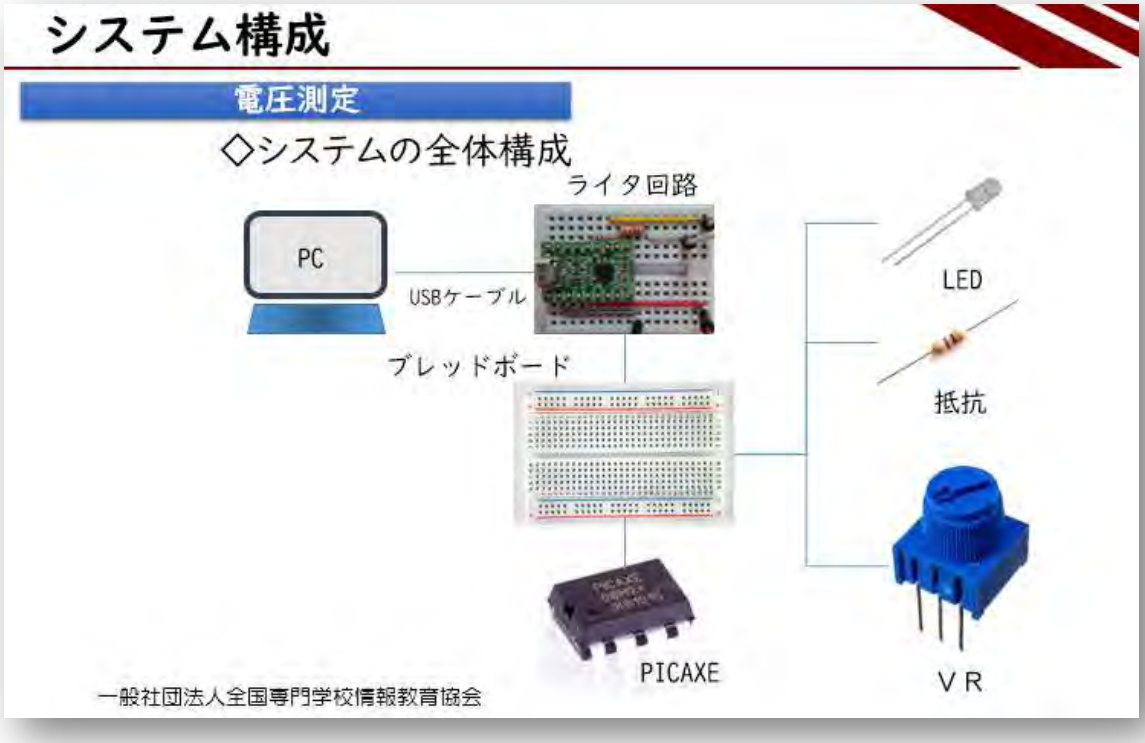


図　89

使用するパーツを図に示します。右側に掲げたパーツで VR だけあれば電圧測定はできますが、LED と抵抗も掲載してあります。後半の応用実習で使用するために前回の LED 点灯回路を残しておきます。

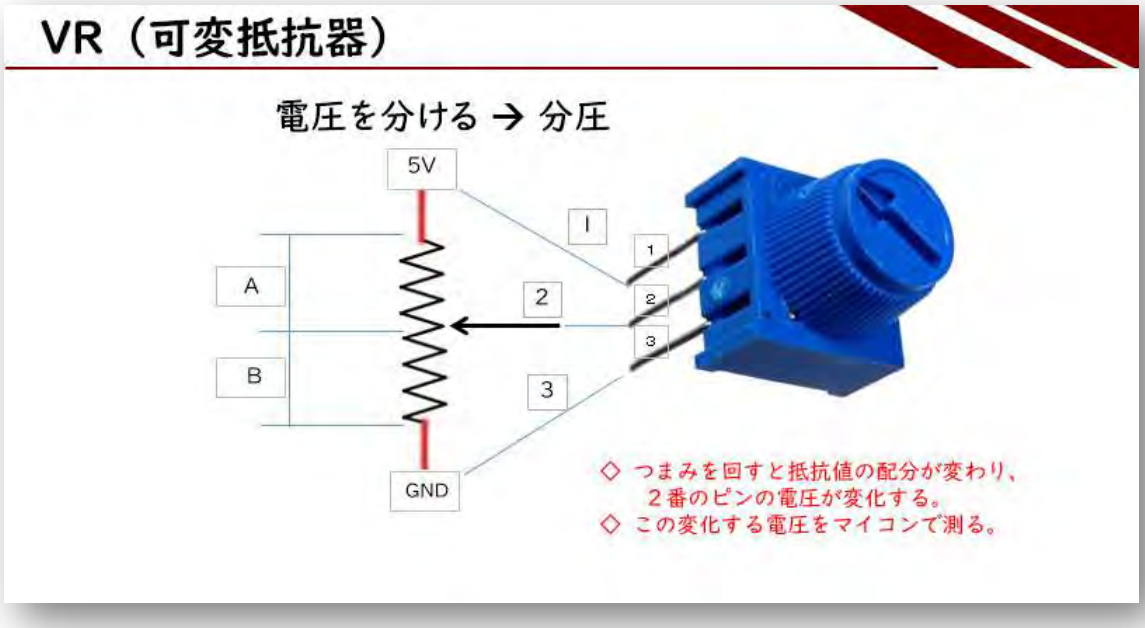


図 90

VR には 3 本の脚ピンがあります。図に示すように番号を付けて、1, 3 番ピンをそれぞれ 5V、GND に接続すると、矢印型の溝があるつまみの回転位置（2 番ピンの図の矢印部分が上下する）に応じて、2 番ピンの電圧が変化します。この電圧【V】は次のように VR 全体の抵抗値（A+B）【Ω】に対する B の抵抗値の割合で計算できます。

$$E = 5 \times \frac{B}{(A+B)}$$

E:2 番ピンの電圧、5：1 番ピンの電圧

この 2 番ピンの電圧をマイコンで測定します。VR は図の矢印部分で電圧を分けています。このことを**【分圧】**と言います。

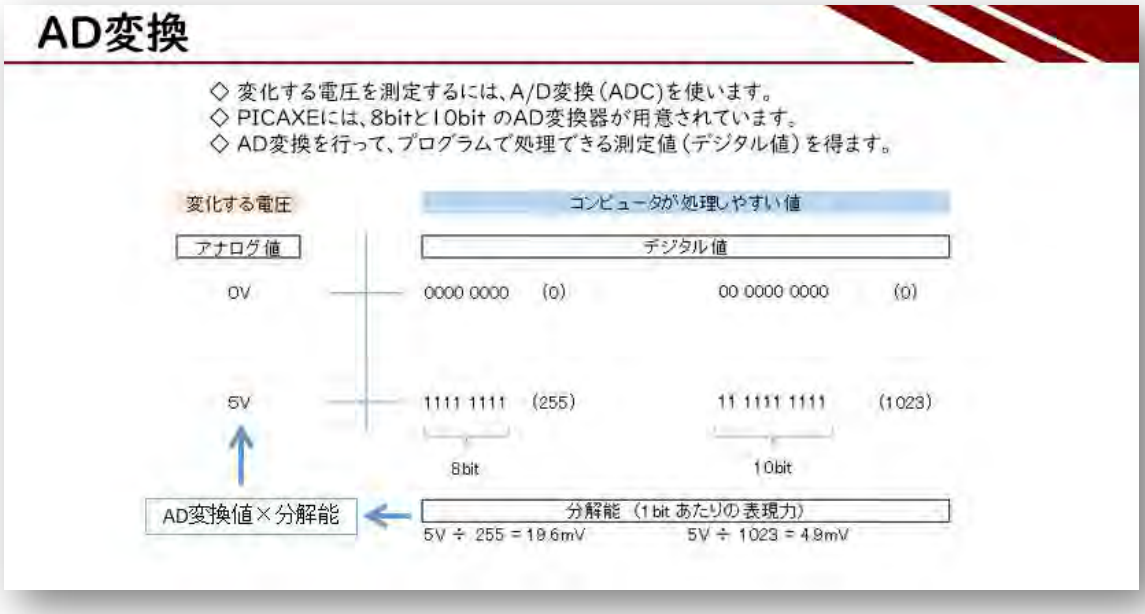


図 91

電圧測定には ADC（Analog Digital Converter）を用います。PICAXE08M2 は 3 番ピンに ADC が内蔵されています。測定の範囲は 0～5V で、プログラムにより 8bit または 10bit のデジタル値として読み込むことができます（A/D 変換）。図に示すように、0V は 8bit、10bit ともに 0 に、5V は 0xFF（8bit）または 03FF（10bit）に変換されます。1bit あたりの分解能は 8bit で 19.6mV、10bit で 4.9mV です。VR から入力される電圧を A/D 変換した値に分解能を掛け算すれば、電圧が求まります。

図 92

VR の 2 番ピンは PICAXE の 3 番ピン（C.4）に接続します。LED は 5 番ピン（C.2）に接続します。PICAXE の電源はこれまでと同様に USB ケーブルを通じて PC からライタ基板に供給されている 5V を使います。4 番ピン（C.3）は応用実験で使用します。

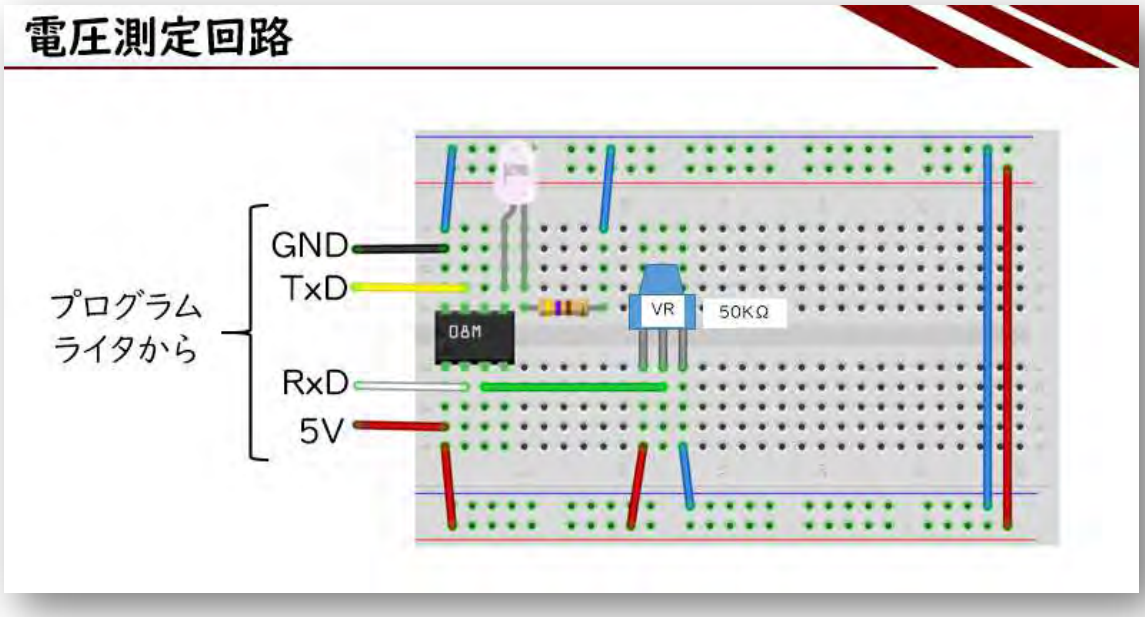


図 93

電圧測定回路は VR と 3 本のジャンパ線を配線するだけです。容易な作業ですが、VR と電源（5V・GND）の接続が間違えていないか、よく確認をしてください。5V と GND の接続を逆配線してパーツが壊れてしまう場合があります。

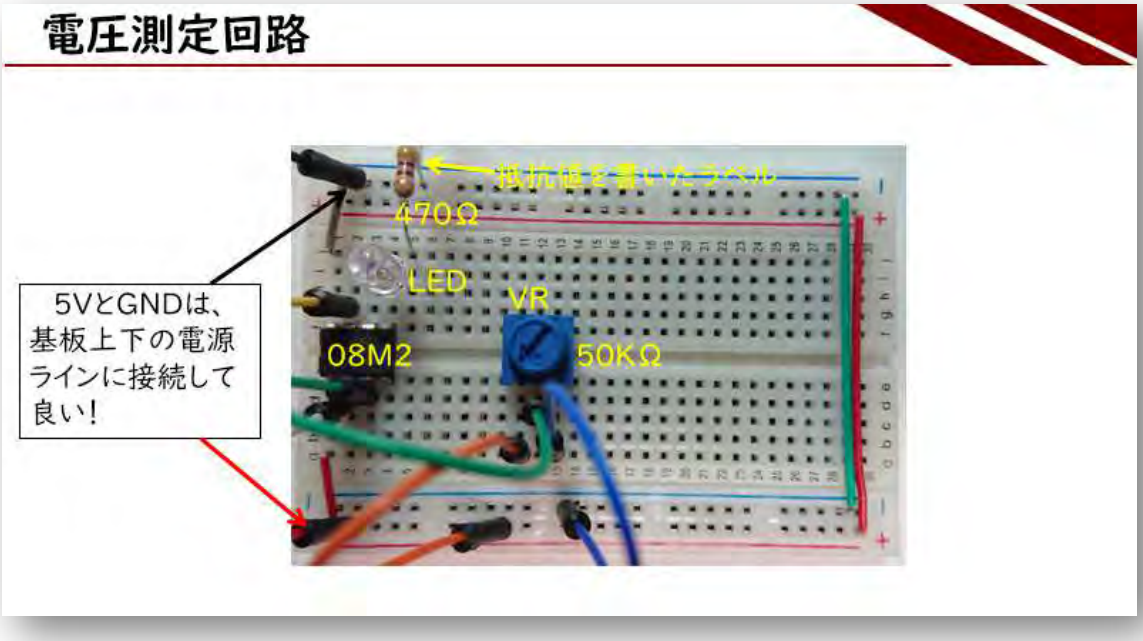


図 94

実際の回路を図に示します。LED 点灯回路は、後半に使います。

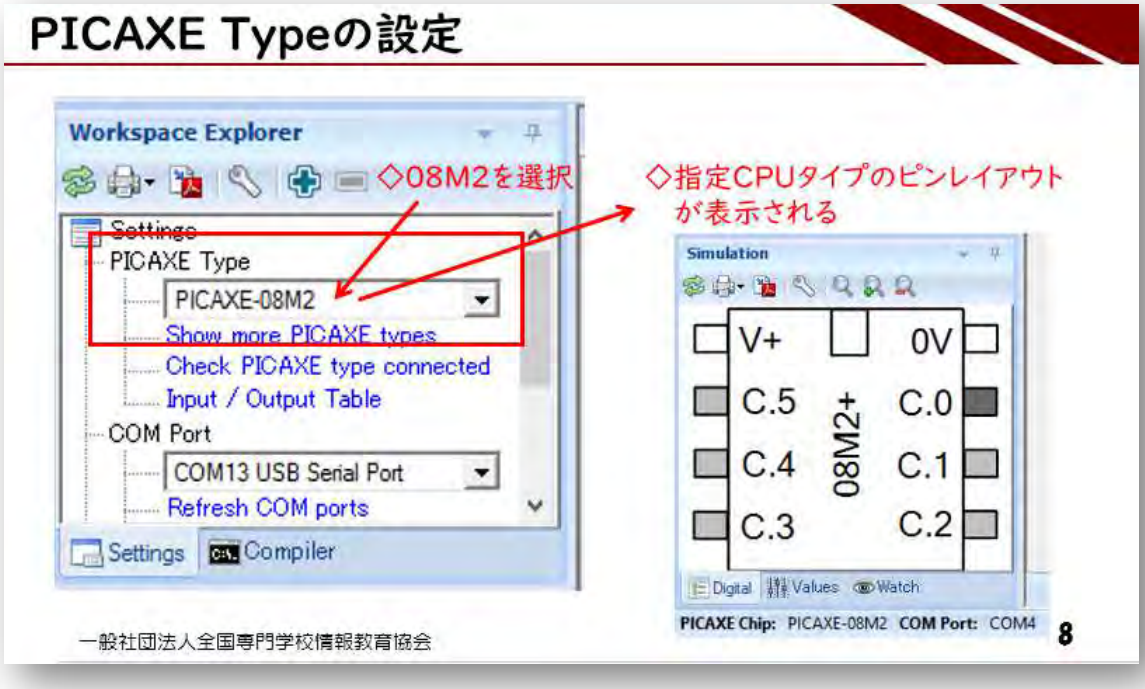


図　95

PICAXE Editor 左上のウインドウで、対象の PICXE08M2 チップを選択します。

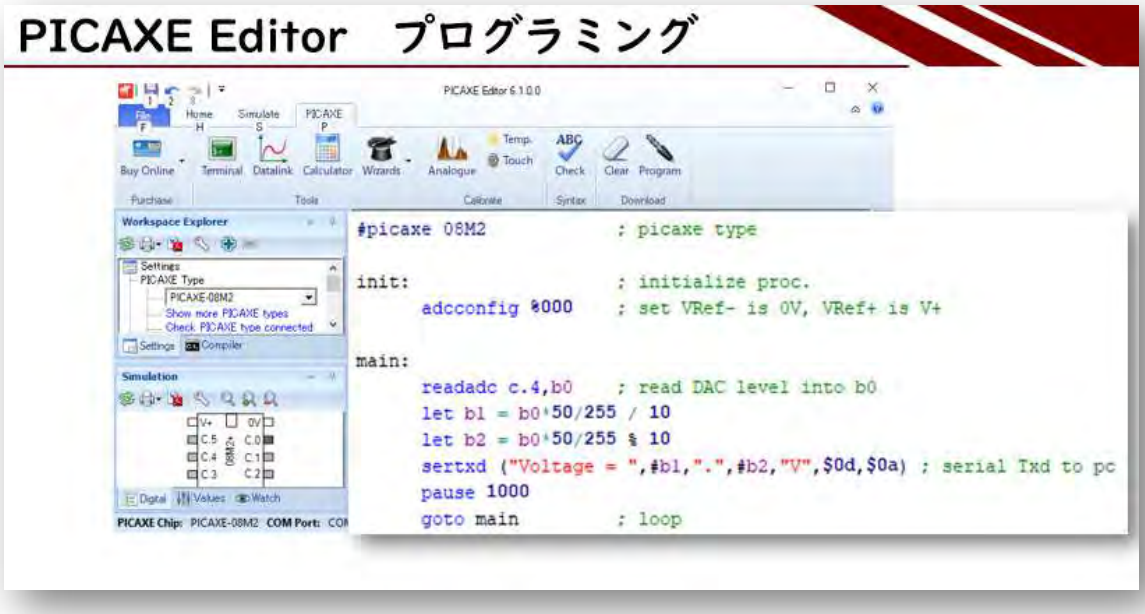


図　96

新しいプログラムを図に倣って記述します。

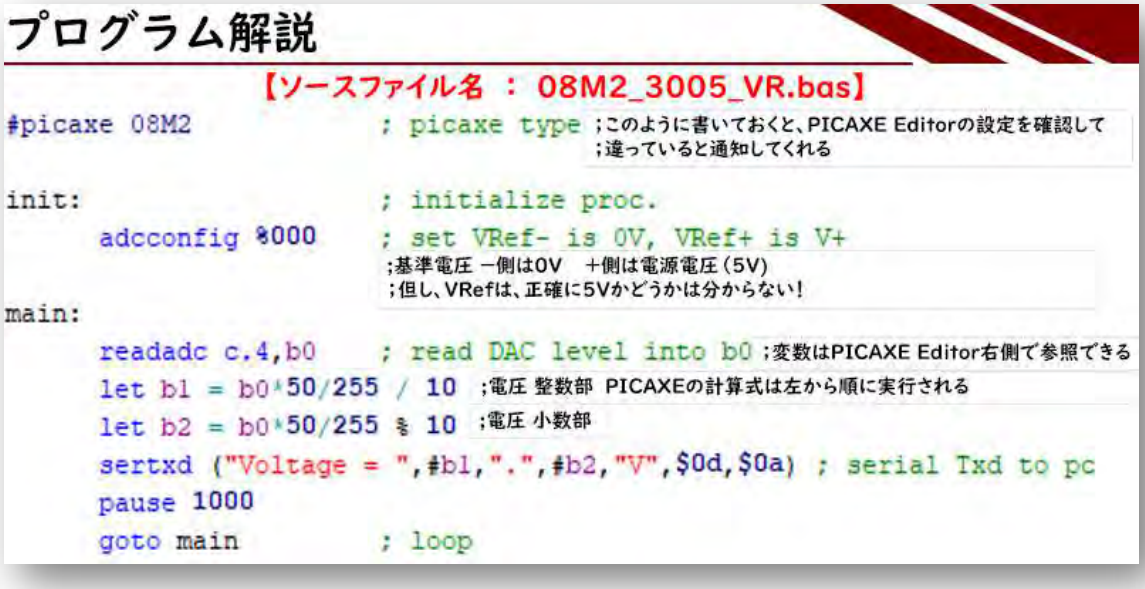


図 97

最初の記述は、コンパイラに CPU タイプを知らせる行です。このように記述しておくと開発環境での指定（プルダウンでの設定）とソースコードの指定を比較して不一致の場合には通知してくれます。init:とラベルが付いている部分では内蔵されている ADC に基準となる電圧を設定しています（マニュアル参照）。main:では 8bit で C.4 ピンの入力を A/D 変換し b0 番のメモリに結果を格納しています。PICAXE Editor の右側にある Code Explorer にマウスカーソルを重ねると使用できるメモリが分かります。10bit で A/D 変換したい場合は readadc10 という命令を使います。b0 に格納した A/D 変換値を整数部と小数部に分けて電圧変換します。5V を 10 倍して（50 にして）10 で割り算しているのは、PICAXE では実数計算ができないからです（let についてマニュアル参照）。sertxd()で測定結果をシリアル出力しています。

ソースコードは実習キット付属の CD に含まれています。

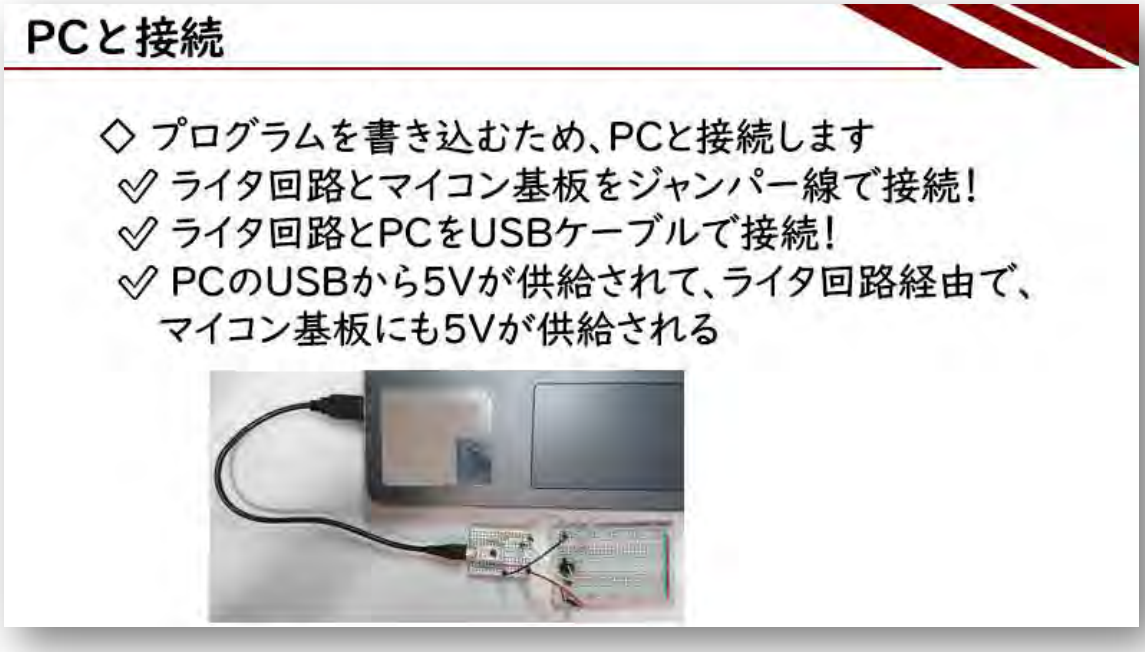


図 98

プログラムが完成したら、コンパイルと書込みを行います。図の手順で PC、ライタ回路、マイコン回路を接続して下さい。

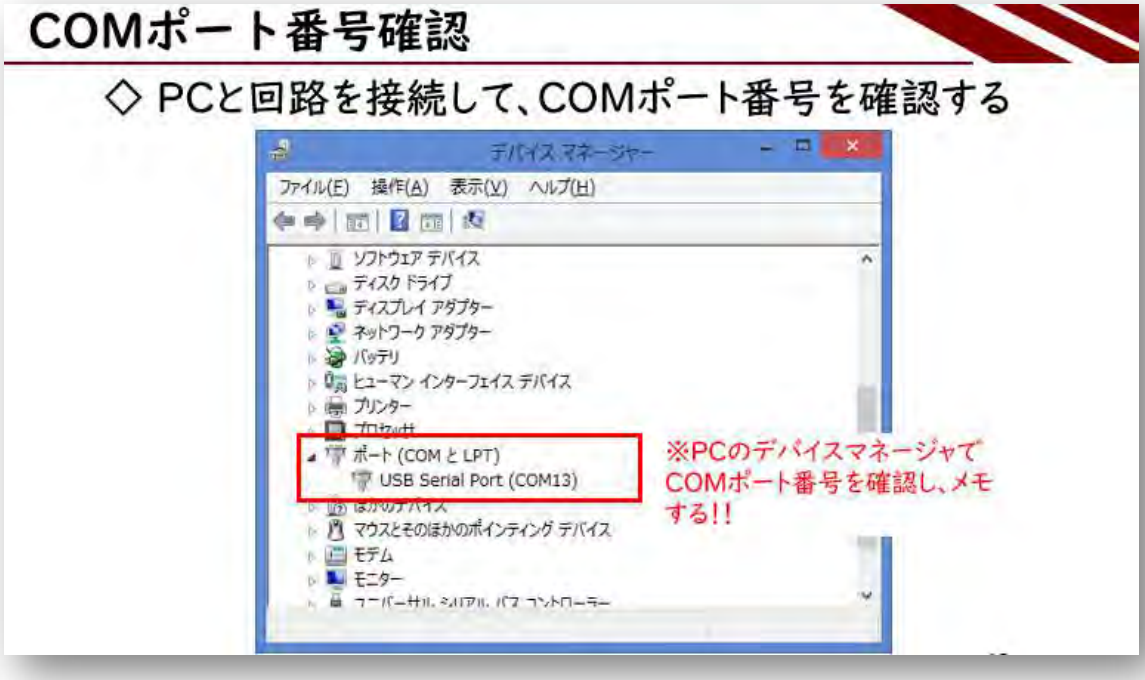


図 99

デバイスマネージャで COM ポート番号を確認します。

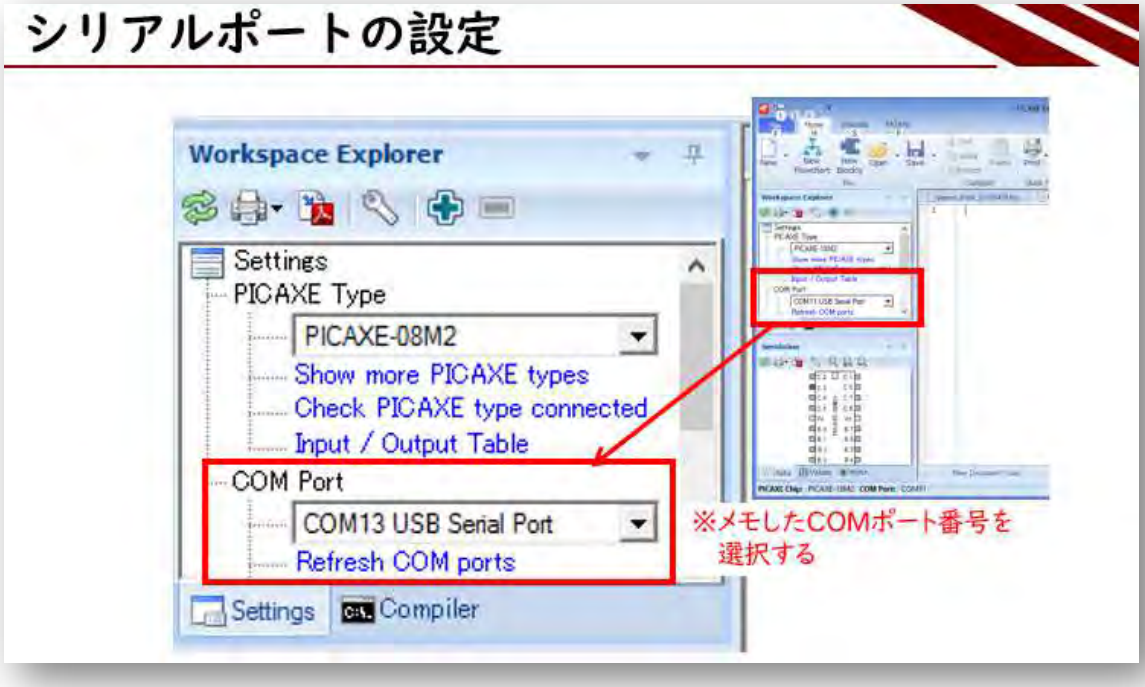


図　100

確認した COM ポートを選択します。

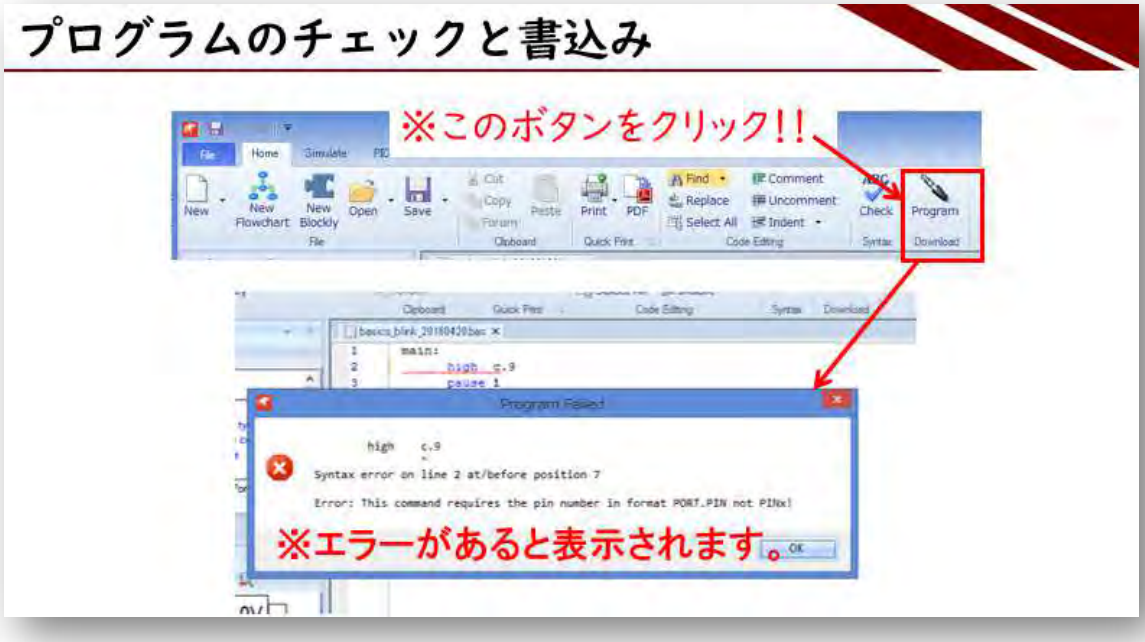


図 101

Program ボタンを押下します。

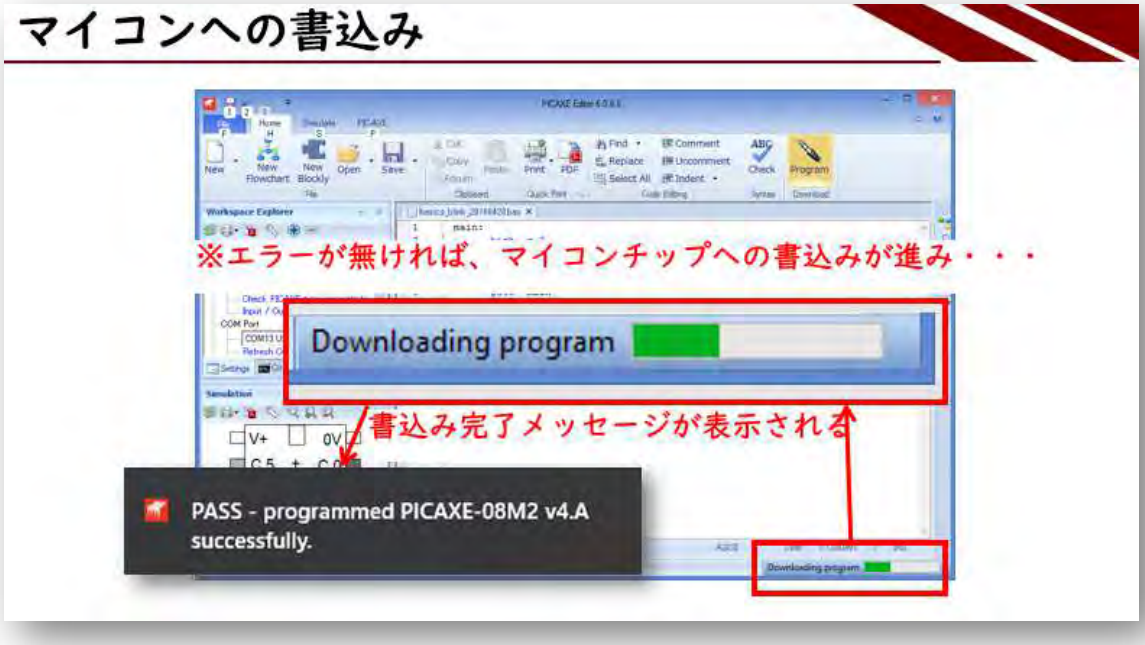


図 102

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色のバーが右に進み、マイコン内部にプログラムが書込まれます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

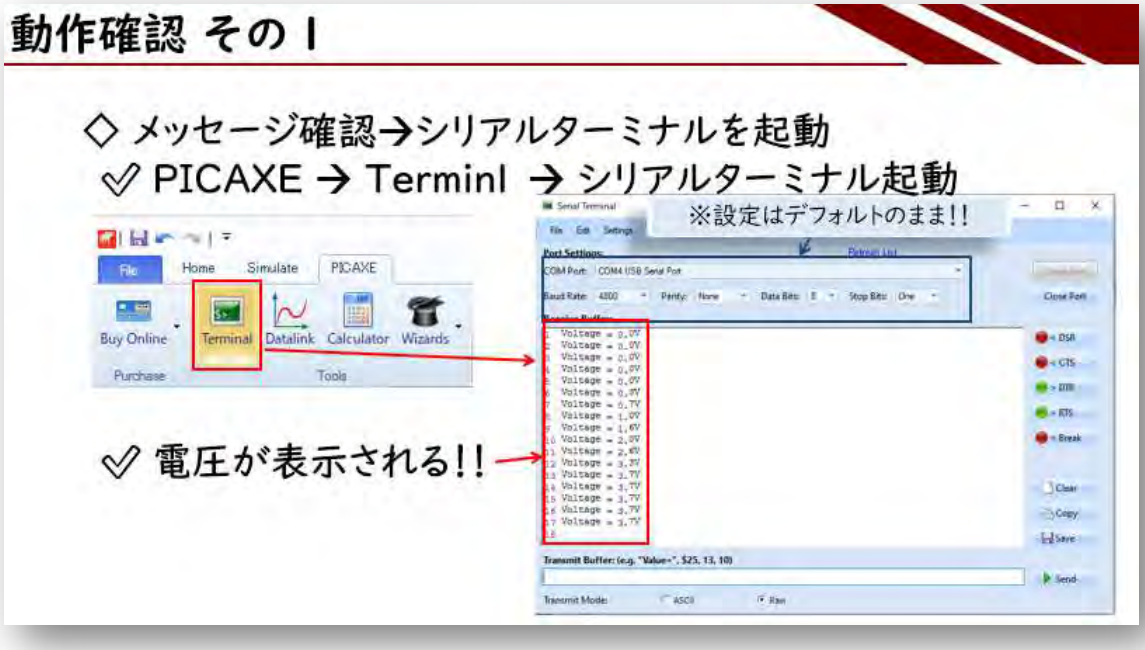


図　103

シリアルターミナルで計測した電圧を確認しましょう。VR のツマミを適当に回すと表示されている電圧が変化します。

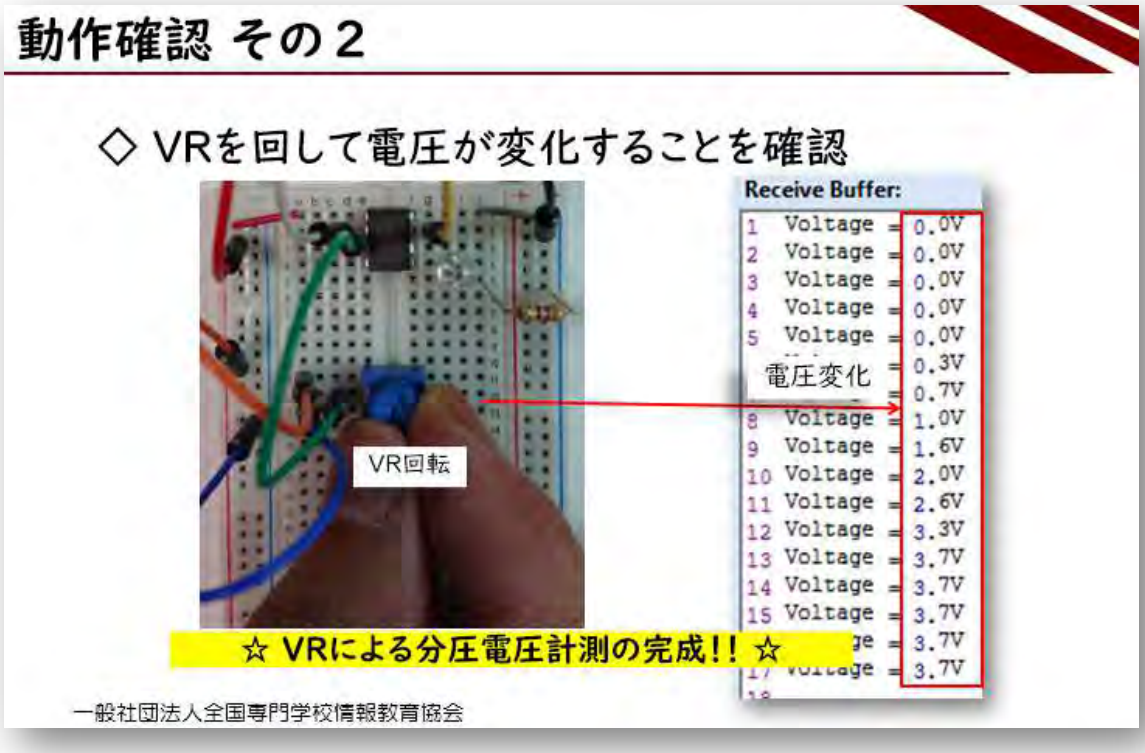


図 104

電圧が変化することが確認できたら、ツマミを一番左に回し切ります。このとき電圧は 0V になるはずです。次に少しずつ右に回していくと、徐々に電圧が大きくなるはずです。一番右に回し切ると、5V になるでしょうか？

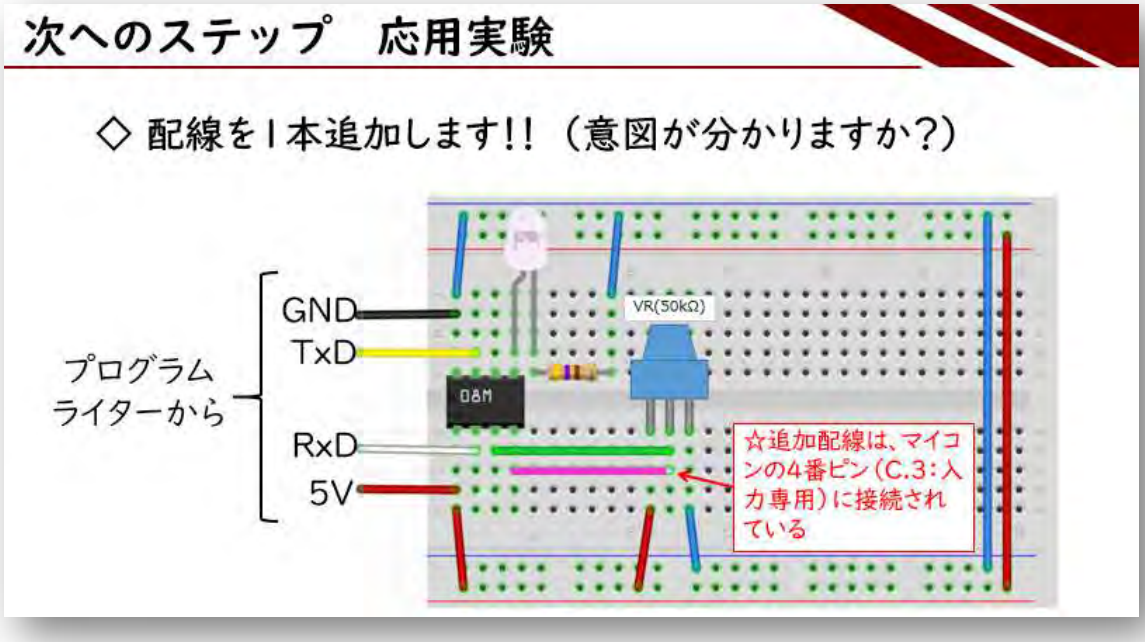


図　105

ここからは、LED を使った実験を行います。図で示す配線を 1 本追加してください。VR の 2 番ピンを PICAXE の 4 番ピンに配線します。VR の 2 番ピンは C.3 と C.4 の 2 箇所配線されました。

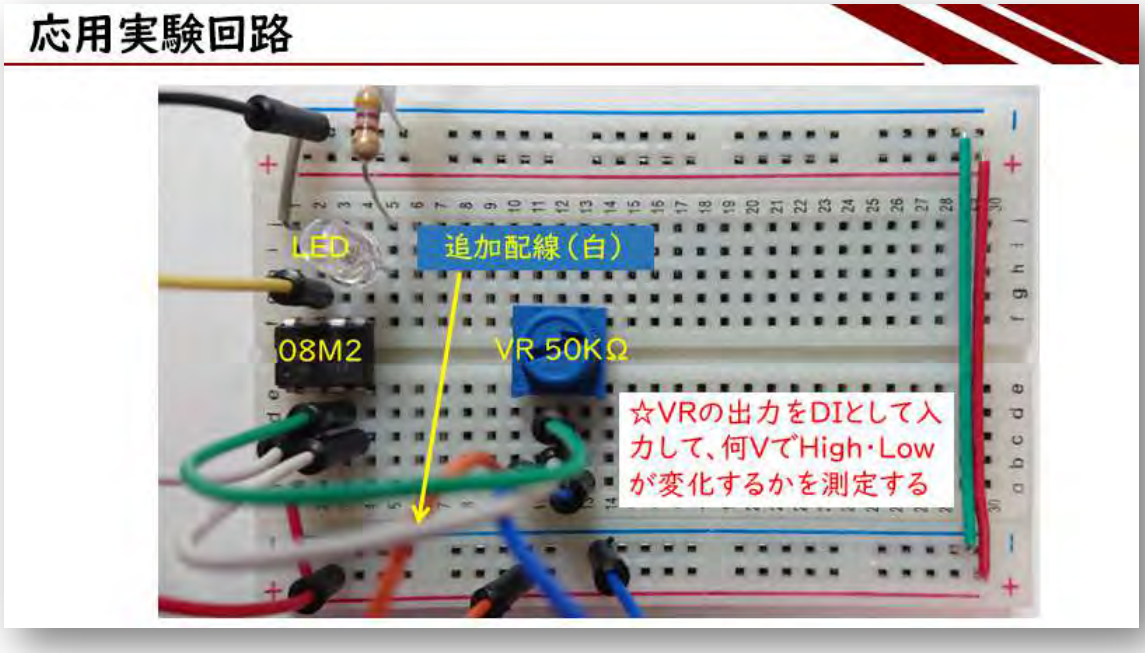


図　106

追加配線をした様子です。

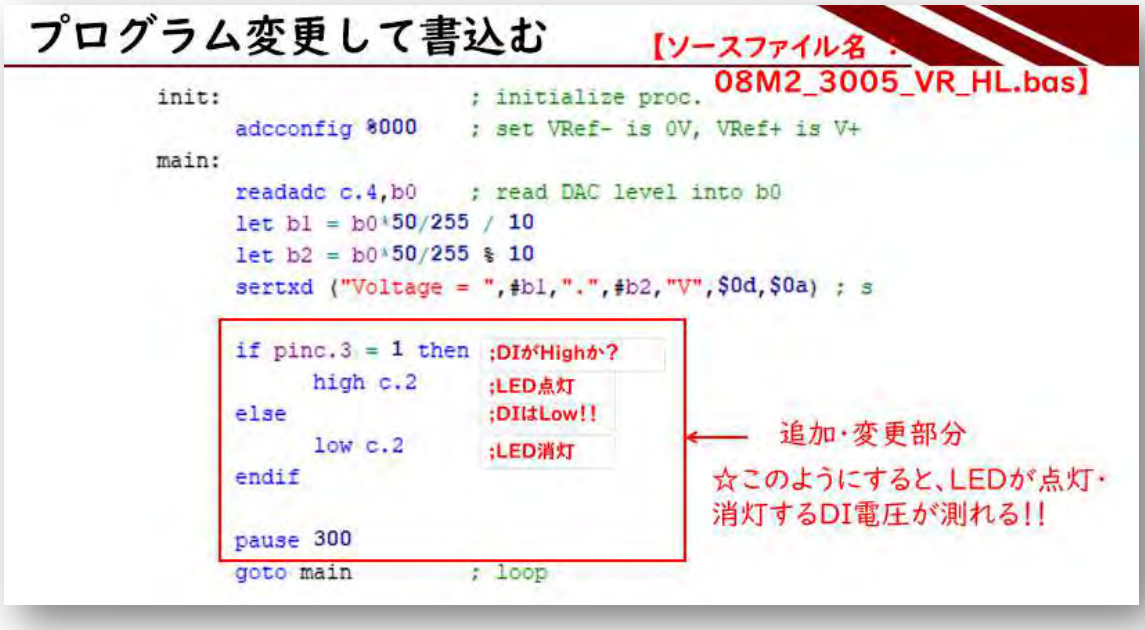


図 107

プログラムを一部追加・変更します。追加プログラムで行うのは、VR の 2 番ピンを DI として読み、High/Low を判断して LED に反映させることです。こうすると LED が点灯・消灯する際の電圧が分かります。つまり、このマイコンが何 V で High または Low と判断しているかが分かります。CPU のデータシートには、このような電気的な特性も記述してありますが、実際には個体差があるものです。また、マイコンの種類が違えばこの電圧も違います。シビアな制御を行おうとする場合は、このような実験をあらかじめ行って、CPU の特性を調べておきます。プログラムの変更が終ったら、前例に倣いコンパイルと書込みを行います。

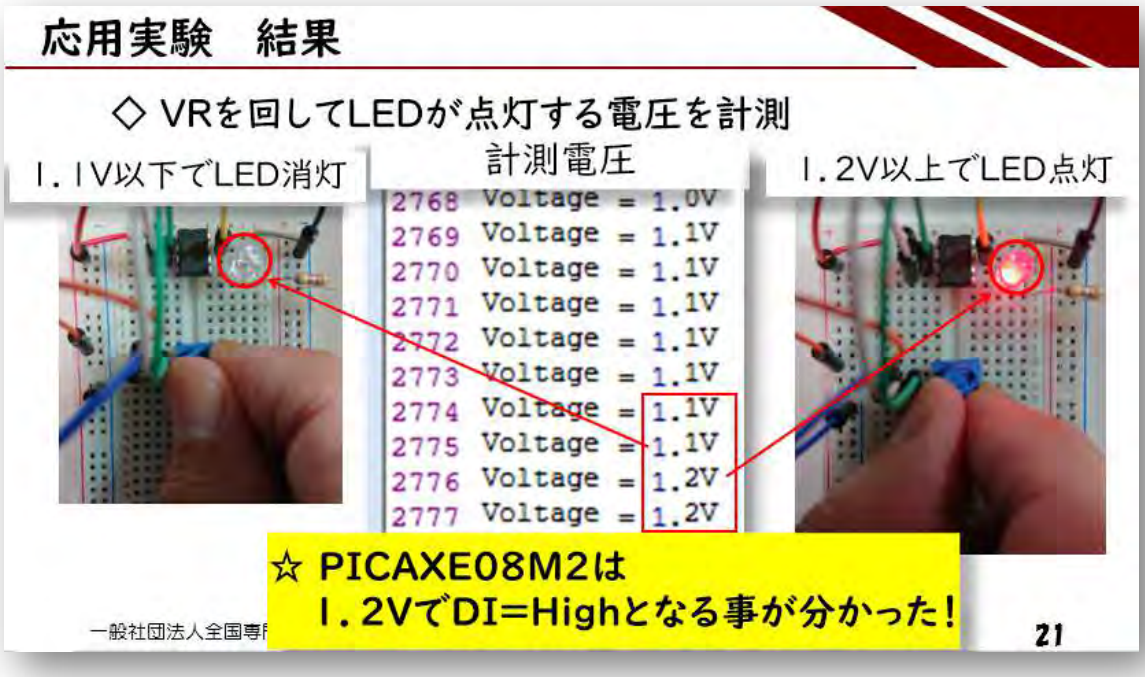


図　108

動作確認をします。まず VR を左に回して電圧を 0V にします。その時 LED は消灯しています。LED も視野に入れて、電圧が上がる事を確認しながら VR を少しずつ右に回していきます。LED が点灯する位置で VR を停め電圧を読みます。次に右に一杯回し切り、電圧が下がるのを確認しながら VR を左に少しずつ回します。LED が消灯する位置の電圧を読みます。私が実験した結果では 1.1～1.2V で LED 点灯・消灯の閾値がありました。プログラムをさらに変更して、もっと細かい値の電圧まで表示するとどんな結果になるでしょうか。

第6回　光SW

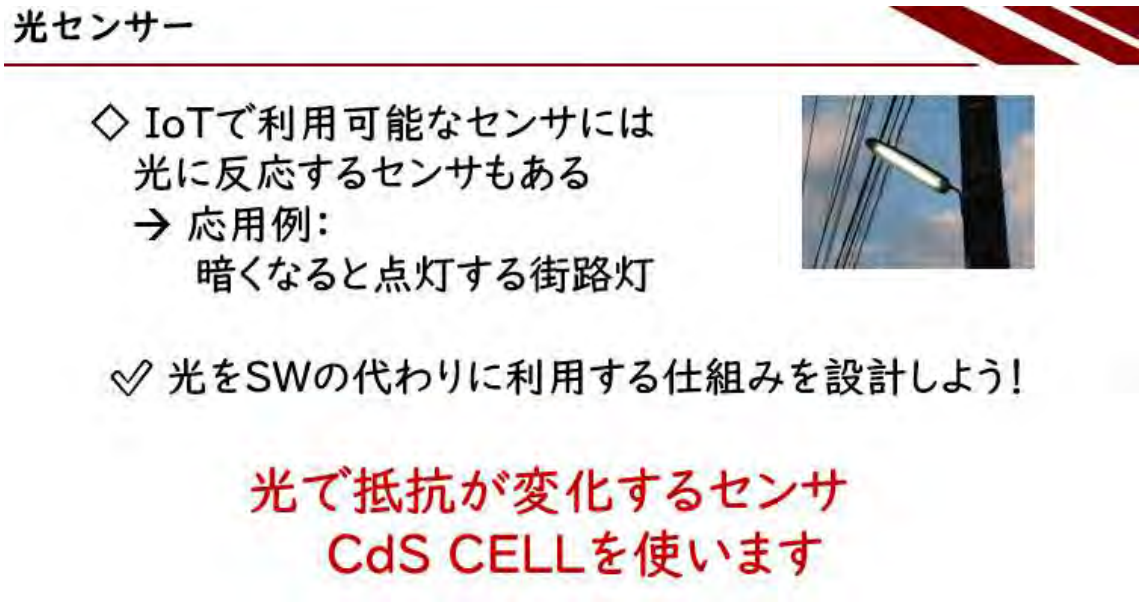


図 109

前回は VR という抵抗値を変化させることのできるパーツを使いましたが、今回は抵抗値が明るさによって変化するパーツ、光センサを使います。VR によって電圧を変化させると、マイコンの DI では High と読める電圧が分かっています。暗くなると、この電圧になる様に抵抗値が変化してくれると都合がいいですね。硫化カドミウム（CdS）という材料を用いた光センサ、CdS CELL がそれです。SW の ON/OFF で LED を点灯・消灯させるのと同じように光量を SW に使うので、光 SW です。

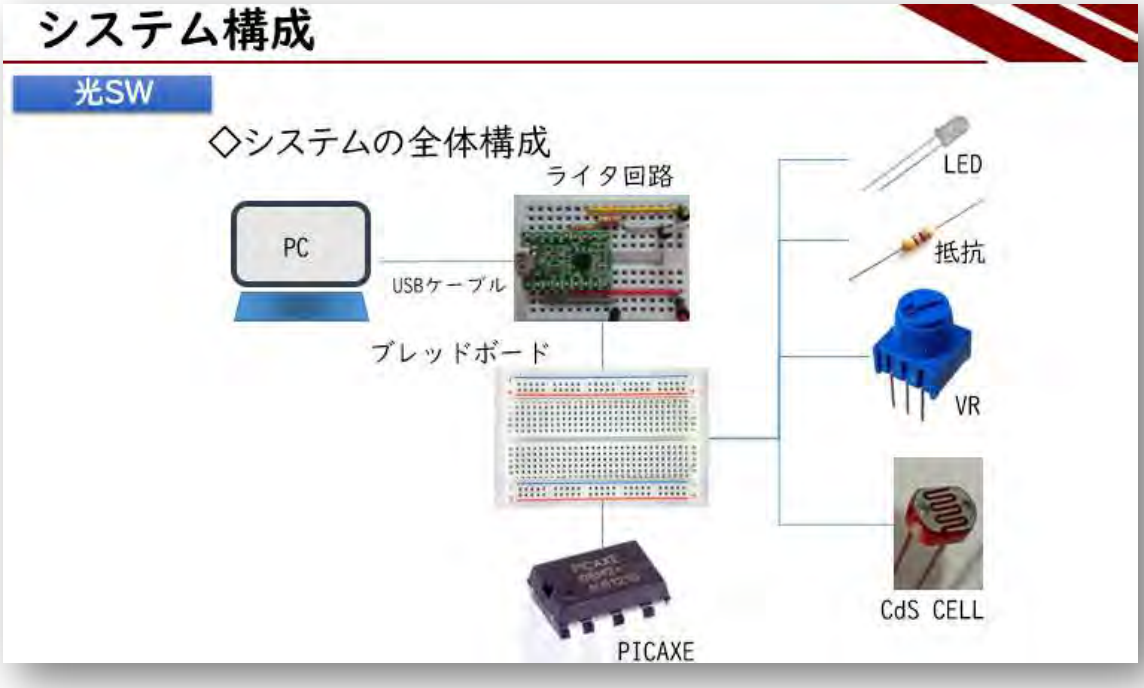


図　110

LED と抵抗器は LED 点灯回路に使います。CdS CELL は光センサです。VR の用途は、CdS CELL で LED を点灯させ暗さの調整用です。前回の VR による電圧測定を理解していれば、このシステムの原理は推測できるでしょう。

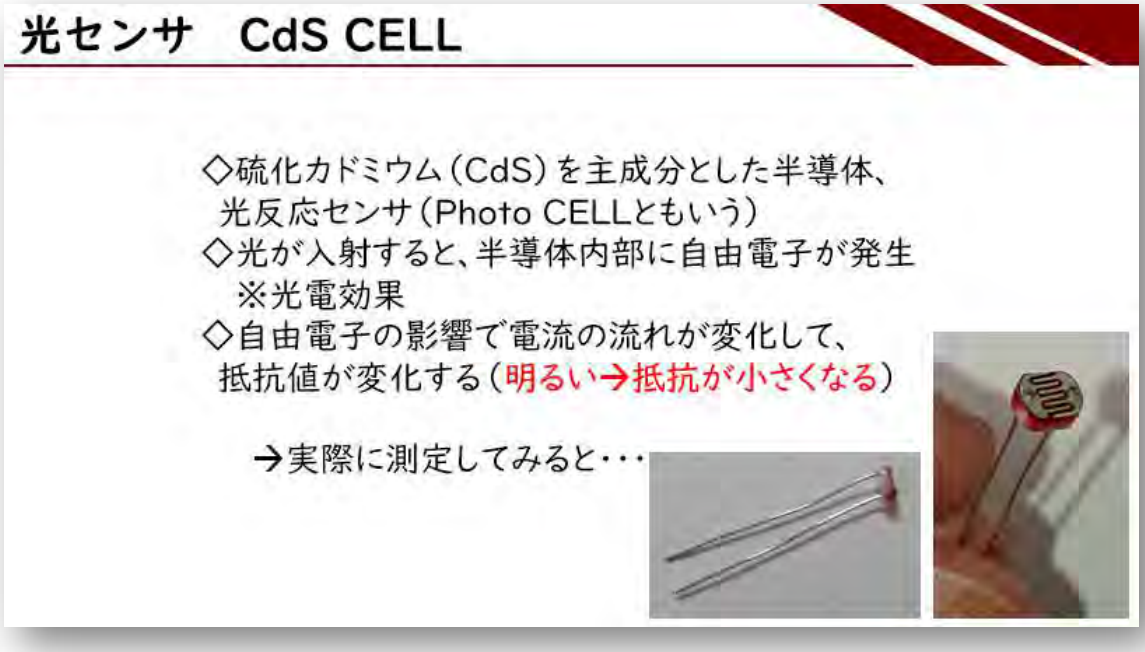


図　111

CdS CELL に光が入射すると内部に自由電子が発生します。この自由電子によって電流が変化します。電流が変化するということは、抵抗値が変化しているということです。抵抗値の変化はどの程度でしょうか。実際に測定してみましょう。

図 112

実験では、通常使用している机の上で部屋の LED 照明を点灯した場合の CdS CELL の抵抗値は約 5kΩ。CdS CELL を手で覆って暗くした状態では約 35kΩでした。CdS CELL は暗くなると抵抗値が大きくなるセンサです。

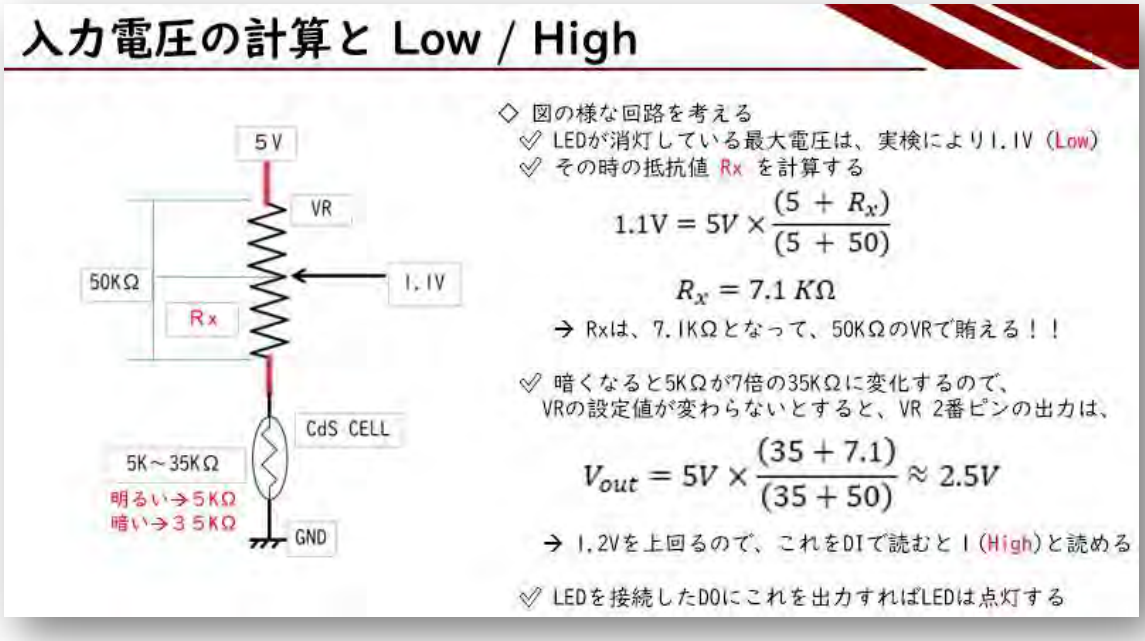


図 113

図に CdS CELL と VR の接続を示します。前回の実験では LED が消灯している最大電圧は 1.1V でした。CdS CELL と VR を直列接続して全体の抵抗値とします。そして VR の 2 番ピンから出てくる電圧が 1.1V のとき LED は消灯、1.2V になると点灯します。1.1V としてその時の抵抗値を計算すると 7.1kΩ です。これは 50kΩ の VR でまかなえる値です。式の分母にある 50 は VR 全体の抵抗値（50kΩ）です。7.1kΩ に VR を固定して、暗くすると CdS CELL の抵抗値は 35kΩ になるので、その値で VR の 2 番ピンの電圧を計算するとおよそ 2.5V になります。これは LED が点灯する 1.2V を超えます。これをマイコンの DI に入力すると 1（High）と読めます。つまり、この回路で VR の 2 番ピンを DI に接続して、その値を LED（DO）に反映させると明るさによる LED 制御が実現します。そして、LED が点灯・消灯する明るさを VR で調節できます。

図 114

使用する CPU のピンは前回と同じです。CdS CELL は VR の GND 側に直列接続して CPU には接続しません。

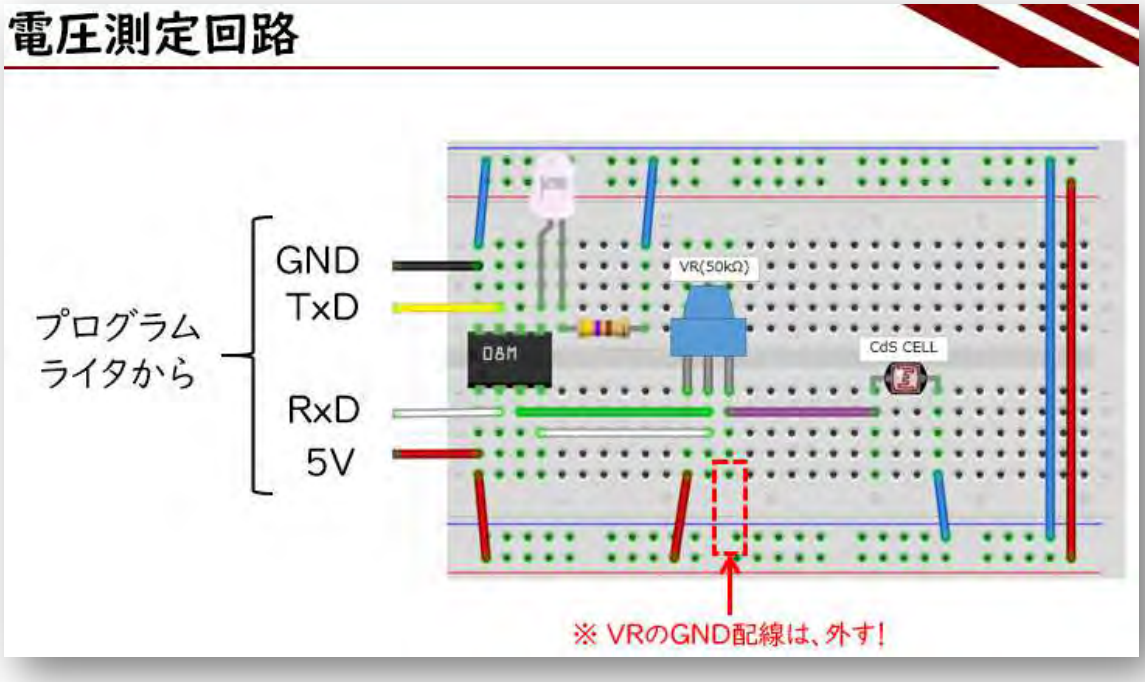


図　115

図の様に、CdS CELL を VR の GND 側に接続します。VR の GND 配線は外します。

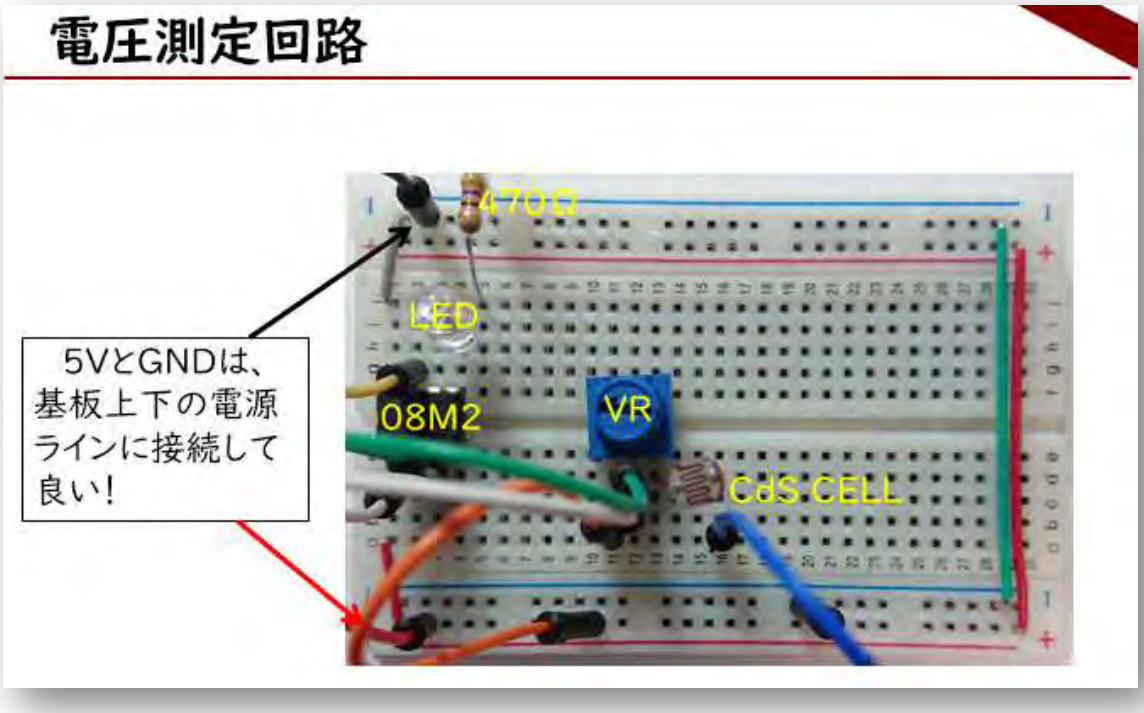


図　116

完成した回路を示します。

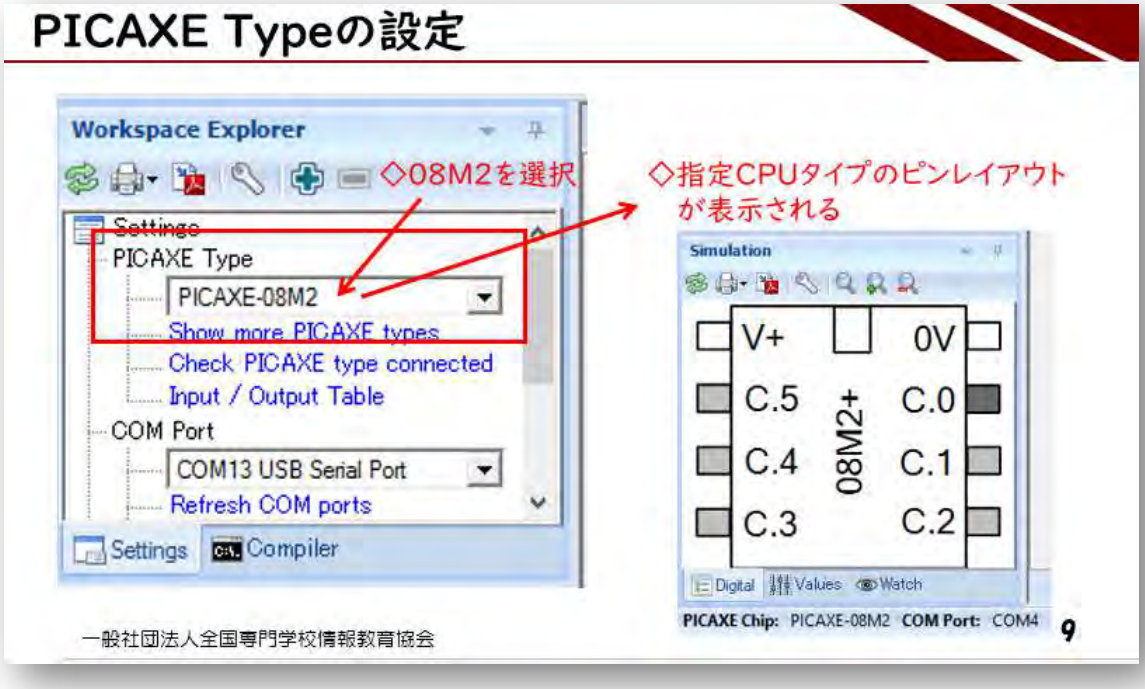


図　117

PICAXE Editor 左上のウインドウで、対象の PICXE08M2 チップを選択します。

図　118

プログラムは、前回使用したものがそのまま使えます。入力する場合は、Home の New ボタンで新しく作ります。

図　119

ソースコードの詳細は、前回を参照してください。

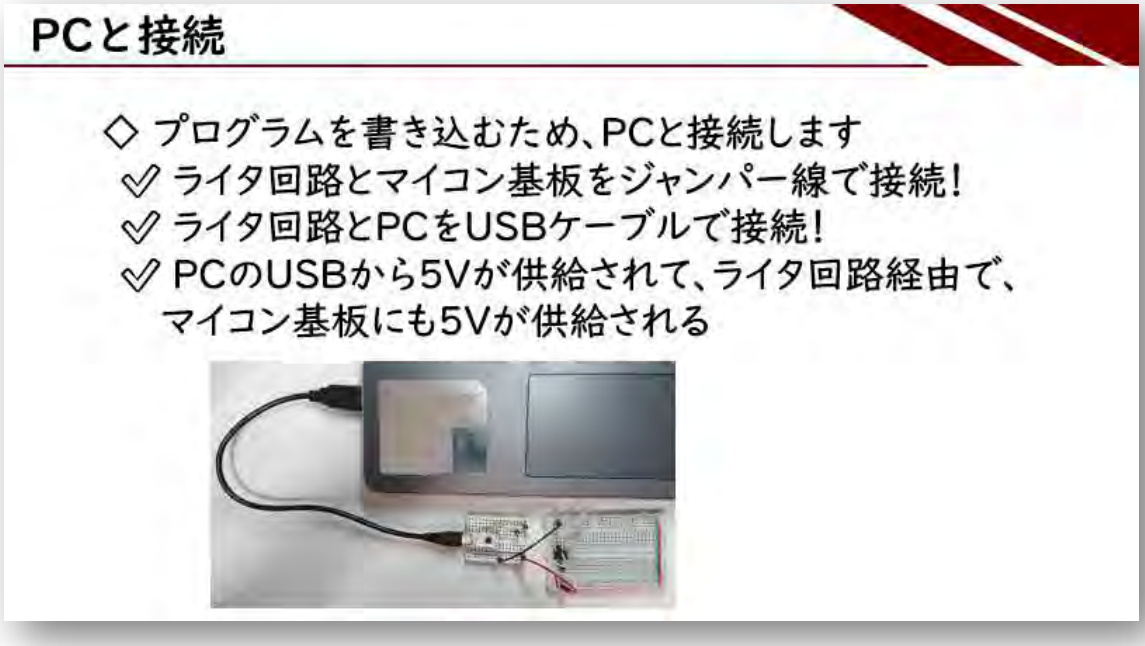


図 120

プログラムが完成したら、コンパイルと書込みを行います。PC、ライタ回路、マイコン回路を接続して下さい。

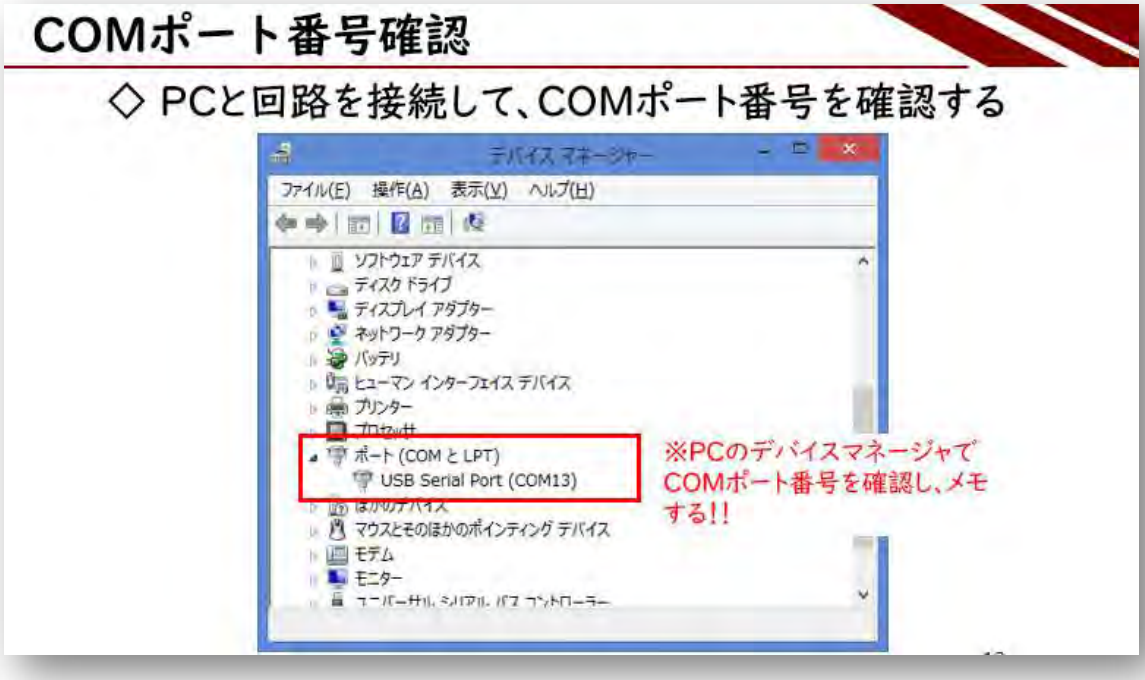


図 121

Windows のデバイスマネージャで、COM ポート番号を確認します。

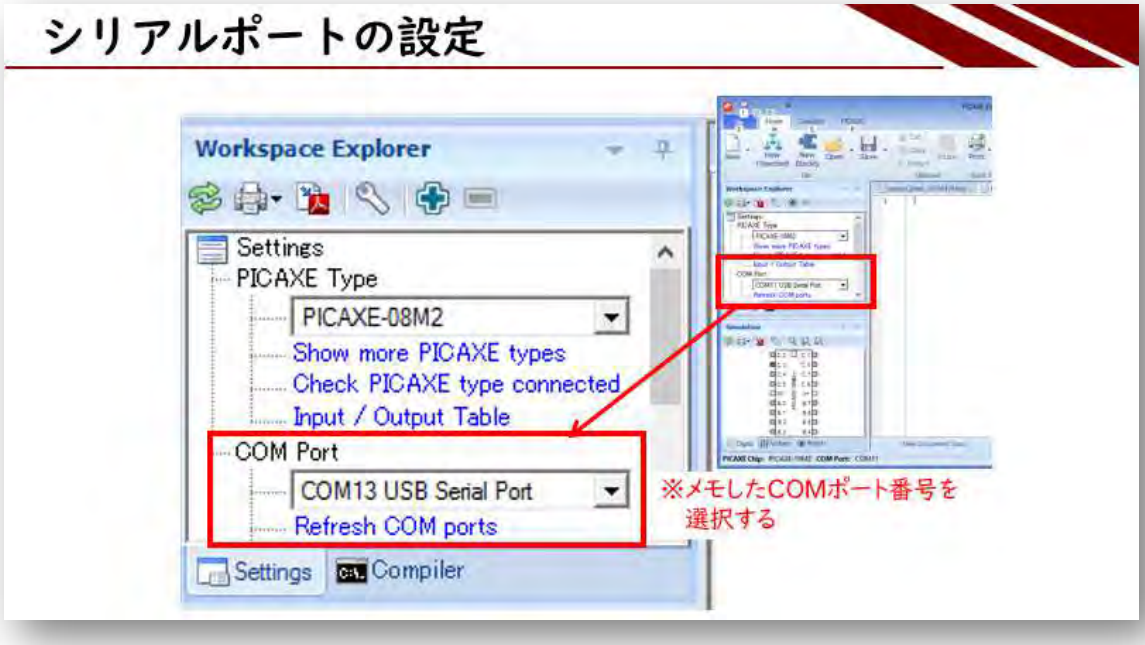


図　122

確認したOCMポート番号を選択します。

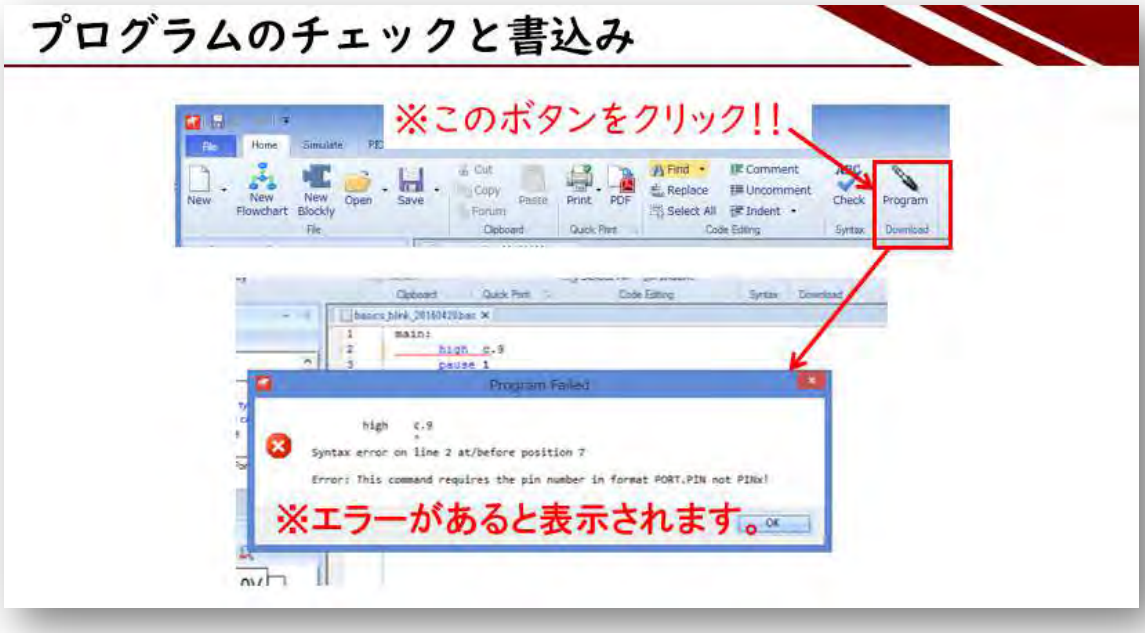


図 123

Home の Program ボタンを押下してコンパイル、書き込みを行います。

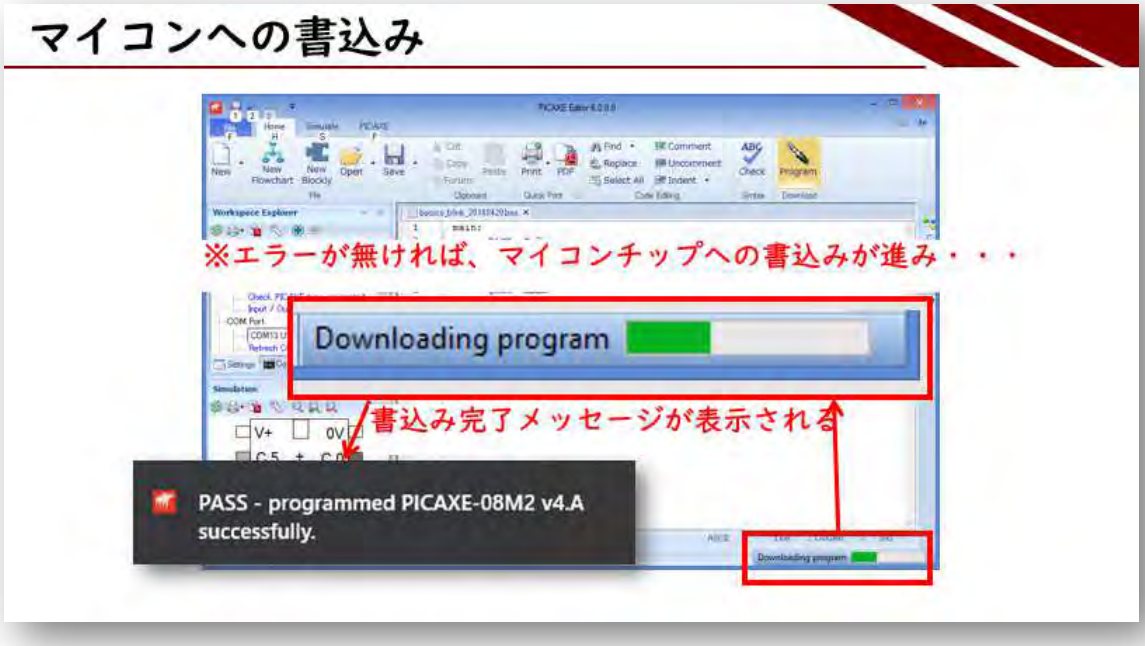


図　124

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

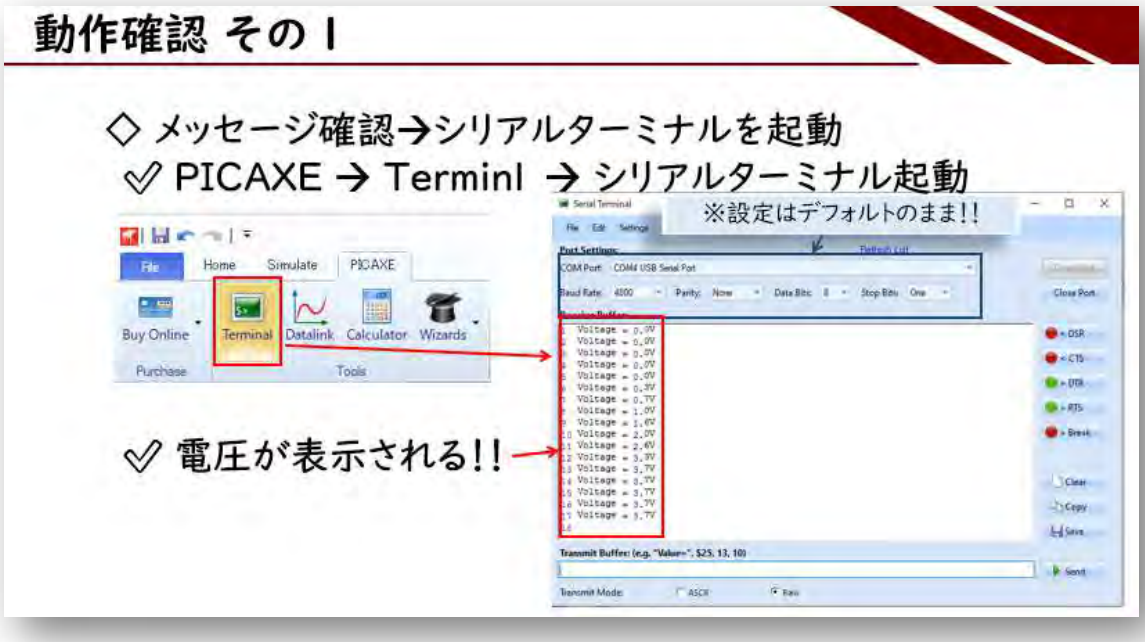


図　125

前回と同様の動作確認を行います。まずシリアルターミナルを起動します。電圧が表示されています。

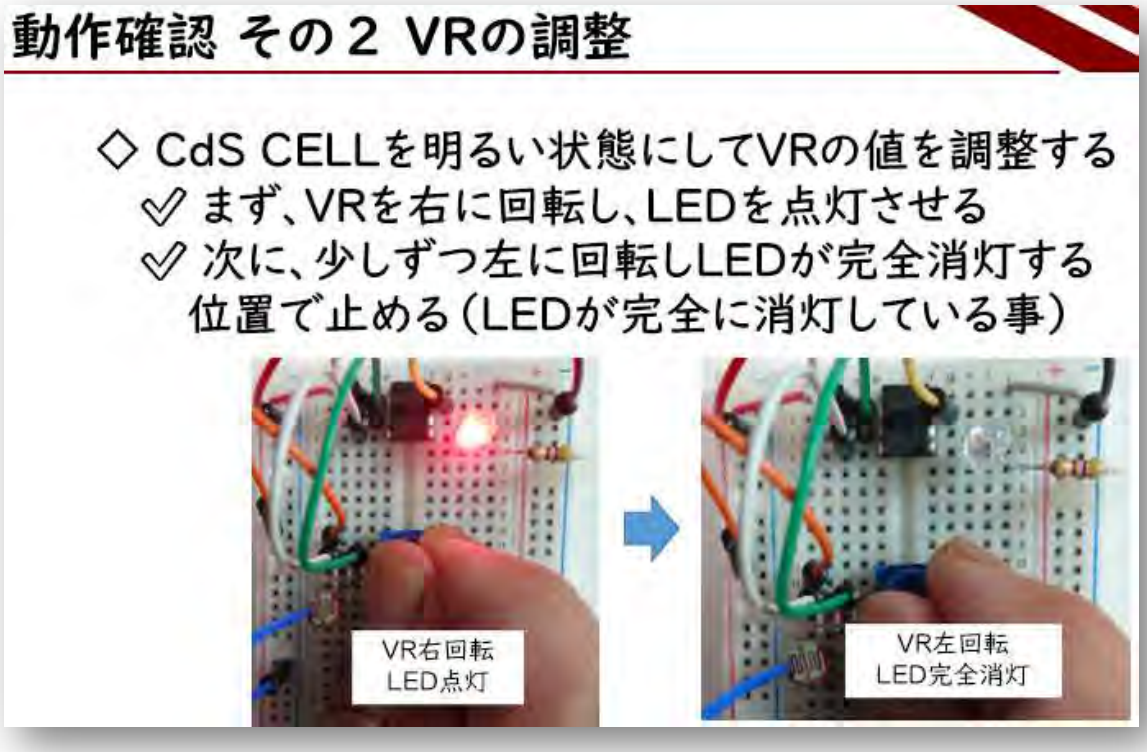


図　126

次に VR 調整を行います。回路全体を屋内の明るい場所に置き VR を右に回して LED を確実に点灯させます。次に、少しずつ左に回して LED が完全消灯する位置で留めます。

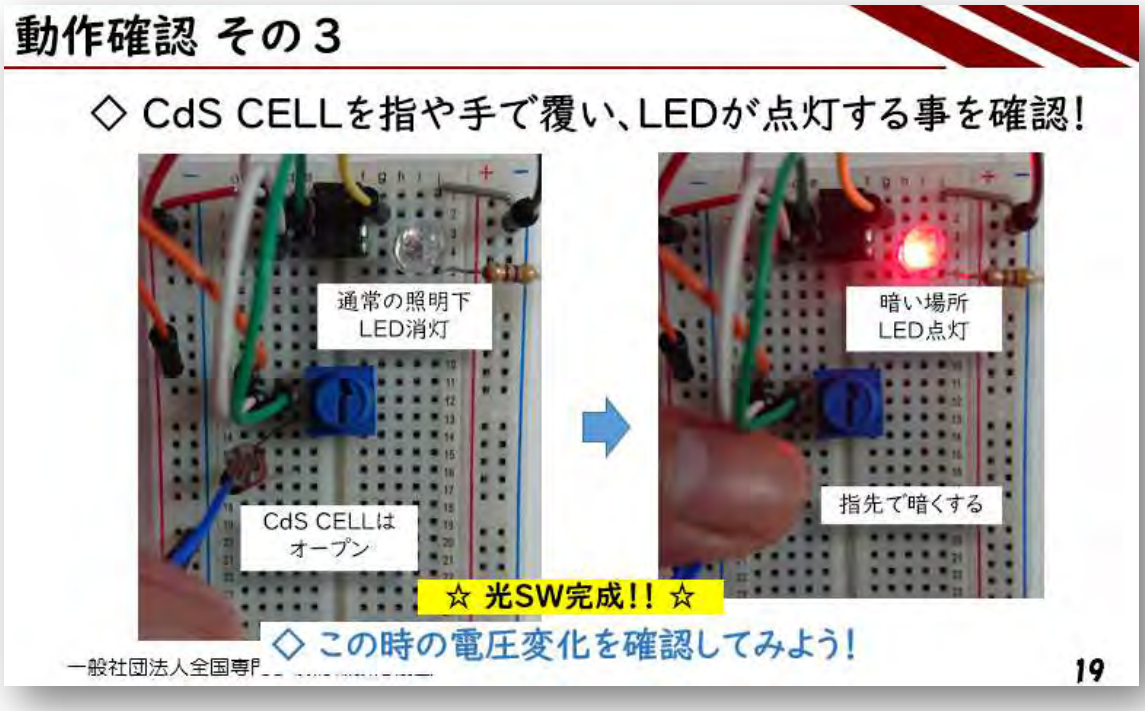


図　127

CdS CELL を手で覆い隠して暗くします。このとき LED が点灯すれば、街路灯などで利用されているような照明の自動点灯システムが完成です！！

第7回　温度センサ

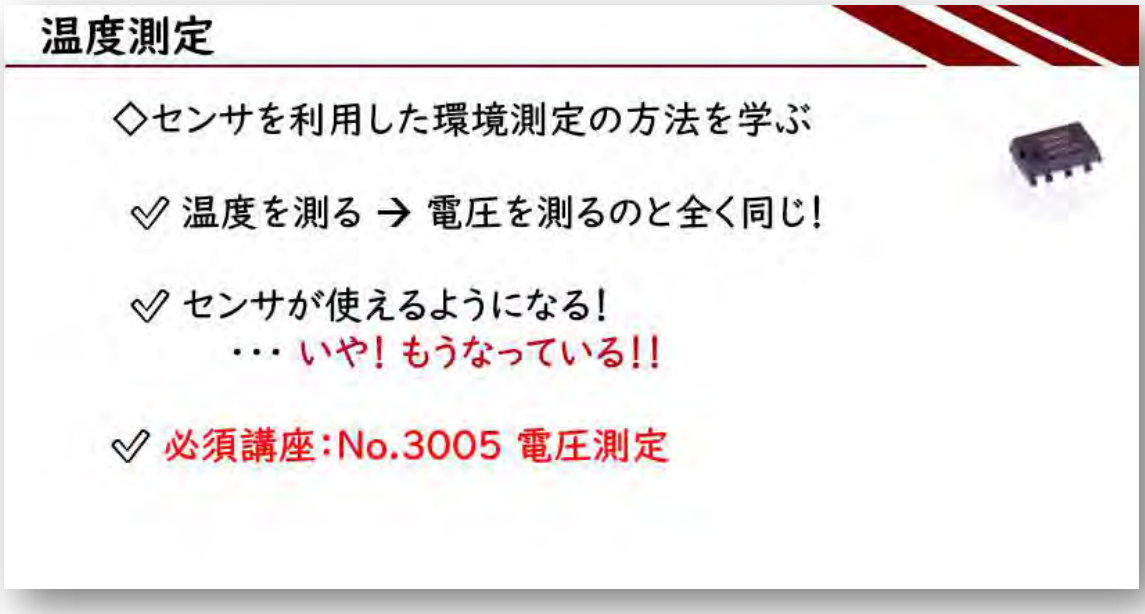


図 128

IoT に求められているものの一つとして、センサの利用があります。今回はその中でも基本的なアナログセンサを使った温度測定を行います。温度を測るのは VR で行ったように、電圧を測ることと同一です。電圧の測定ができれば温度が測れることを実証します。

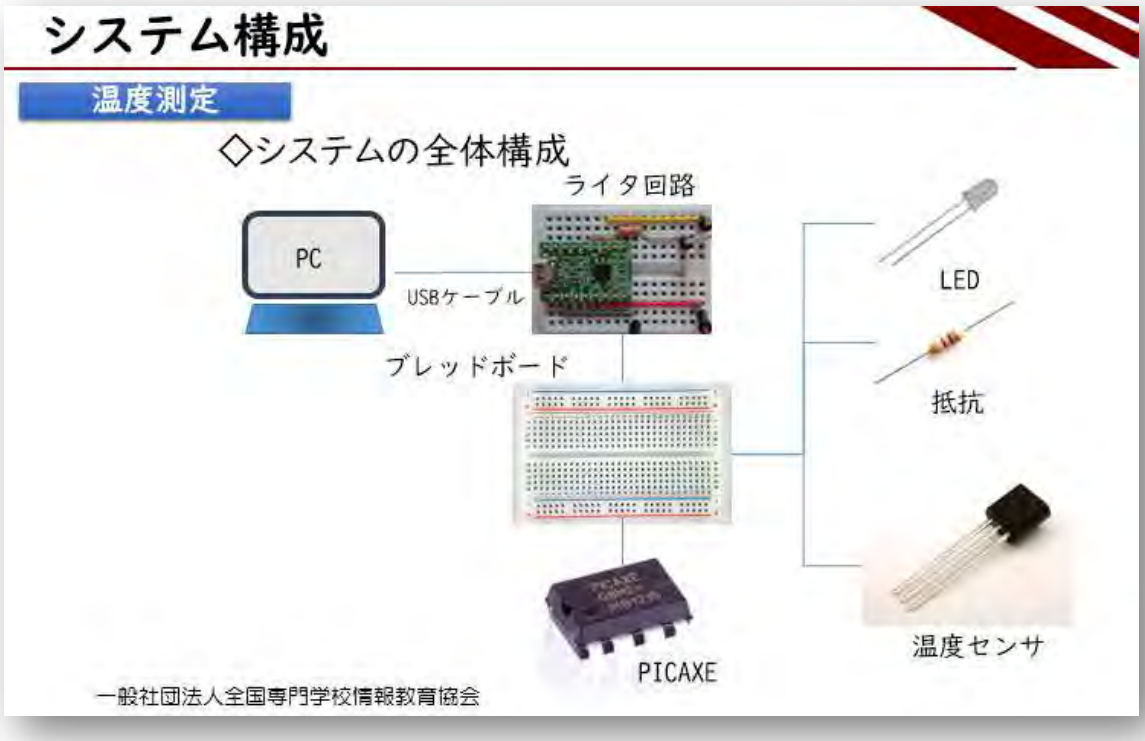


図 129

使用するパーツは、VR の代わりに（又は、VR と CdS CELL の代わりに）アナログ温度センサを使います。LED と抵抗器は未使用ですが、残しておきます。

図 130

図は温度センサのデータシートの一部です。このセンサは LM61CIZ という型番です。リニアな特性というのは、出力が直線的に変化することを意味しています。温度係数（+10mV／℃）と 0℃＝600mV がその直線の方程式の係数などに該当します。室温や外気温を測る目的であれば十分な測定温度範囲です。センサ出力電圧から温度を知るには、図に示す式で計算します。

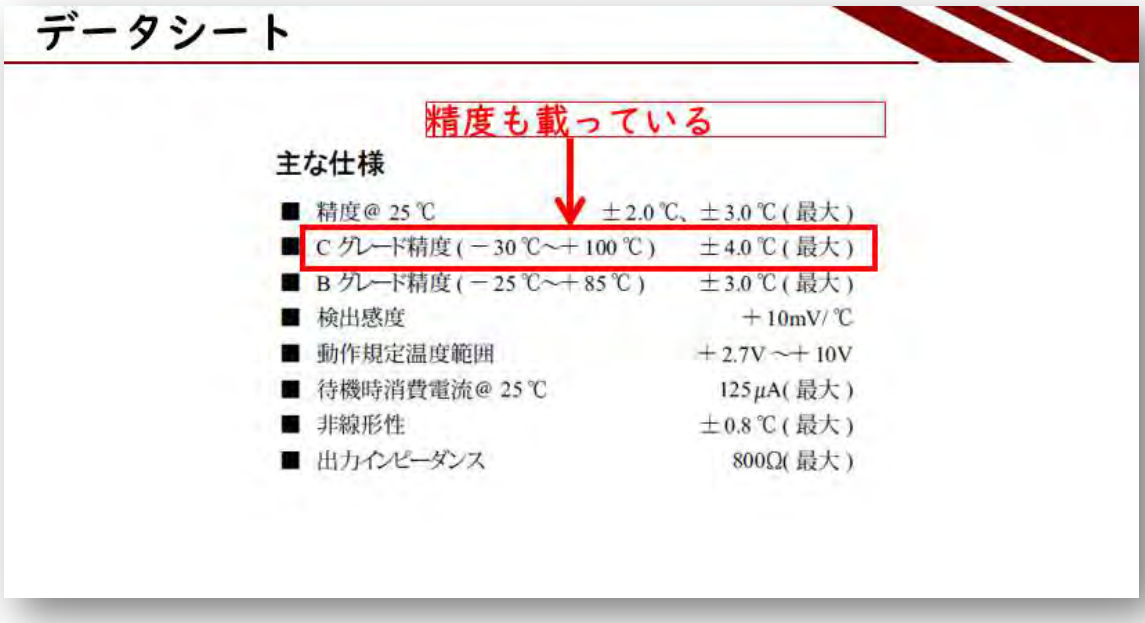


図　131

型番にある CIZ の C は、精度を意味するコードです。センサと言うともっと高精度なものをイメージするかと思いますが、半導体などの温度に対する特性のばらつきは、意外に大きいものです。

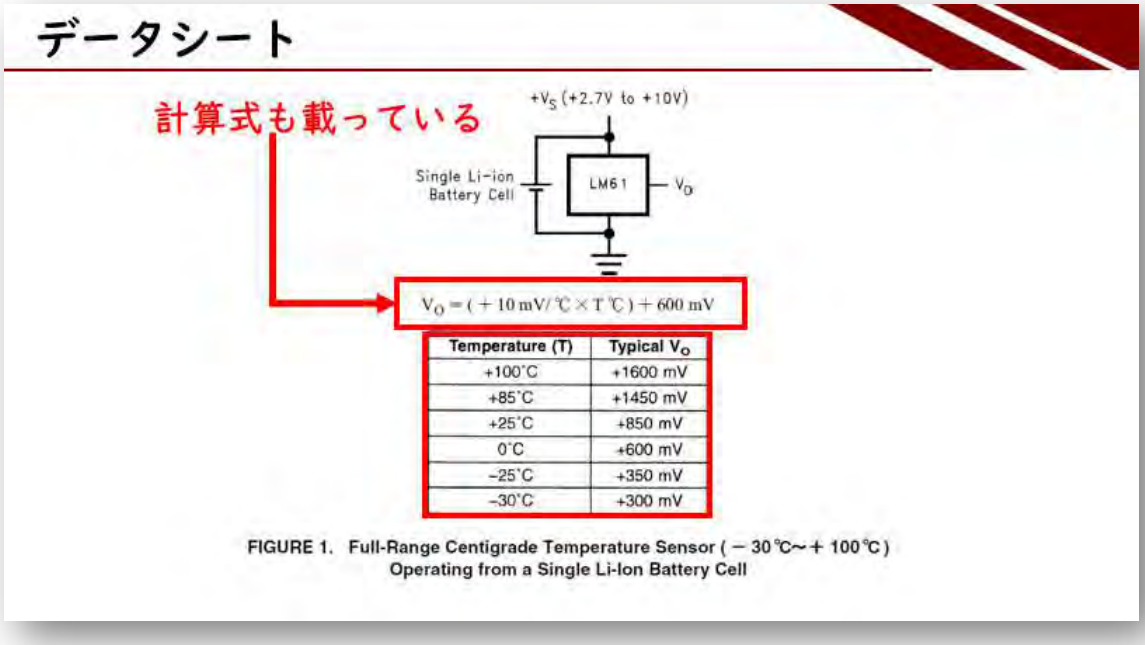


図 132

前の図で示した計算式はデータシートに記載されているものを温度について書き直したものです。計算式で求めた温度に対する出の表も掲載されています。これをグラフにしたものが次の図です。

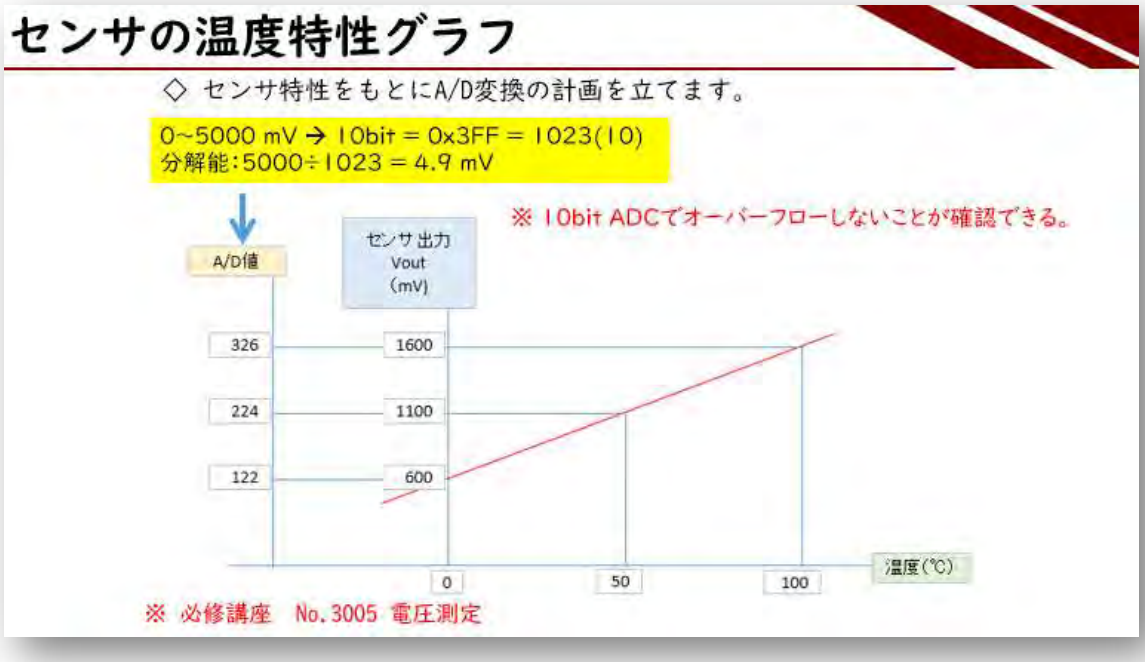


図 133

温度に対する出力電圧は当然、直線のグラフになります。このセンサ出力電圧を 10bit で A/D 変換した場合の値も縦軸に記載しました。PICAXE は 5V まで A/D 変換できるので、対象温度が 100℃の時の出力 1600mV は、余裕で測定できます。

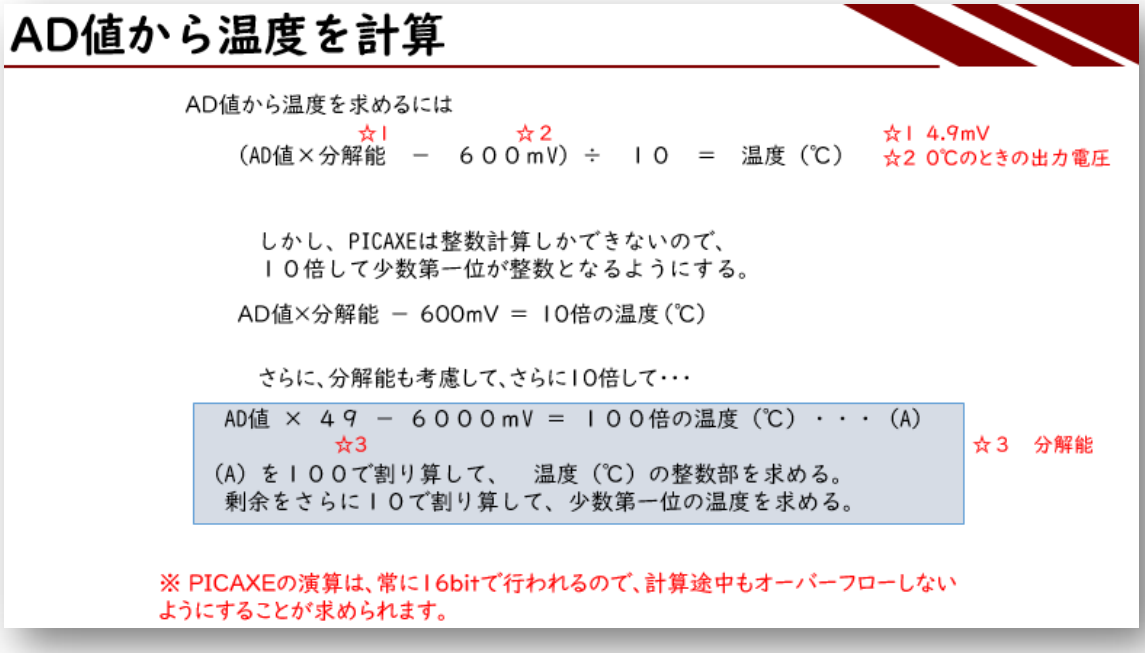


図 134

10bit での分解能は 4.9mV/bit ですから、A/D 変換した値から温度を求める計算は図に示すように行います。PICAXE の演算は、整数演算で常に 16bit で行われます。また記述した式の左から計算されます。普通の計算式のように括弧内の計算を優先する機能はありません。ですから、計算の途中で 16bit から溢れる（オーバーフロー）することが無いように事前に確認をしておきます。

図 135

データシートには、センサの外形と共にピン配置も記されています。これを基にして配線を考えます。アナログ温度センサは、電源と GND を反対に接続すると、高熱を発して壊れてしまいますので注意してください。図の【BOTTOM VIEW】は【下側から見た図】という意味です。

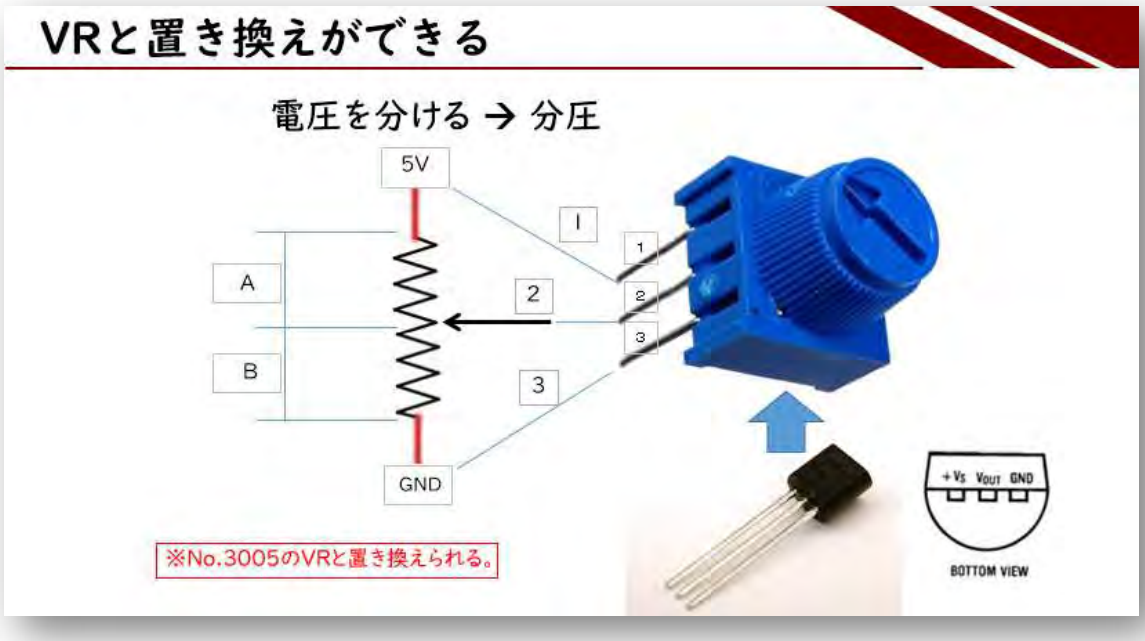


図 136

BOTTOM VIEW をみて分かるように、このセンサの左側のピンは電源（＋Vs）中央が出力、右端が GND ですから、既に使用した VR と全く同じピン配置であることが分かります。つまり、電圧測定回路の VR と置き換えれば、この温度センサが使えます。

図 137

温度センサは ADC のある C.4（3 番）ピンに接続します。LED は未使用ですが、これまでと同じように C.2（5 番）ピンに接続しておきます。

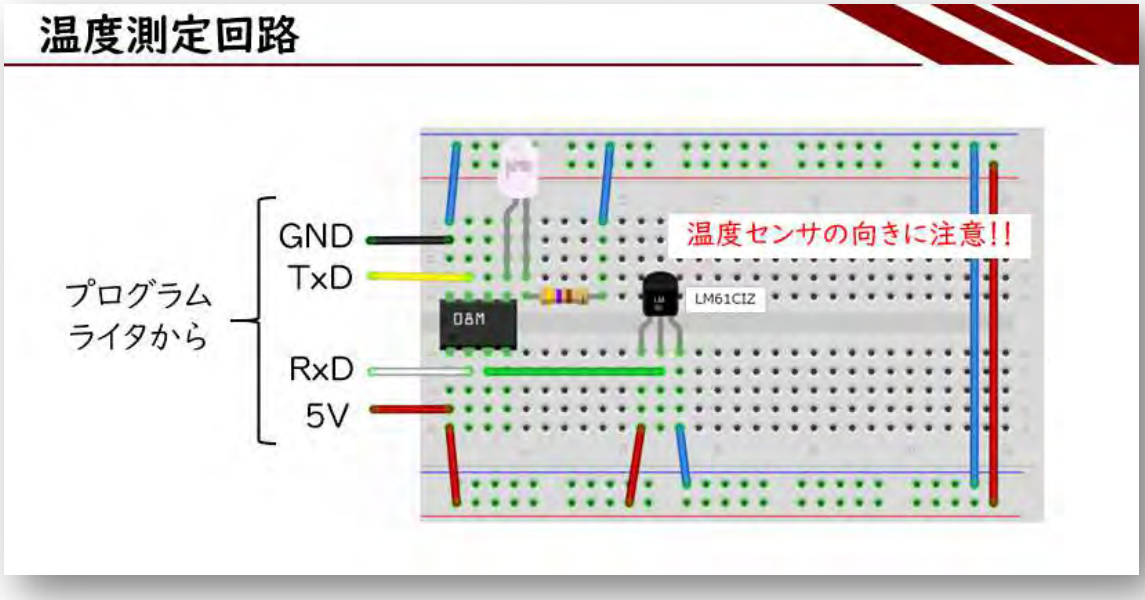


図 138

温度センサは VR よりずっと小さいトランジスタ形状のパーツです。蒲鉾型をしており、平らになっている部分に型番が小さく印刷されています。この面を図では下側に向けて配置してください。向きを間違えると、発熱して壊れてしまいます。

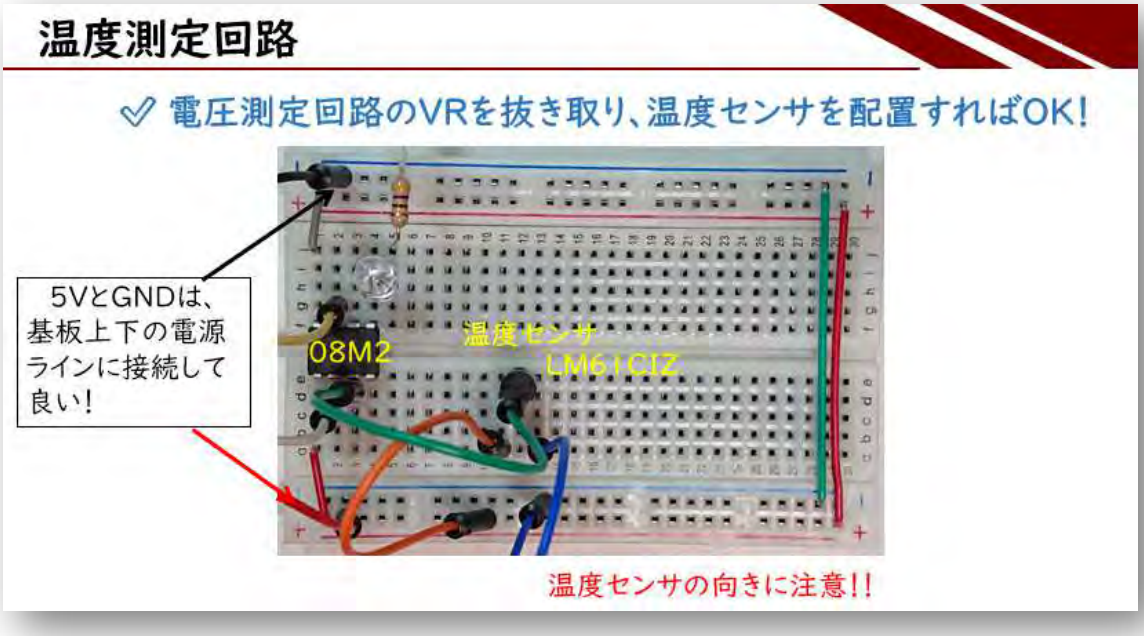


図 139

実際の温度測定回路を図に示します。

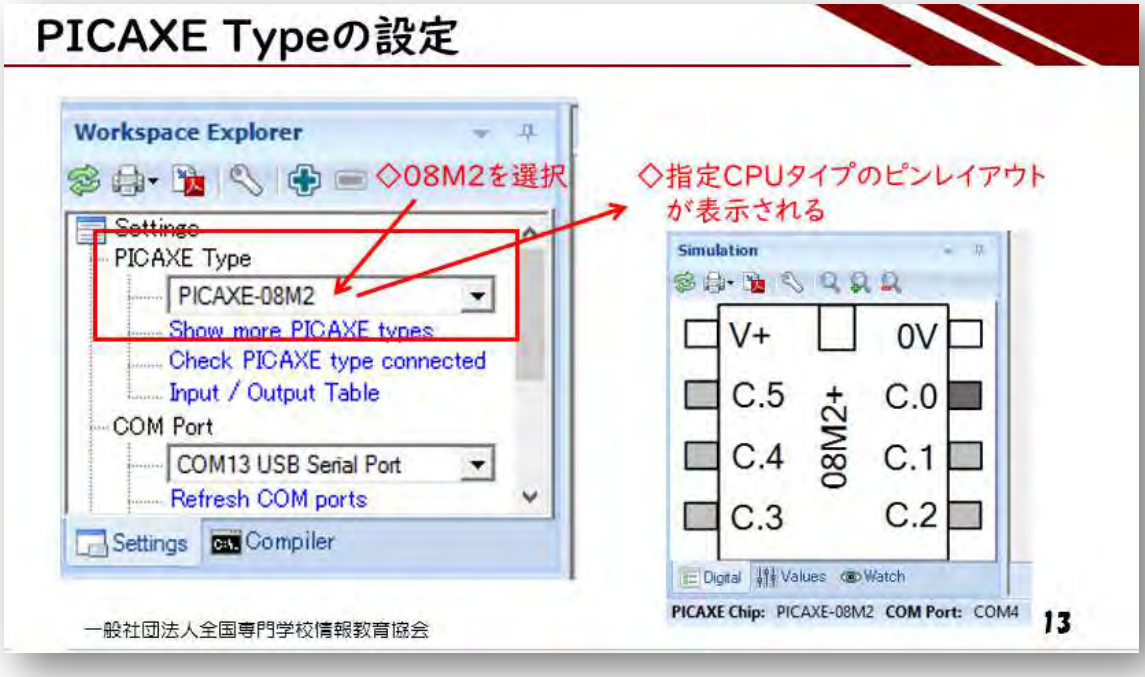


図 140

PICAXE Editor 左上のウインドウで、対象の PICXE08M2 チップを選択します。

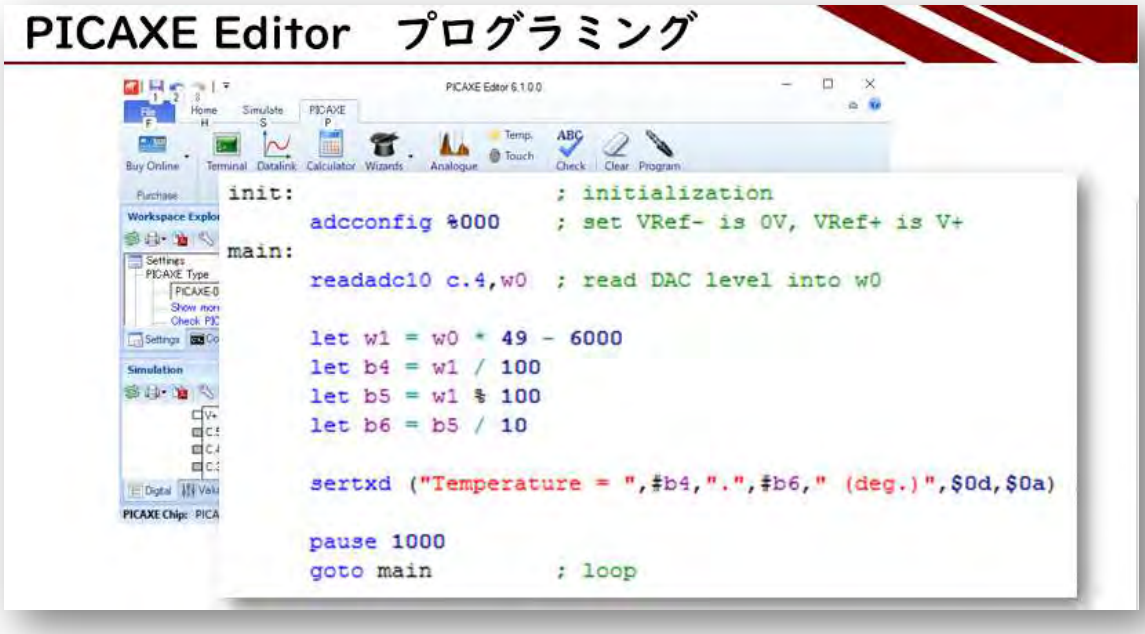


図　141

Home---->New で新しいプログラムを作ります。

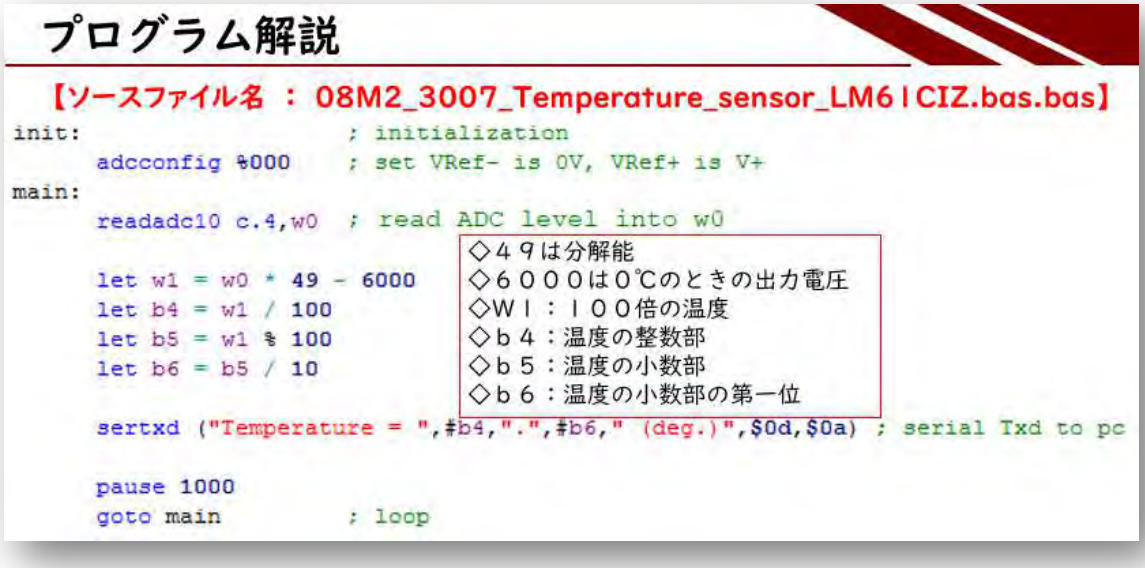


図 142

VR の場合とほとんど同じプログラムです。A/D 変換は 10bit を使用するので、readadc10 という命令を使い、A/D 変換値は 2 バイトのエリア（w0）に格納しています。w0 というエリアは、PICXE Editor の右端にある Code Explorer にマウスカーソルを重ねれば確認できます。PICAXE では自由に使えるワークエリアが限られていて、1 バイトの b0,b1 をつないで w0 として使います。事前に検討しておいた計算の手順で A/D 変換値を温度に換算します。

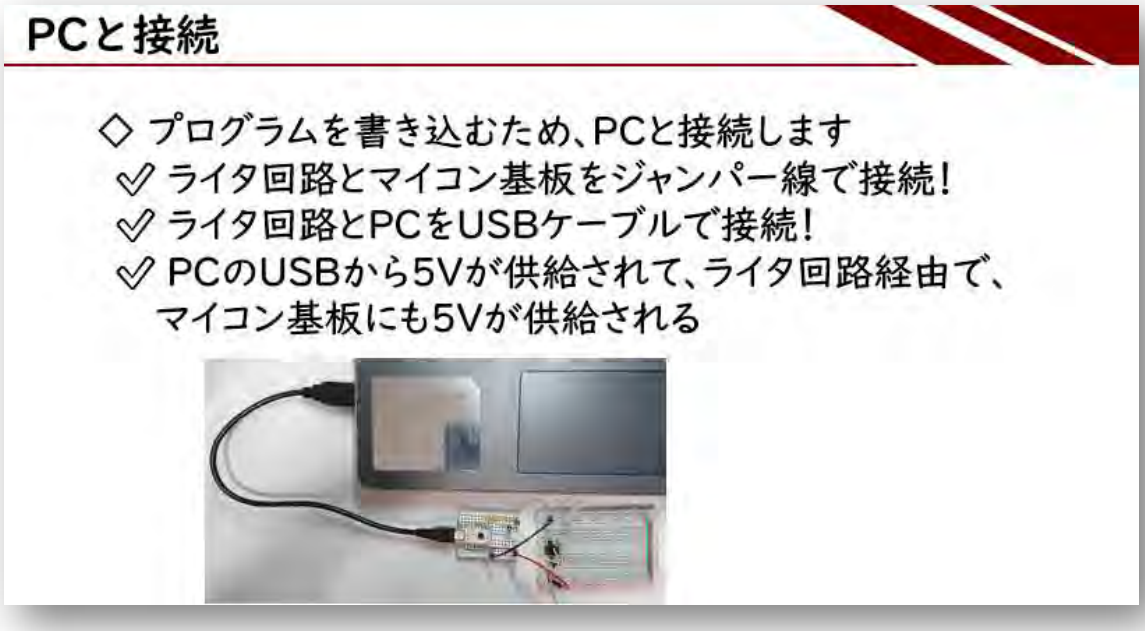


図 143

プログラムが完成したら、コンパイルと書込みを行います。PC、ライタ回路、マイコン回路を接続して下さい。

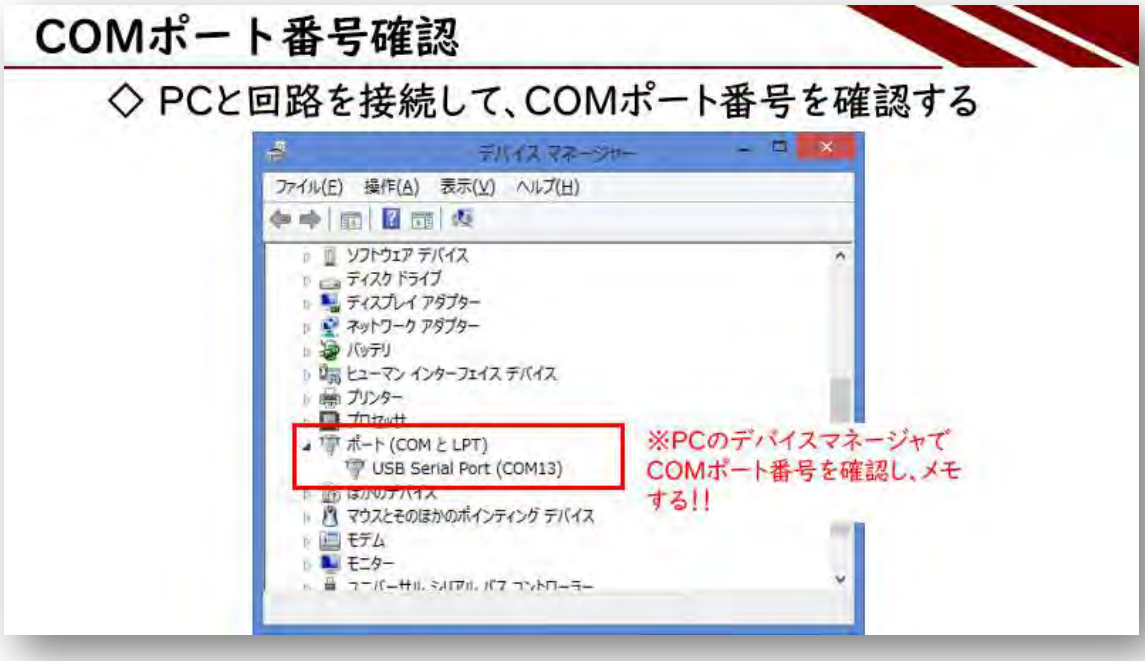


図 144

デバイスドライバで COM ポート番号を確認します。

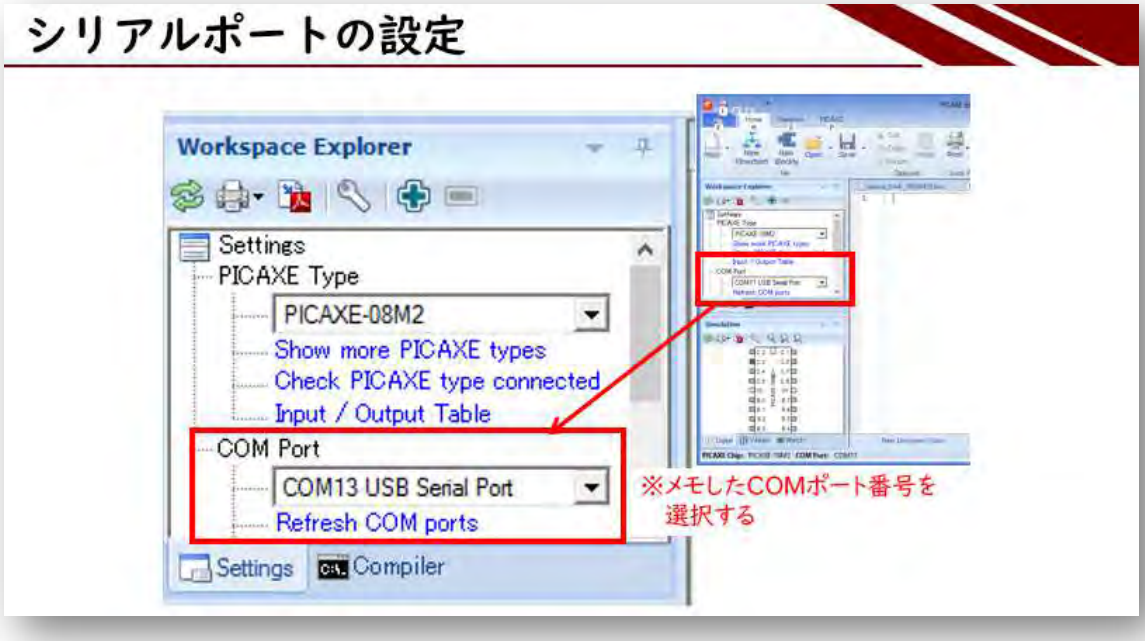


図　145

確認した COM ポート番号を設定します。

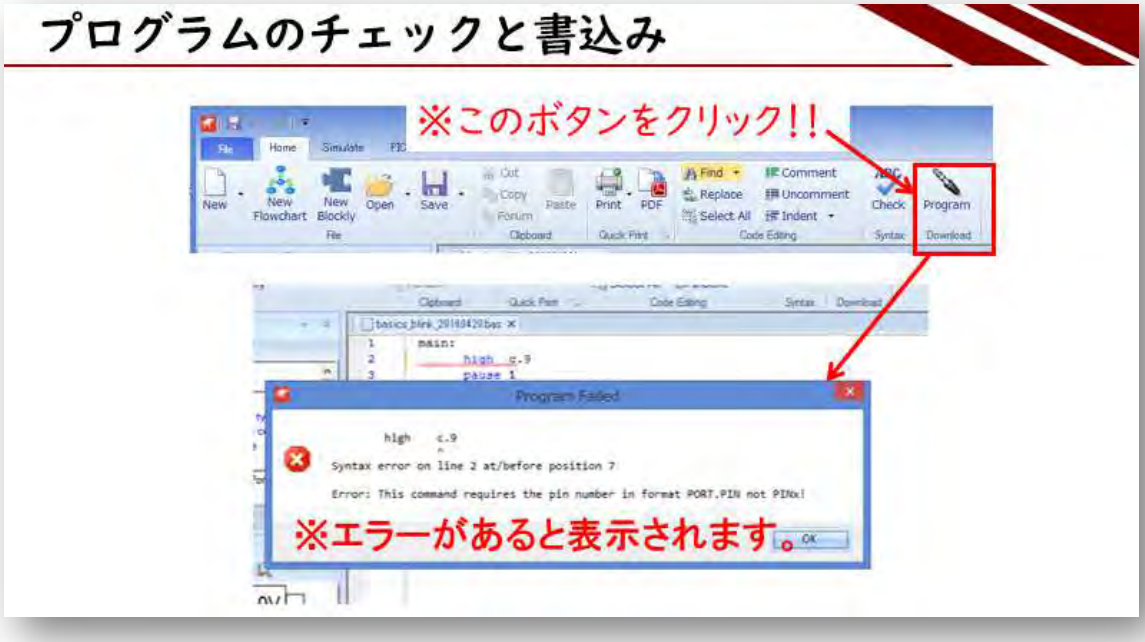


図 146

Home の Program ボタンを押下して、コンパイルと書込みを行います。

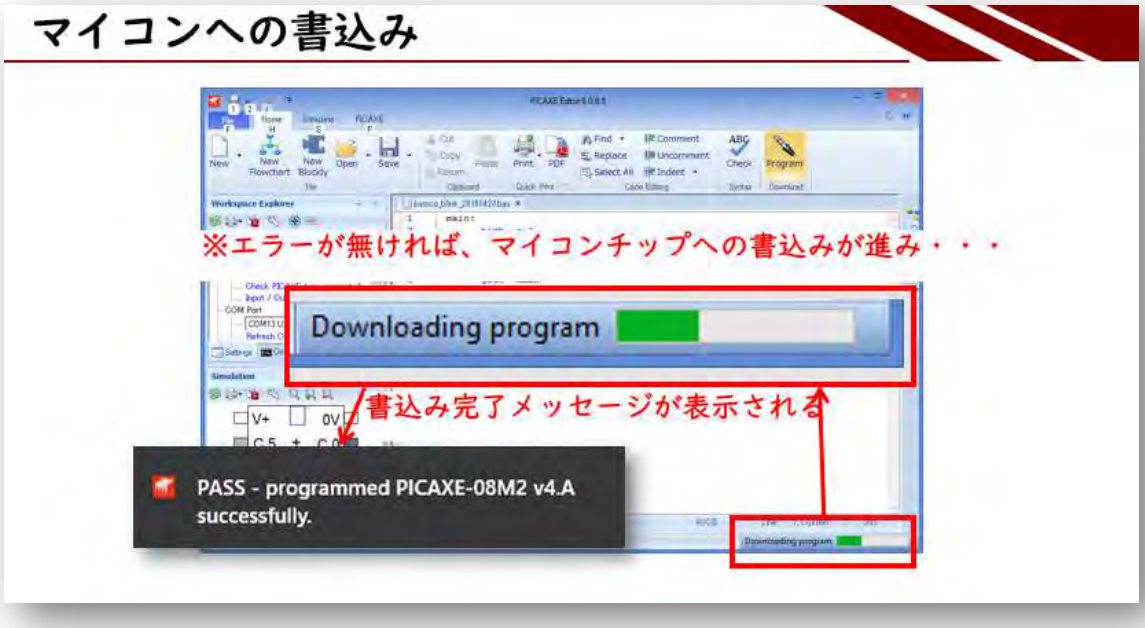


図　147

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

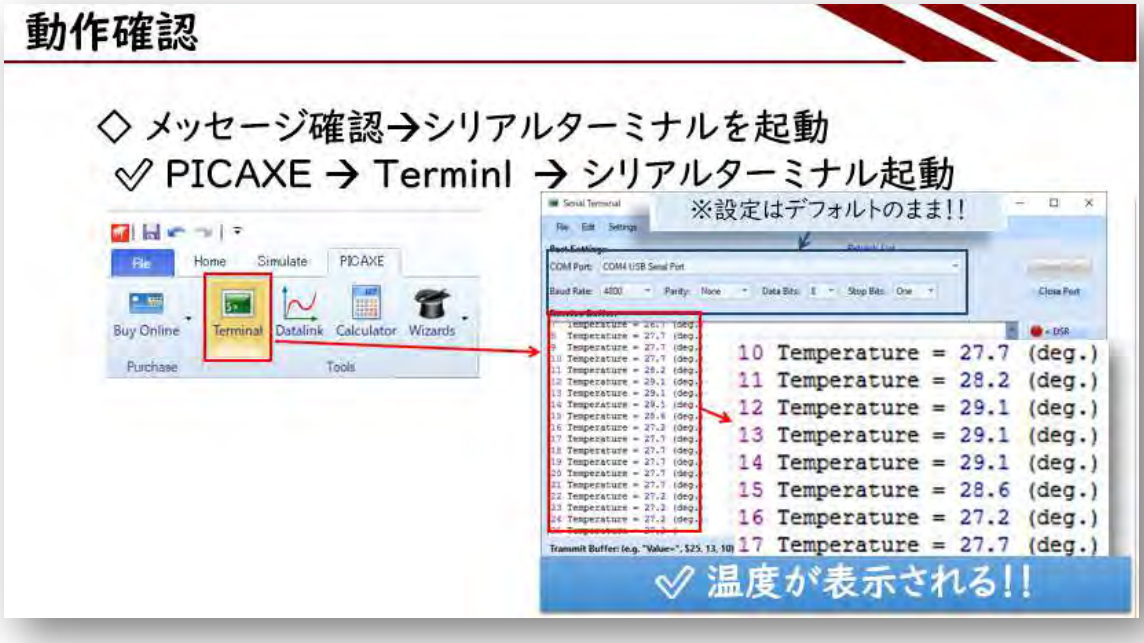


図　148

動作確認はシリアルターミナルを起動して行います。ウインドウに温度が一定時間ごとに表示されます。温度センサに指で触れると、温度が上昇するのが見えるでしょうか。温度センサのピンに導線を半田付けしてセンサ自体をアルミチューブなどに封入し、水中や土中に置くこともできます。色々な箇所の温度を測定するにはどのようにセンサを固定するかも検討課題です。IoT で求められるのは、測定ができるだけではなく、実際にフィールドで実用になるまでをサポートできる技術です。

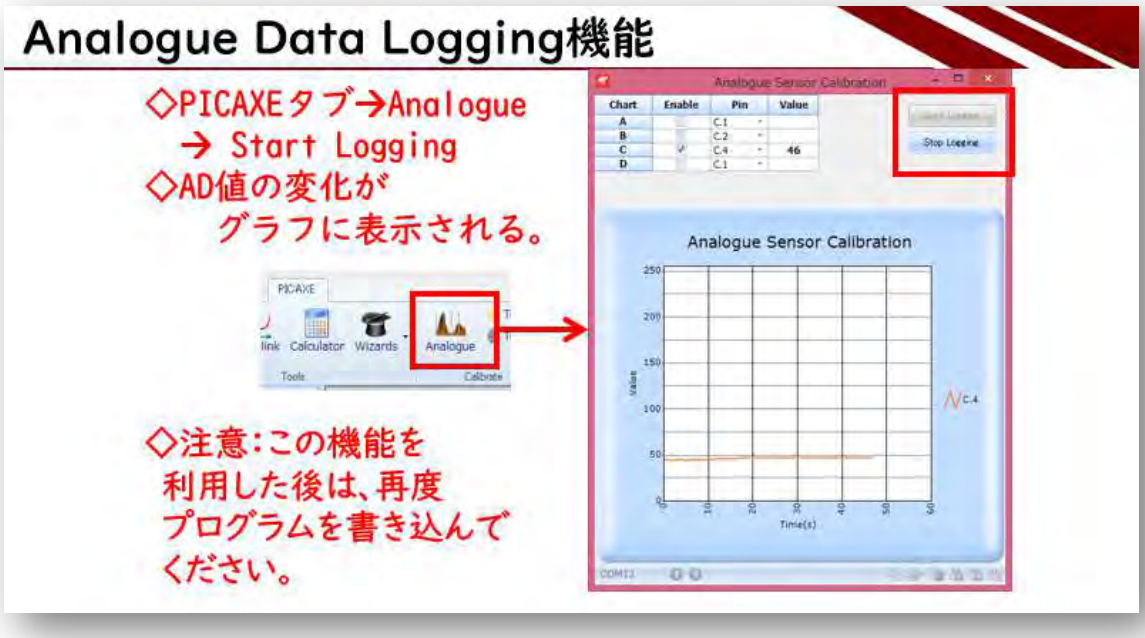
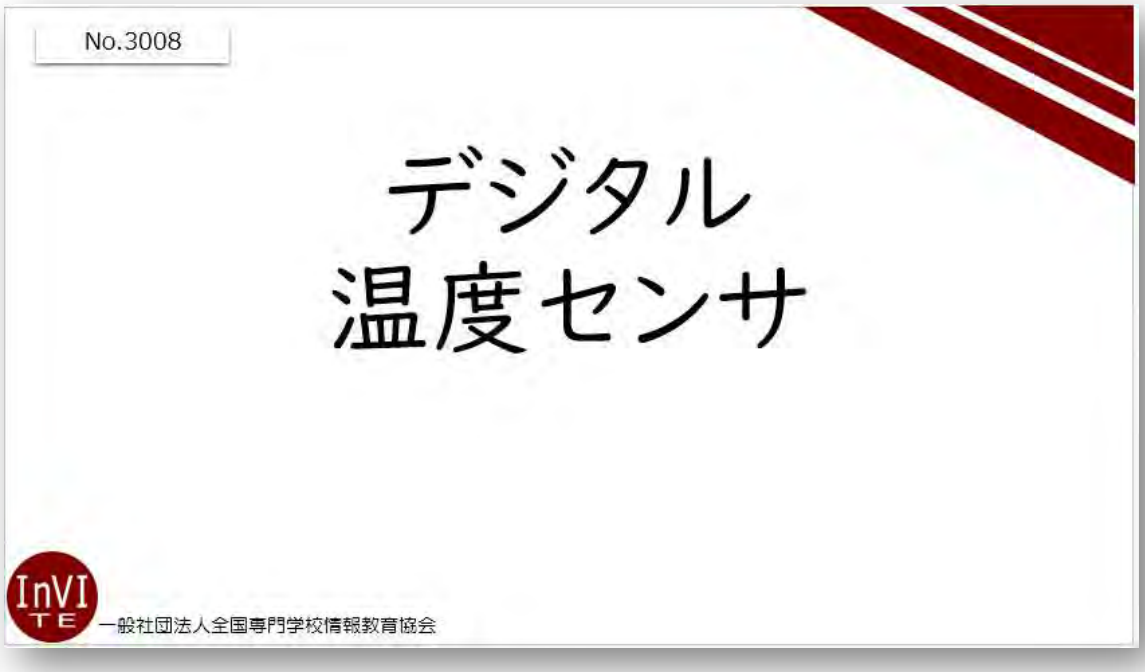


図 149

センサを用いる場合、PICAXE タブの Analogue から Start Logging を押下すると図のようなウインドウが開き、温度変化をリアルタイムのグラフとして見る事ができます。これは、センサ校正の為の機能ですが、便利に使えます。

【注意】この機能は、専用のプログラムを PICAXE に書き込むので、利用した後は、再度プログラムの書き込みを行って下さい。

第8回　デジタル温度センサ



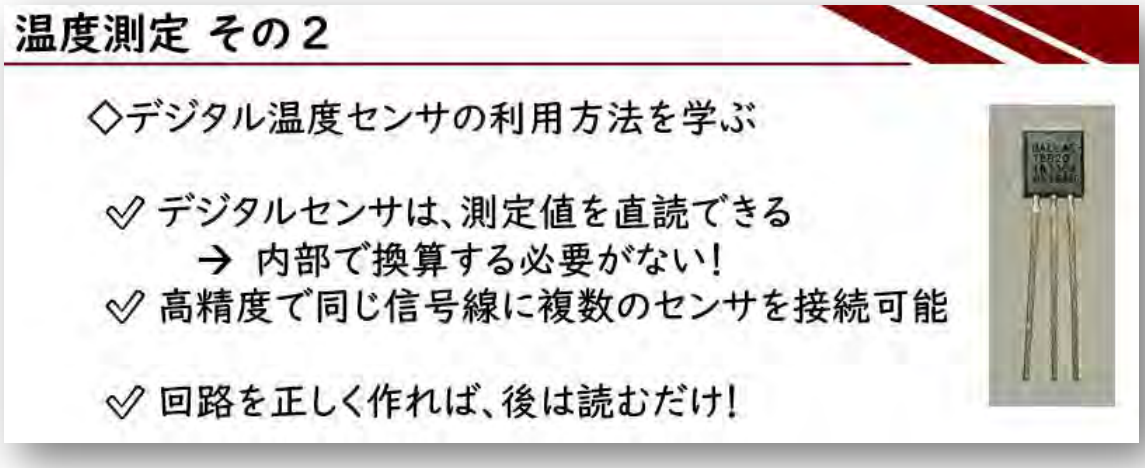


図 150

温度測定の二回目はデジタルセンサを用いた測定です。デジタルセンサは、対象の測定結果をデジタル値で読めるので、変換の計算は必要ありません。アナログセンサの様に ADC を占有せずに、複数のセンサを配置することができて、高精度なものが広く使われています。回路が正しければ、センサからデータを読むだけですが、センサごとに異なるデータの読み込み方があるので、予めデータシートを良く読むことが必要です

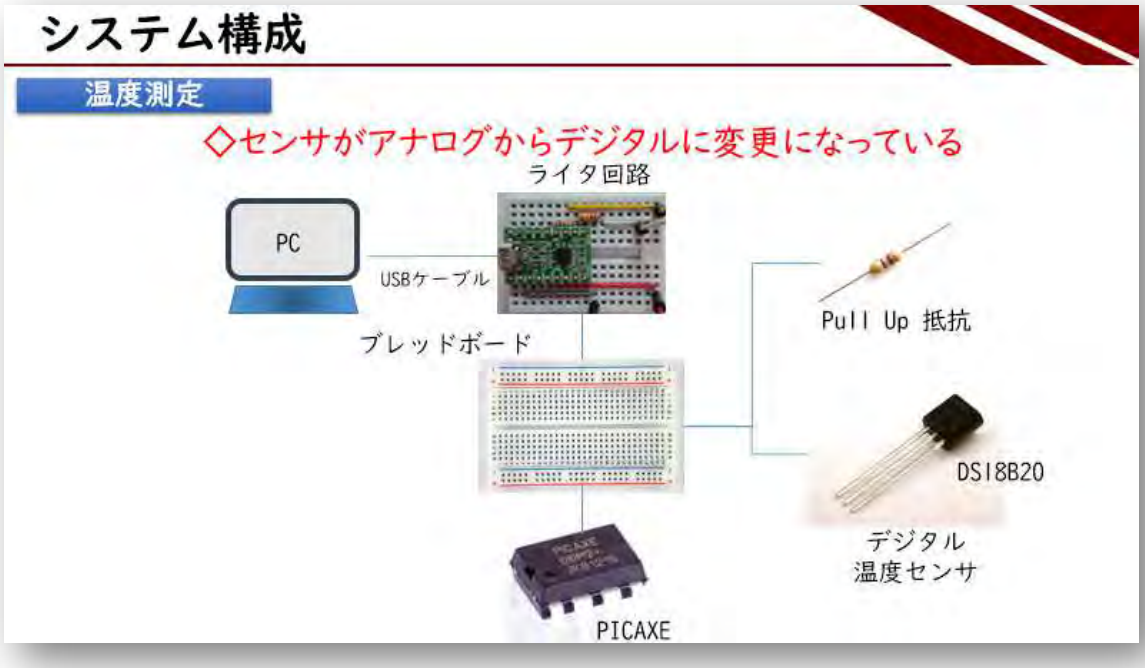


図　151

必要なパーツは外見では同じですが、温度センサはデジタルのものに置き換わります。LED は使用しません。

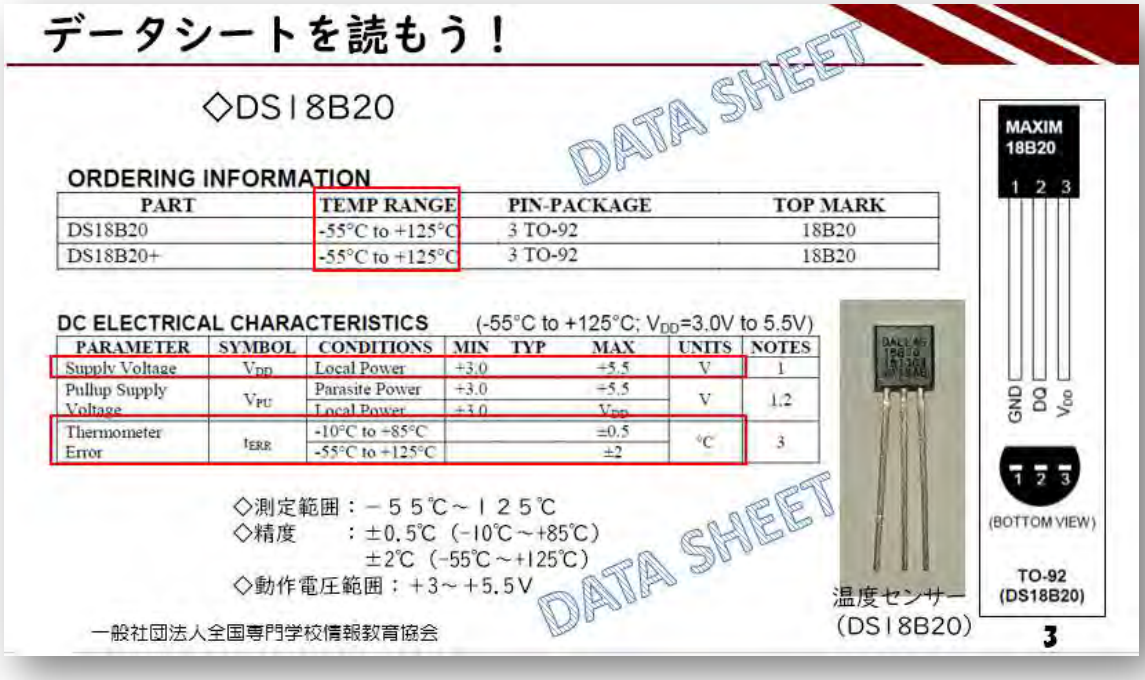
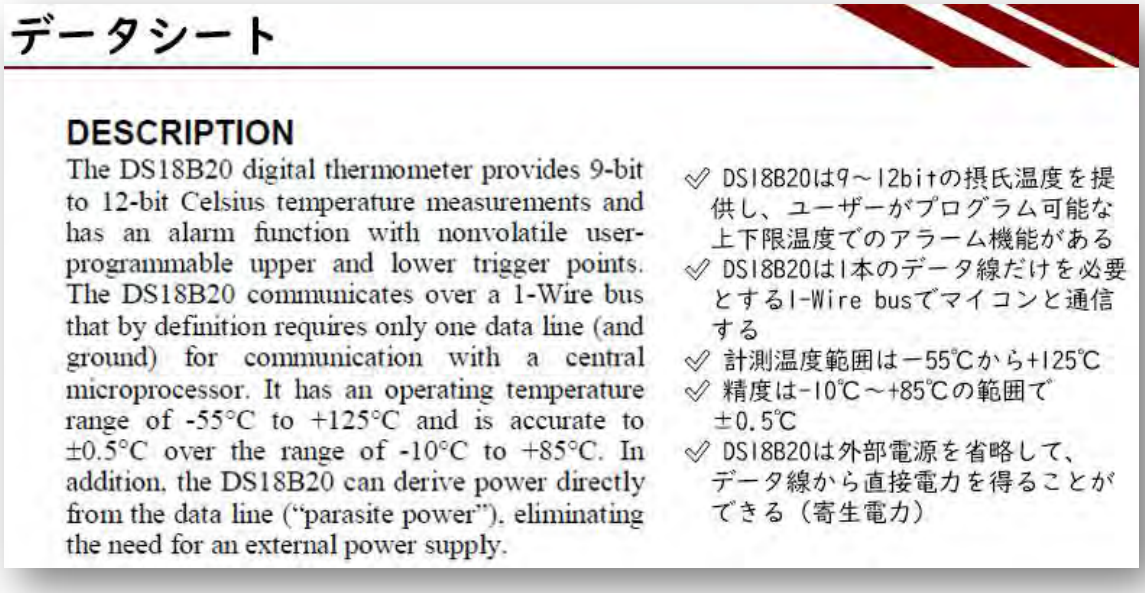


図　152

図はデータシートの一部です。温度範囲は-55～+125℃と広範囲で使用できます。電源電圧は+3～+5.5V で、5V の USB 供給で駆動可能です。誤差は-10～+85℃の範囲で±0.5℃と高精度です。



データシート

DESCRIPTION

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of -55°C to +125°C and is accurate to ±0.5°C over the range of -10°C to +85°C. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

✓ DS18B20は9～12bitの摂氏温度を提供し、ユーザーがプログラム可能な上下限温度でのアラーム機能がある
✓ DS18B20は1本のデータ線だけを必要とする1-Wire busでマイコンと通信する
✓ 計測温度範囲は－55℃から+125℃
✓ 精度は-10℃～+85℃の範囲で±0.5℃
✓ DS18B20は外部電源を省略して、データ線から直接電力を得ることができる（寄生電力）

図　153

ある温度になるとアラーム信号を出力する機能がありますが、今回は未使用です。マイコンとは１本の信号線で通信を行います。

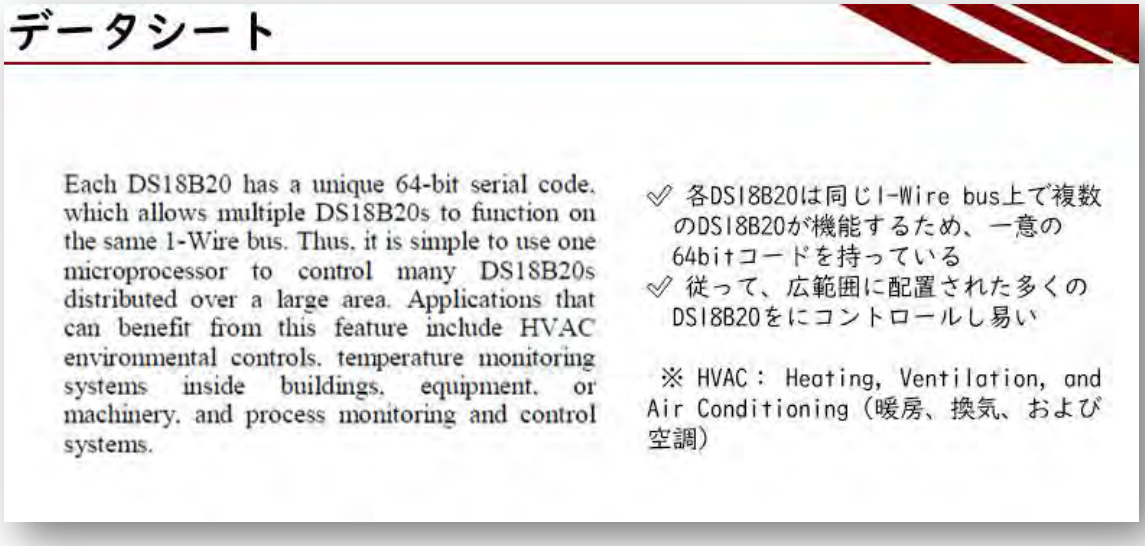
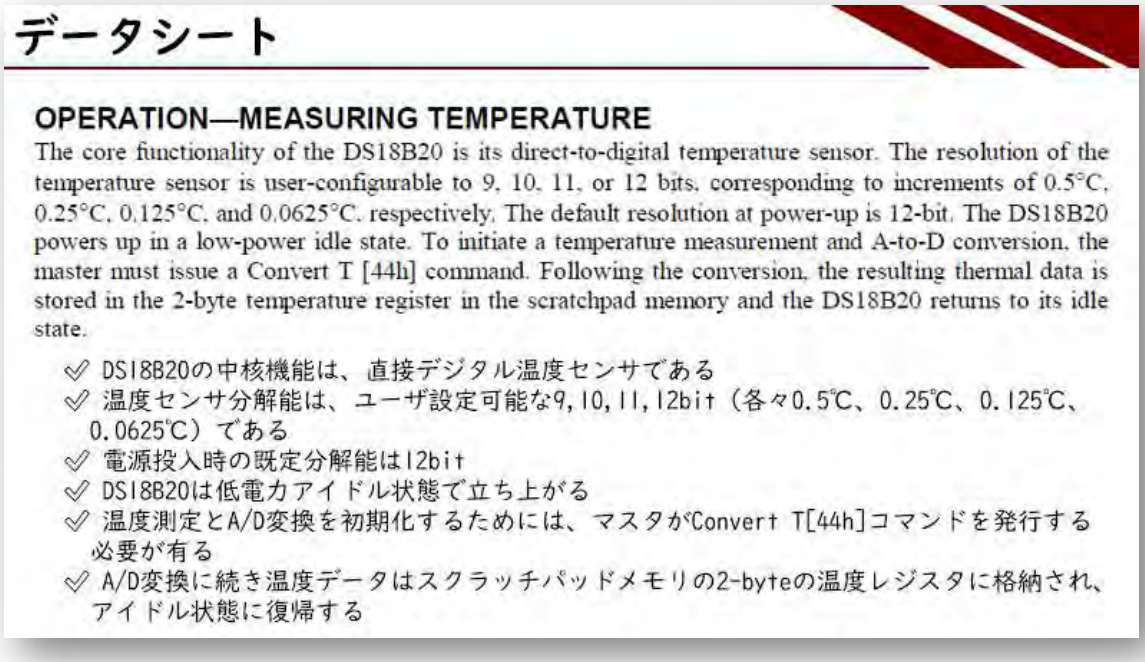


図　154

1本の1-Wire busに複数のセンサを接続出来ますが、センサを識別するため、センサ毎に異なる64bitのコードを持っています。


データシート

OPERATION—MEASURING TEMPERATURE

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state.

- ✓ DS18B20の中核機能は、直接デジタル温度センサである
- ✓ 温度センサ分解能は、ユーザ設定可能な9,10,11,12bit（各々0.5℃、0.25℃、0.125℃、0.0625℃）である
- ✓ 電源投入時の既定分解能は12bit
- ✓ DS18B20は低電力アイドル状態で立ち上がる
- ✓ 温度測定とA/D変換を初期化するためには、マスタがConvert T[44h]コマンドを発行する必要が有る
- ✓ A/D変換に続き温度データはスクラッチパッドメモリの2-byteの温度レジスタに格納され、アイドル状態に復帰する

図 155

センサ分解能は 9bit で 0.5℃から 12bit で 0.0625℃まで、設定ができます。今回は、最高分解能 12bit で使用します。

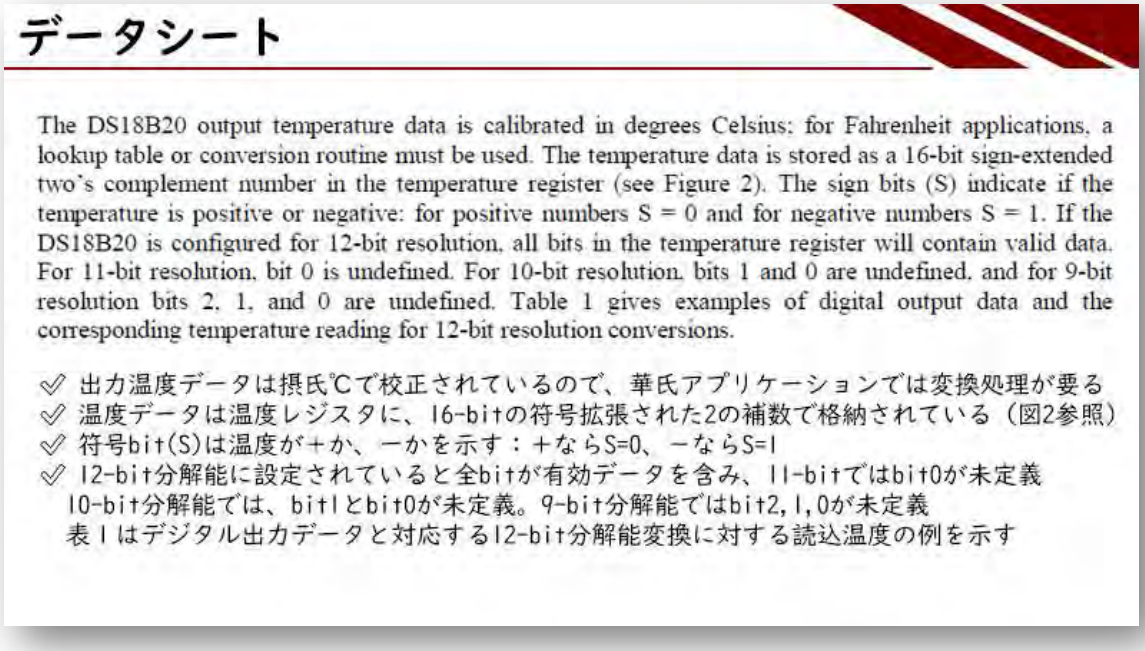


図 156

データは摂氏温度で出力されるので、変換の必要は有りません。この PICAXE はこの温度センサに対する専用の読み出し命令 readtemp（後で解説）を持っているので細かな初期設定が要りません。一般のマイコンでこのセンサを使うためには、図で示すような細かな点を確認し、プログラムで対応しなければいけません。

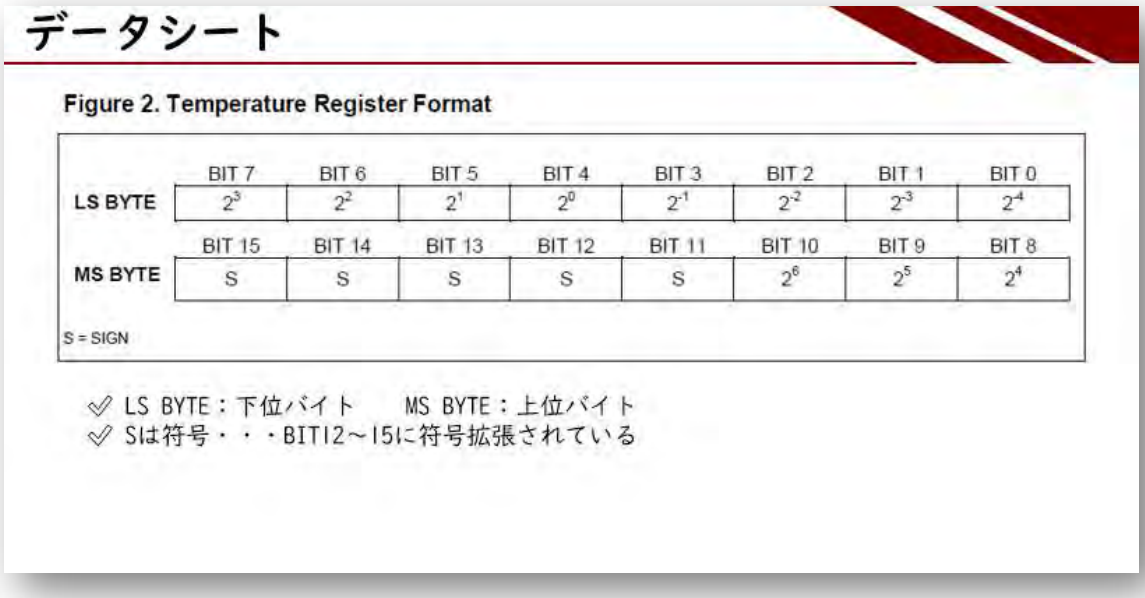
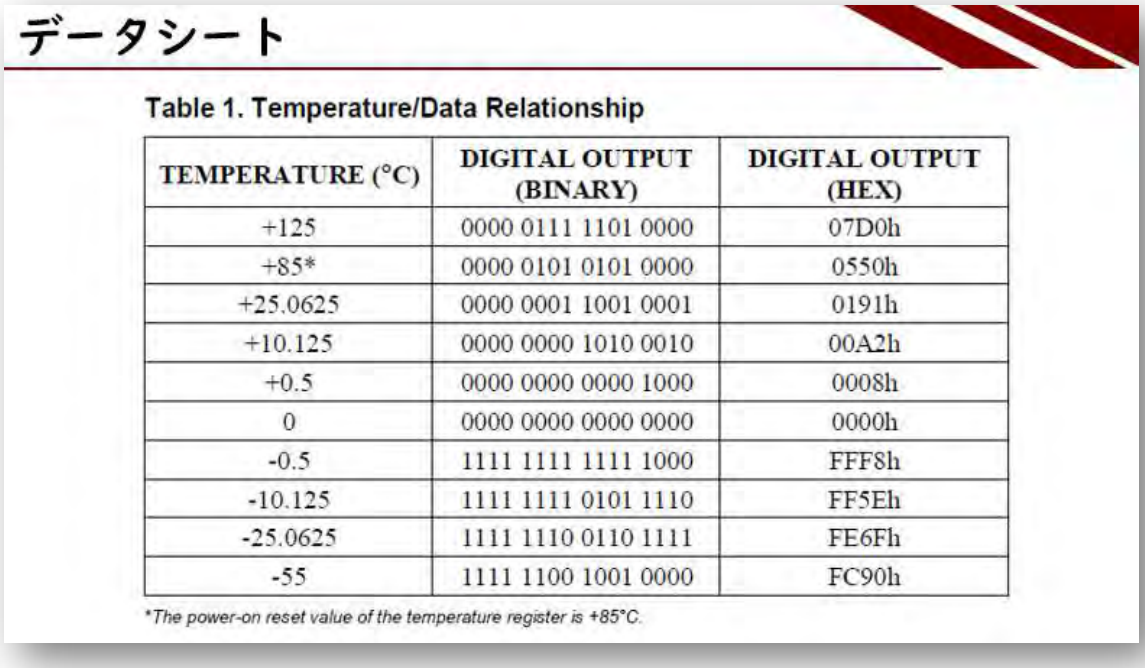


図　157

図は温度レジスタの内容です。LS BYTE の BIT0 は 2 の-4 乗ですから、0.0625 になります。



データシート

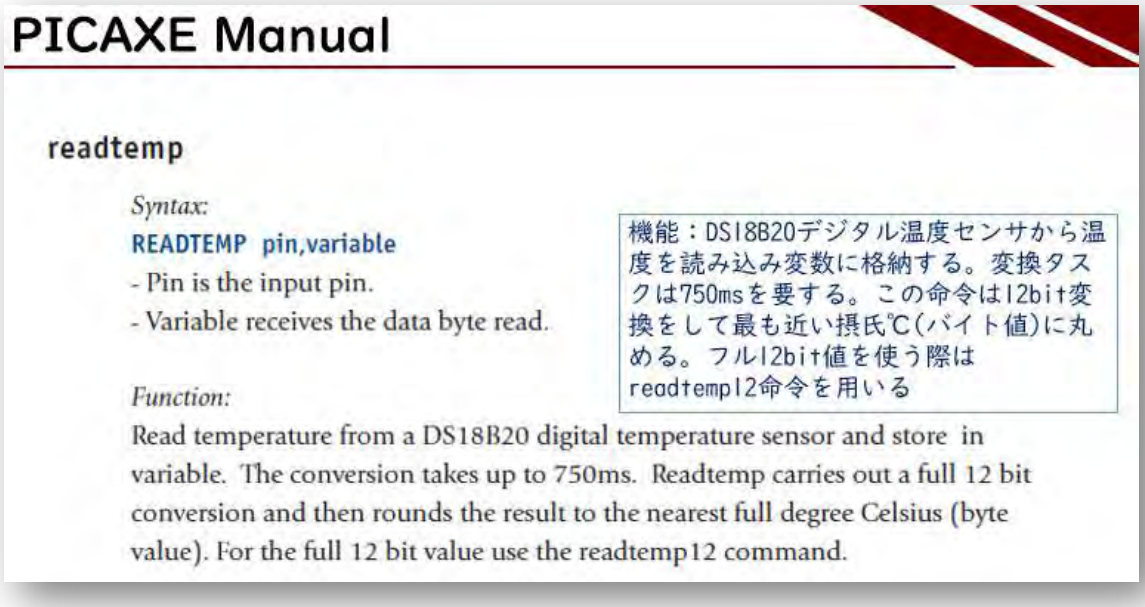
Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

*The power-on reset value of the temperature register is +85°C.

図 158

図は代表的な温度とセンサの出力の値を 2 進数と 16 進数で示した表です。



PICAXE Manual

readtemp

Syntax:

READTEMP pin,variable

- Pin is the input pin.
- Variable receives the data byte read.

機能：DS18B20デジタル温度センサから温度を読み込み変数に格納する。変換タスクは750msを要する。この命令は12bit変換をして最も近い摂氏℃(バイト値)に丸める。フル12bit値を使う際はreadtemp12命令を用いる

Function:

Read temperature from a DS18B20 digital temperature sensor and store in variable. The conversion takes up to 750ms. Readtemp carries out a full 12 bit conversion and then rounds the result to the nearest full degree Celsius (byte value). For the full 12 bit value use the readtemp12 command.

図 159

PICAXE でこの DS18B20 を使用する際には、readtemp という専用の命令が準備されています。センサを 12bit 分解能で使用し、計測したデータを摂氏温度に丸めて返してくれる命令です。12bit 分解能をそのまま用いる場合には readtemp12 という命令を使い、bit 分解能による演算が必要になります。readtemp、readtemp12 について、マニュアルを参照してください。

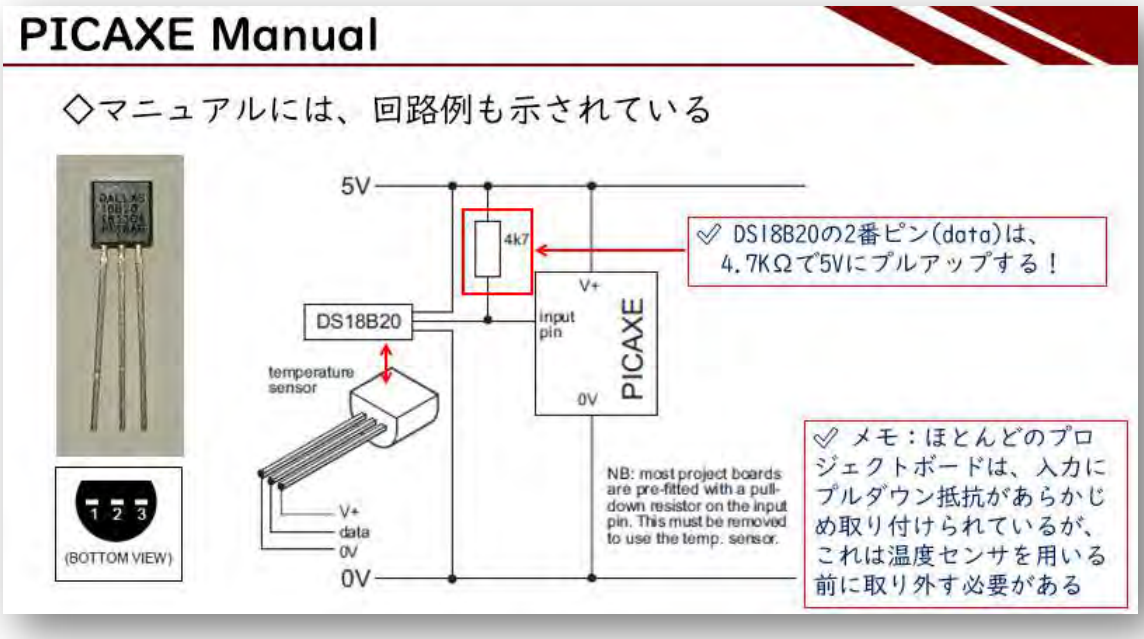


図 160

PICAXE のマニュアル（BASIC Commands）には、図のような回路の例示があります。例に従い回路を設計します。

図 161

デジタル温度センサ DS182B20 は 3 番ピン（C.4）に接続します。

【注意】マニュアル（BASIC Commands）の readtemp のページに、PICAXE08M2 は C.0,C.3,C.5 が使用できないと記載があります。

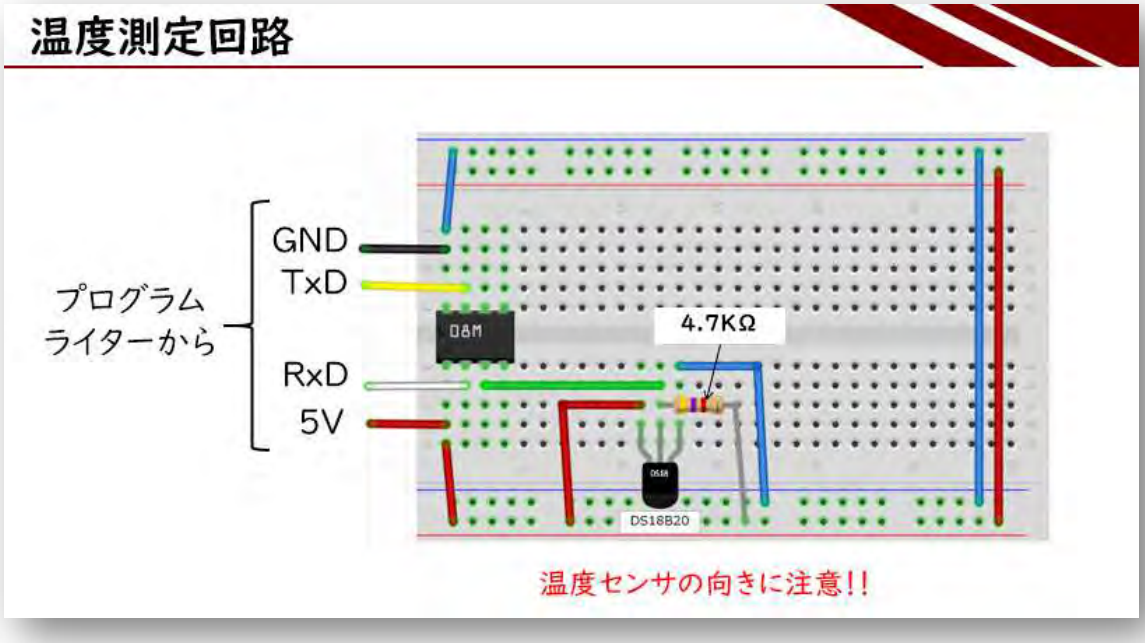


図　162

デジタル温度センサの向きに注意してください。センサの平らな面が図の上に向くように配置します。このデジタル温度センサ(DS18B20)は、アナログ温度センサ(LM61CIZ)と向きが反対ですが、中央のピンでマイコンと通信するので、アナログ温度センサの置き換えには好都合です。

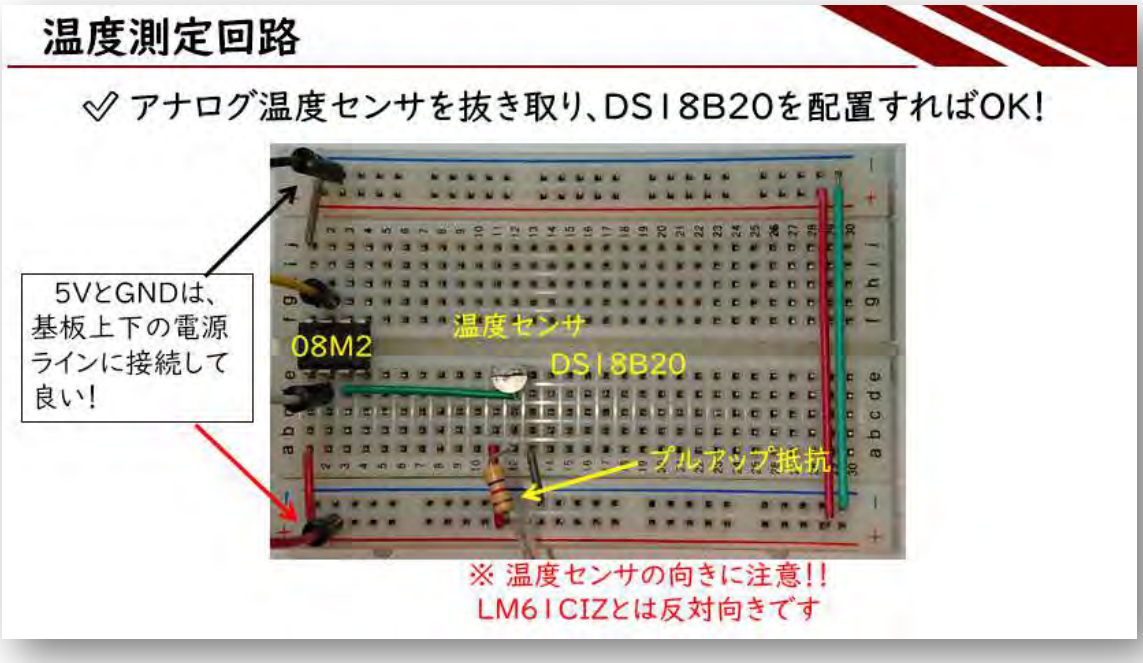


図　163

実際に作成した回路を示します。アナログ温度センサを抜き取り、デジタル温度センサ(DS18B20)を差し込みますが、向きが反対です。注意して下さい。

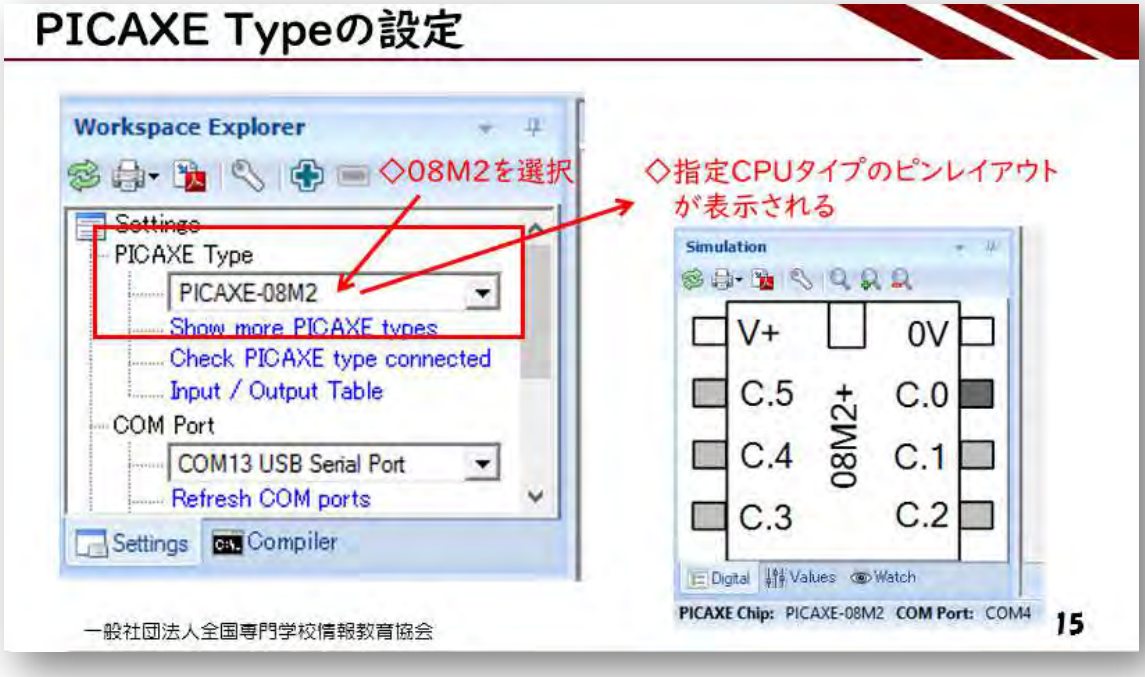


図　164

PICAXE Editor 左上のウインドウで、対象の PICXE08M2 チップを選択します。

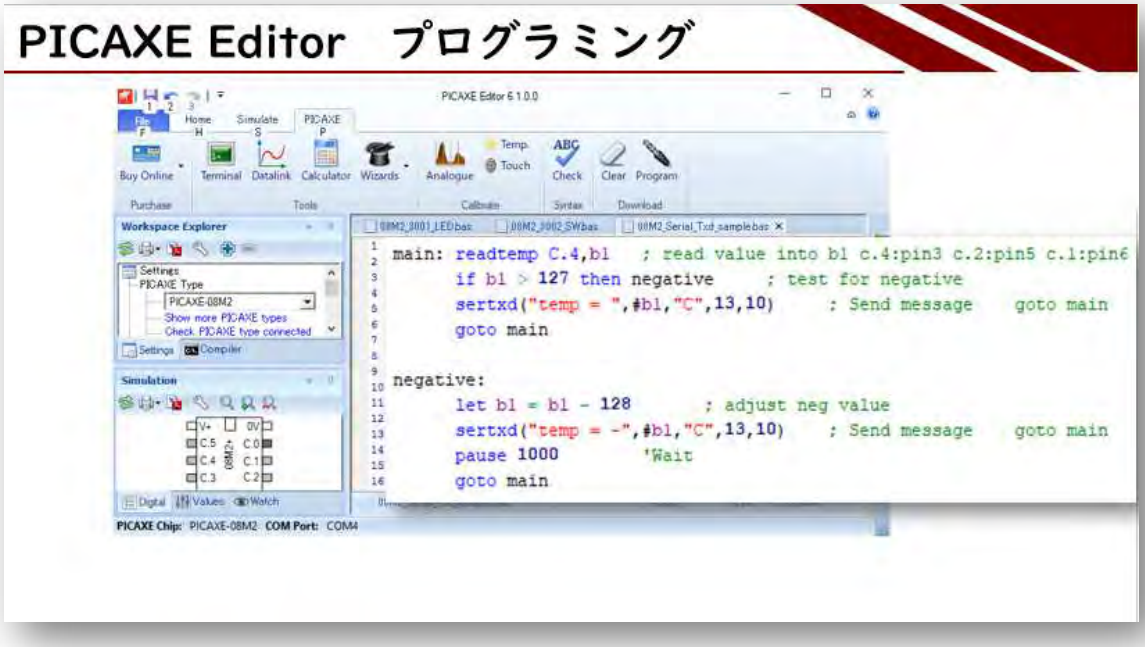


図　165

例に倣ってプログラムを作ります。

プログラム解説

【ソースファイル名 ： 08M2_3008_Digital_Temp_Sensor_DS18B20.bas】

```
main: readtemp C.4,b1    ; read value into b1 c.4:pin3 c.2:pin5 c.1:pin6
      if b1 > 127 then negative      ; test for negative
      sertxd("temp = ",#b1,"C",13,10)      ; Send message
      pause 1000          'Wait
      goto main

negative:
      let b1 = b1 - 128         ; adjust neg value
      sertxd("temp = -",#b1,"C",13,10)     ; Send message
      pause 1000          'Wait
      goto main
```

図 166

まず、readtemp で温度を読み変数 b1 に格納します。読込温度が 127 より大きい場合は負の処理を行います。正であればそのまま温度としてシリアル出力します。**負の場合は b1 から 128 を減算し（b1 に 128 の補数【0x100】を加えて）得た結果(補数)を元に戻して、マイナス符号をつけてシリアル出力します。**この演算は、図 159 の表で-55℃の値を利用して計算してみると分かるでしょう。

※1 の補数は、元になる数値のビット列を反転したものです。2 の補数は、1 の補数に 1 を加えたものです。

readtemp 命令では、DS18B20 から読み取った 12bit 表現の温度を近い整数に丸めていますので、小数部を得ることができませんが、簡単な回路とプログラムでデジタル温度を読めるのがメリットです。

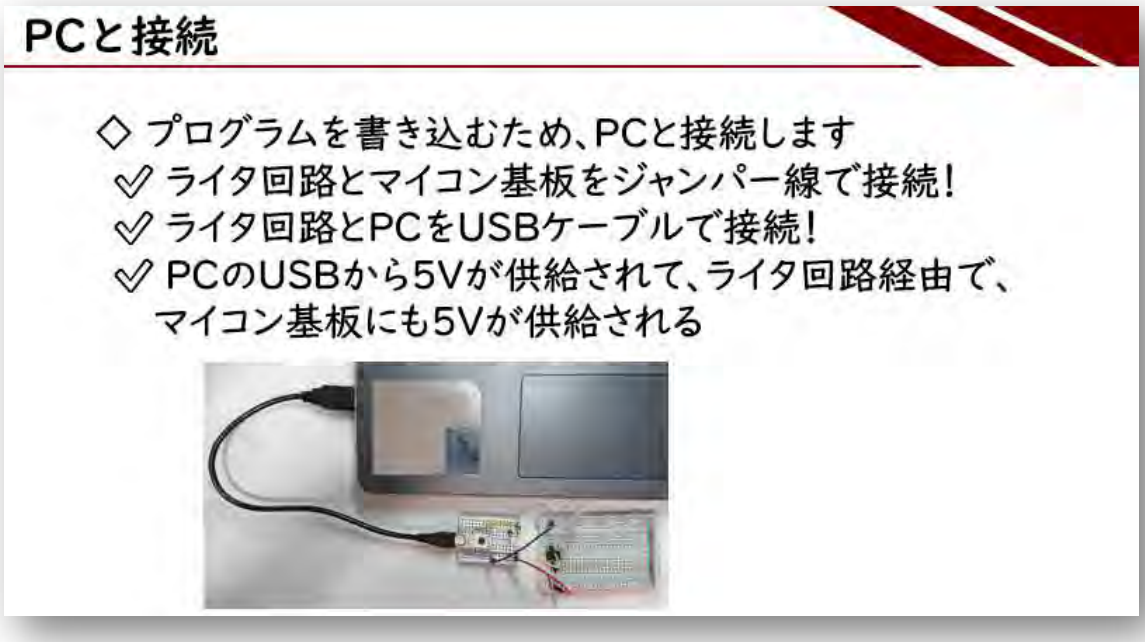


図 167

プログラムが完成したら、コンパイルと書込みを行います。PC、ライタ回路、マイコン回路を接続して下さい。

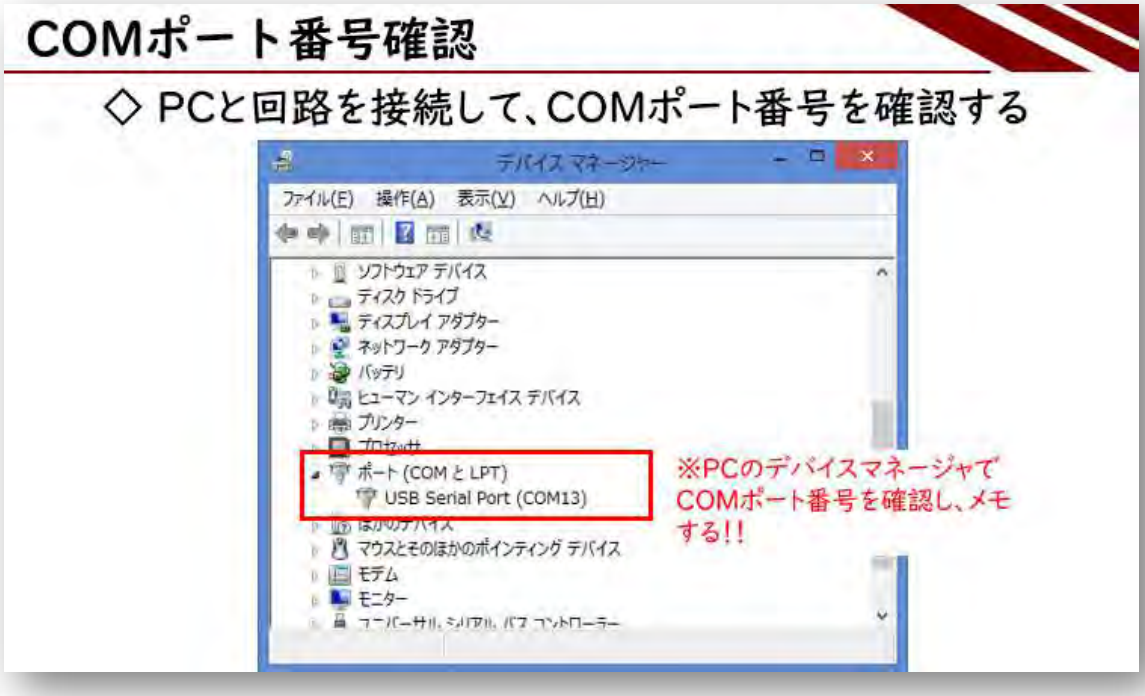


図 168

デバイスマネージャで COM ポート番号を確認します。

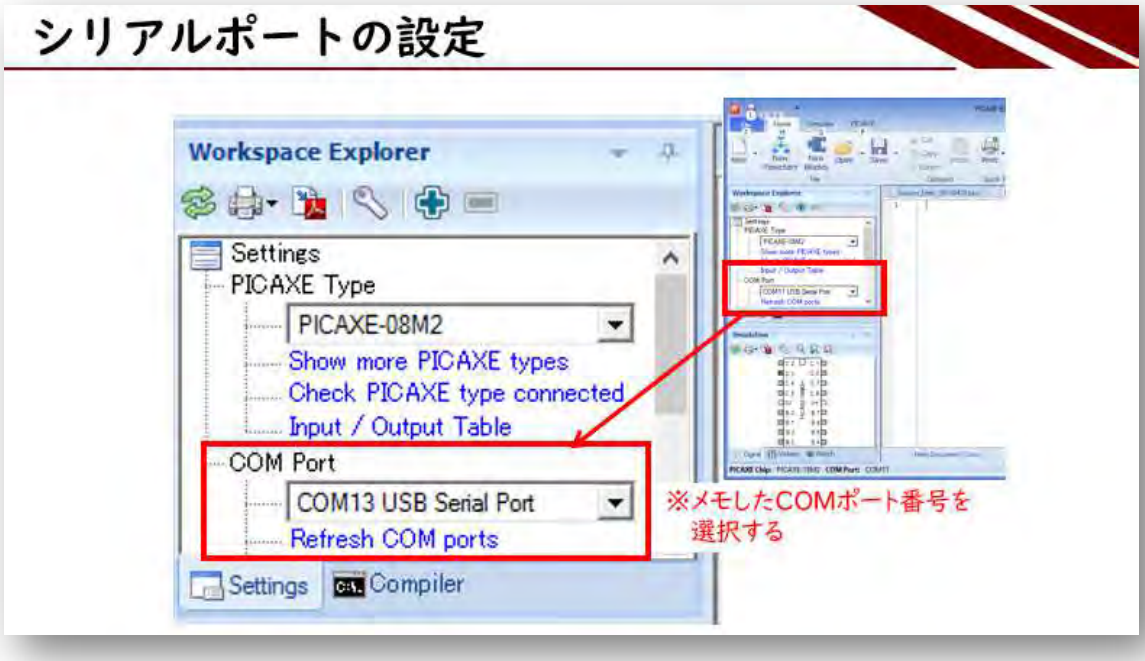


図 169

COM ポート番号を設定します。

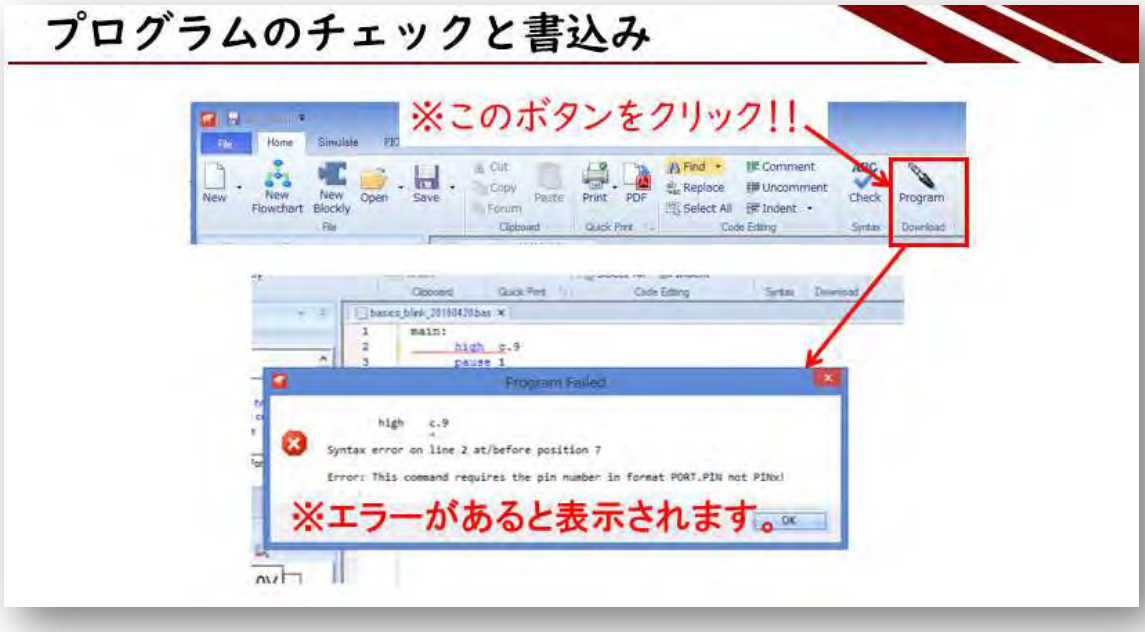


図 170

Home の Program ボタンを押下してコンパイルします。

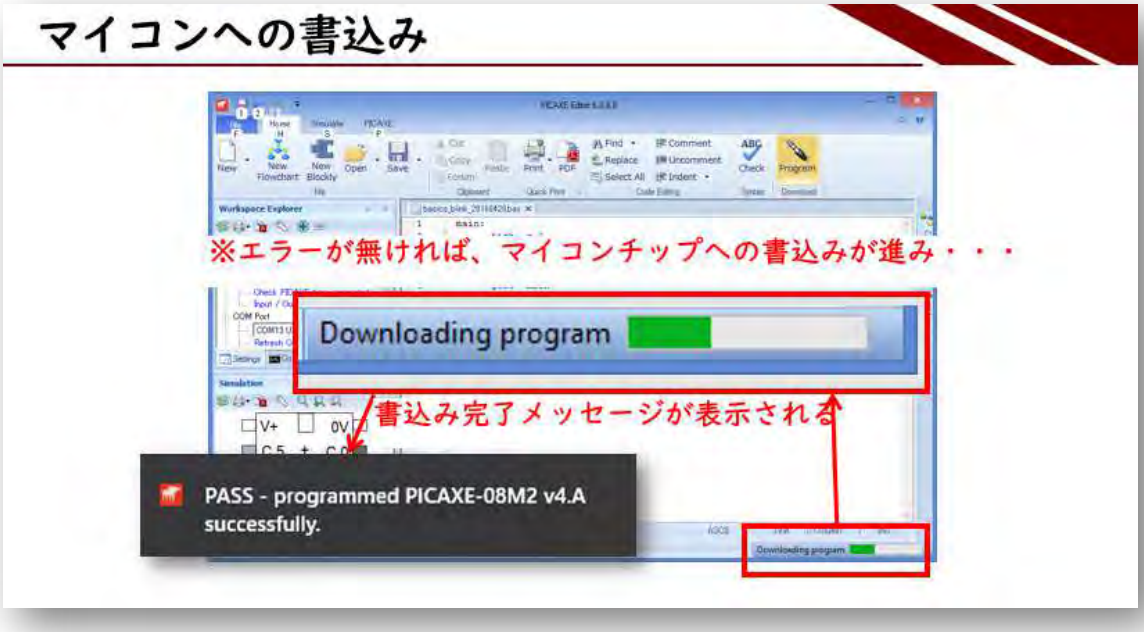


図　171

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

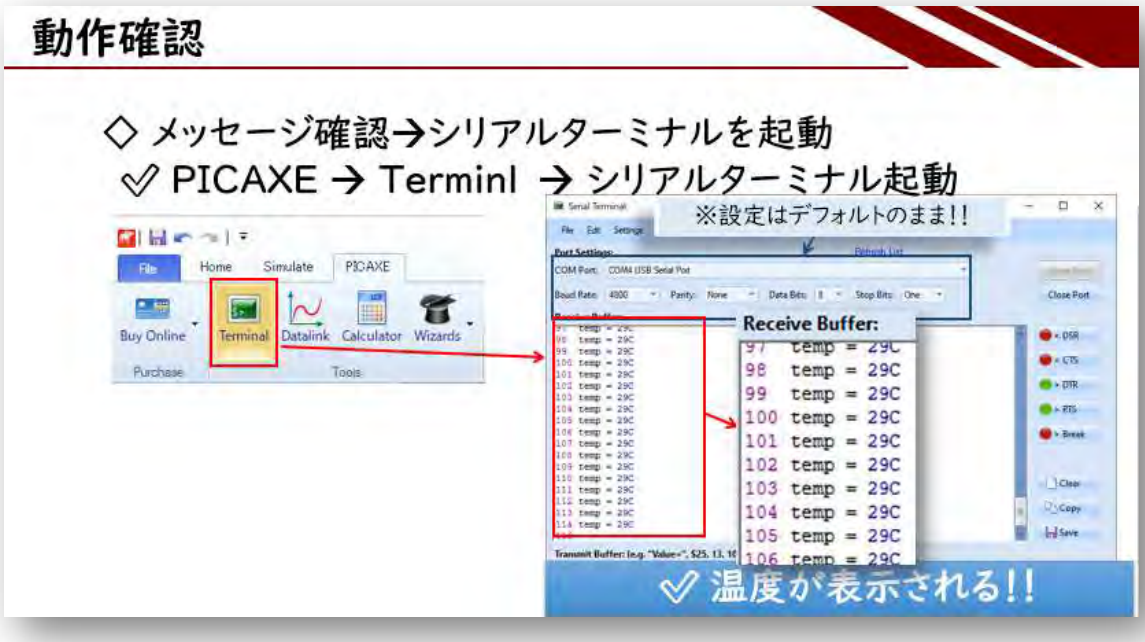


図 172

シリアルターミナルで動作確認します。ウインドウに温度が表示されていることを確認します。次にセンサを指でつまむなどして、表示される温度が変化することを確認します。

第9回　LCD

図 173

これまでの実験では、マイコンから出力される数値データは、シリアルターミナルを介して確認をしていましたが、これは PC を高価な表示装置として使っていることになります。マイコンのシステムだけで文字表示できる液晶表示器（LCD）を使ってみましょう。

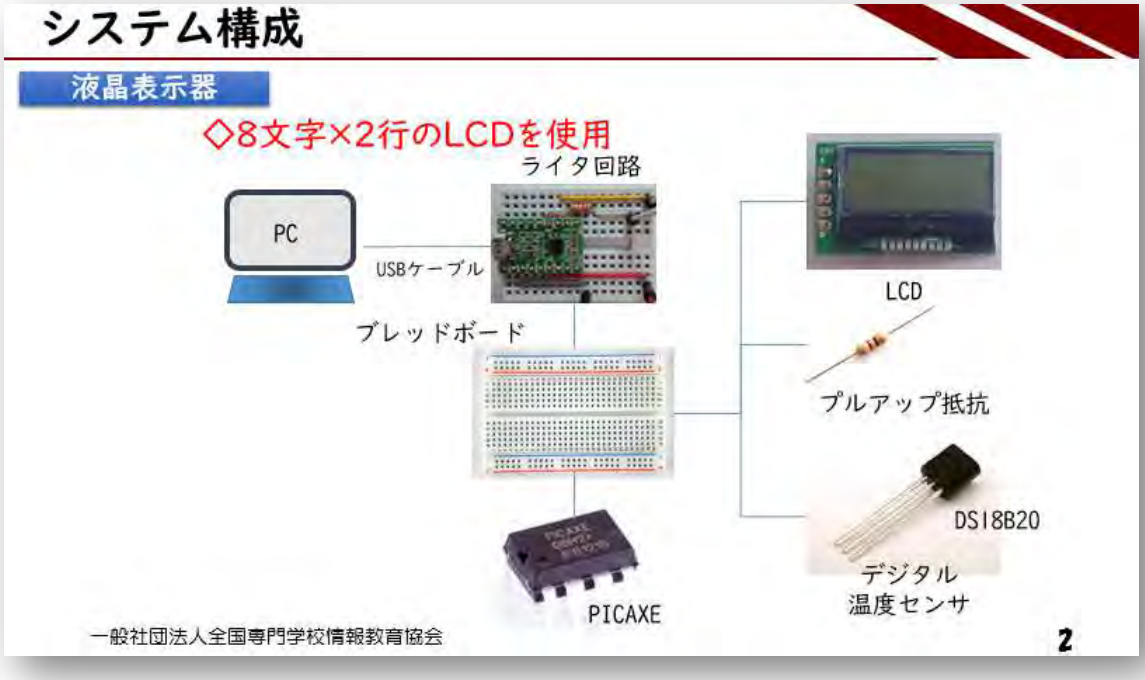


図　174

LCD は多くの種類がありますが、その中で小さいものを使い、マイコンから文字列を表示してみます。前回使用したデジタル温度センサは、回路の中にそのまま残しておきます。

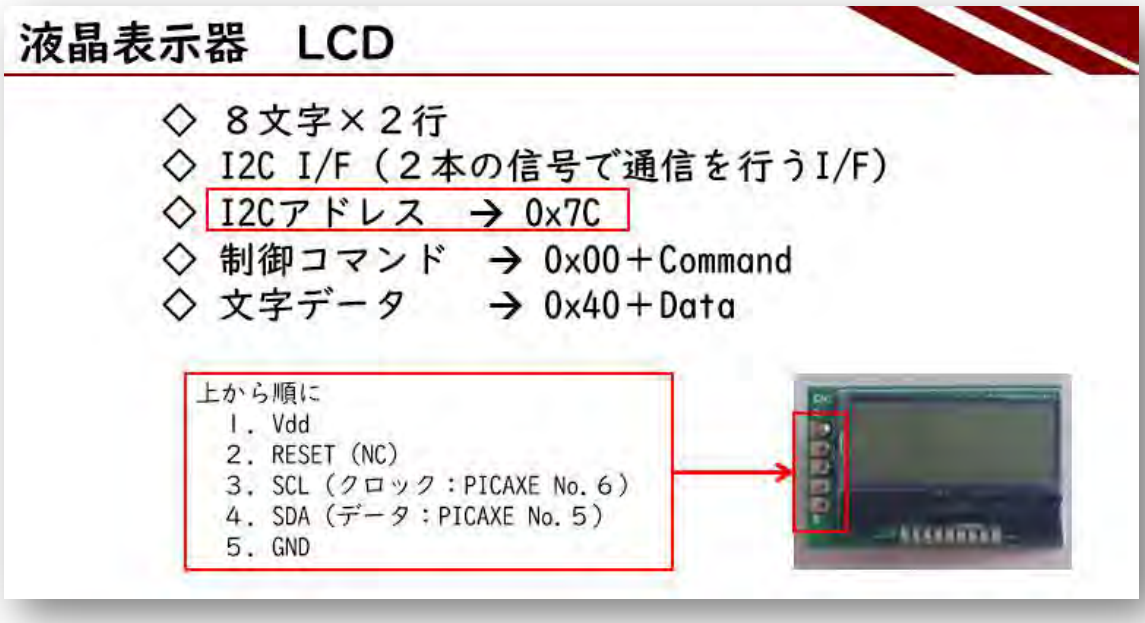


図　175

使用するLCDは8文字×2行表示のモノクロ液晶表示器です。この表示器はI2C I/Fでマイコンと接続します。I2C(Inter-Integrated Circuit)は、フィリップス社が提唱したシリアル通信の方式で周辺デバイスとの通信に用います。このI/Fは2本の信号線（I2C BUS）に複数のデバイスを配置することができるI/Fです。そのためマイコンが通信する相手を識別するためにI2Cアドレス（スレーブアドレス）が設定されています。LCDをコントロール（初期化やカーソル位置制御、画面消去など）するコマンドと、文字列を表示するコマンドがあり、これらをI2C I/Fを使ってマイコンから送信します。LCDには5本のピンがあり上から1～5ピンと番号が付いています。各々の信号はPICAXEのピン配置に記述されている信号と照らし合わせるとどのピンに接続するか分かるでしょう。

図 176

I2C I/F の制御信号は 2 本あり、図では hi2cscl と表記された 6 番ピンと、hi2csda と表記された 5 番ピンを使用します。hi2cscl は I2C 通信の同期クロックで hi2csd は通信データです。

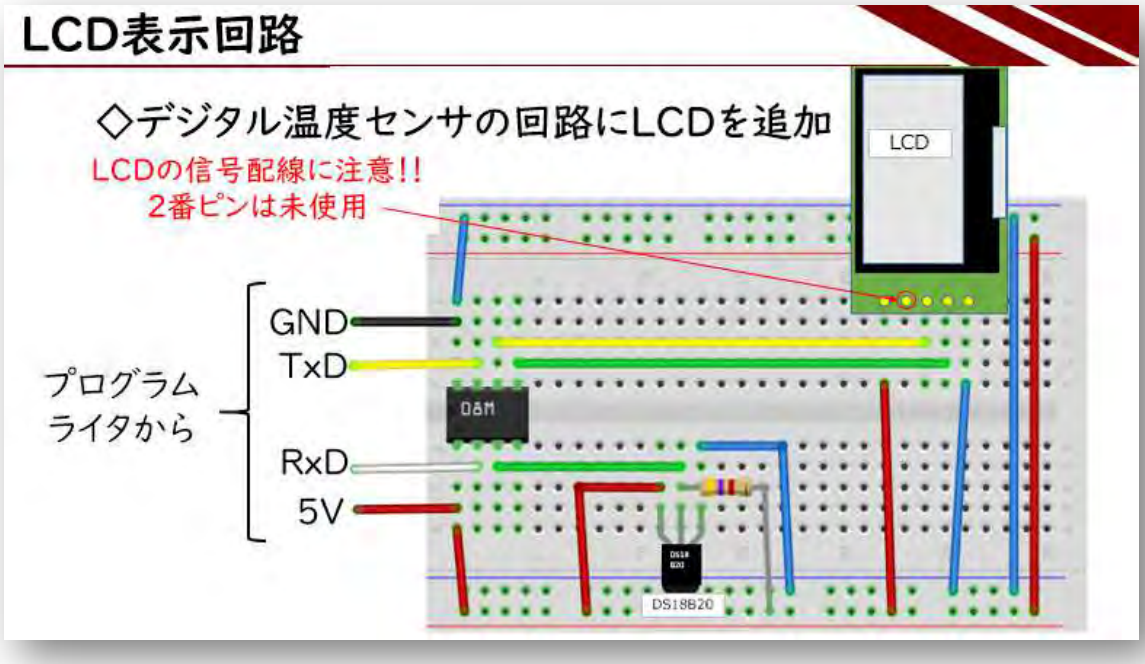


図　177

LCD 表示回路を追加します。LCD の基板裏側に各ピンの信号名称が記載されているので、よく確認して配線しましょう。デジタル温度センサの回路は次回に使用するので、そのまま残しておきます。

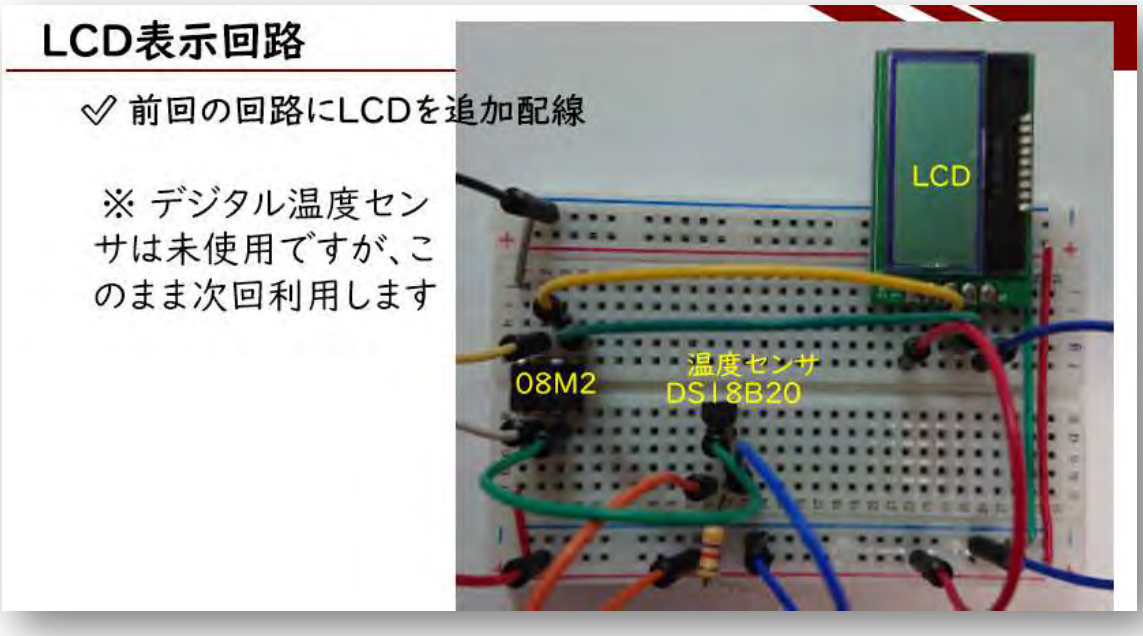


図　178

実際に配線すると鵜の様になります。

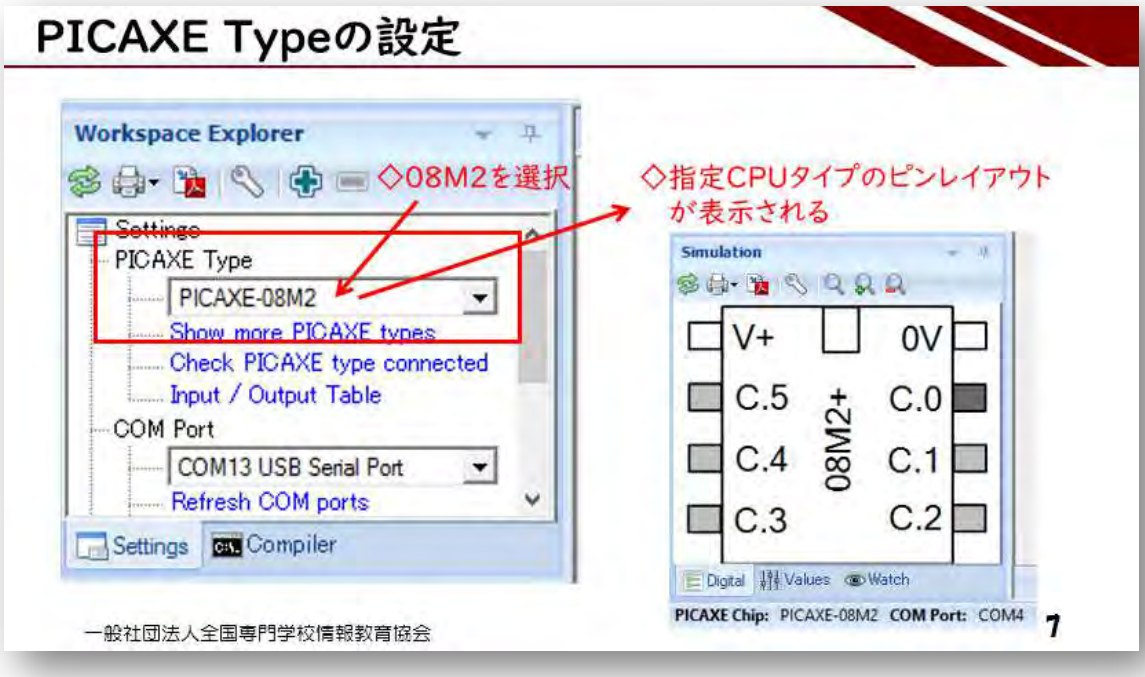


図 179

使用する PICAXE チップのタイプを設定します。

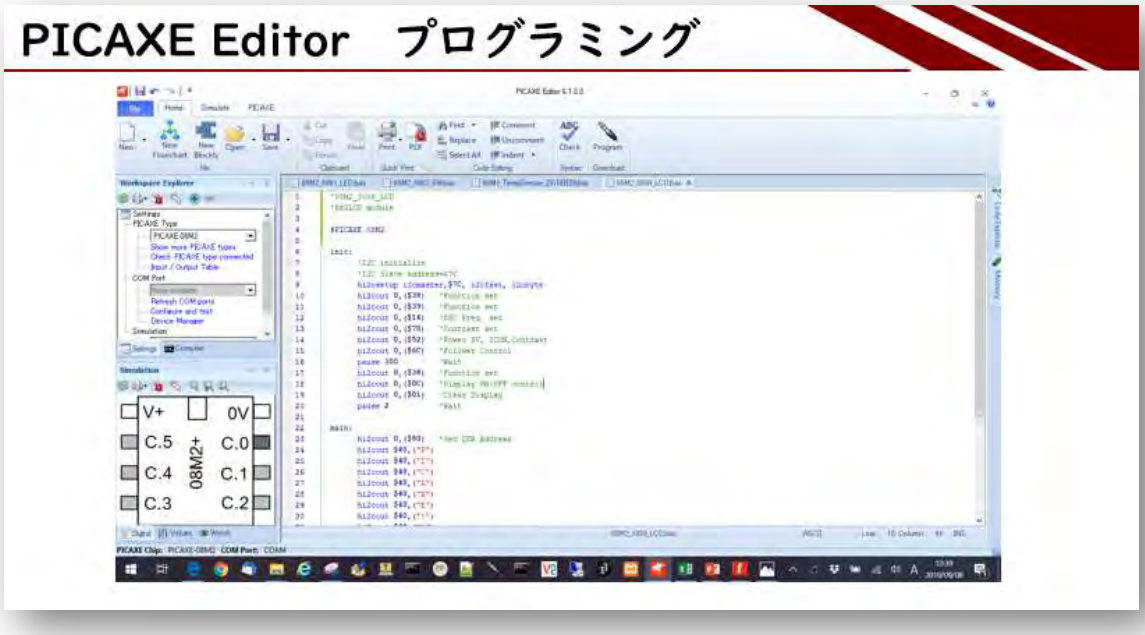


図　180

Home→New とボタンを押下して新しいプログラムを作成します。今回は LCD の初期化のためのプログラムがあるので、少し長くなります。

図 181

図は、LCD 初期化ルーチンです。hi2csetup 命令では、CPU の I2C マスタ動作、LCD のスレーブアドレス、I2C 通信速度、I2C スレーブアドレスの幅を設定しています。hi2cout で送信するコマンドは 0 に続けて送られているので、制御コマンドと分かります。$38 から始まる一連のコマンド送信は、LCD のデータシートに掲載の初期化コマンド群です。

図 182

main:以後は、文字表示コマンドです。()内に記述した文字が LCD の表示用メモリに書き込まれます。先頭の($80)と中程の($C0)はメモリのアドレスを指定しています。メモリのアドレスは、LCD のデータシートに記載されています。このプログラムが実行されると、LCD 上段に、PICXE、下段に 01234567 と表示されるはずです。

図 183

データシートに記載の初期化コマンドのシーケンスです。これを基にして初期化ルーチンを作りました。

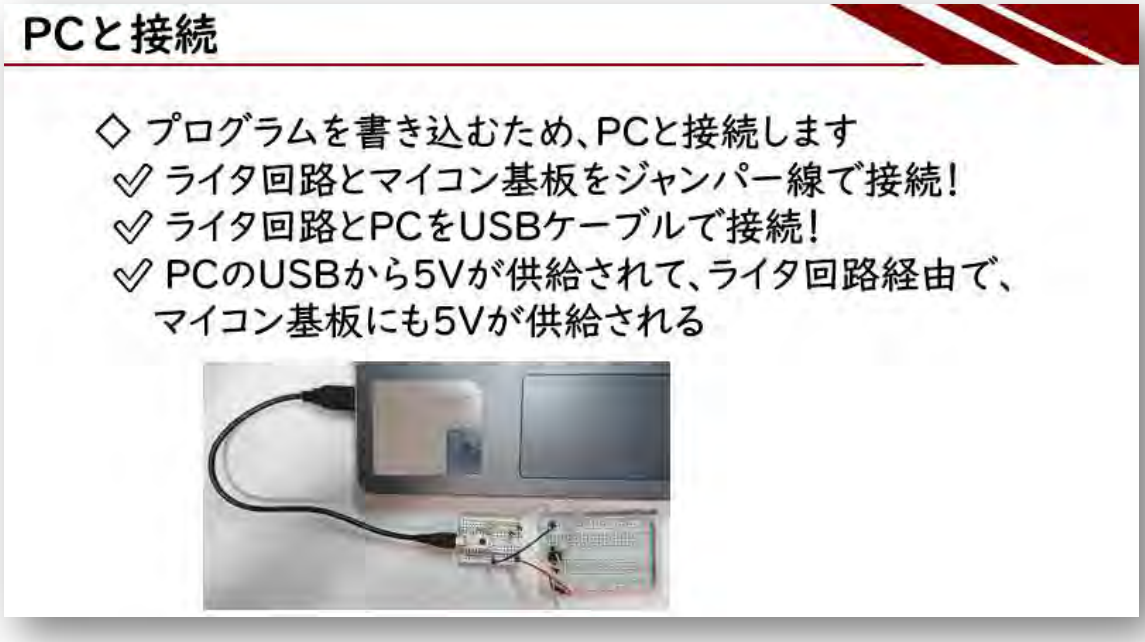


図 184

プログラムが完成したらコンパイルと書込みを行います。図のようにPC、ライタ回路、マイコン回路を接続して下さい。

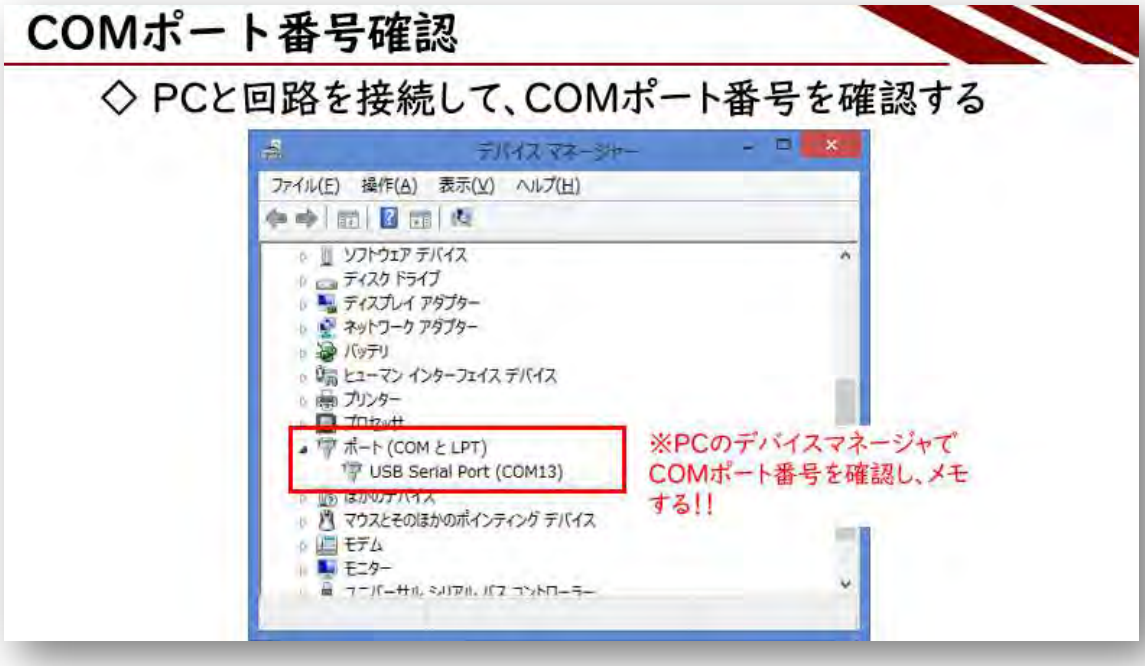


図 185

Windows デバイスマネージャで COM ポート番号を確認します。

図 186

確認した COM ポート番号を設定します。

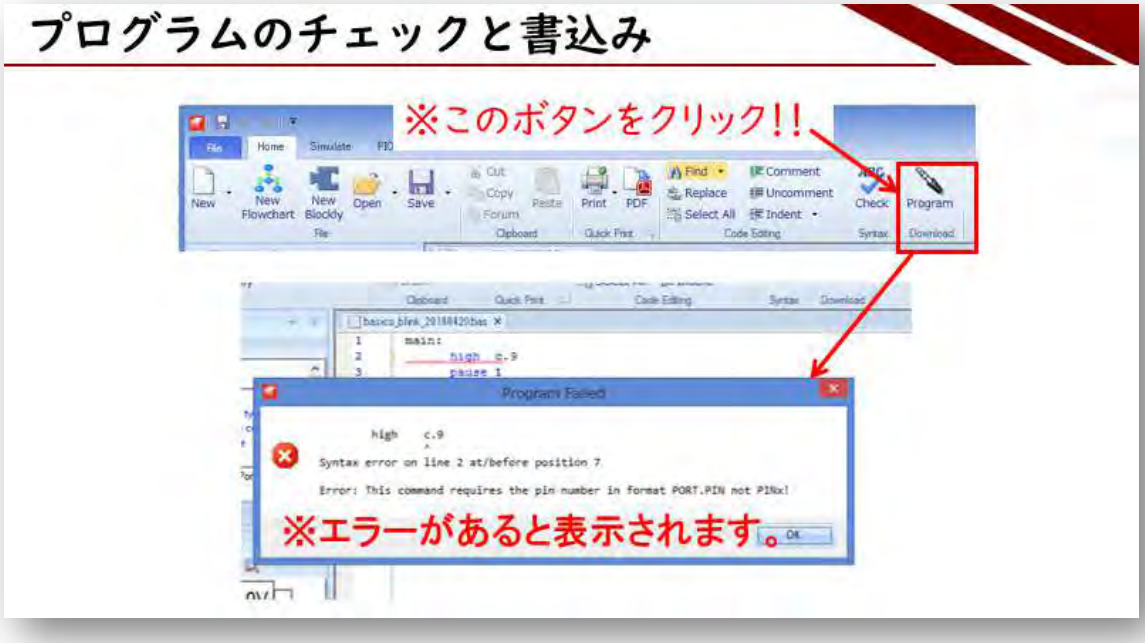


図 187

コンパイルと書込みを行います。Homeの Programボタンを押下します。

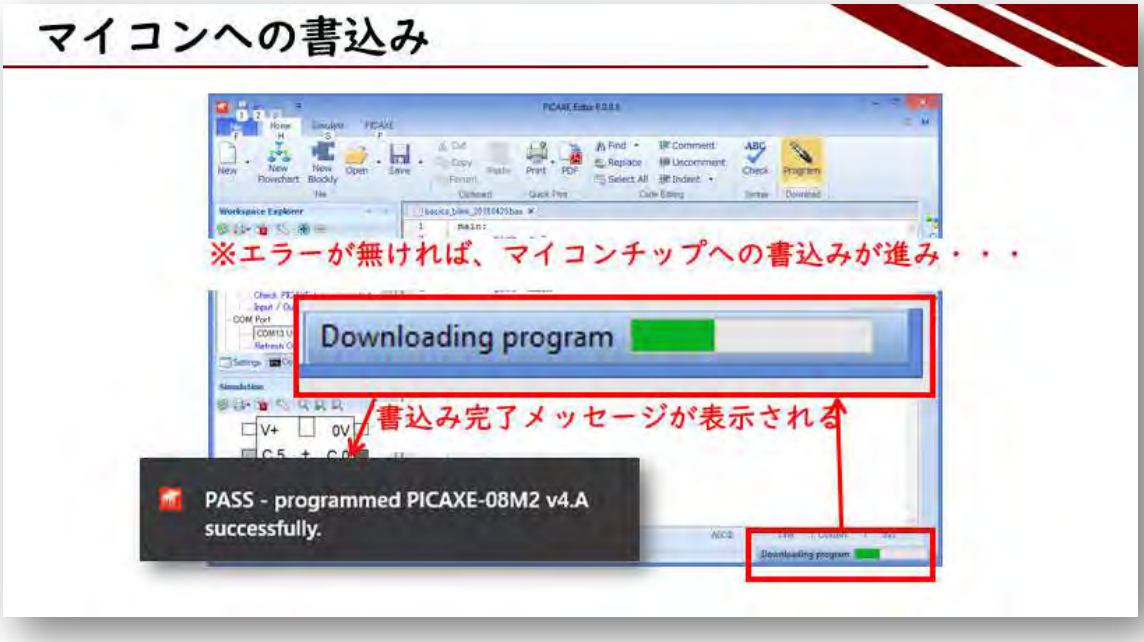


図　188

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

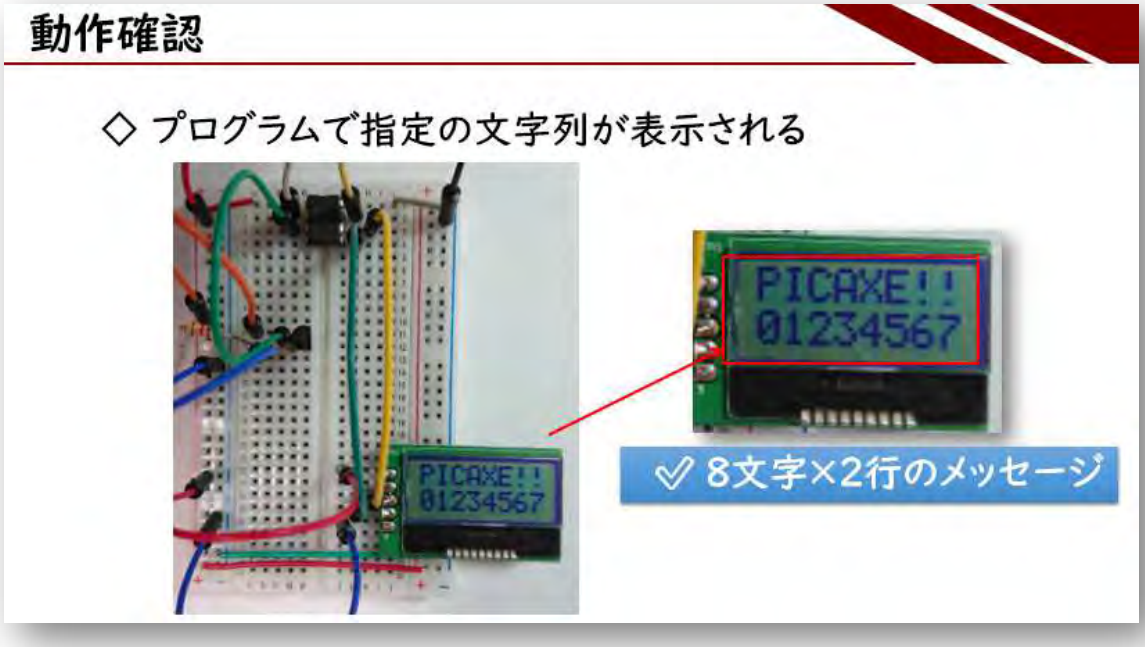


図 189

プログラムは既に実行されて、LCD に図の様に文字列が表示されます。この LCD には、ASCII コードに該当する文字（英数）と記号フォントが内蔵されています。必用に応じて記号などを用いることができます。

第10回　デジタル温度計

図 190

前回は LCD(液晶表示器)を使用して固定の文字列を表示しました。文字通り LCD はマイコンのディスプレイに相当します。今回は既に実装済みのデジタル温度センサの測定温度を LCD に表示する、独立したデジタル温度計を開発してみましょう。加えて PC にもシリアル通信で送信します。

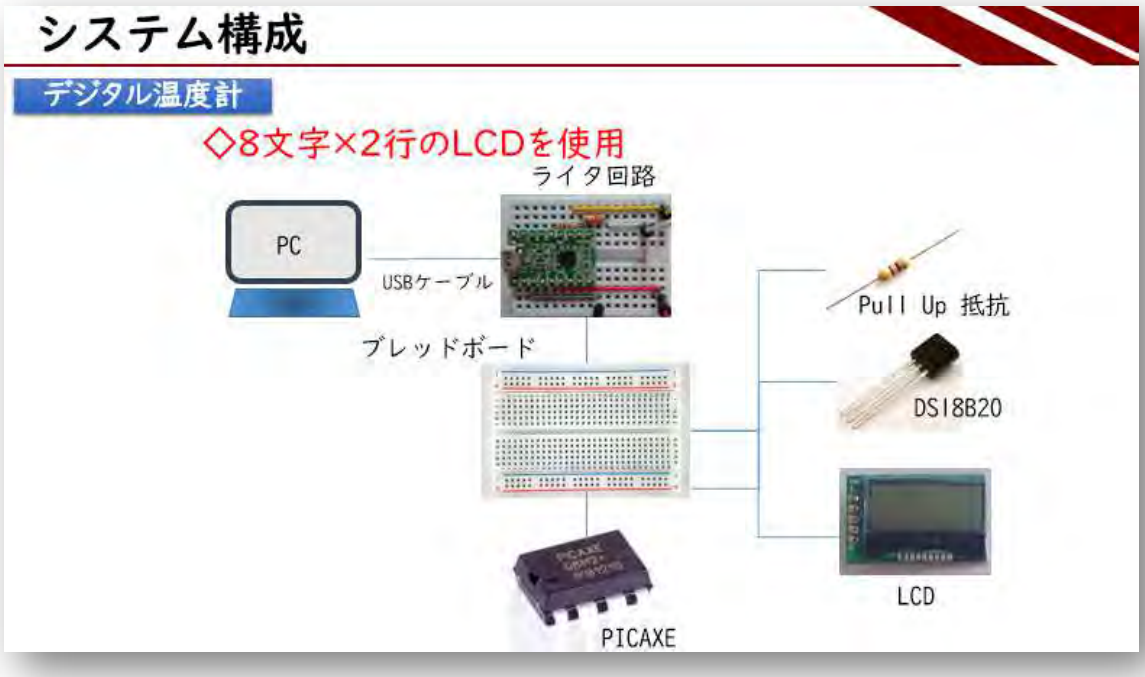


図 191

前回 LCD の回路を作成した際、デジタル温度センサを残しておいたのは、いずれデジタル温度計を作る目的があったからです。ですから、使用パーツは前回と変わりません。

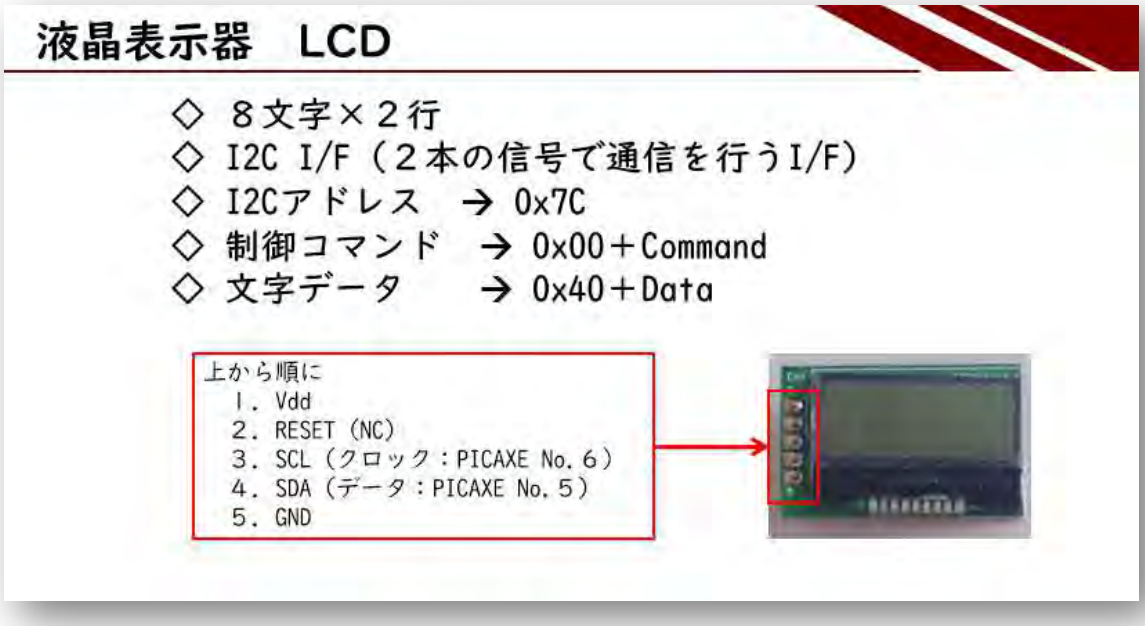


図　192

液晶表示器(LCD)の仕様を再掲します。LCDには5本のピンがあり、PICAXEと直結可能です。各々の信号をマイコンのどのピンに配線するかは、PICAXEの信号名称と比較すれば分かります。

図 193

PIXACE の 3 番ピンには DS18B20 の出力を、6 番ピンには LCD の scl を、5 番ピンには LCD の sda を接続します。前回と同じです。

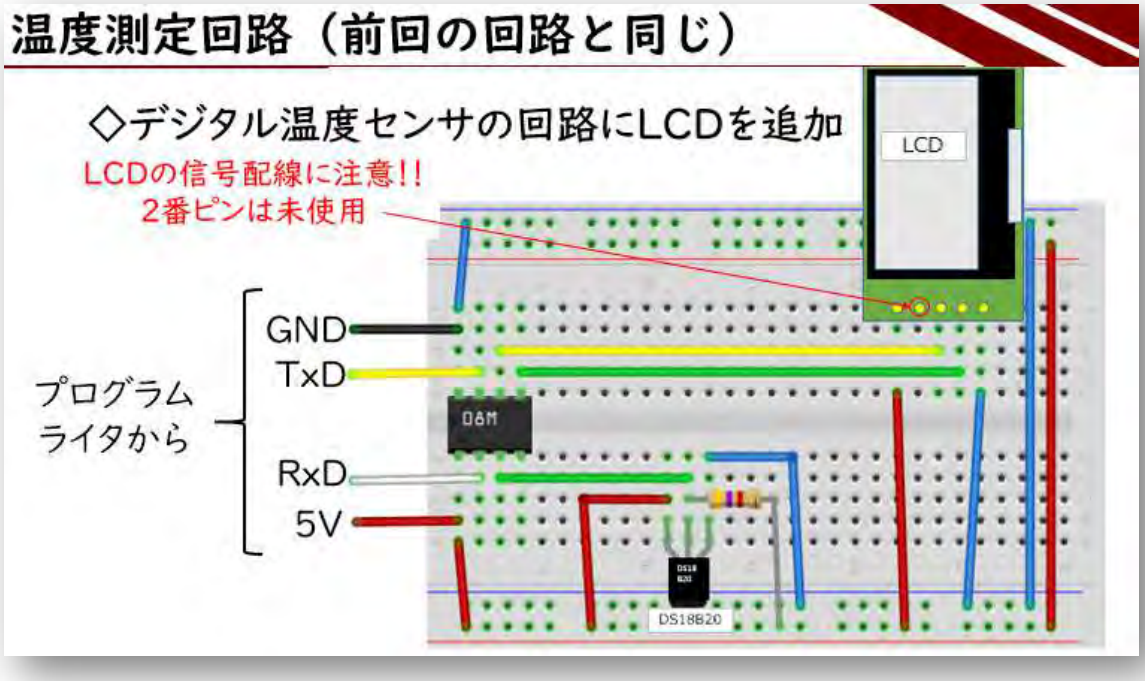


図　194

既に作成済みの回路がそのまま使えます。LCD 表示が成功した回路であれば、容易に動かすことができるはずです。各配線をよく確認してください。

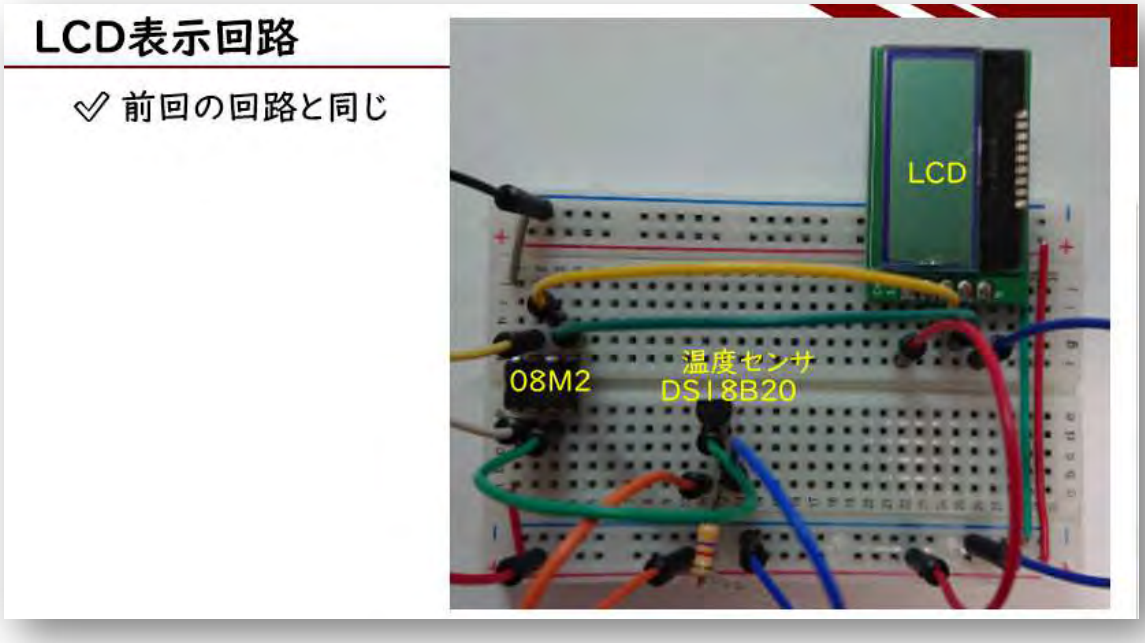


図　195

出来上がりの図を示します。

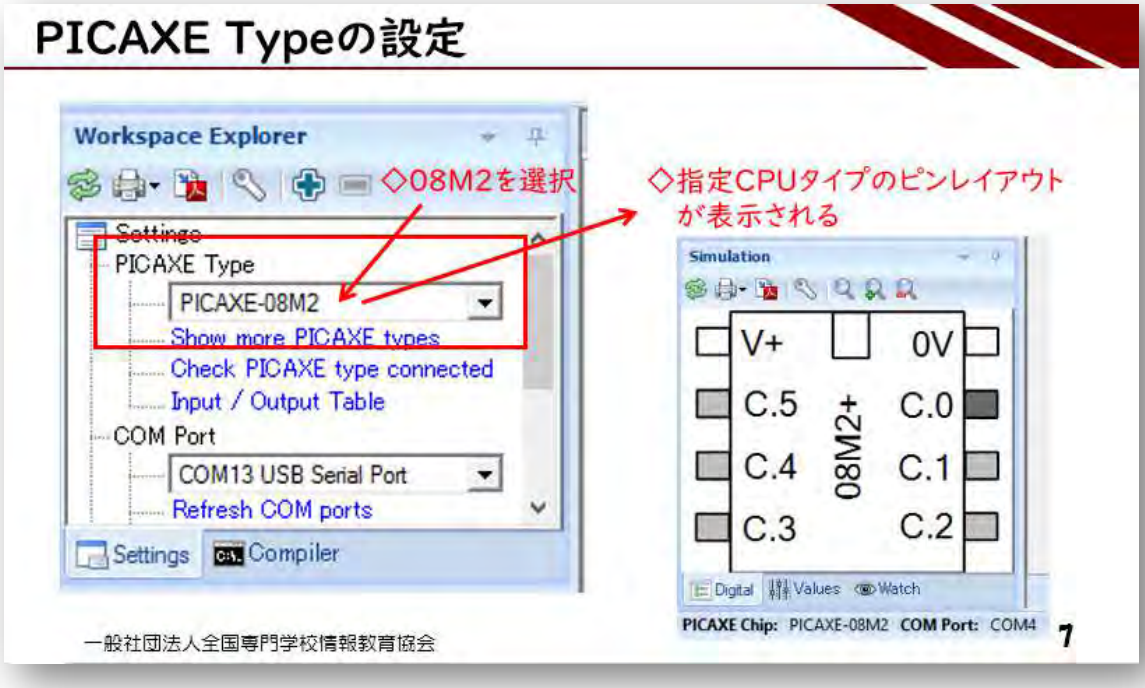


図 196

PICAXE Editor 左上のウインドウで、対象の PICXE08M2 チップを選択します。

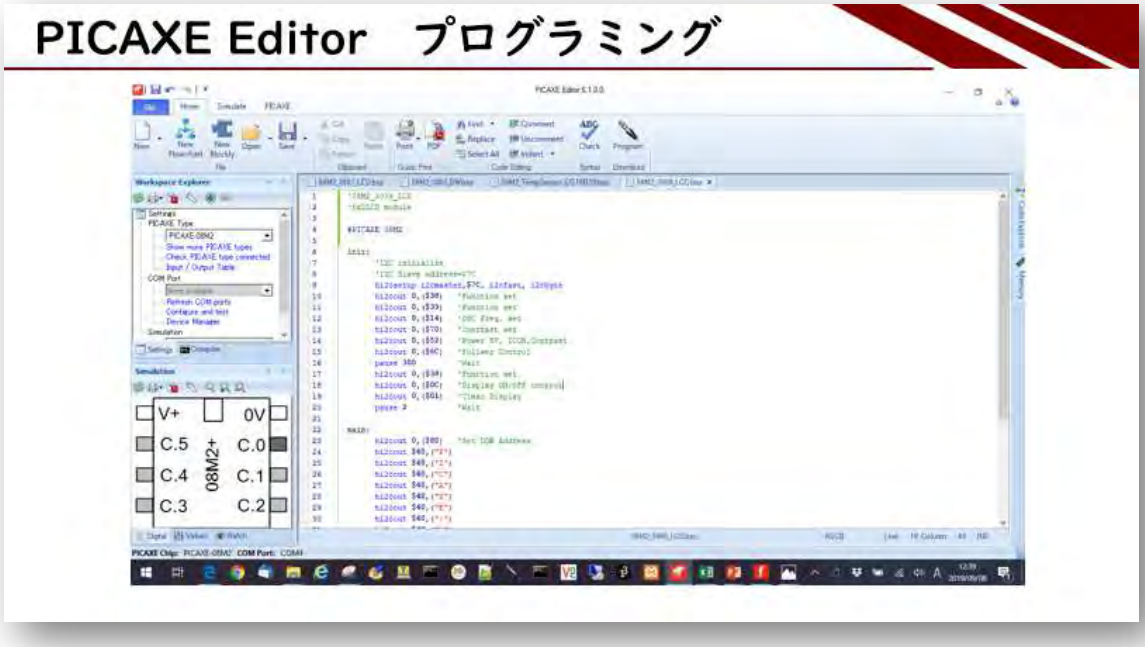


図　197

今回のプログラムは、デジタル温度センサと LCD が合体したものになるので、回路は既にできていましたが、ソースコードは少し長めになりました。

※ソースファイルは実習キット付属の CD に含まれています。(次頁で図中央に示すファイル名)

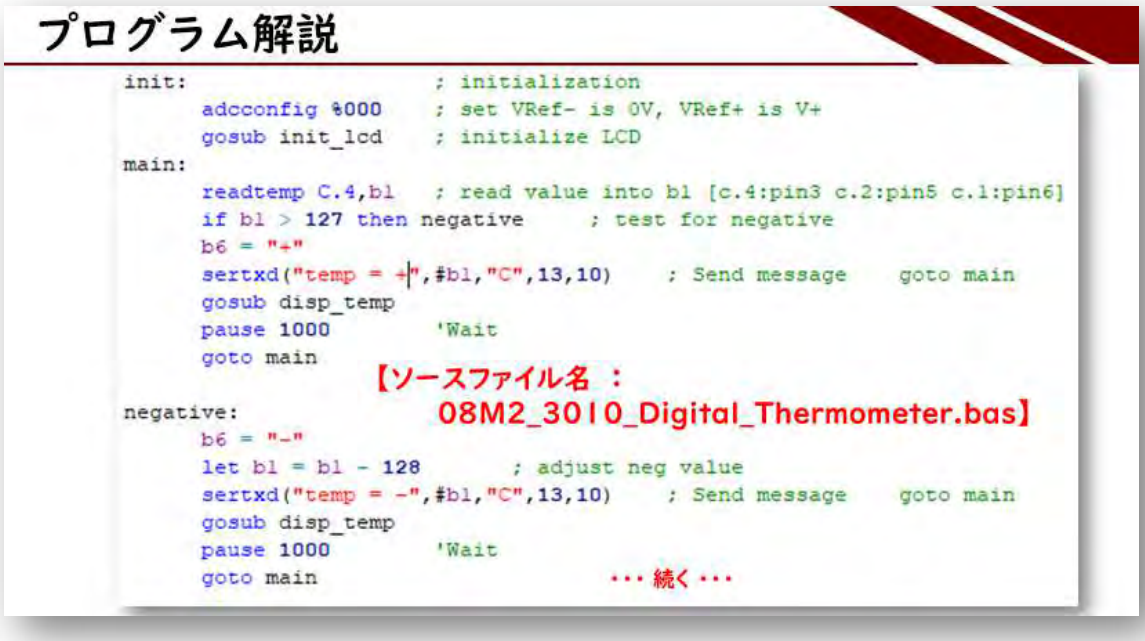


図 198

init:の部分は ADC と LCD の初期化です。main:部分では、まず温度を読み、値の正・負により符号部分の処理を行い、シリアル通信で出力します。次に disp_temp というサブルーチン（次頁で解説）で LCD に表示する文字列を編集し、I2C I/F で出力します。

図中央に、作成済みのソースファイル名を示します。

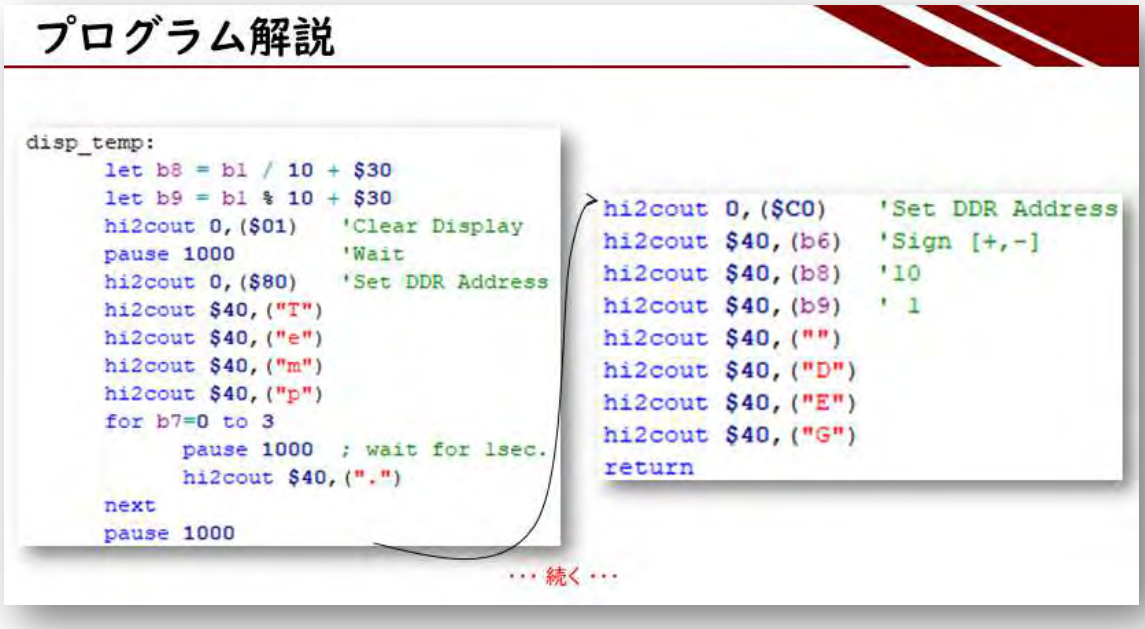


図 199

disp_temp は、変数 b1 に格納されているセンサ出力値を整数部と小数部に分けて演算後、LCD 表示文字列を作り出力しています。表示は、少し工夫してあります。Temp 表示の後に【・】が 1 秒ごとに 1 個ずつ増えて最後に温度＋DEG を表示するようにしています。上手く動いている様に見えるでしょうか。

図 200

init_lcd は、LCD 初期化サブルーチンです。発行しているコマンド群は、LCD のデータシートに沿ったものです。既に LCD の固定文字列表示は成功しているので、問題ないはずです。

今回は少し長めのプログラムになりました。プログラムが出来たら、名前を付けて保存することを忘れない様にしましょう。

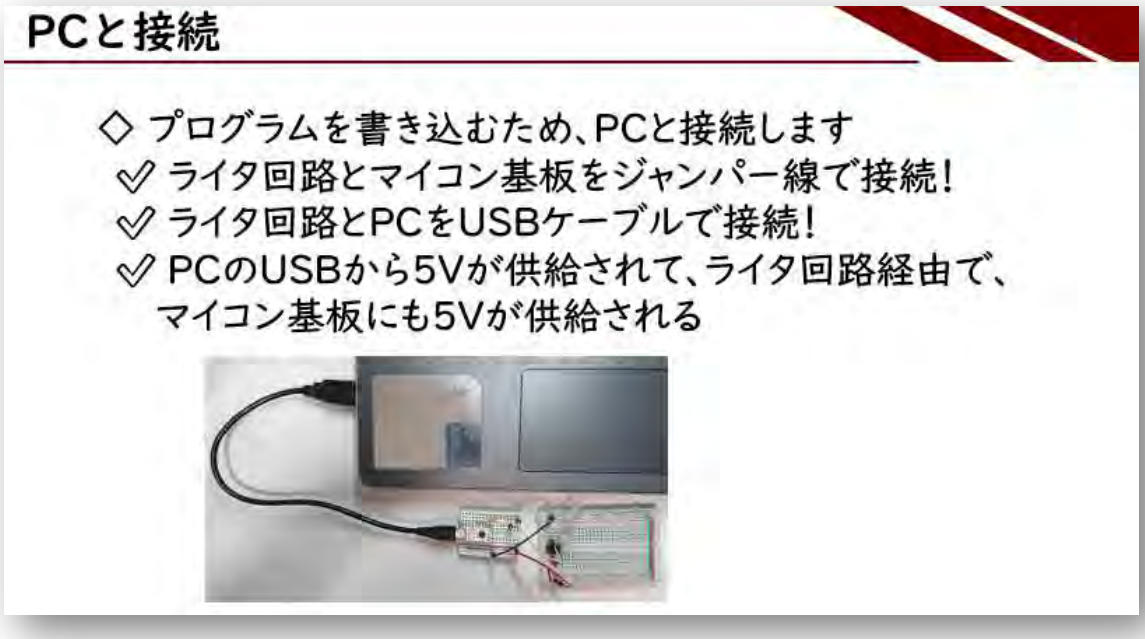


図 201

プログラムが完成したら、コンパイルと書込みを行います。PC、ライタ回路、マイコン回路を接続して下さい。

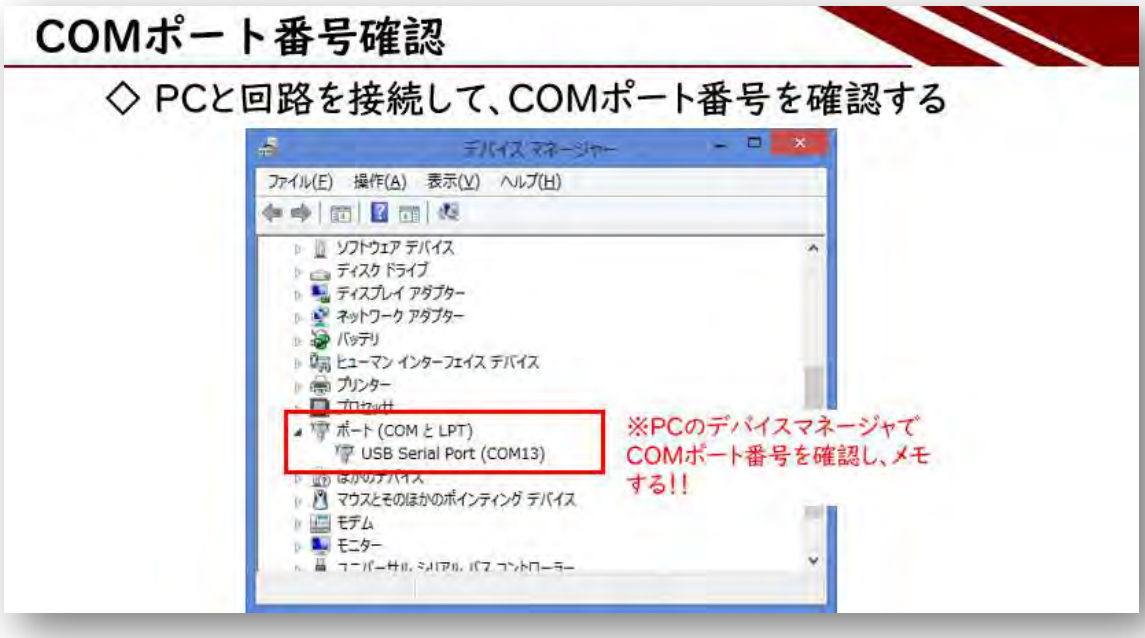


図 202

デバイスマネージャで COM ポート番号を確認します。

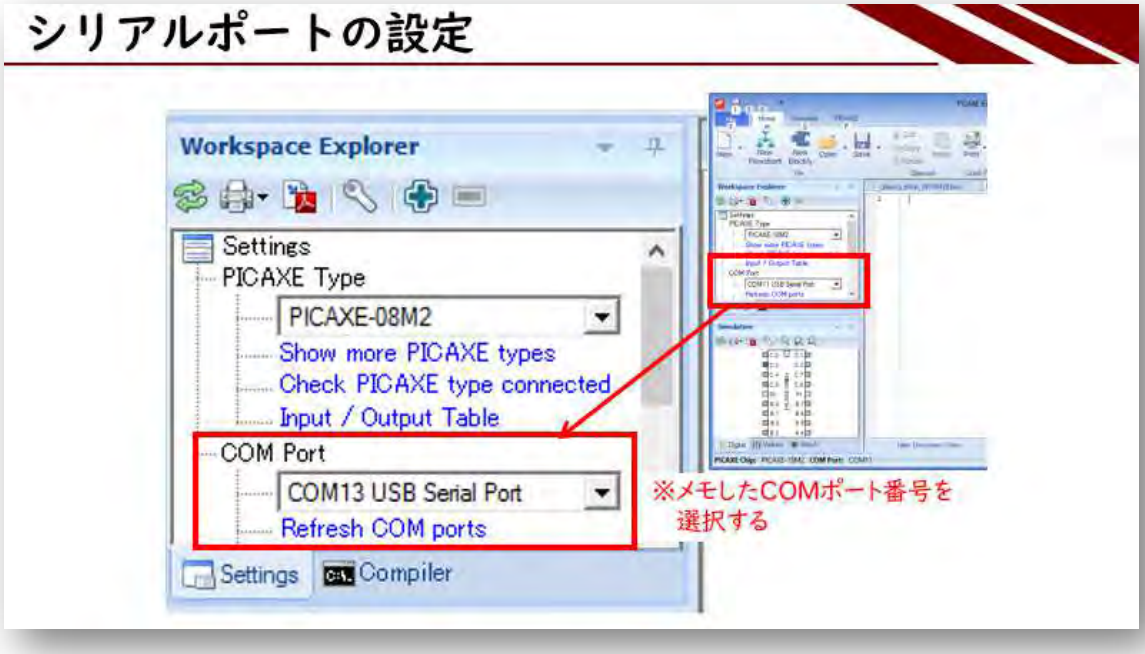


図 203

PICAXE Editor で COM ポート番号を設定します。

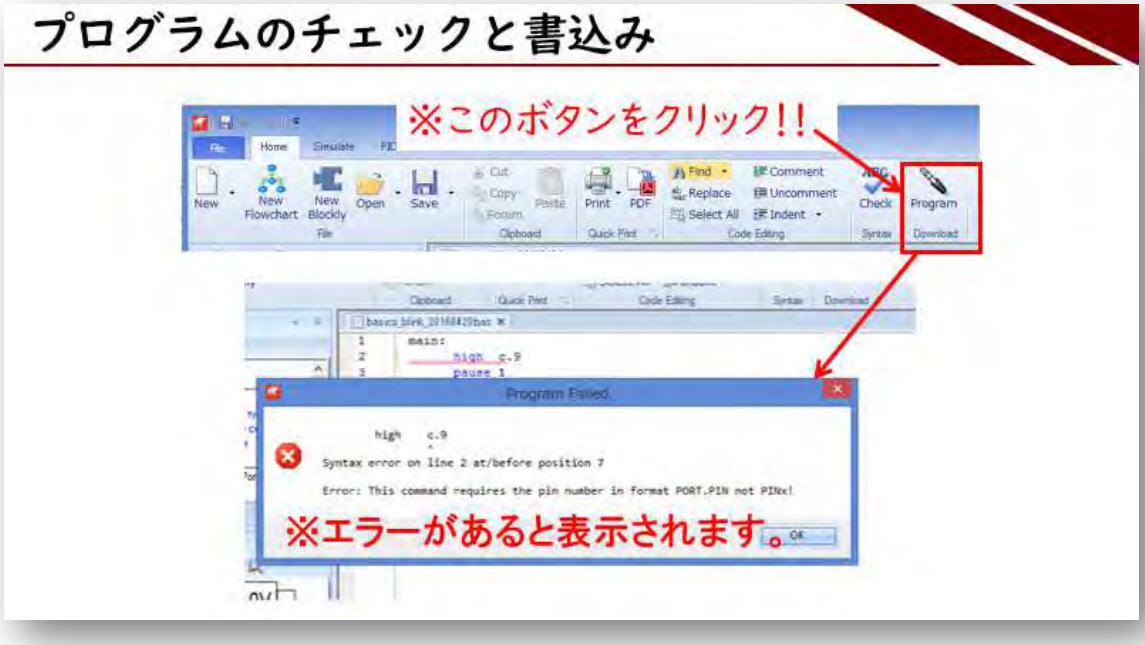


図 204

Home の Program ボタンを押下して、コンパイルとマイコンへの書込みを行います。

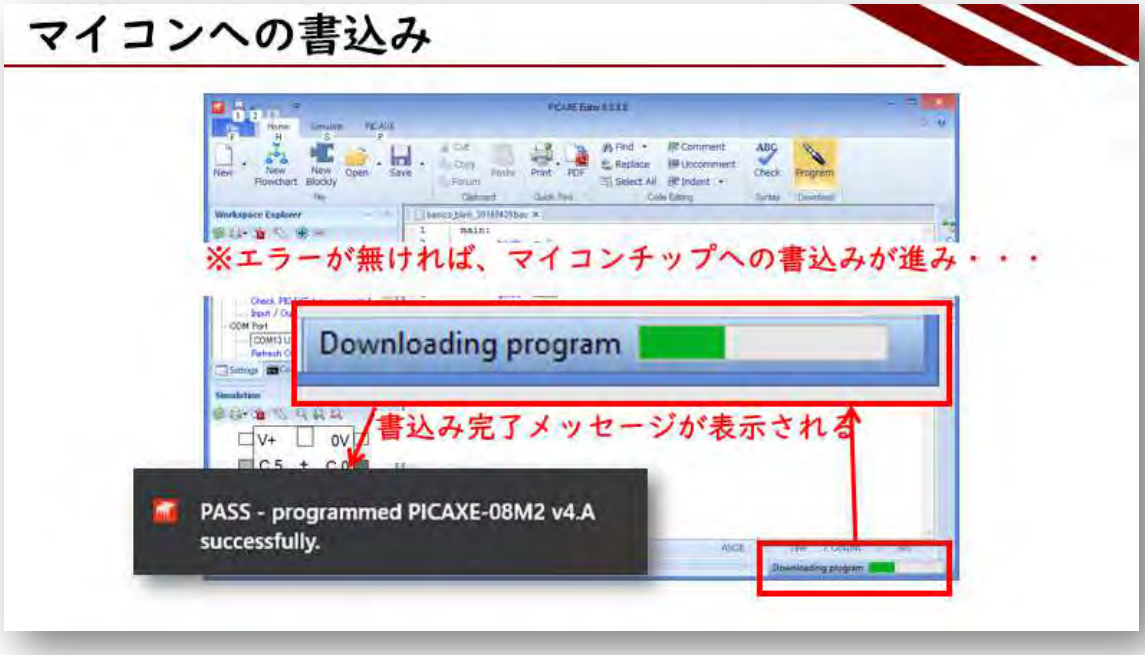


図 205

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されれば書込み終了です。終了と伴に、マイコンが Reset され、書き込んだプログラムの実行が開始されます。

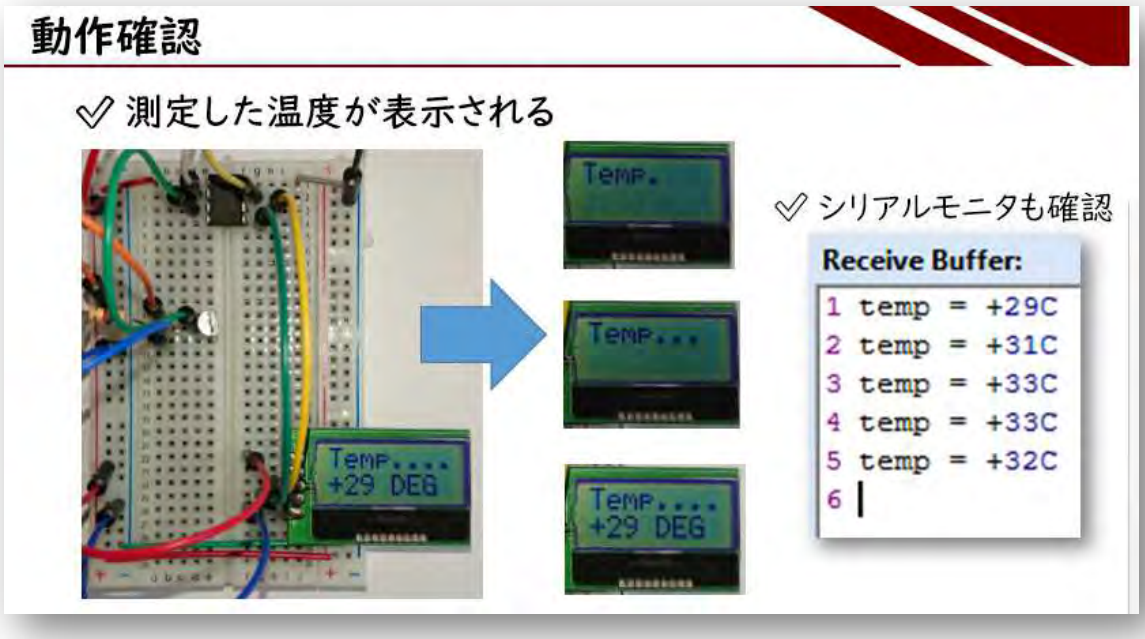


図 206

動作確認では 2 つの事を確認します。シリアルモニタ上に温度が表示されていることを確認しましょう。次に LCD の温度表示がシリアルモニタと一致しているかを確認します。

【温度補正】　補正をする場合は校正用温度計を準備します。その表示温度および LCD 表示の差を記録します。校正温度を変えて何点か行います。記録した校正温度毎の差を見てプログラムに反映します。いくつかの補正方法が考えられますが、計測した温度間の比例計算による補正が容易です。PICAXE は整数計算しかできない事を前提にプログラム化しなければいけません。また、数式は左から順に処理されて、括弧による計算の優先が行われないことも考慮します。

第11回　複数 I/O【IoT の種】

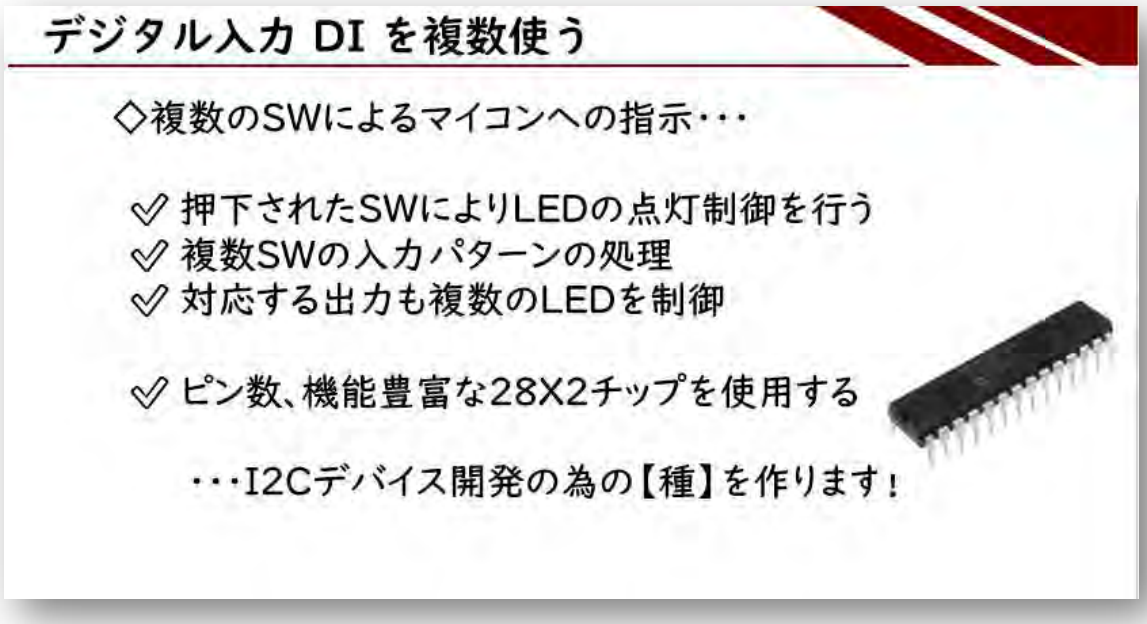


図 207

ここまでは、マイコンの取扱いが初めてでも容易に完成できたと思います。今回からピン数の多い PICAXE(28 ピン)を使い実戦に近い複数の DI・DO に取り組みます。本講座の目的の一つ【I2C デバイス開発】の【種】を作ります。

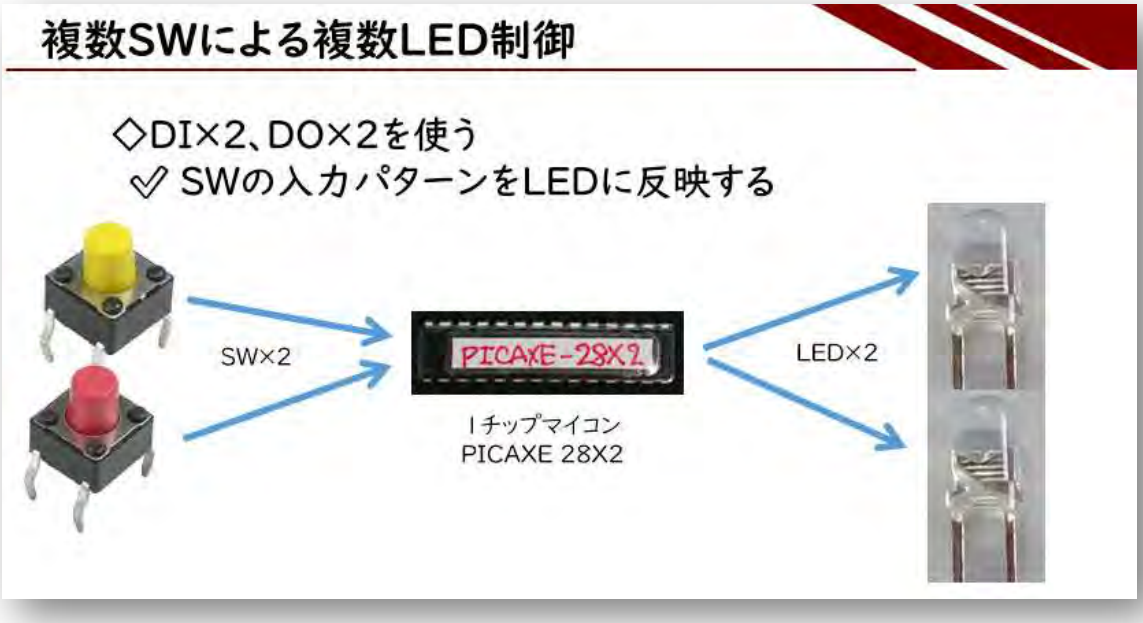


図 208

SW と LED を 2 組使用して、それぞれ関連付けした LED に SW の状態を反映してみます。第 2 回で行った、SW の状態を LED に反映するシステムの拡張版です。

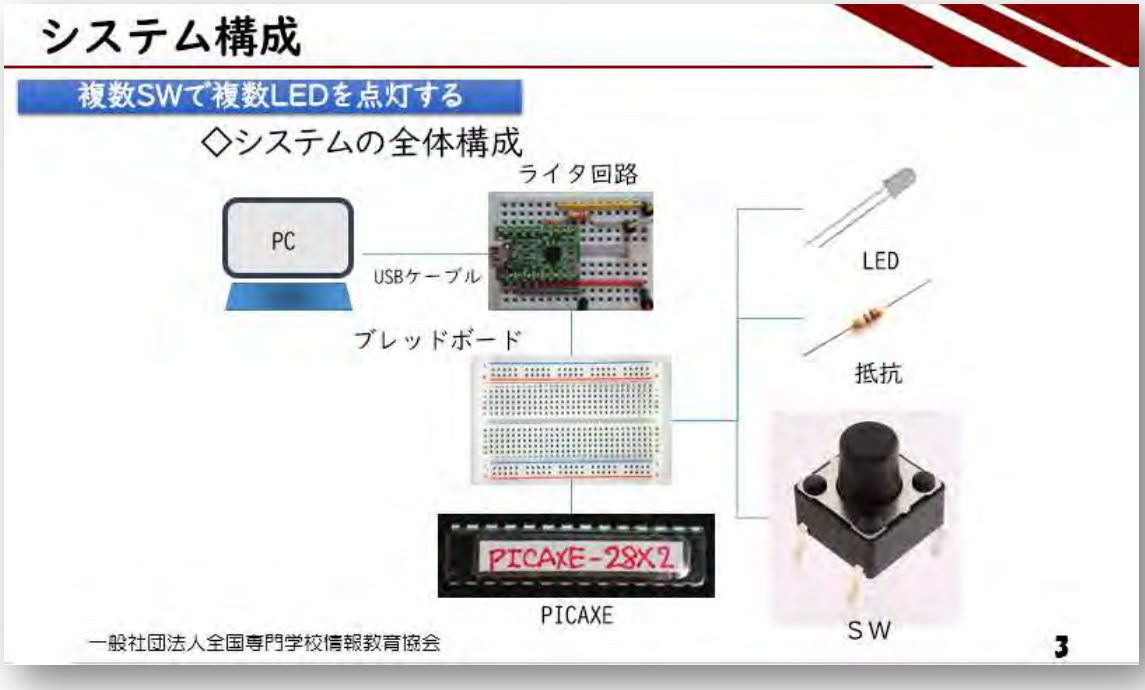


図 209

講座を始めたころに行った実験と使用するパーツが似ていますが、LED、抵抗器、SW が 2 つずつ使用します。マイコンもピン数が多く多数の DI・DO が使える 28 ピンの PICAXE 28X2 を用います。

図 210

PICAXE 28X2 のピン配置は図の様になっています。今回は、電源（19，20 番ピン）とシリアル通信（6、7 番ピン）以外に DI として 11，12 番ピンと、DO に 17，18 番ピンを使います。28 ピンですから、それら全てを使い切るのは大変です。

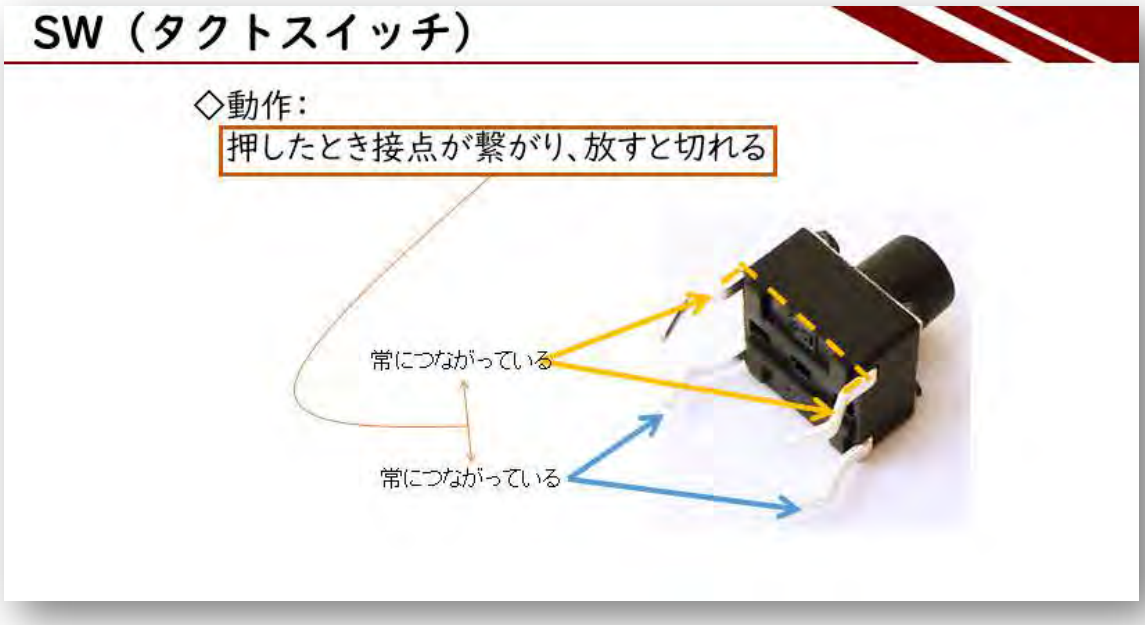


図　211

図は第 2 回で説明したタクト SW です。内部の様子が想像できるようになったでしょうか。湾曲した脚の対がボタンを押下することによって電気的に接続される A 接点を持つ SW です。

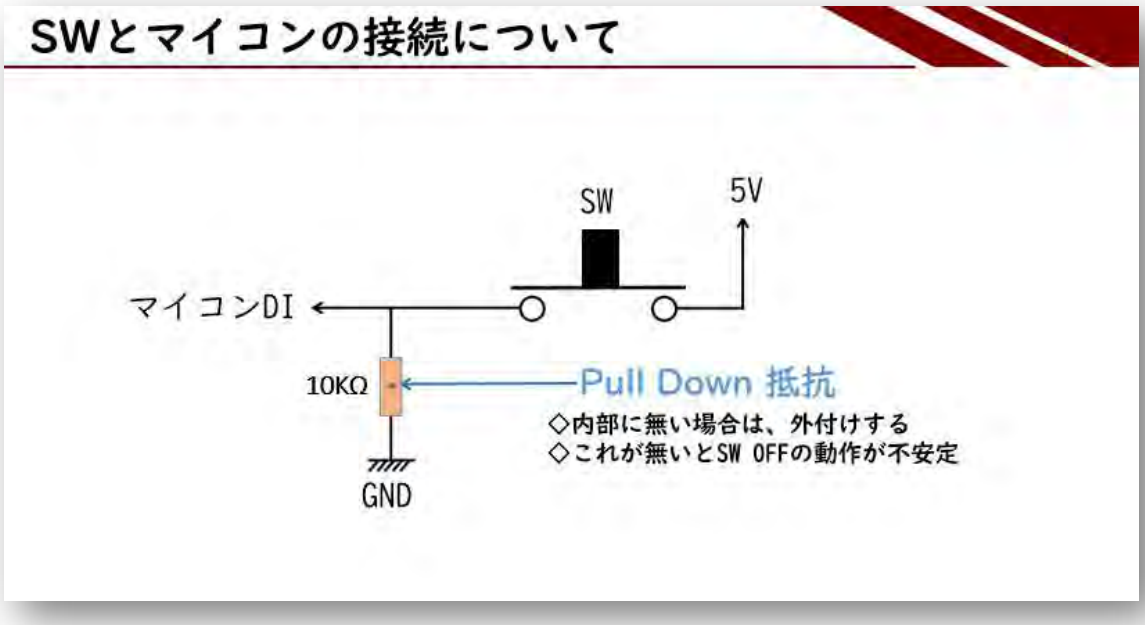


図 212

SW を DI に接続する際には、接点解放時の DI の状態を確実に 0 にして、安定させる目的で Pull Down 抵抗を接続することを説明しました。マイコンの種類によっては、内蔵の Pull Down 抵抗があり、プログラムで設定できるものもあります。残念ながら PICAXE はそのようになっていないので、SW の回路に抵抗を使います。

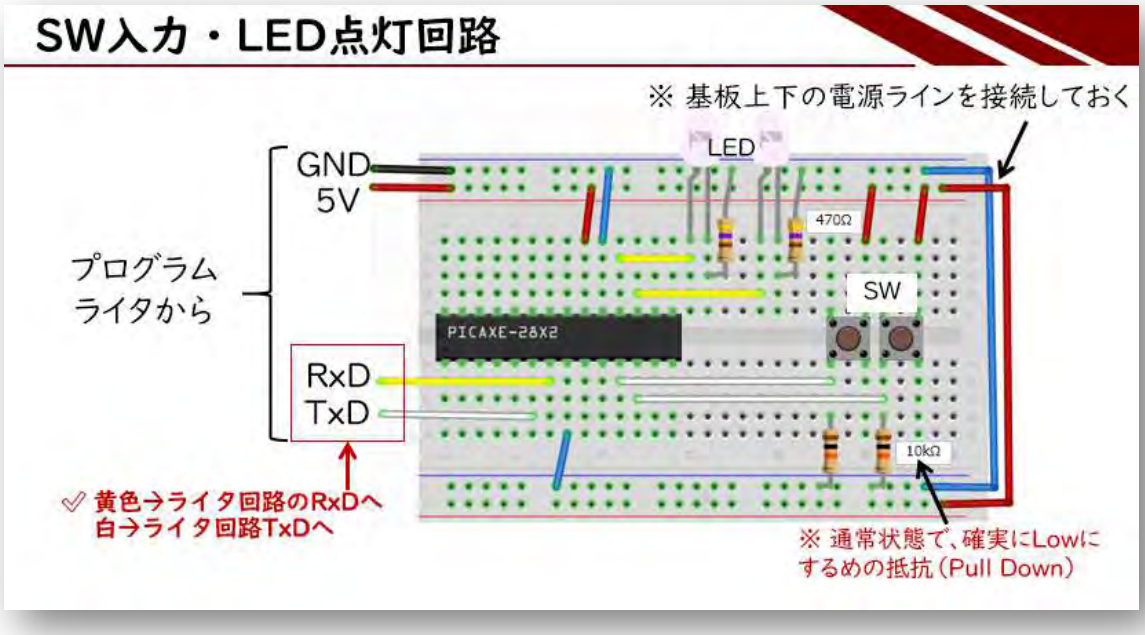


図 213

2 組の SW 入力と LED 点灯回路を図に示します。配線が混雑するので、配線が済んだものを引掛けてしまったりしないように注意しながら回路を作成してください。抵抗器が 2 組ありますが、それぞれの抵抗値が異なるので注意が必要です。図で PICAXE の GND（0V）を接続している配線が 2 本あります。CPU 内部で GND は共通になっていますので、どちらか一方でも動作します。ピン数の多いマイコンでは、外部回路に近くの GND を配線し易いように GND のピンが複数あります。

図 214

実際の回路の様子を図に示します。抵抗器が 2 種類あるので、それぞれの抵抗値を間違えないように選択してください。

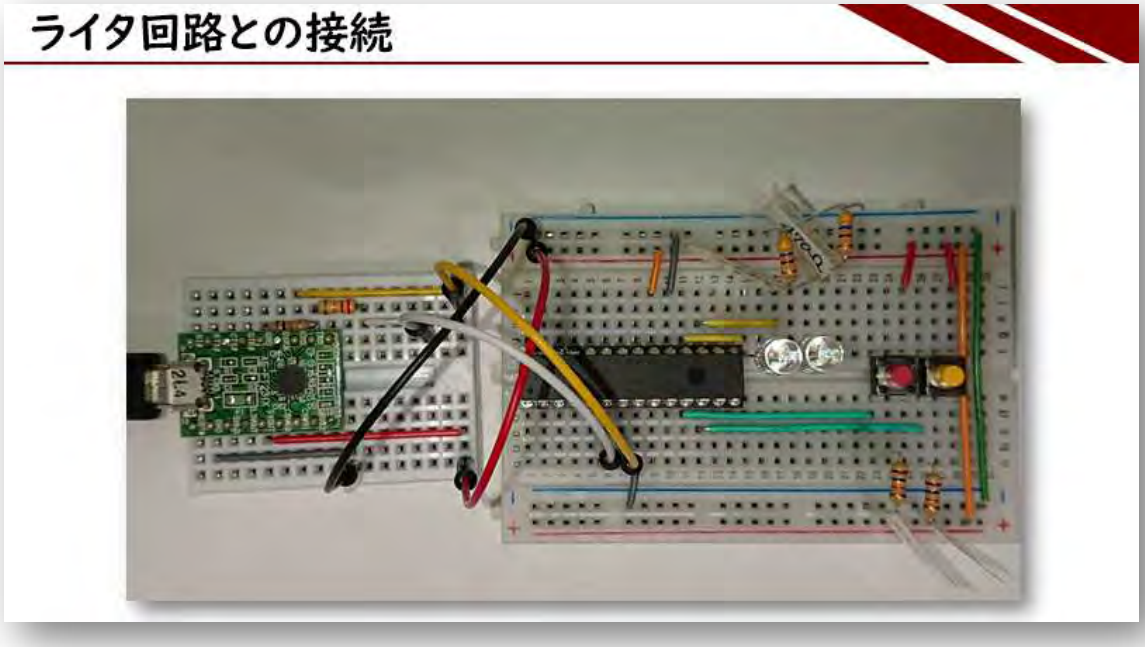


図 215

ライタ回路との接続はこれまのPICAXE08M2とは別のピンですから、間違えていないか確認してください。ライタ回路とマイコンとの接続が窮屈な場合は、長めのジャンパ線を使いましょう。

図　216

PICAXE Editor 左上のウインドウで、対象の PICXE28X2 チップを選択します。

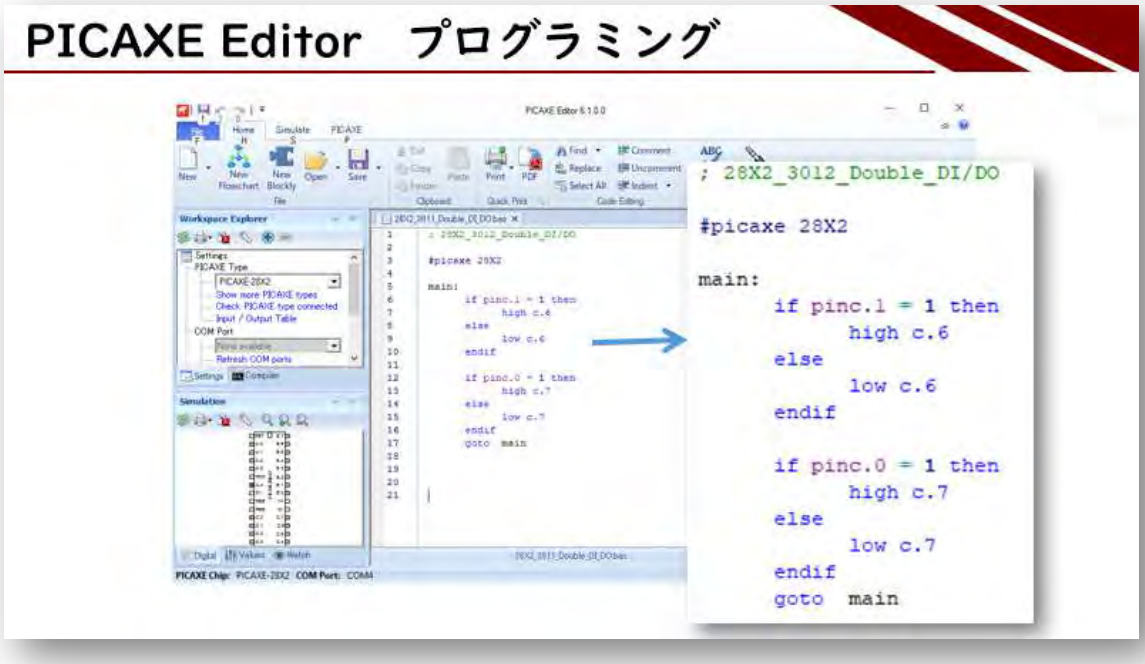


図　217

28 ピンマイコンで初めてのプログラミングです。これまで同様 Home Tab の New で新しいプログラムを作成します。初めから 2 組の DI・DO を用いていますが、練習として各 1 個の SW と LED で始めてから、2 組に増やしても良いと思います。その際は使用するピンに注意して、適宜ソースコードを考えてください。

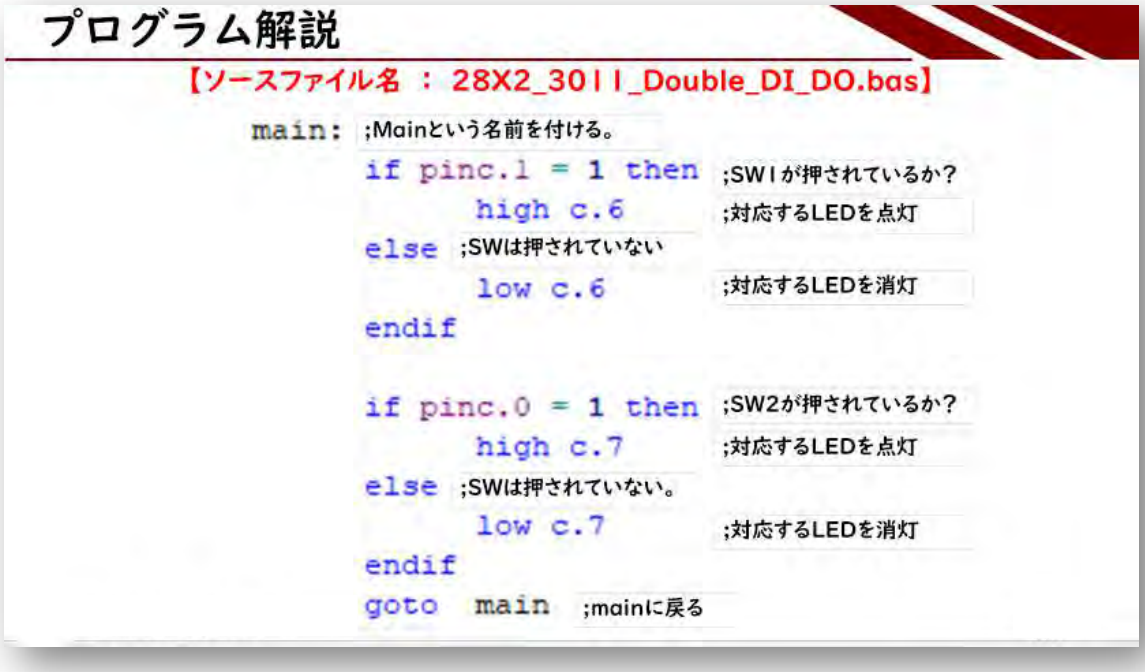


図 218

PICAXE の種類が異なるので、同じピン番号でも対応する内部の番号が異なることに注意してください。SW1 と SW2 それぞれの状態を対応する LED に反映させています。SW1（C.1）が LED1（C.6）に、SW2（C.0）が LED2（C.7）に対応しています。

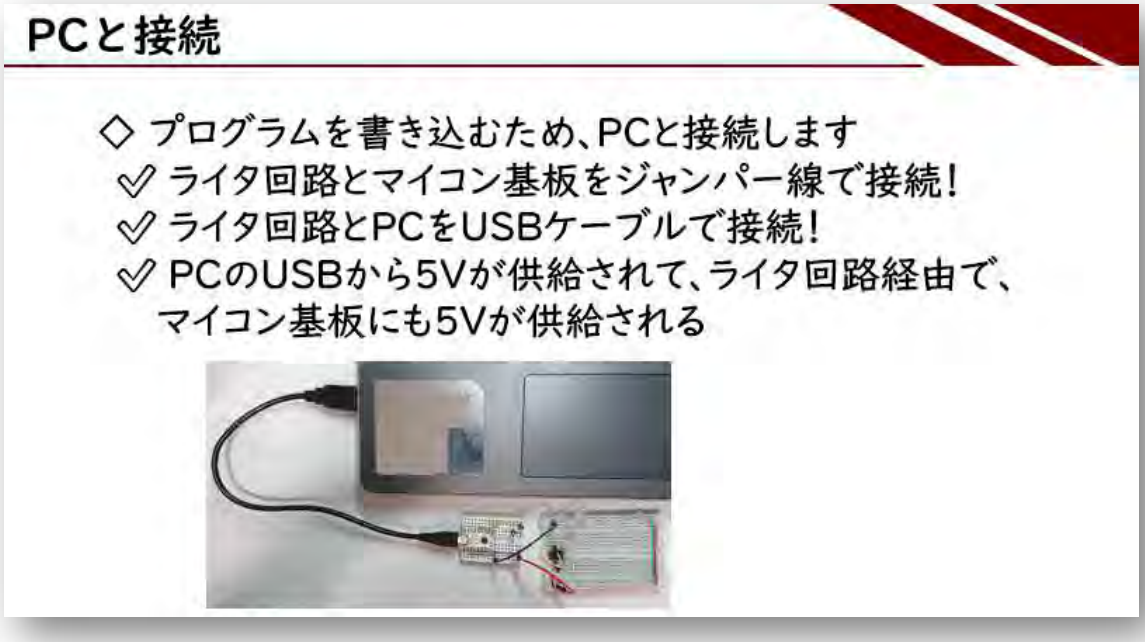


図 219

プログラムが完成したら、コンパイルと書込みを行います。図の手順で PC、ライタ回路、マイコン回路を接続して下さい。

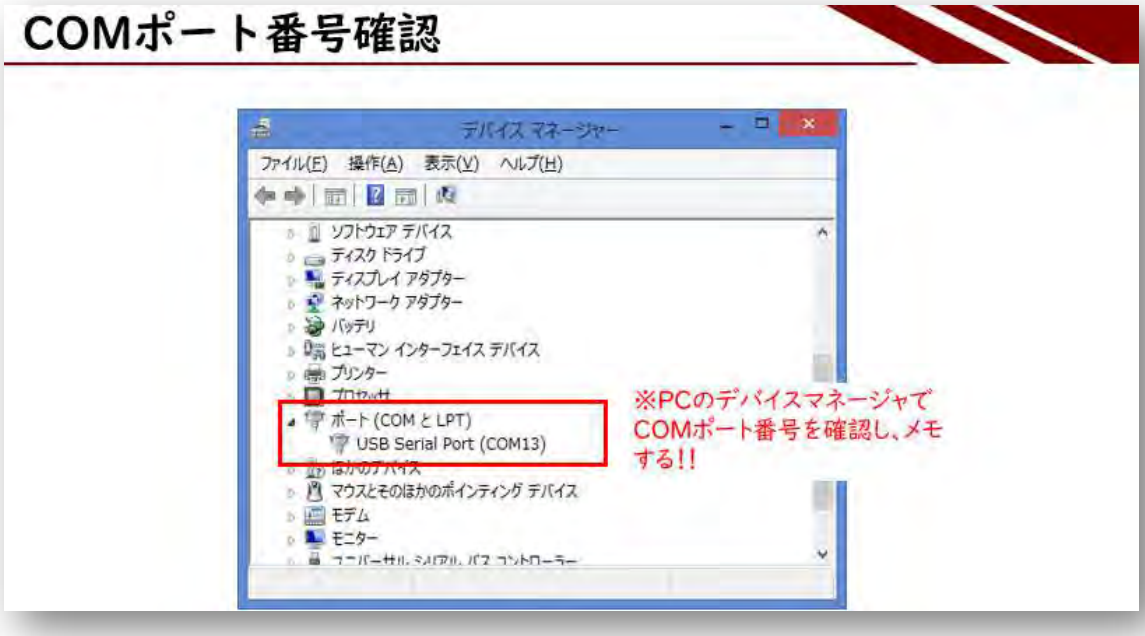


図 220

デバイスマネージャで COM ポート番号を確認します。マイコンの種類が変わっても、ライタが同じであれば COM ポート番号は変わりません。

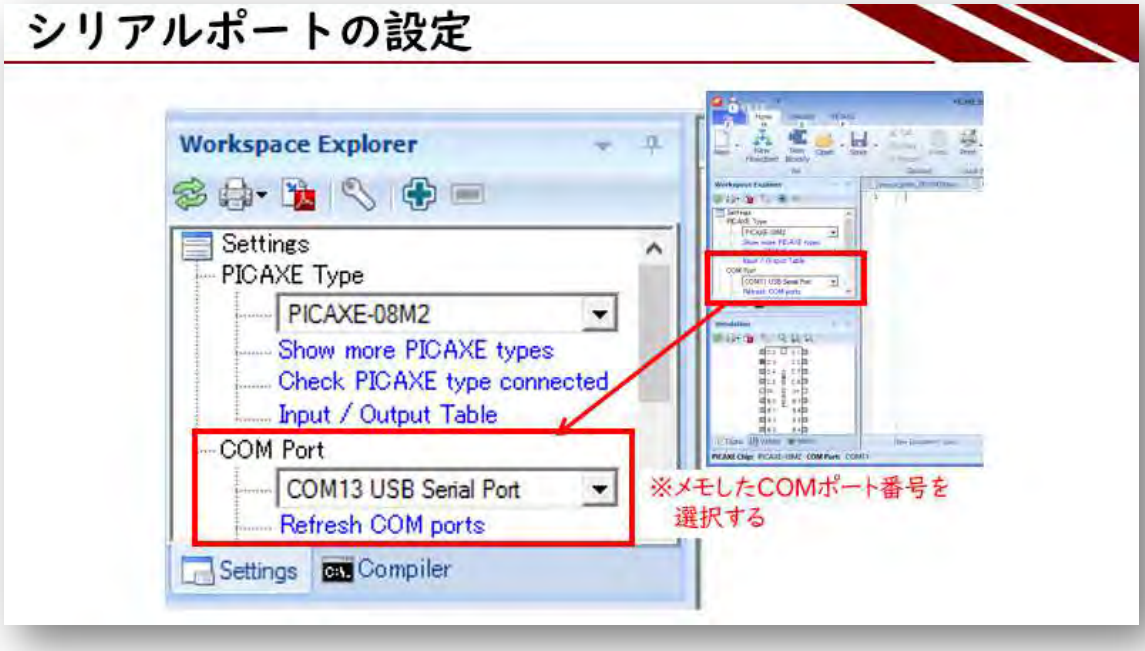


図　221

確認した COM ポート番号を設定します。

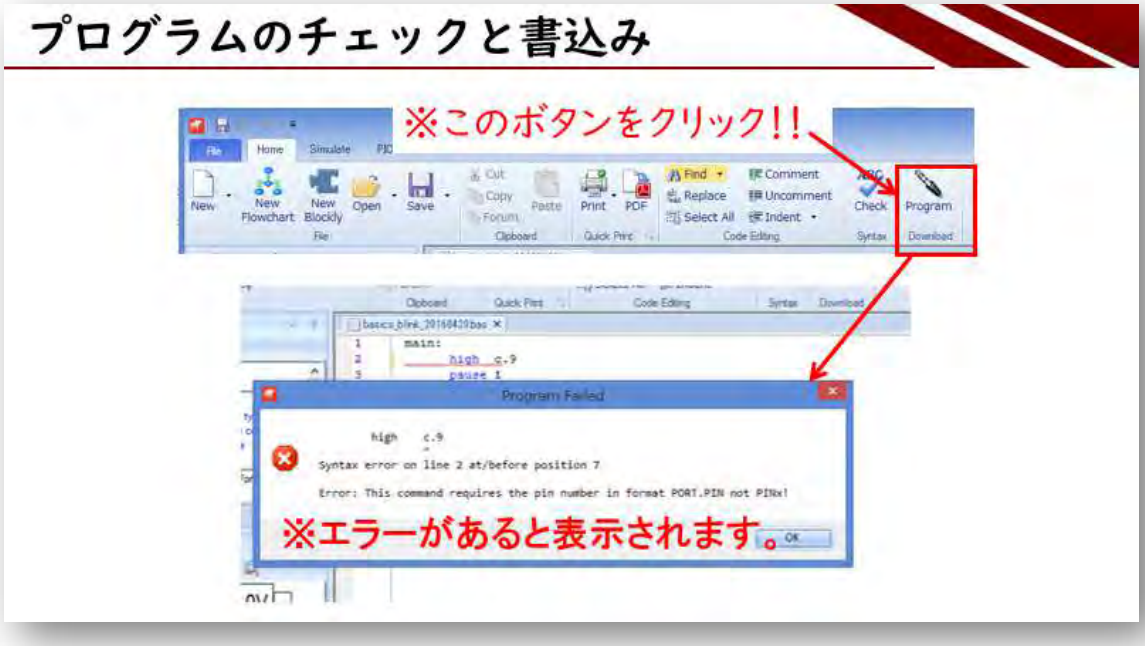


図 222

Homeの Programボタンを押下してコンパイルとマイコンへの書き込みを行います。PICAXEの種類が変わっても、コンパイル等の操作は変わりません。

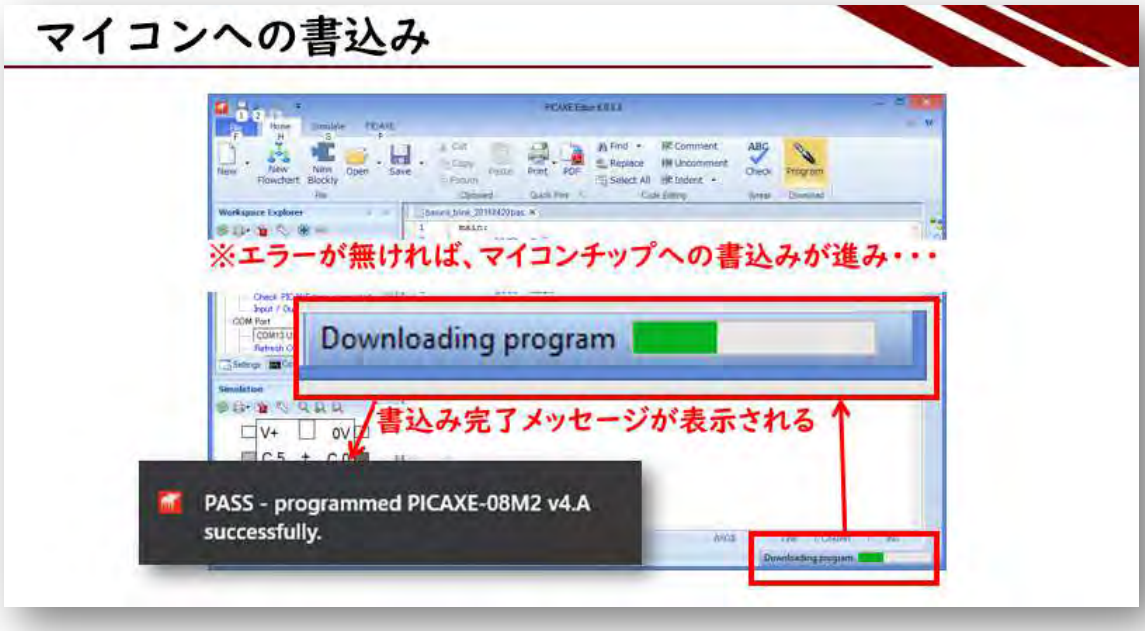


図　223

エラーが無ければ PICAXE Editor の右下にある Download program という表示の緑色バーが右に進み、マイコン内部にプログラムを書込みます。図の様な PASS－＊＊＊のメッセージが表示されると書込み終了です。書込み終了後、マイコンが Reset され、書き込んだプログラムが実行されます。

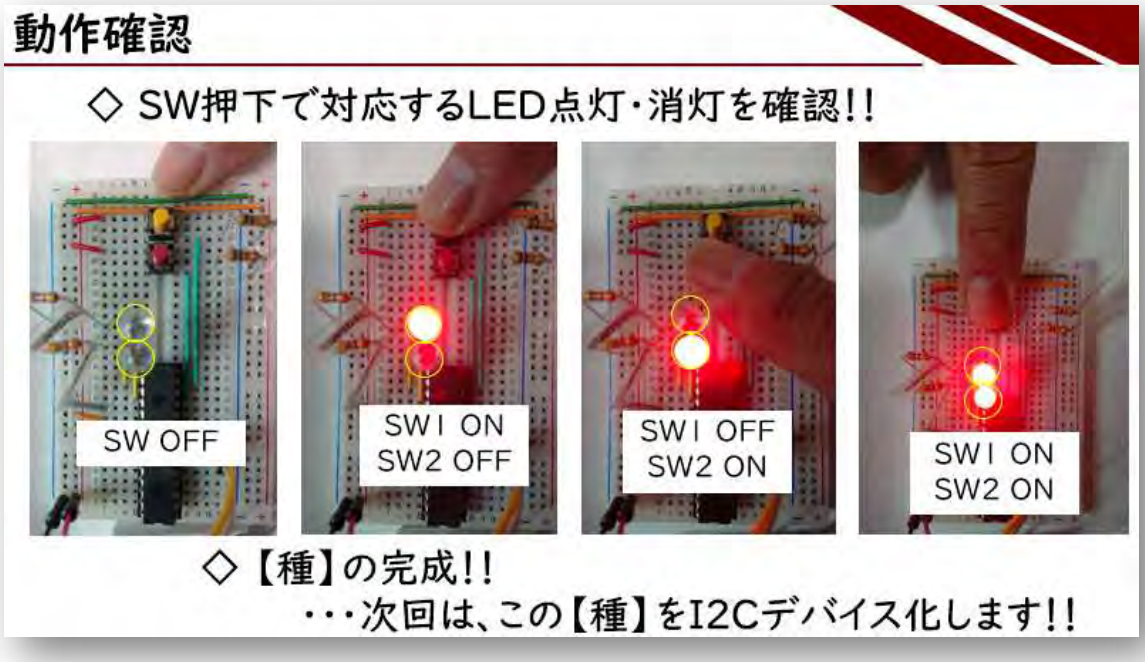


図 224

動作確認は、複数の確認事項があります。通常状態では 2 つの LED は消灯しています。SW 1 を押下すると対応する LED1 が点灯します。同様に SW2 を押下すると LED2 が点灯します。2 つの SW はバラバラに押下しても、対応する LED が点灯するか確認します。プログラムでは、1 組目、2 組目と順番に処理をしているのですが、目に見える様子では各々の組が独立に SW 状態を見ているかのように動いています。もし、もっと多くの SW と LED を配置してみたらどうなるでしょうか。空いている信号を使用して沢山の組を配線して実験してみて下さい。

【割込み】組込み系システムでは、多くの SW 状態をそれぞれシステムに反映させるために割込みを使用することもあります。

第12回　I2Cデバイス開発

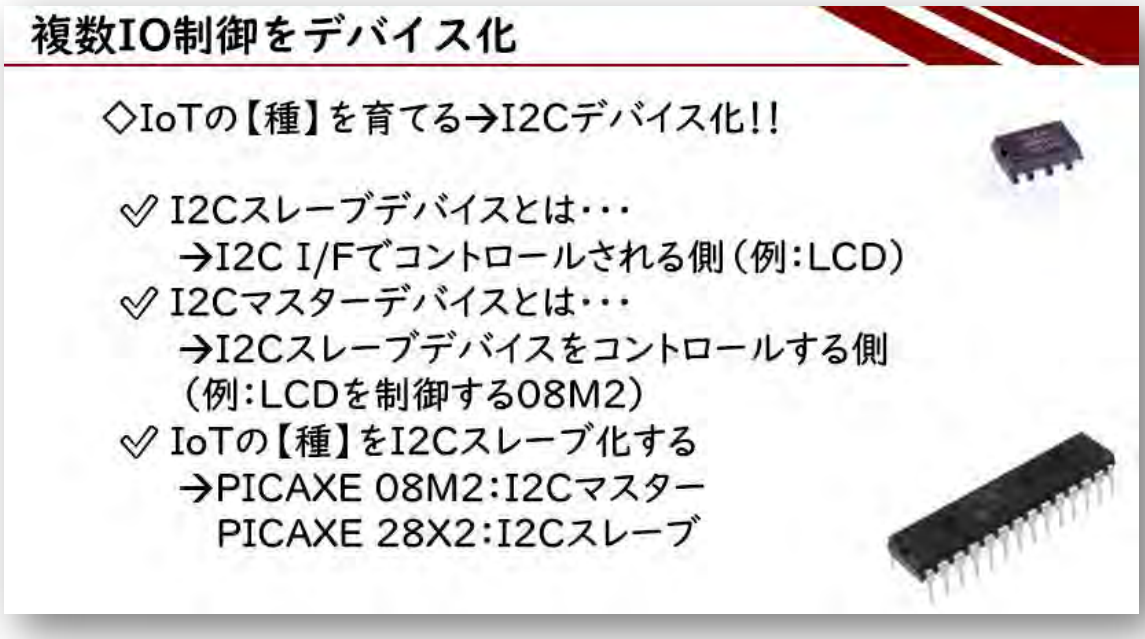


図 225

第 11 回で作った【IoT の種】を育てます。PICAXE が備えている I2C 機能を用いて I2C スレーブデバイス開発を行います。【IoT の種】の I2C スレーブ化です。スレーブデバイスに対するマスタデバイスも必要です。PICXE08M2 をマスタ、PICAXE28X2 をスレーブとして 2 つのマイコンが連携して動くシステムを開発します。

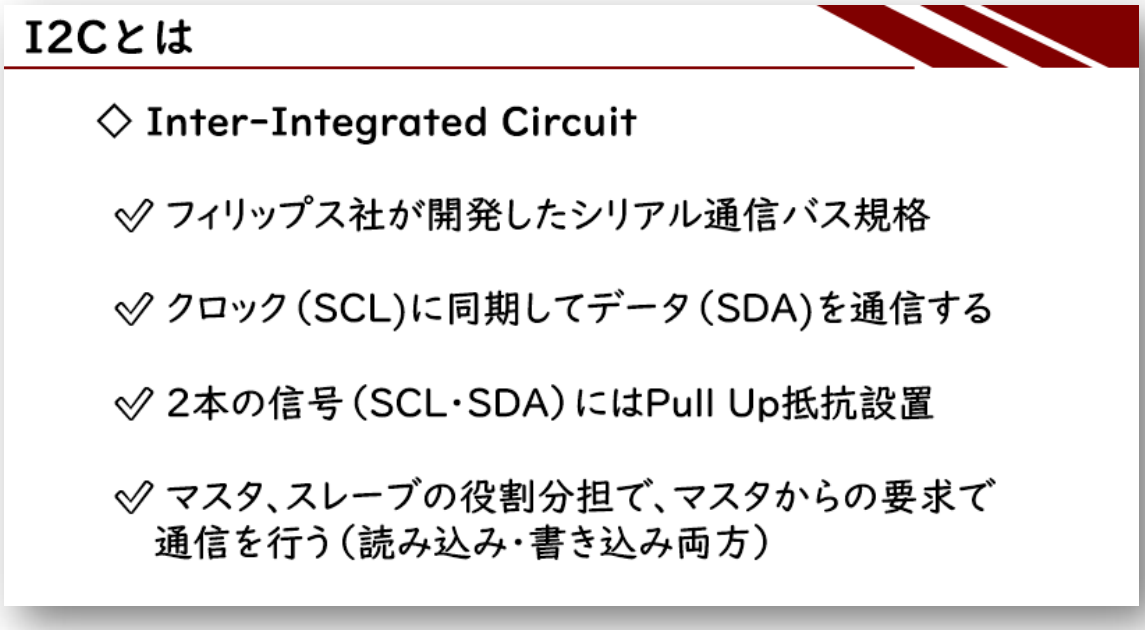


図 226

I2C とは、Inter-Integrated-Circuit の頭文字を採り IIC ということで、これを I2C と表現しています。私はアイ・ツー・シーと読んでいますが、アイ・スクエアド・シーと読む方もいます。フィリップス社がマイコンと周辺デバイスとの間の通信向けに開発した規格で、センサなどに広く用いられていますが、本講座でもすでに LCD で I2C I/F を使用しました。2 本の信号（SCL,SDA）（I2C BUS）で通信を行い、複数のデバイスをこの信号に接続することができるので、配線が少なくなるというメリットがあります。マスタとスレーブという役割があり、通常マイコンがマスタ、周辺デバイスがスレーブになります。I2C I/F を複数持つマイコンもあります。

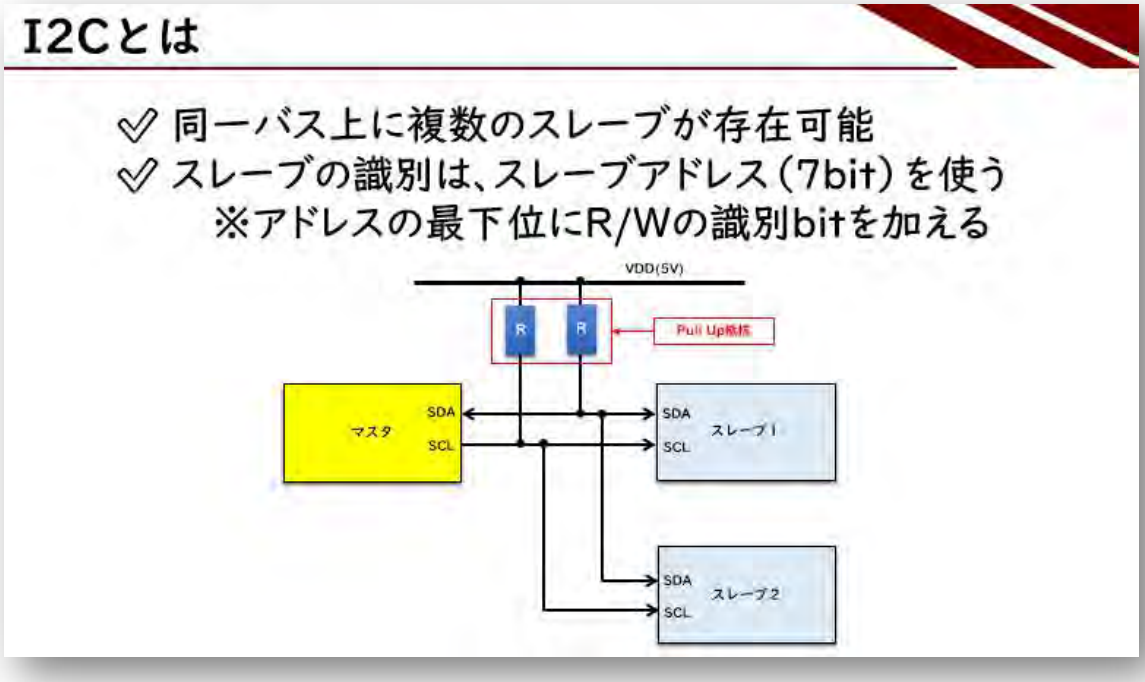


図　227

マスタが同じ I2C BUS にあるスレーブデバイスを識別するために、スレーブアドレスを用います。スレーブアドレス（7bit）の最下位にスレーブに対する書込/読込の識別 bit を付けて 8bit でリクエストを出すことで通信が始まります。データシートにはスレーブアドレスが記載されています。通常 7bit で記載されていますが、識別 bit を 0 にした 8bit 記載のものがあるので注意が必要です。このように、I2C I/F はマスタ主導で通信が始まり、その通信中は他のデバイスは I2C BUS を利用できません。

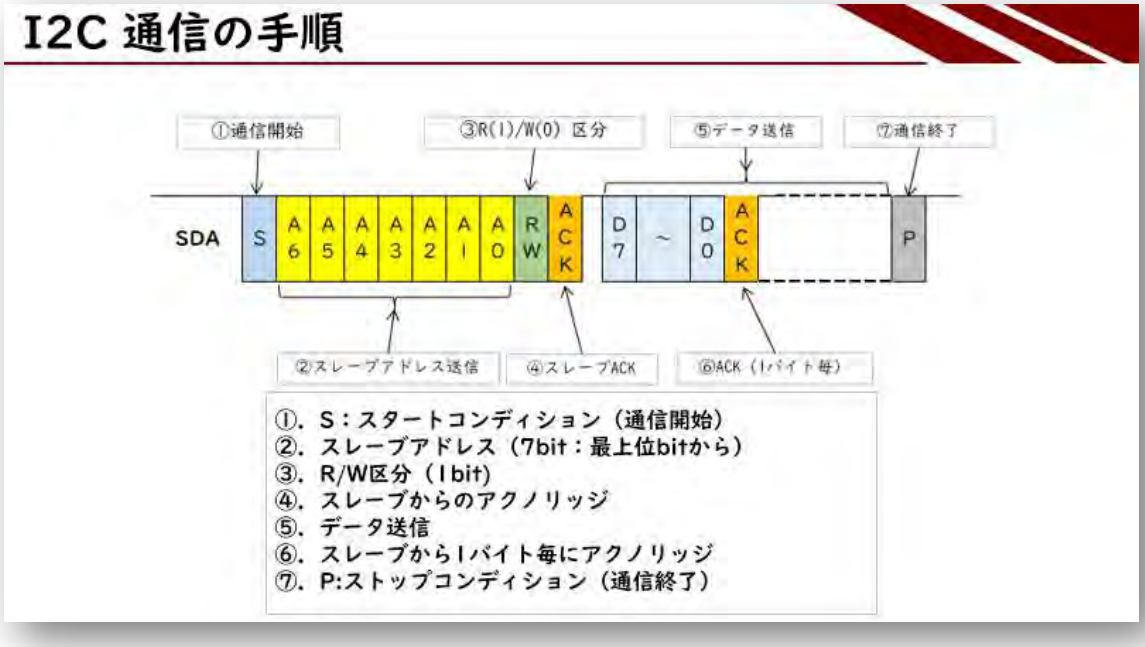


図 228

図は 1 組の単純な I2C 通信です。マスタが通信開始のスタートコンディションに続けてスレーブアドレスと RW の識別 bit を SDA に出力すると、該当するアドレスのデバイスが ACK を返します。その後 RW データが 8bit 出力され、そのデータを取り込んだ側が ACK を出力します。マスタがストップコンディションを出力して、1 組の通信が終了します。

図 229

図の③で示す RW の識別 bit でデータと、それに続く ACK の送信方向が変わります。I2C はあくまでもマスタ主導の通信なので、スタートコンディションとスレーブアドレス＋RW、及びストップコンディションはマスタが出力します。スレーブデバイスは常に I2C I/F を監視して、自分のアドレスが SDA に現れたら、ACK を返して通信状態に入ります。

図 230

I2C 通信の詳細を理解しました。今回は、PICAXE を用いて I2C のスレーブデバイスを開発します。前回の複数 DI・DO を持つシステムを I2C スレーブデバイスへと変身させます。I2C ではマスタが必用ですから、PICAXE08M2 をマスタとして開発します。マスタは DI に SW を用い、その状態を I2C でスレーブに伝え、スレーブは受診したデータを元に DO に接続した 2 つの LED 点灯制御を行います。I2C のマスタ機能を持つマイコンは多いのですが、スレーブ機能を持つマイコンはあまりありません。PICAXE28X2 はスレーブ機能を持つマイコンの中では利用し易いものです。今回は 2 種類の PICAXE を使用する開発で難易度が上がります。PICAXE の開発には慣れてきたところですが、**【慣れによる間違いをしない様に慎重に開発を進め】**て、一発でシステムを完成させて下さい。

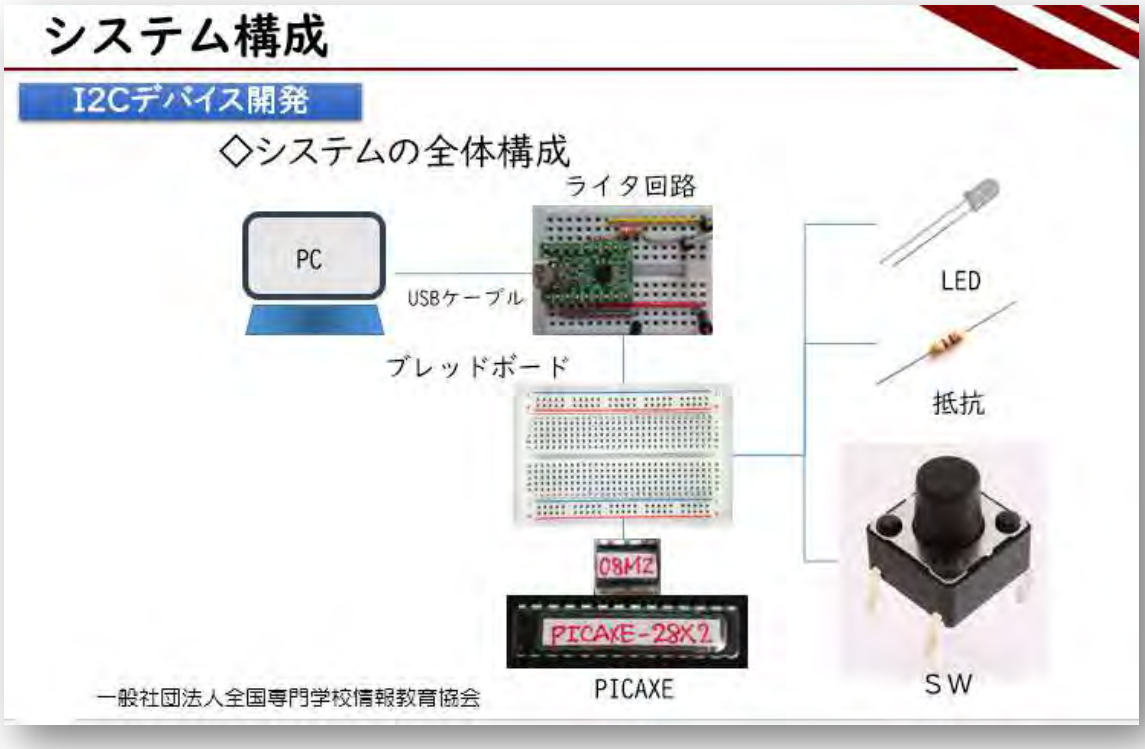


図　231

使用するパーツを図に示します。これまでに使用してきたものですが、CPU は 2 種類になっています。

図 232

今回は I2C マスタの役割を担う PICAXE08M2 は 8 本のピンがありますが、今回から全てのピンを使用することになります。それぞれのピンには複数の機能が割り当てられているので、使用する機能を□で囲んであります。

図 233

PICAXE28X2 にはたくさんのピンがあります。利用するピンは前回の複数 DI・DO とほぼおなじですが、DI の代わりに hi2c scl と hi2c sda を用います。使用する機能を□で囲んであります。

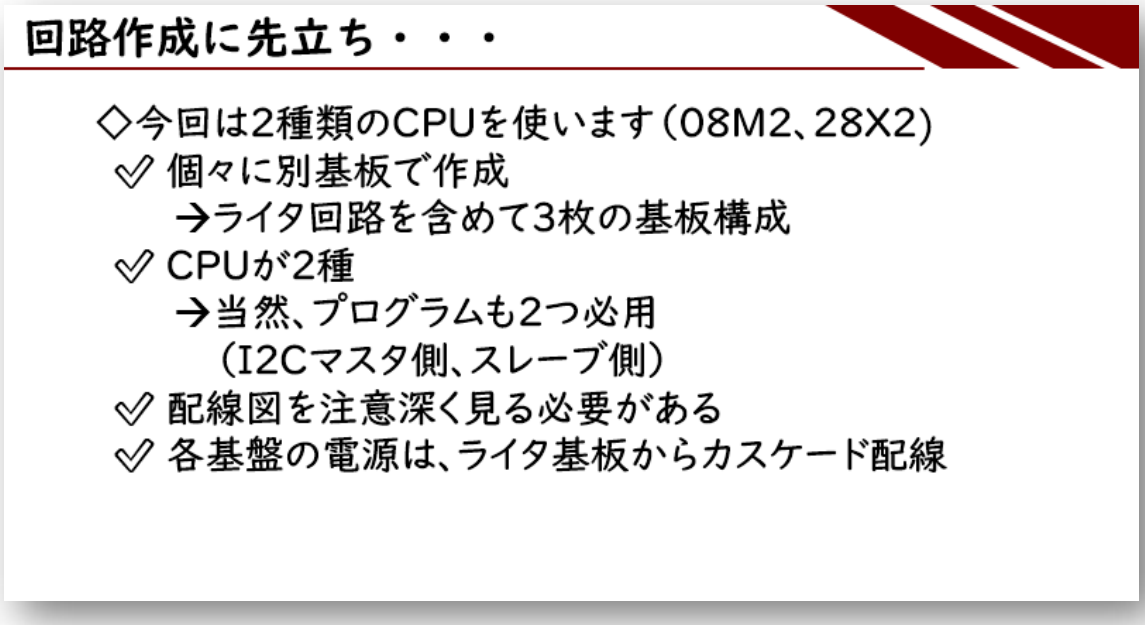


図 234

回路を作成する前に、全体を整理しておきます。今回使用するのは PICAXE08M2（I2C マスタ）、PICAXE28X2（I2C スレーブ）の 2 種類です。CPU 毎に回路を分けて別基板で作成します。ライタ回路もありますから、基板は全部で 3 枚になります。電源の大元は PC の USB からライタ回路に与えられる 5V をします。ライタ回路を経由して 1 枚目の CPU 基板へ、そして 2 枚目の CPU 基板へと、カスケード接続します。

図 235

新しいマイコンや IC の脚は、製造過程で脚が広がって出来上がります。図は横から見た様子です。この脚の広がりを右の様に成型して、基板に挿すようにします。特に 28 ピンの場合は、成型をしないとブレッドボードに挿し込むことができません。この広がりを直さずにブレッドボードに挿し込むと、脚が大きく曲がってしまい、場合によっては折れてしまうことも有りますので、注意して取り扱ってください。ブレッドボードに挿し込んだ後は、横から見てチップのお腹がブレッドボードに並行で近くなるよう、押し込んでください。ブレッドボードから取る外す場合は、ピンセットなどで少しずつ浮かして取り外します。

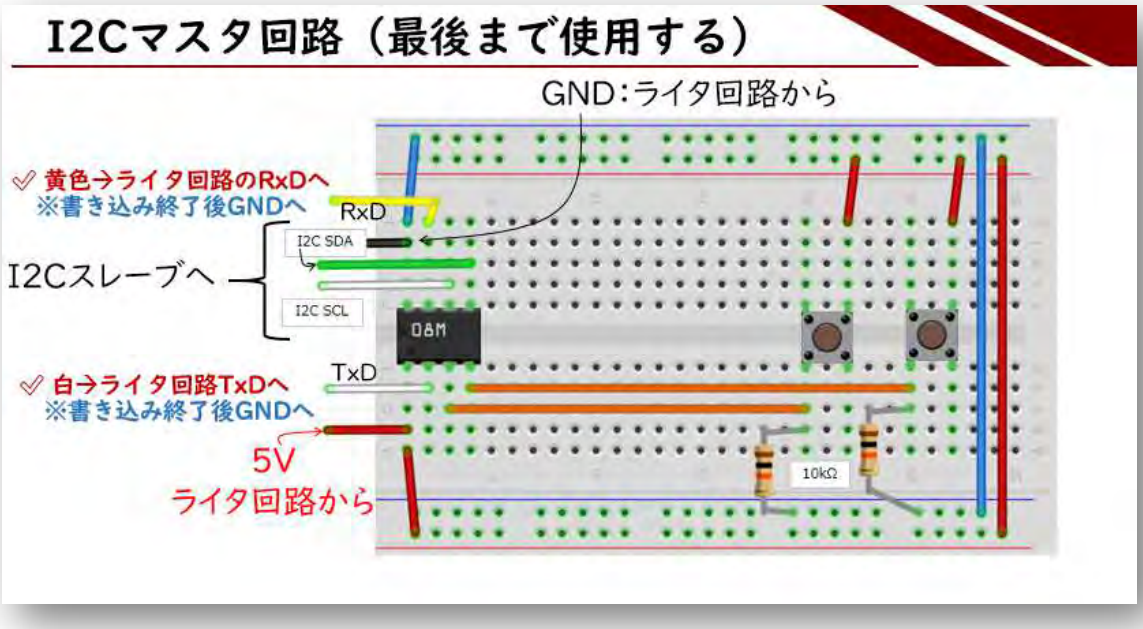


図 236

電源の大元はライタ回路ですが、システム全体を接続する際に I2C スレーブ基板から配線します（実際の回路を参照）。この回路の配線では最大の注意が必要です。SW やプルダウン抵抗の配線は図に従い行って下さい。TxD、RxD は下記の注意に従って配線してください。**書込時と動作時の配線が異なります。**

【注意】TxD と RxD の配線は注意が必要です。マイコンプログラム書き込みの際は TxD、RxD 各々をライタ回路の該当するピン（図に記載）に接続して下さい。書込み終了後は GND に接続して下さい。

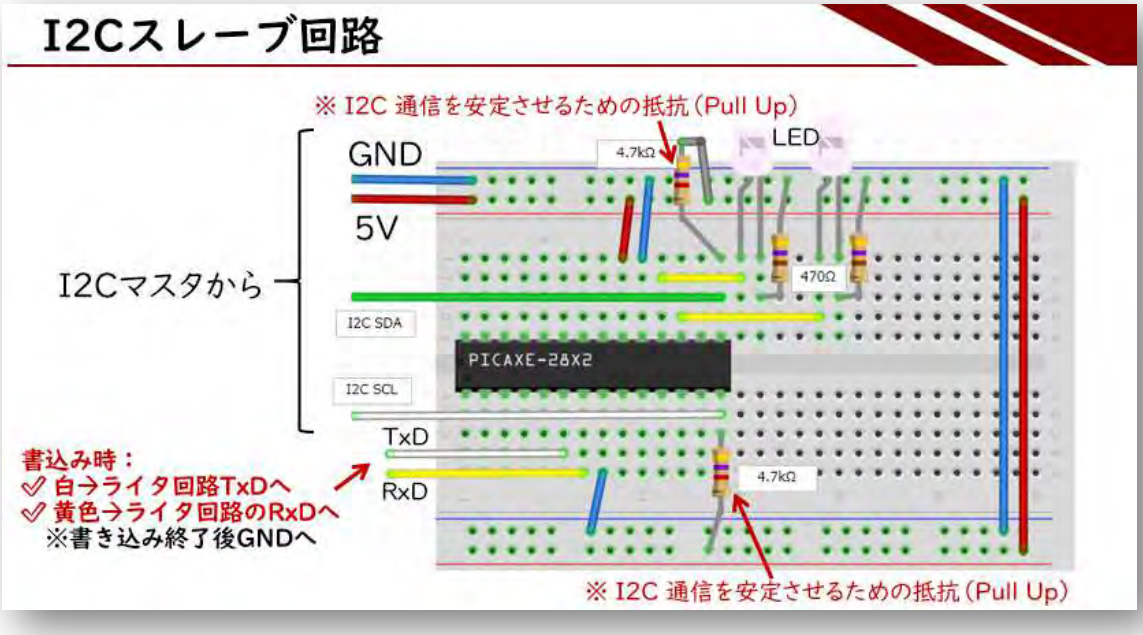


図 237

電源はライタ回路から配線します。SCL と SDA は I2C マスタ（PICAXE08M2）の該当するピンに配線します。SCL、SDA 各々は、4.7kΩで Pull Up します。プルアップ抵抗は I2C の通信を安定させるために使用します。通常状態で High の状態に信号線を保つことが目的です。4.7kΩという抵抗値は PICAXE のマニュアルに記載の値です。

【注意】TxD と RxD の配線は注意が必要です。マイコンプログラム書き込みの際は TxD、RxD 各々をライタ回路の該当するピン（図に記載）に接続して下さい。書込み終了後は GND に接続して下さい。

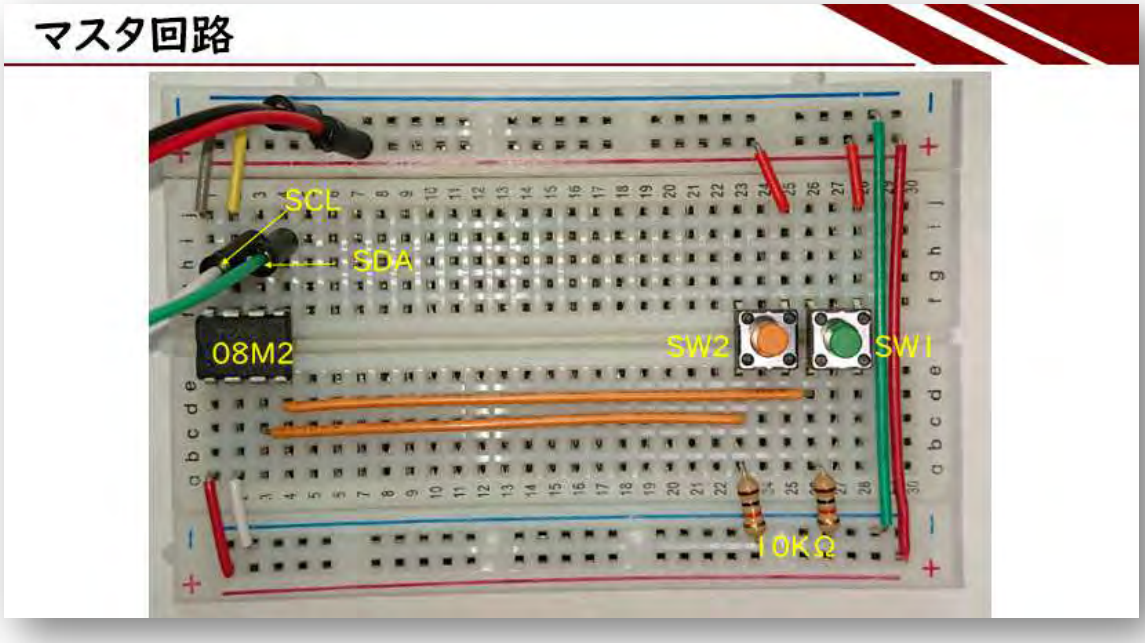


図 238

電源は I2C スレーブ（PICAXE28X2）から配線します。SCL と SDA は、I2C スレーブの該当するピンから配線します。SW の Pull Down 抵抗は 10kΩ です。

【注意】TxD と RxD の配線は注意が必要です。マイコンプログラム書き込みの際は TxD、RxD 各々をライタ回路の該当するピン（図に記載）に接続して下さい。**書込み終了後は GND に接続して下さい。**

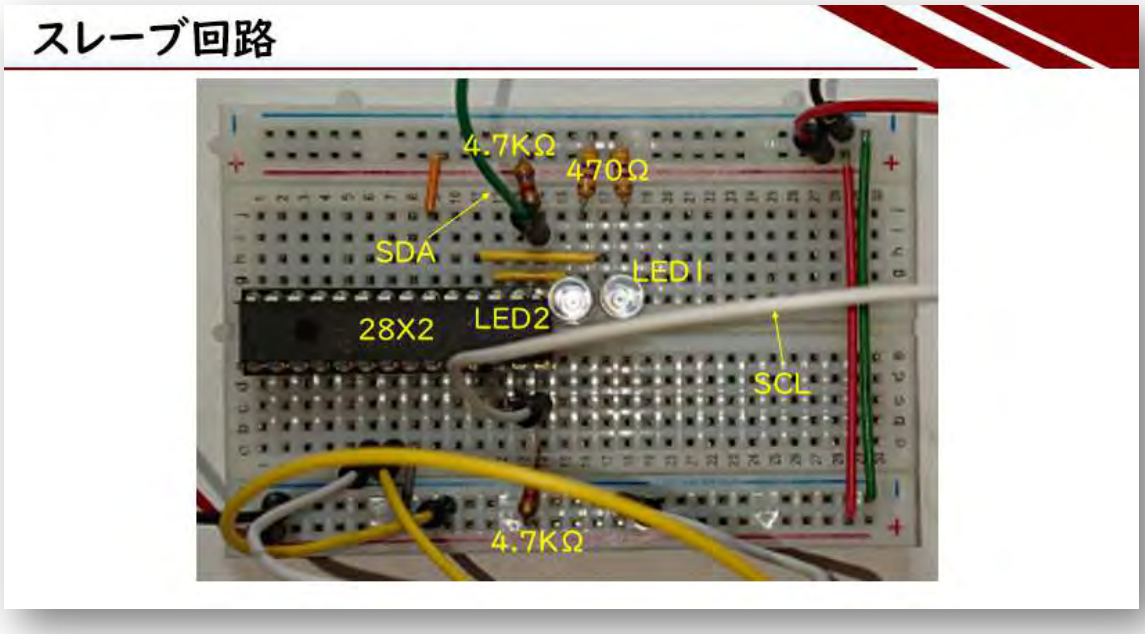


図 239

電源はライタ回路から配線します。この基板を経由して図の右上の配線で I2C マスタとなる PICAXE08M2 にカスケード接続しています。

【注意】TxD と RxD の配線は注意が必要です。マイコンプログラム書き込みの際は TxD、RxD 各々をライタ回路の該当するピン（図に記載）に接続して下さい。**書込み終了後は GND に接続して下さい。**

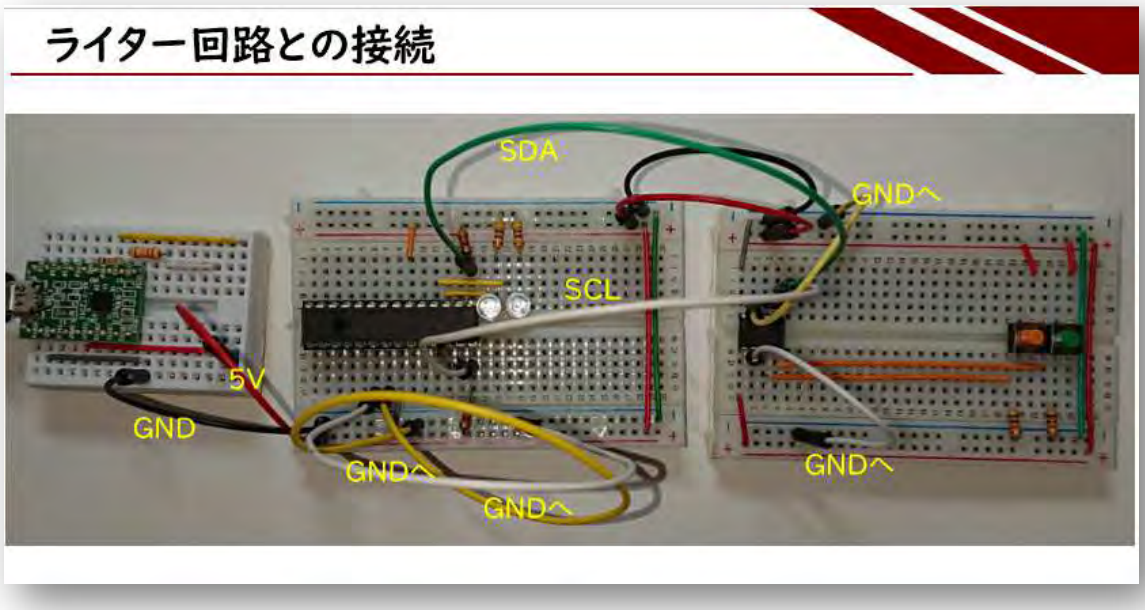


図 240

図は最終的な全体接続の様子を示しています。ここで慌てないでください。図はプログラムの書き込みが終った後の全ての基板を接続した様子です。まだ書込みは行っていませんので、前掲の回路図と上図を参考に配線を念入りに確認してください。各々のマイコンプログラムを書込む場合は、回路図に示したライタ回路との接続をして書込みを行います。2 つのマイコンへの書き込みが終了したら図の様に全体を接続します。プログラムを説明した後に全体の接続について、再び説明します。

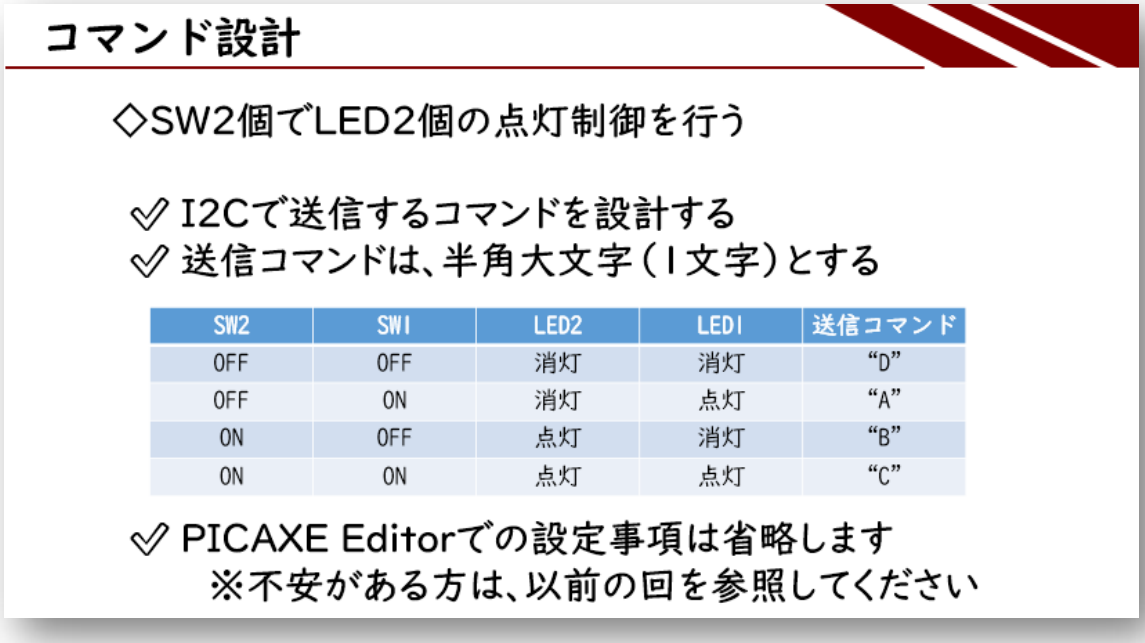


図 241

I2C でマスタから送信するコマンドを設計します。コマンドといっても 1byte の文字なので簡単です。図の表を見てください。マスタにある SW とスレーブにある LED の状態に該当する送信コマンドが示されています。コマンドは A、B、C、D の各 1 文字で LED の点灯パターンをスレーブに通知します。

図　242

マスタ（08M2）のプログラムを示します。#NO_DATA と記述すると、マイコン内部のデータ領域への書き込みを行わないので、書込み時間が短くなります。symbol というのは C 言語でいう#define に相当する、定義文です。初期化部分では hi2csetup で CPU をマスタに設定しています。スレーブアドレスは 0x70、i2c の通信速度は slow、i2c スレーブアドレスの幅を 1byte に設定しています。あらかじめ SW の状態を D（SW は全て OFF、LED も消灯）にしておきます。

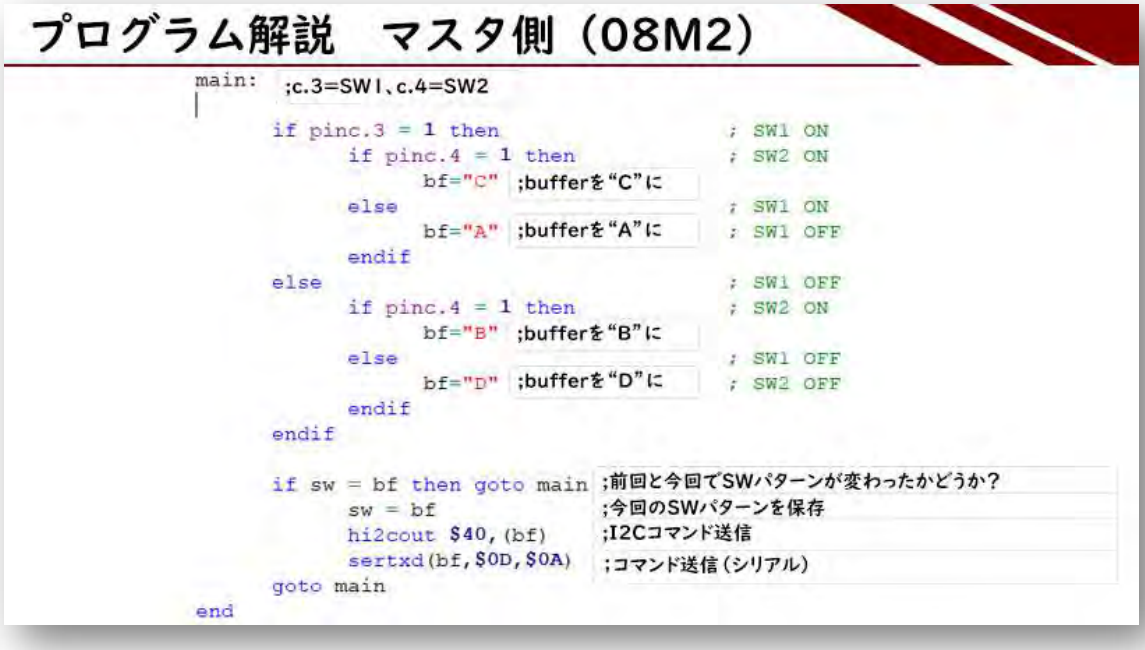


図　243

C.3 は SW1 に、C.4 は SW2 に接続されています。main:部分はでは、SW の押下状態に応じて、事前に設計した表に従い、該当コマンドを決定します。前回調べておいた SW 押下状態と現在の状態が変化した場合に、現在の状態を表すコマンドを hi2cout 命令で送信しています。LCD に対する制御コマンドと同じにするために$40（0x40）を付加しています。sertxd 命令で PC 向けにコマンドを送信しているので、PC を接続すれば動作の様子がモニタできます。

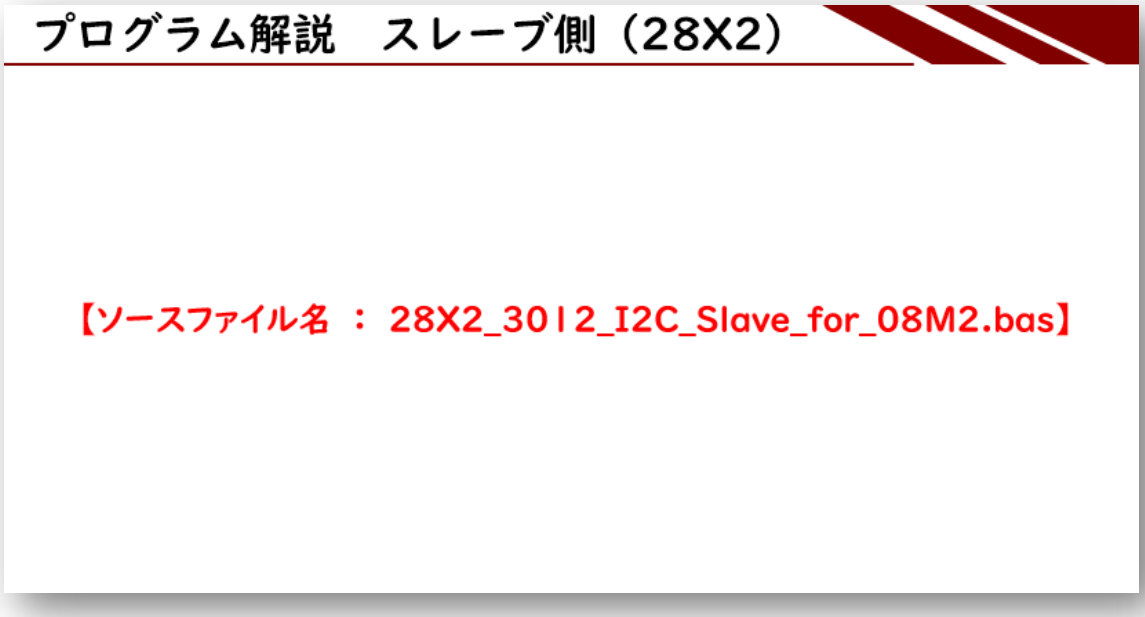


図 244

次頁から説明するスレーブ側プログラムは、図のファイル名で、実習キット付属 CD に含まれています。

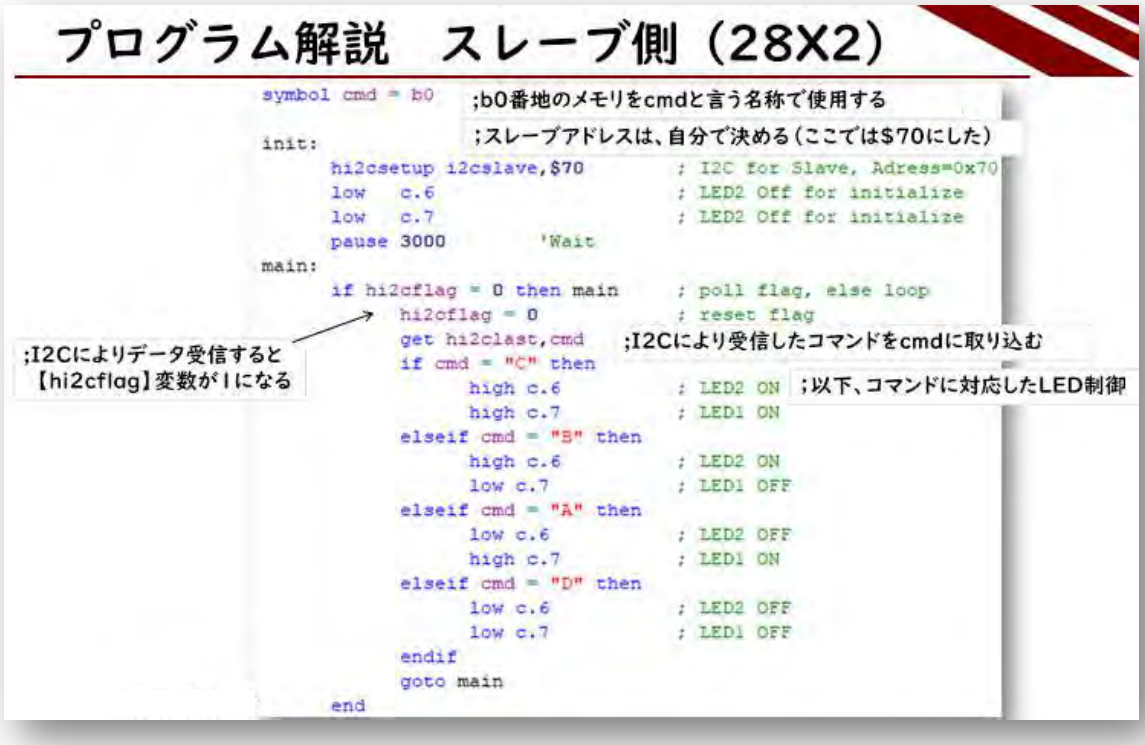


図　245

symbol 命令で変数を cmd（コマンド格納用）と言う名で定義しています。init:部分では、hi2sestup ではスレーブアドレス$70（0x70）のスレーブデバイスとしてシステムを設定します。初めは、LED は消灯しておきます。main:部分では hi2flag という PICAXE が内部に持っている I2C の受信フラグを確認します。このフラグは PICAXE 内部で I2C I/F を通じての受信があるとフラグが 1 になります。マスタから送信されるのは、0x40（制御コマンド）に続いて LED の点灯御パターンを示すコマンド 1byte です。hi2flag が 1 になったとき、最後に受信したデータが hi2clast 変数に格納されているので、それを get 命令で取り出します。取り出した LED 点灯パターンコマンドを調べて、対応する LED の点灯制御を行います。

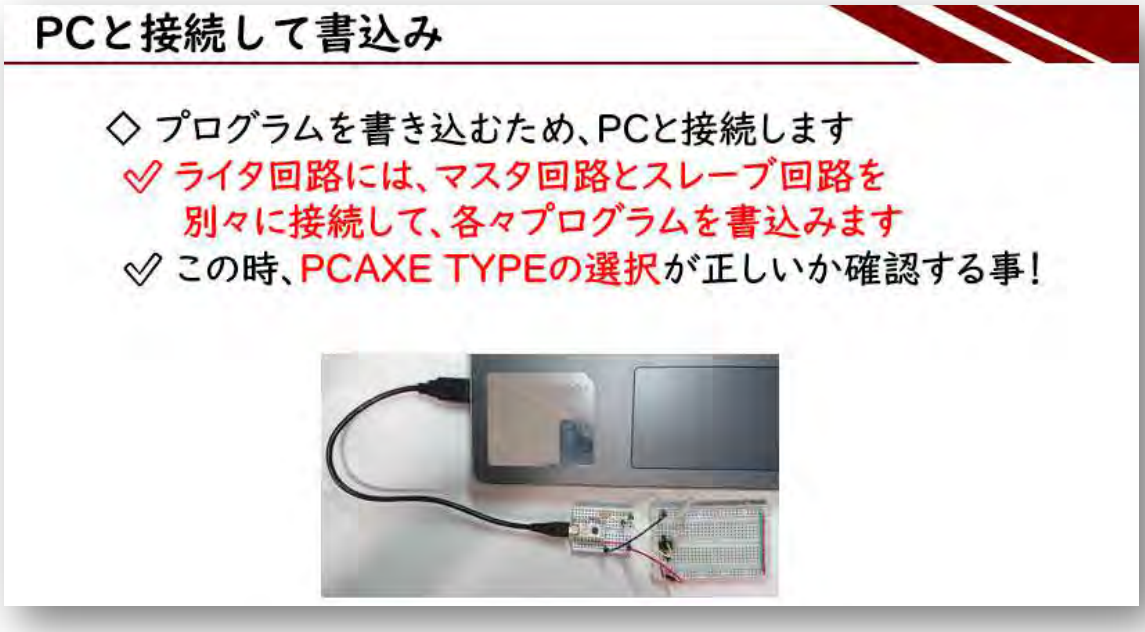


図 246

プログラムが完成したら、コンパイルと書込みを行います。PC、ライタ回路、マイコン回路を接続して下さい。

【注意】プログラムはマスタ回路用とスレーブ回路用がありますので、各々をライタに接続して書込みを行います。プログラムと対象回路を間違えないように、慎重に作業しましょう。

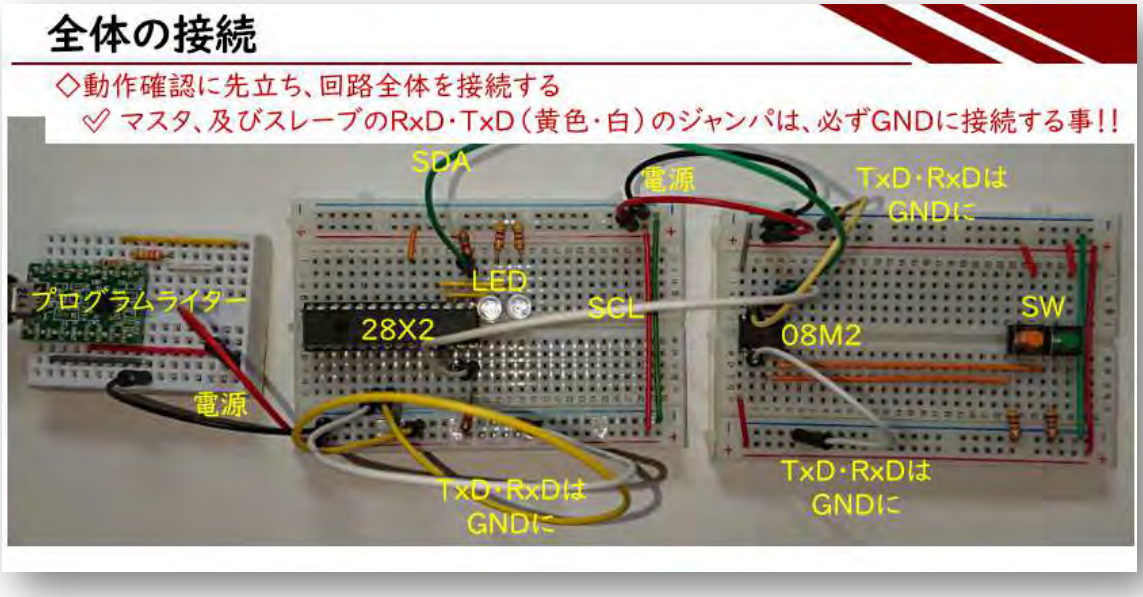


図 247

既に一度説明しましたが、書き込みが終了したら図の様に全体を接続します。その際、書き込みに使用したマスタ・スレーブの TxD・RxD は GND に接続して下さい。このように配線しておくことで、マイコンが PC からの書き込み要求を検出しない様にしています。この配線を行わないと、I2C の通信が途中で中断してしまいます。PICAXE 内部で書き込み要求を検出する処理を行うのが原因と考えられます。

これはマニュアルにも記載がなく、実際の動作を見ながら対応した結果です。実際の組込みシステム開発では、この様なことがよくあります。上手く動かなくても、執念深く確信をもって色々な工夫をしてみることが必要です。

USB ケーブルは動作確認の直前に PC に接続して下さい。

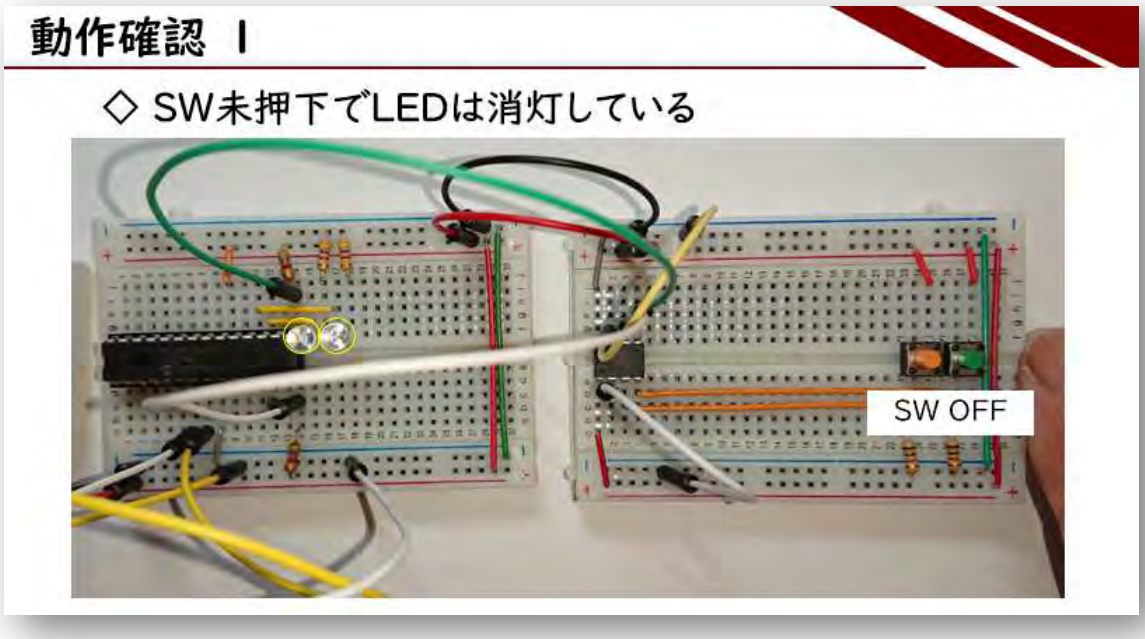


図 248

色々な作業がありましたが、遂に動作確認です。PC からの電源が供給されると、スレーブ側 init:の部分で指定の 3 秒を経過した後、動作が始まります。

SW 未押下では、LED は消灯しています。

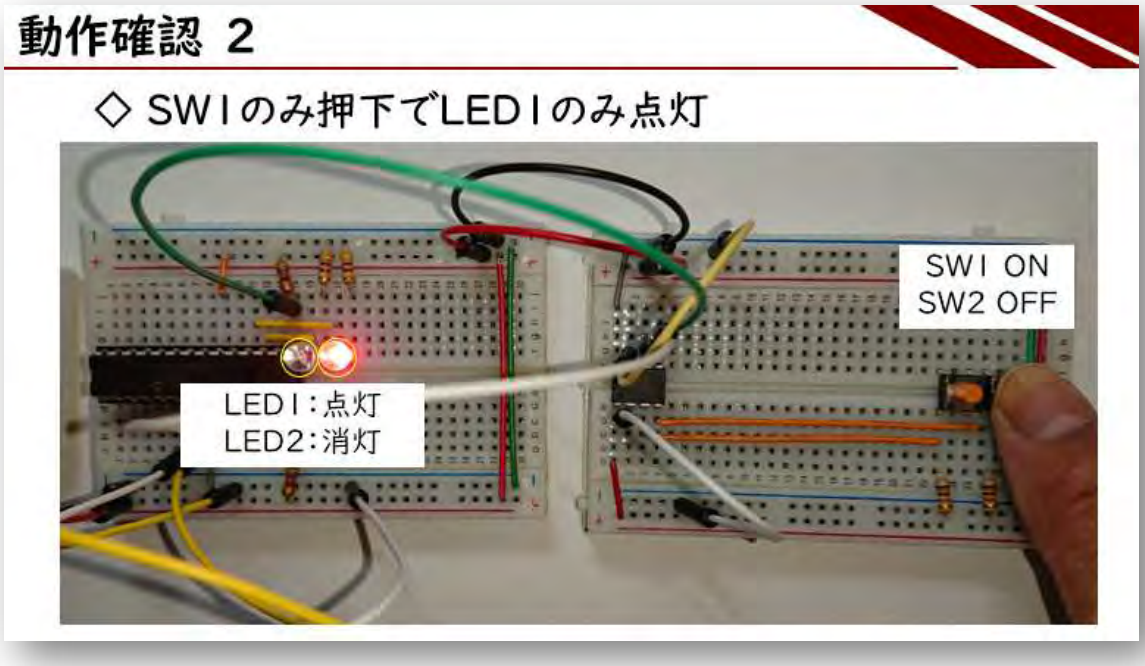


図 249

SW1 を押下すると、LED1 が点灯、LED2 は消灯です。SW1 から指を離せば LED は消灯します。

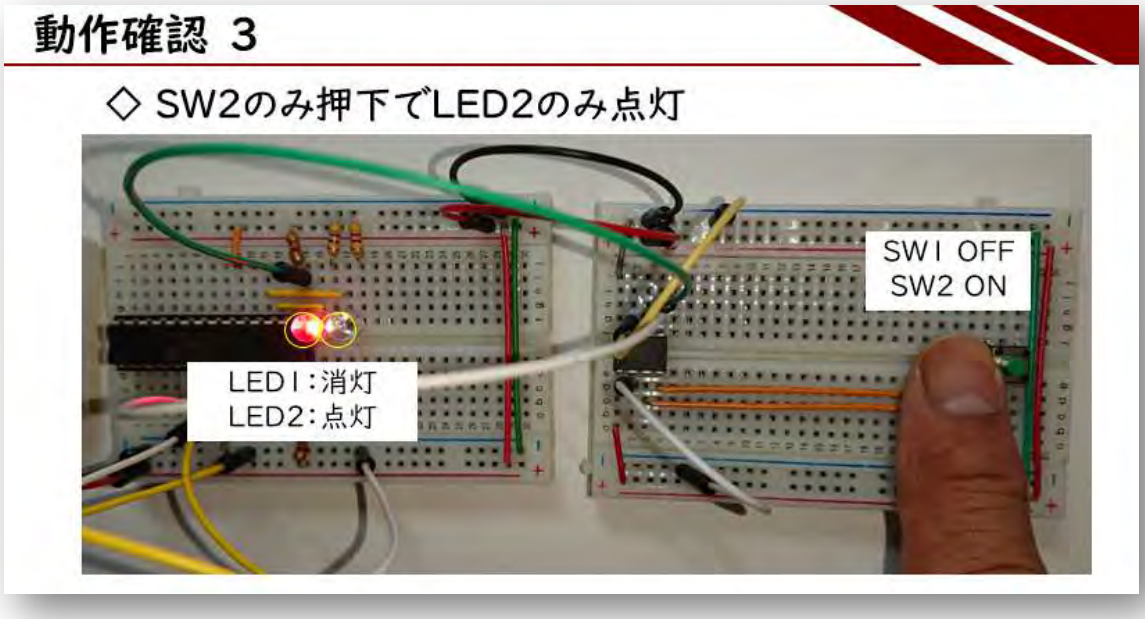


図 250

SW2 を押下すると、LED1 は消灯、LED2 が点灯します。指を離すと LED は消灯します。

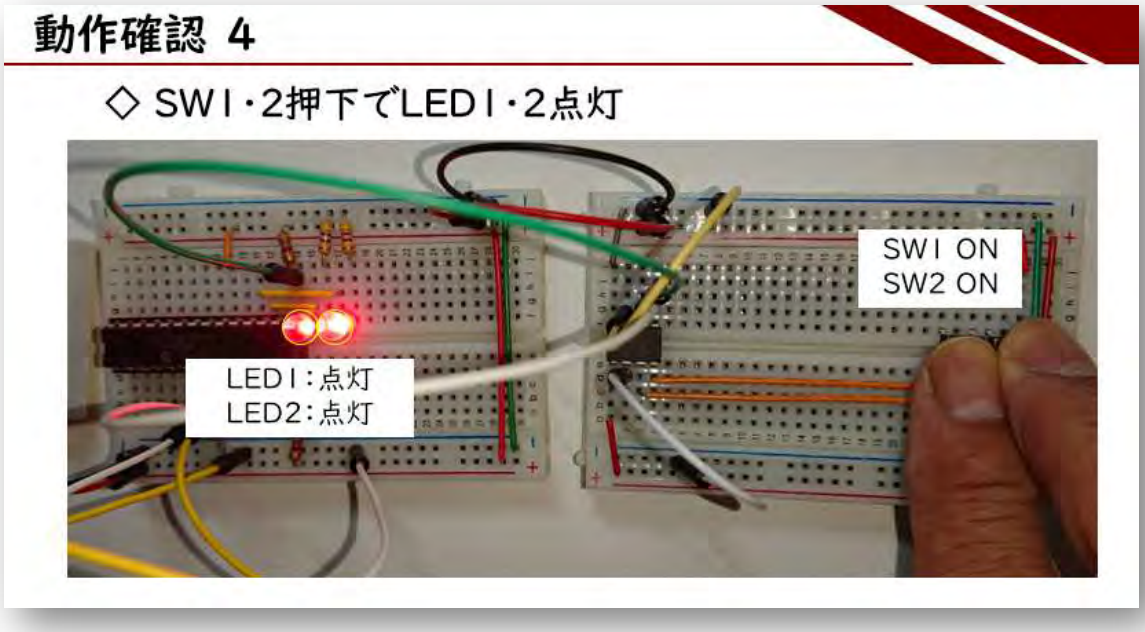


図 251

次に SW1 と SW2 を両方押下します。LED1、LED2 共に点灯します。指を離せば LED は消灯します。

動作確認はこれで終了です。あらかじめ設計したコマンドの通りに SW の押下パターンが I2C I/F を通じてスレーブマイコンに通知されて LED に反映されました。I2C スレーブデバイスの 1 例を開発しました。

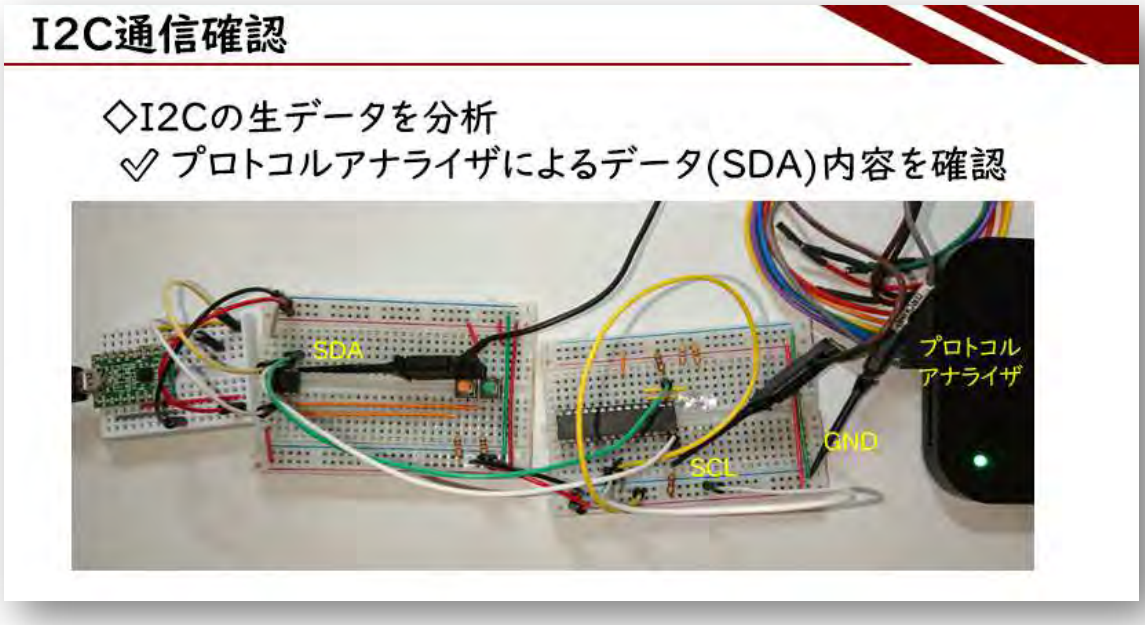


図 252

ここで、実際の I2C 通信の様子をプロトコルアナライザで観察してみましょう。プロトコルアナライザに SCL と SDA を接続します。

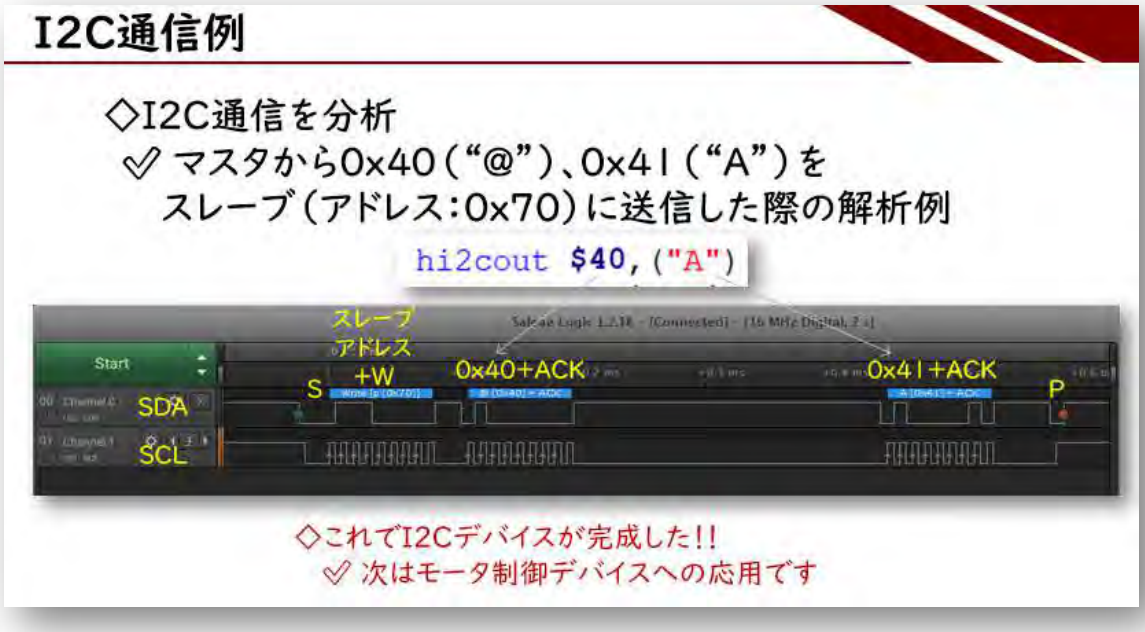


図 253

プロトコルアナライザで検出した SCL と SDA の様子を示します。図はマスタから 0x40 に続き文字 A を 0x70 のスレーブに向けて送信した場合です。スタートコンディション S に続きスレーブアドレス 0x70 が送信され ACK が返された後、しばらくして文字 A(0x41)を送信して、スレーブから ACK が返され、ストップコンディション P が送られて、通信を終了しています。

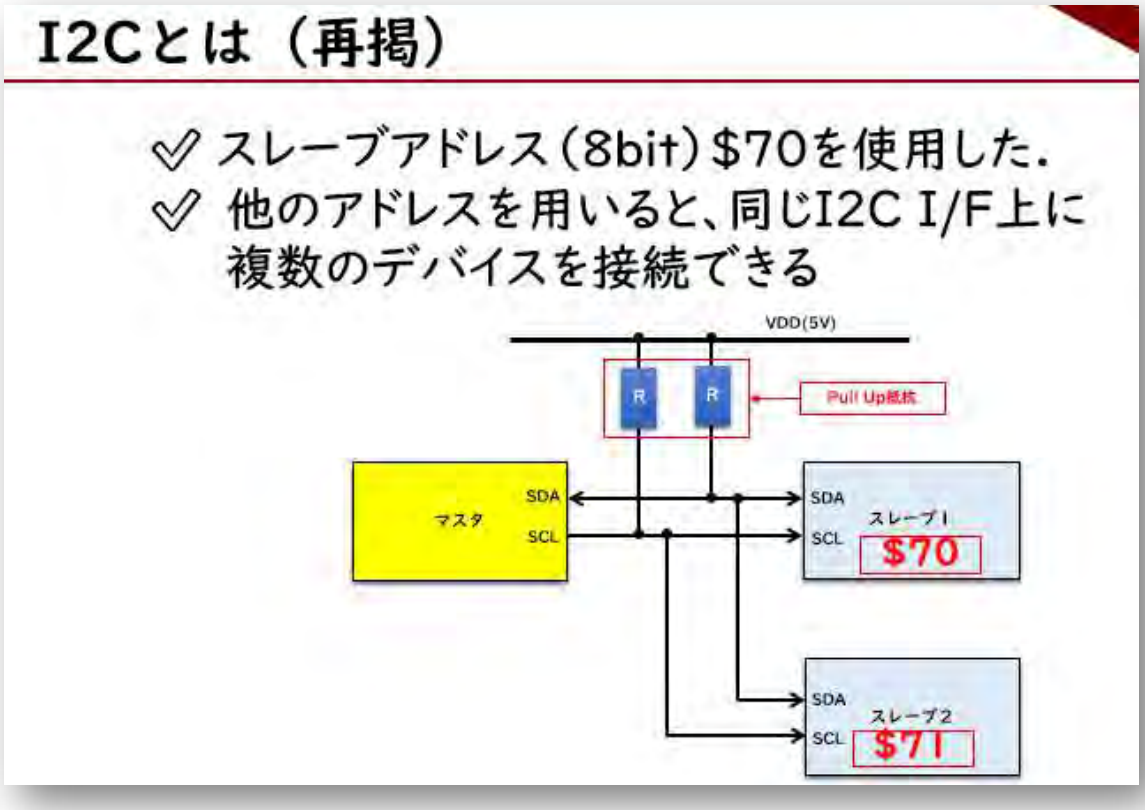
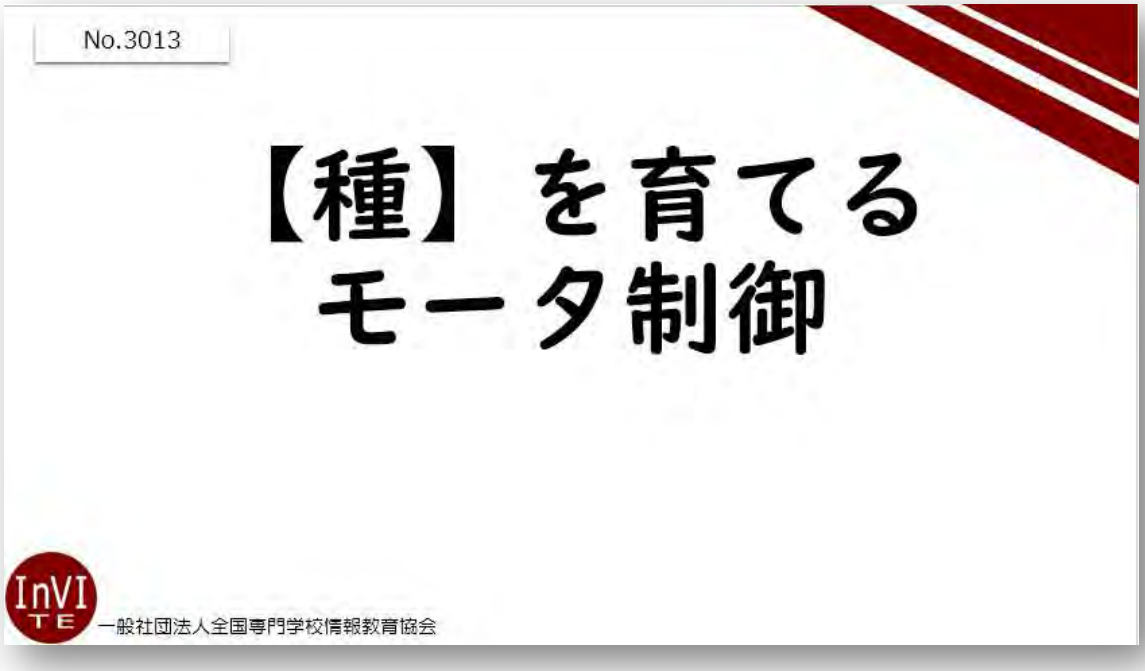


図 254

PICAXE28X2 では、スレーブアドレスを i2csetup 命令で自由に設定できます。今回は$70 を使用しましたが、例えば$71 も使えます。スレーブデバイスを複数開発して、マスタ側で SW を増設するなどして、送信先を変えれば、複数のデバイスをコントロールすることができるようになります。また内容の異なるスレーブシステムを同じ I2C BUS に接続できるので、開発可能なシステムの範囲が飛躍的に広がります。まさに【IoT の種】です。

※特定のスレーブアドレスは用途が限定されています。詳細は PICAXE マニュアルを精読してください。

第13回　モータ制御デバイス開発



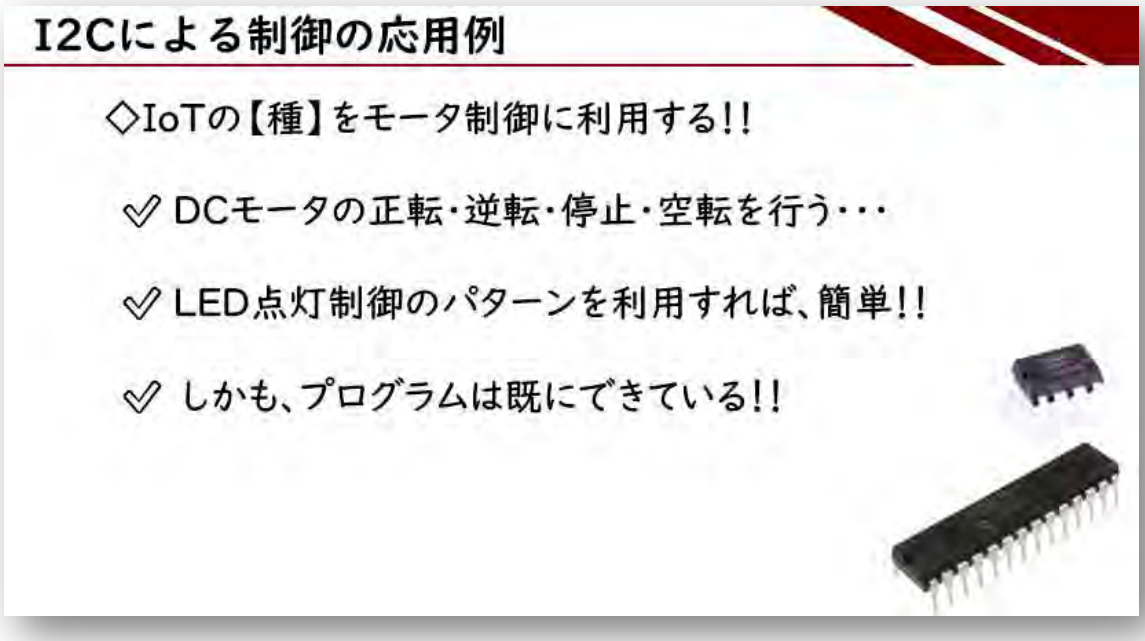


図 255

前回開発した【IoT の種】を育てるのが今回の狙いです。I2C デバイスを利用したモータ制御システムを開発してみましょう。単に回転・停止だけではもったいないので、正転・逆転・停止・空転の 4 モード制御を行います。【IoT の種】で行っている LED の点灯パターンを利用すれば簡単に実現できます。既に開発したプログラムがそのまま利用できます。

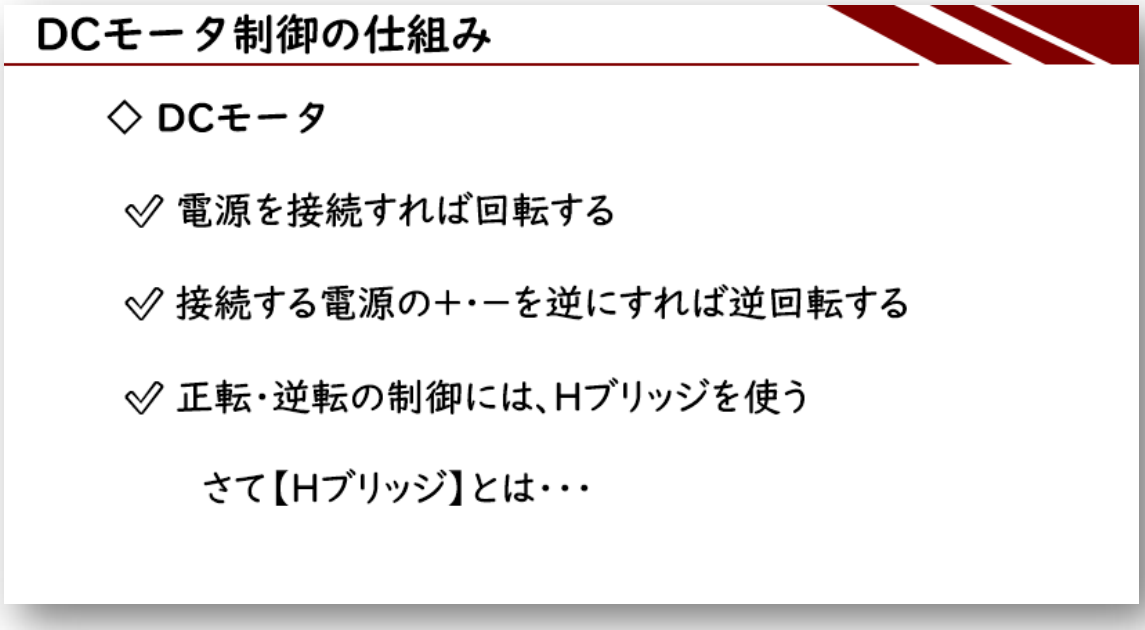


図 256

DC モータは直流電源を接続すれば回転する動力源です。直流には＋－がありますが、これを反対に接続すれば逆方向に回転します。通常回転（正転）と逆回転（逆転）の制御を行うには、＋－の接続を反対にしなければいけません。さて、モータを接続したままこれを行うにはどうすれば良いでしょうか？

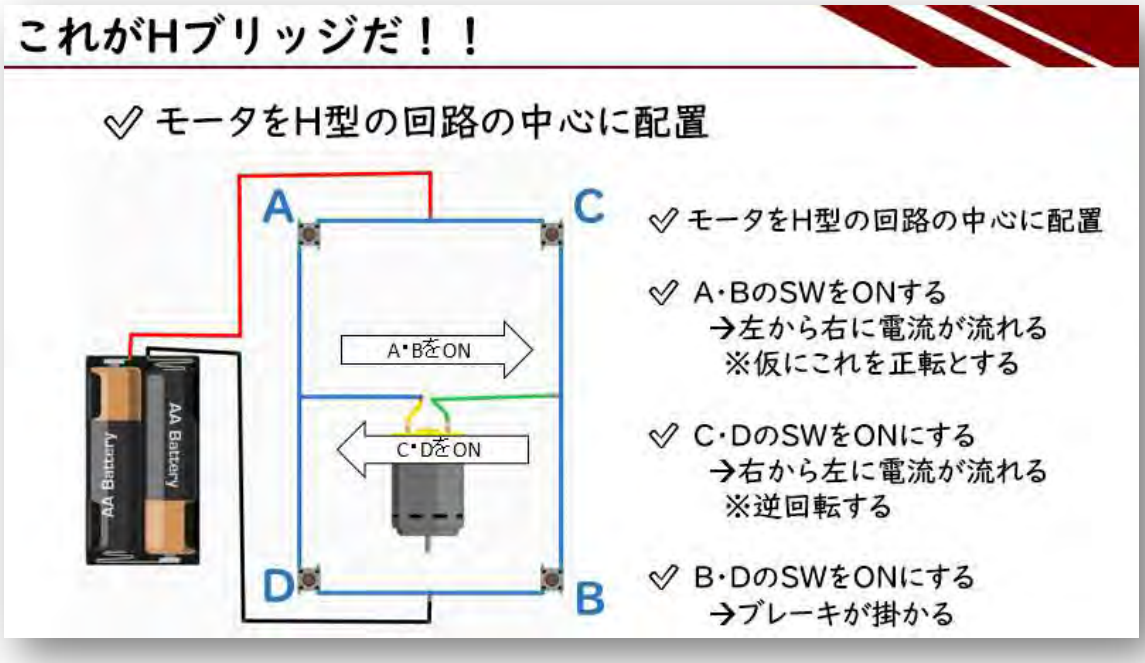
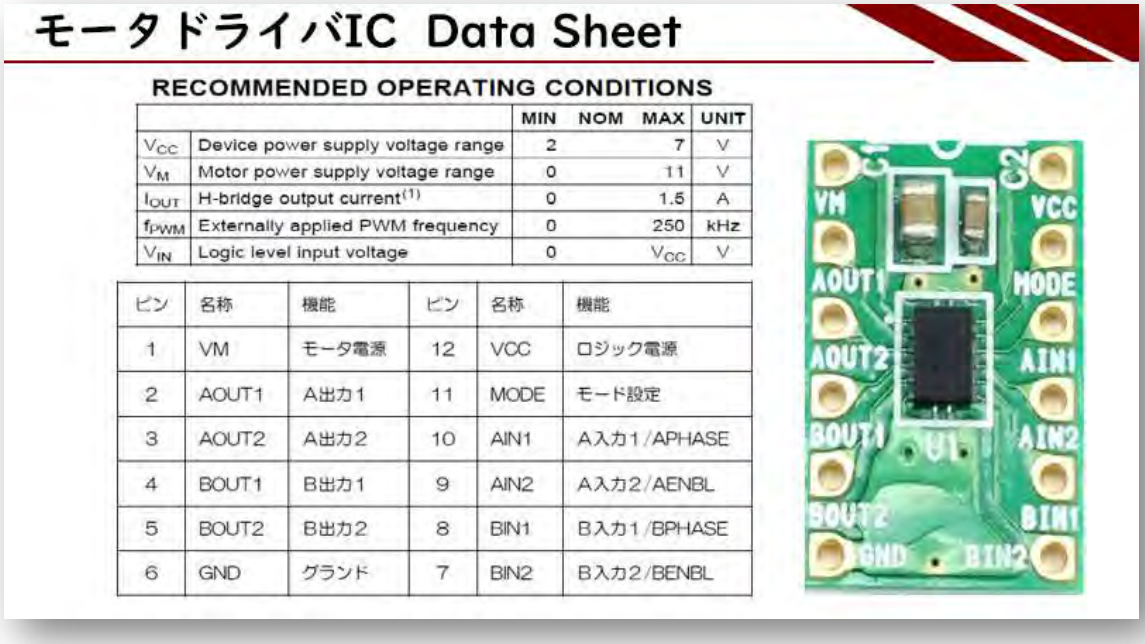


図　257

H の文字の形に回路を組んで、その中央部にモータを配置する H ブリッジという回路があります。H の文字の四隅に SW を置き中央にモータを接続します。SW がすべて OFF の場合は、回路に電流が流れないのでモータは空転状態です。モータに流れる電流の向きを左から右、右から左に変化させるには、SW の押下パターンをどうすれば良いかを考えましょう。

図 258

SW の代わりをして、H ブリッジを構成してくれるモータドライバ IC があります。SW 機能をトランジスタなどで実現すると 4bit の DO が必要です。モータドライバでは 2bit で済みます。半分の配線でモータの回転を制御できます。モータドライバ IC のデータシートの一部を図に示します。前回設計したコマンドの表にそっくりです。表の FUNCTION が前回では ABCD のコマンドになっていました。モータドライバ IC の入力（マイコンからの出力）は図の x IN1、x IN2 です。前回の表の LED の点灯パターンに相当します。この制御は前回開発した【IoT の種】で賄えます。



モータドライバIC Data Sheet

RECOMMENDED OPERATING CONDITIONS

		MIN	NOM	MAX	UNIT
V_{CC}	Device power supply voltage range	2		7	V
V_M	Motor power supply voltage range	0		11	V
I_{OUT}	H-bridge output current(1)	0		1.5	A
f_{PWM}	Externally applied PWM frequency	0		250	kHz
V_{IN}	Logic level input voltage	0		V_{CC}	V

ピン	名称	機能	ピン	名称	機能
1	VM	モータ電源	12	VCC	ロジック電源
2	AOUT1	A出力1	11	MODE	モード設定
3	AOUT2	A出力2	10	AIN1	A入力1/APHASE
4	BOUT1	B出力1	9	AIN2	A入力2/AENBL
5	BOUT2	B出力2	8	BIN1	B入力1/BPHASE
6	GND	グランド	7	BIN2	B入力2/BENBL

図 259

【このページは重要です】

データシートをさらに見ると、この IC の電源は 2～7V となっています。マイコンの電源 5V で駆動できます。モータ電源は MAX 11V で、H-bridge 電流は 1.5A となっています。小型の DC モータでこの規格に入るものはたくさんあります。これまで、システムの電源は PC の USB 電源に頼っていましたが、モータ電源は別に用意して PC の USB I/F に負荷をかけないように配慮します。図の下にある表はこのドライバ IC を搭載した基板のピン配置です。11 番のモードは、前頁の表にある、MODE が 0 となっていますので GND に接続します。入出力が 2 組あり内部は 2 回路になっています。モータは起動時に大きな電流が流れます。2 組の回路を入・出力を同時に使ってこれに備えます。AIN1 と BIN1、AIN2 と BIN2、AOUT1 と BOUT1、AOUT2 と BOUT2 をそれぞれ接続して 1 つの回路として使用します。

図　260

前回 LED に対して出力していた DO 2bit をモータドライバ IC に対して出力しています。スレーブの LED がモータドライバ IC と DC モータに置き換わった回路になることが図で分かります。

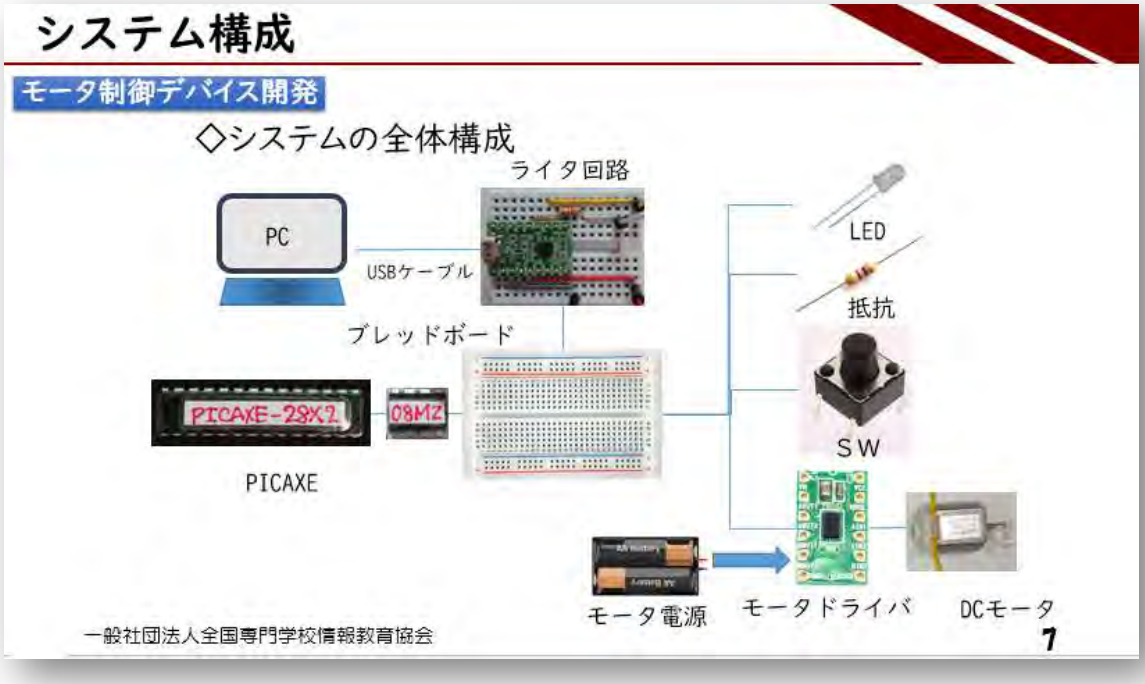


図　261

モータ関連パーツが追加されます。前回の LED はそのまま、ドライバ IC への出力モニタとして使います。モータ電源は別途乾電池で供給します。

図 262

PICAXE08M2（I2C マスタ）は前回と全く同じで、変更の必要はありません。

図 263

PICAXE28X2（I2C スレーブ）も前回と同じで変更はありません。

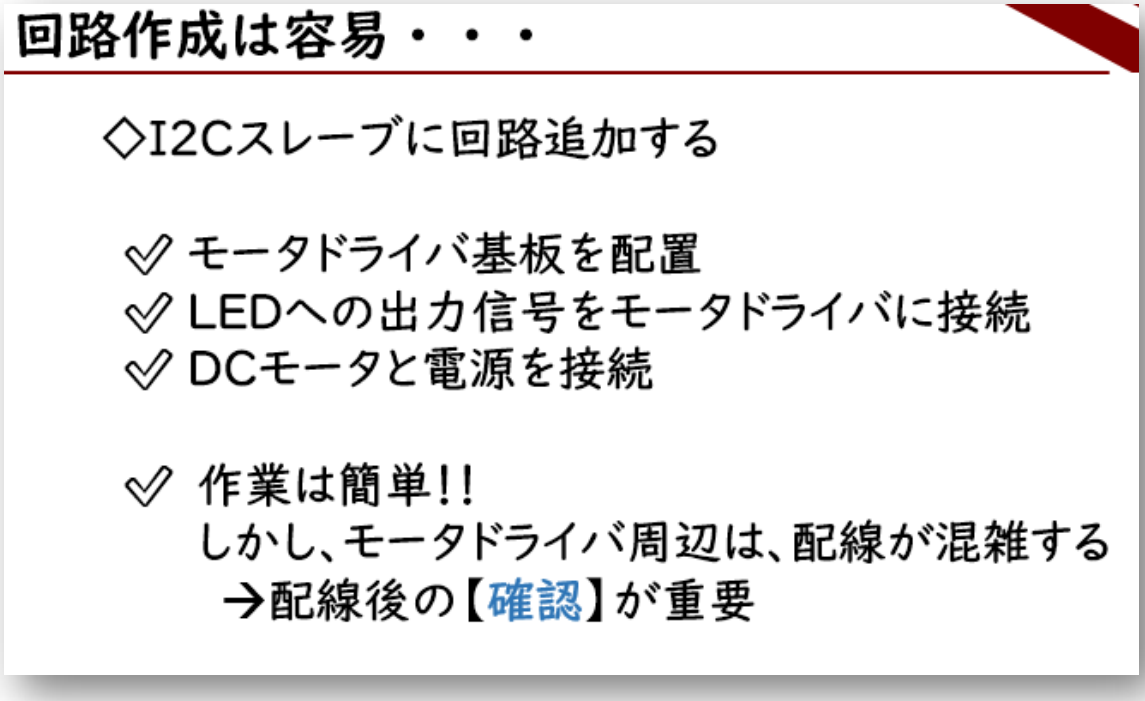


図 264

今回の回路についてまとめると次の様になります。

1．I2C スレーブに追加回路があり内容は以下の通り。

2．モータドライバ基板を配置

3．LED への出力信号を分岐してモータドライバ基板に入力

4．モータドライバ基板の出力を DC モータに接続

5．DC モータ用電源をモータドライバ基板に接続

今回の追加配線は複雑ですから、完成後の確認は念入りに行って下さい。

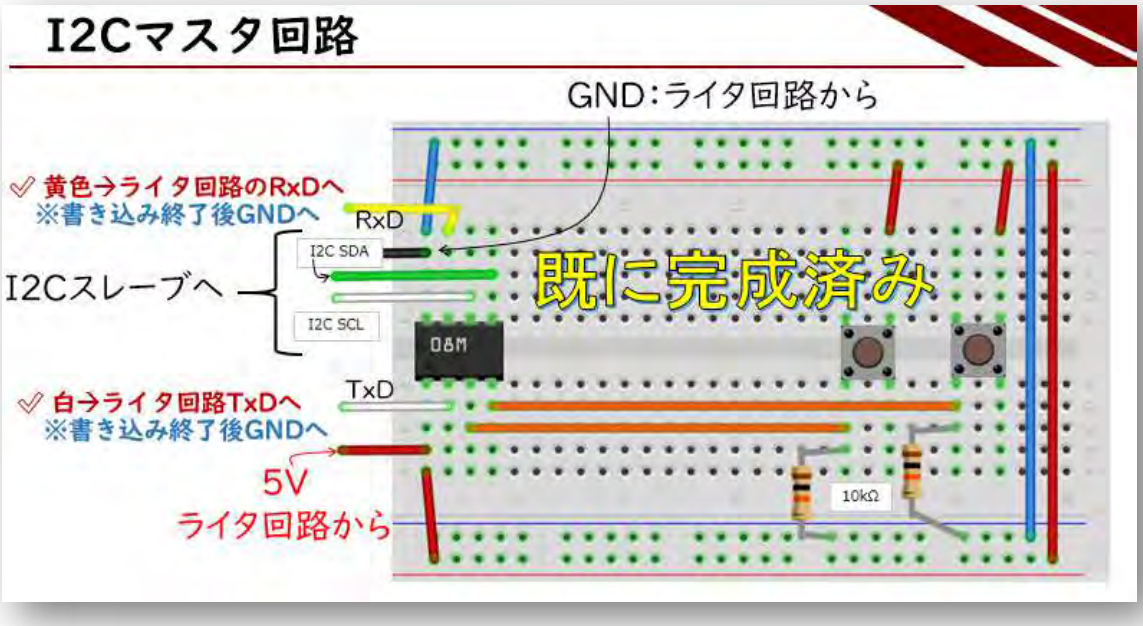


図 265

I2C マスタとしての回路は変更がありません。配線の緩みなどを点検します。

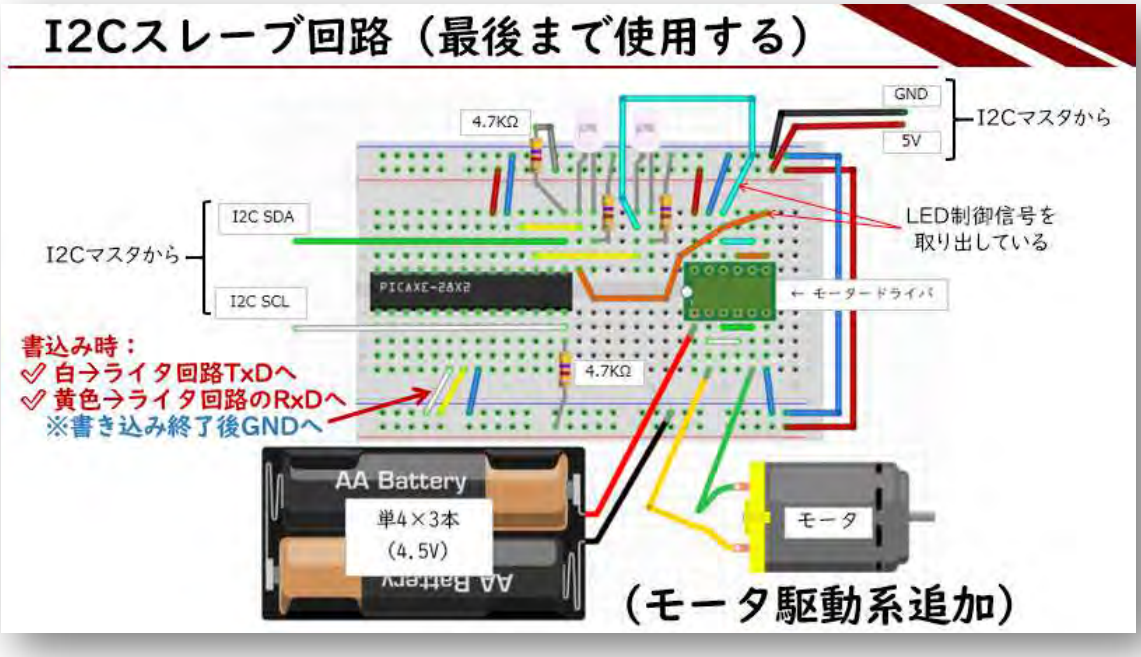


図 266

スレーブ回路は少し複雑です。じっくりと丁寧に配線してください。時間をたっぷり掛けて作業しましょう。特に、LED の点灯信号をモータドライバ基板に接続する部分、及びモータドライバ基板の入力・出力信号の取りまとめを 273 ページで十分理解した後でなければ、作業を始めないでください。また、不用意にモータ電源の電池ボックス SW を ON にしないでください。

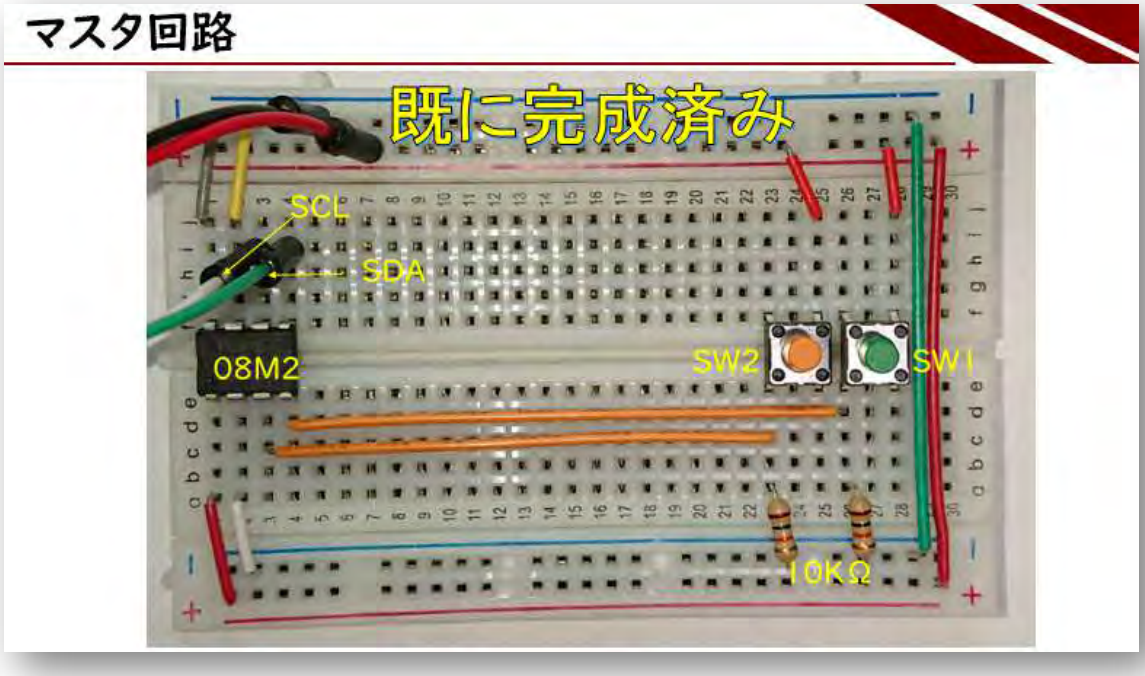


図 267

マスタ回路は変更がありません。前回と全く同じです。

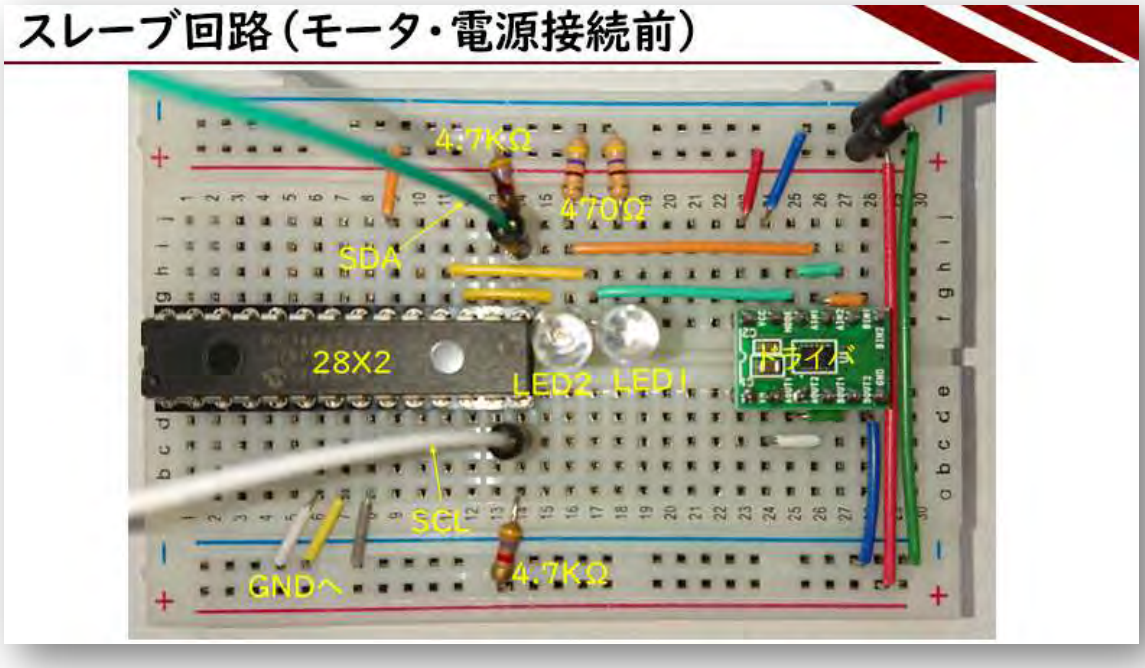


図 268

モータ、電源接続前のスレーブ回路を示します。十分な確認を行って下さい。

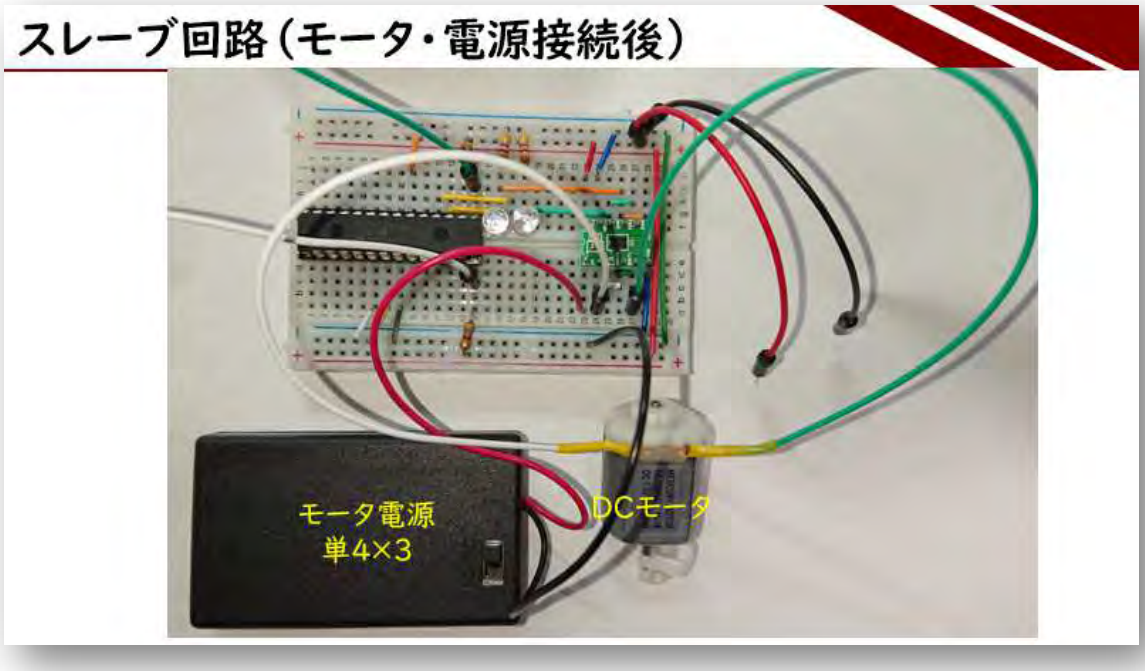


図 269

モータ、電源を接続するとこのようになります。

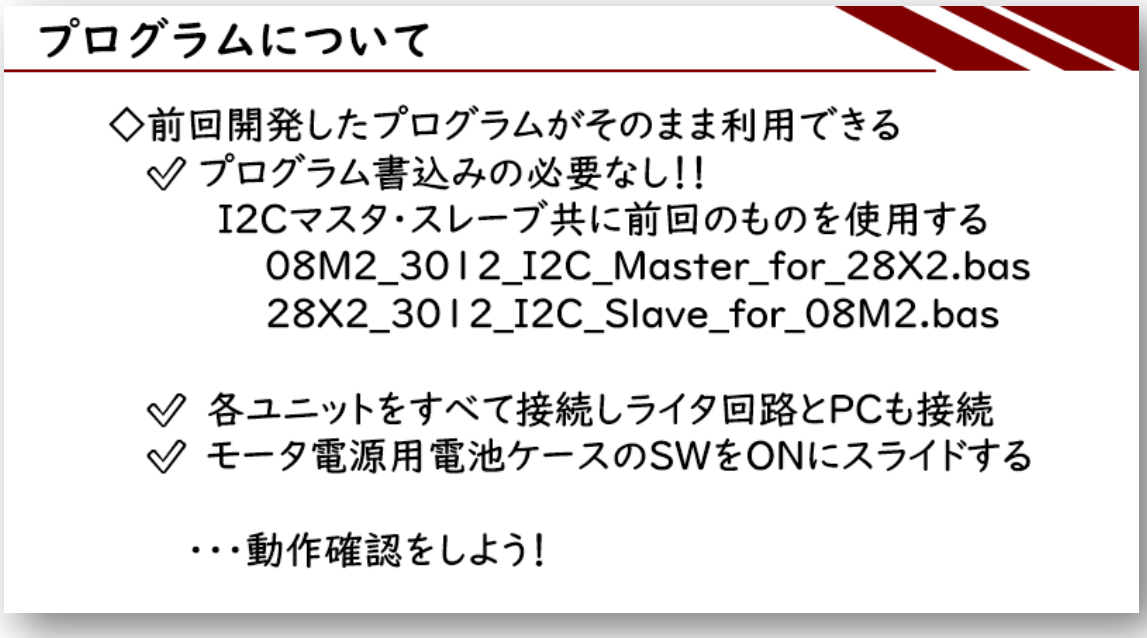


図 270

各マイコンのプログラムは今回作製する必用はありません。前回で動作確認が完了していれば、このシステムにそのまま使えます。回路の確認ができたら各ユニットを全て接続し、ライタ回路、PC も接続します。最後にモータ電源の SW を ON にスライドします。

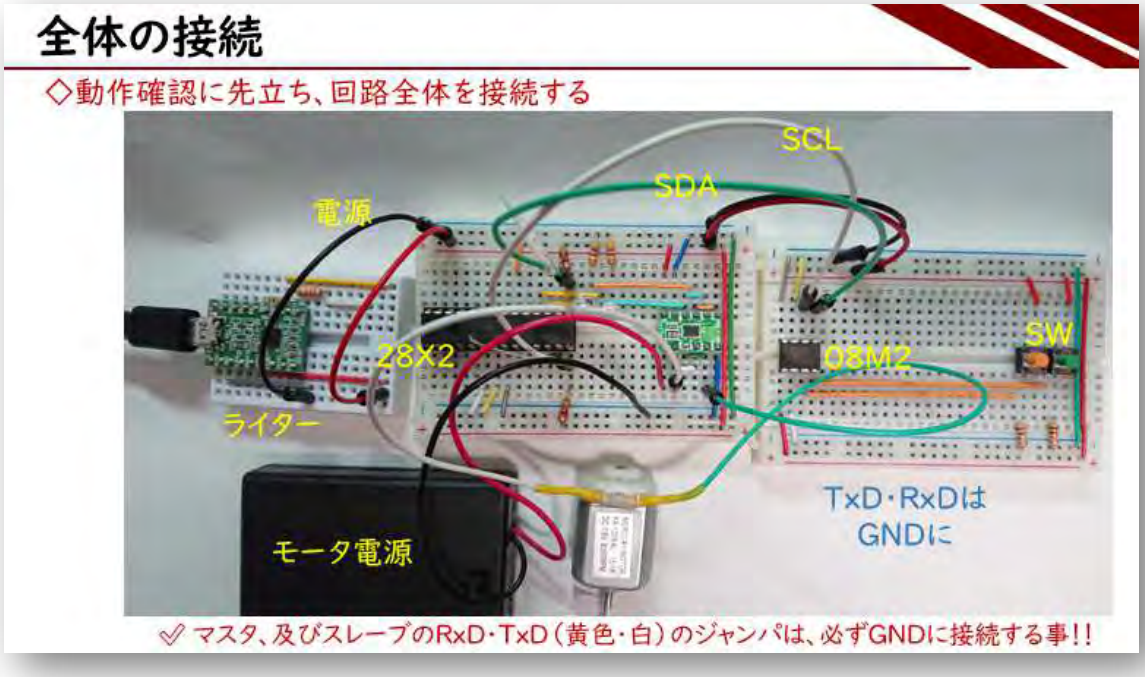


図　271

図はライタも含めた全ユニットを接続した様子です。各マイコン基板の TxD、RxD が GND に接続されているか、再確認してください。準備が整ったら動作確認です。

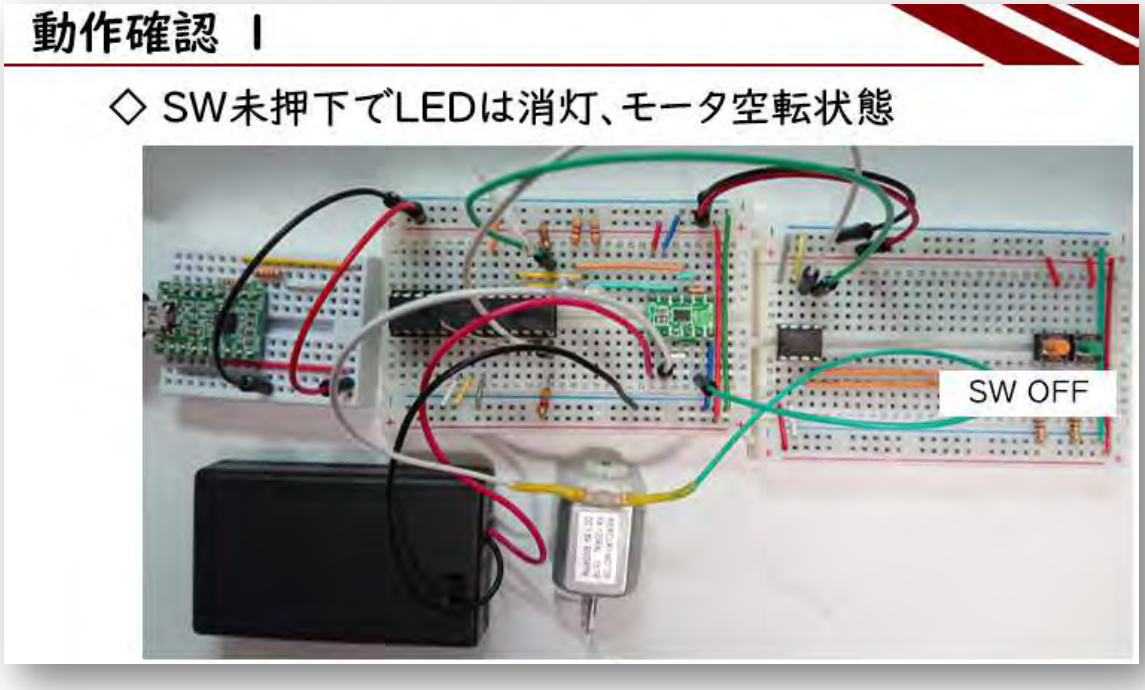


図 272

まず、そのままの状態（通常状態）では LED は消灯しモータは空転状態です。モータの主軸は手で容易に回せます。

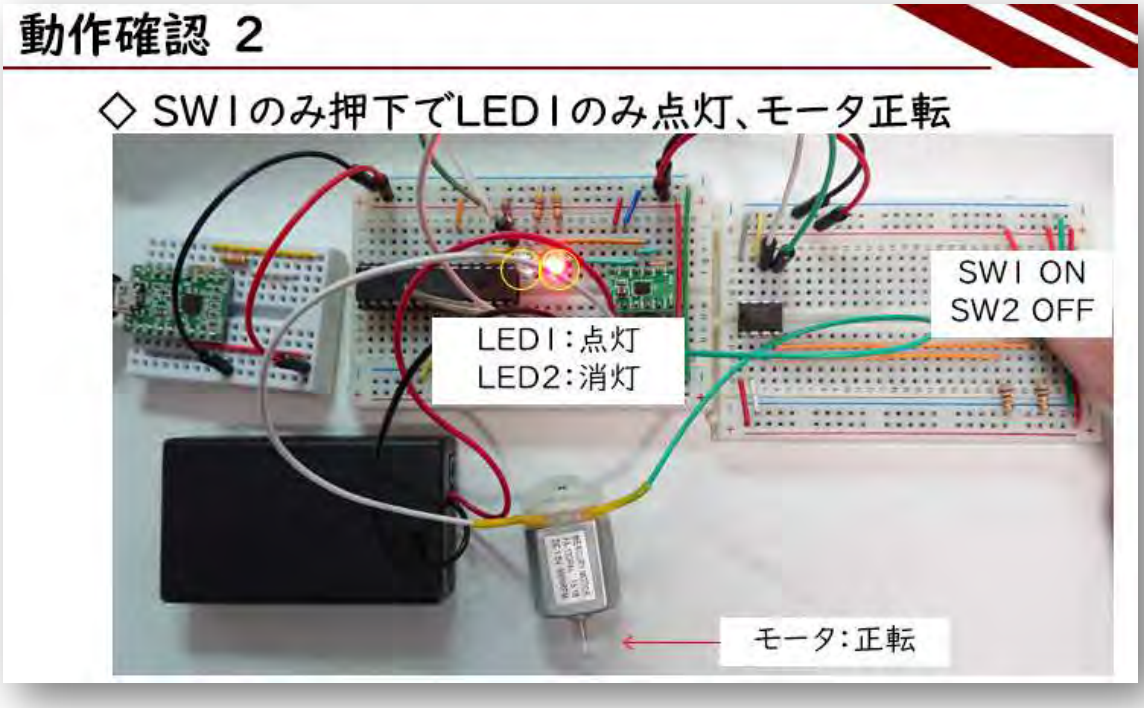


図 273

SW1 のみ押下します。LED1 が点灯し、同時にモータが正転します。※この時の回転の向きを正転とします。（DC モータのリード線とドライバ基板の配線によって、回転の向きが変わります。）SW から指を離せば LED は消灯してモータも停止します。

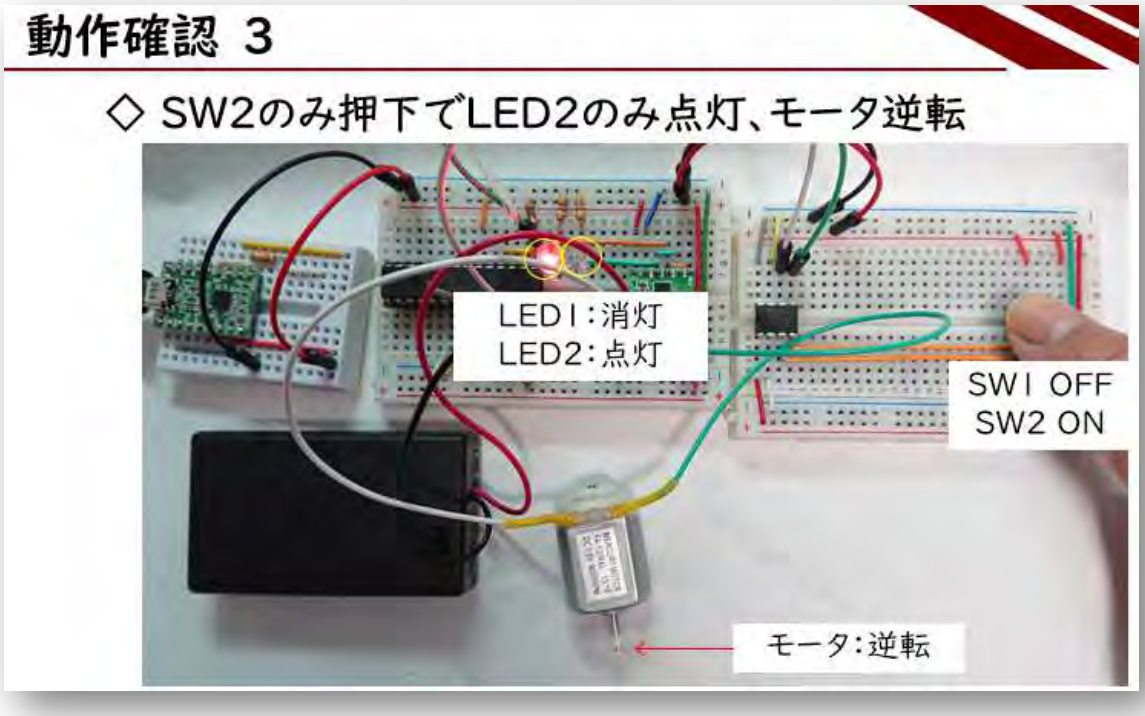


図 274

SW2 のみ押下します。LED2 が点灯し、同時にモータが逆転します。SW から指を離せば LED は消灯してモータも停止します。

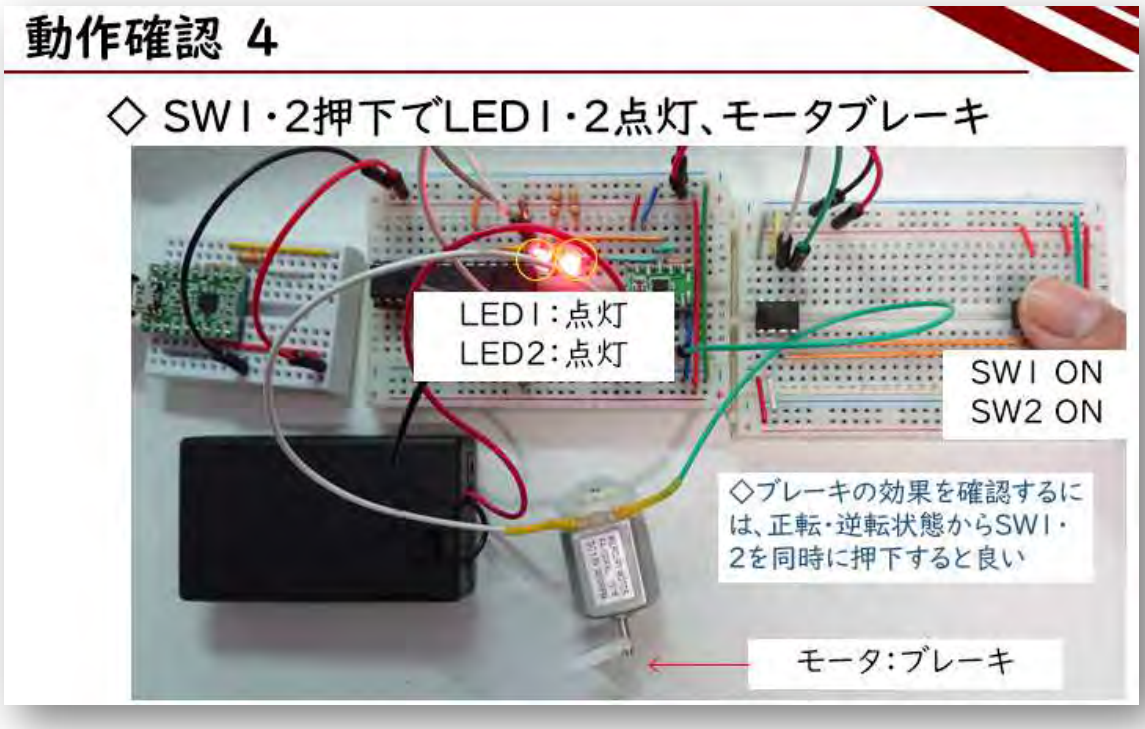


図 275

SW1・SW2 を同時に押下します。LED1・LED2 ともに点灯します。モータはこの時ブレーキが掛かった状態になります。主軸を指で回すと、トルクを感じる状態です。

※正転または逆転の状態から、SW1・SW2 を同時に押下すると、この様子が分かり易い。

SW から指を反せば LED は消灯して、モータは空転状態になります。最期にモータ電源の電池ボックス SW を OFF にスライドしてください。

I2C デバイスとして開発した【IoT の種】LED の点灯制御システムが、モータ制御システムへと育ちました。このようなステップを踏むことで、柔軟性を持った開発力が身に付きます。

第14回　外部モータ制御

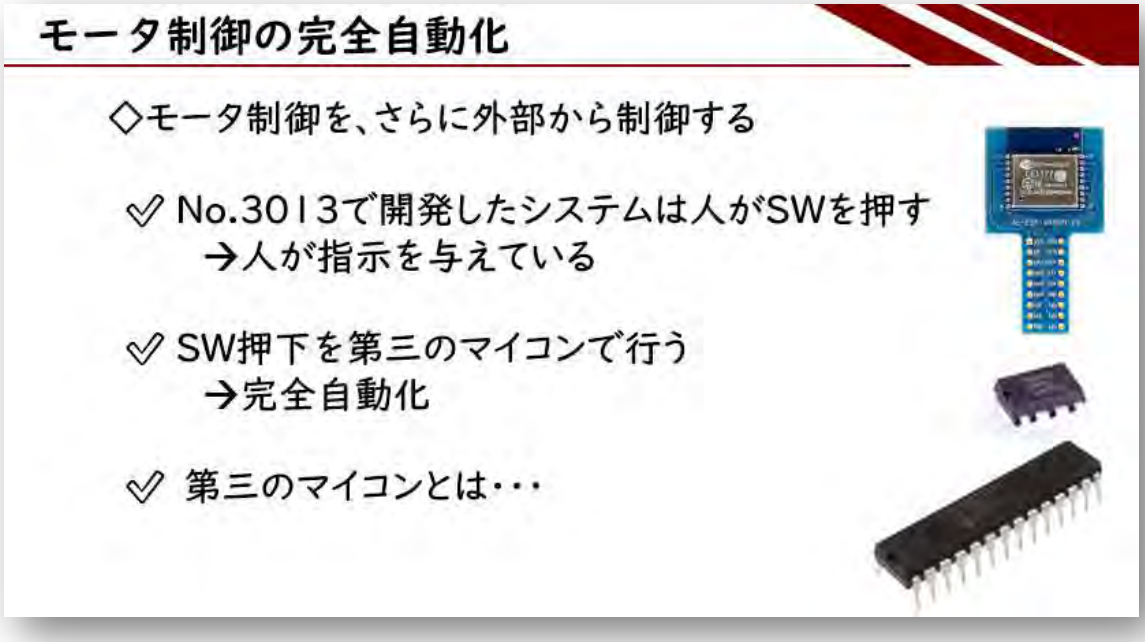


図 276

前回（第 13 回）に開発したシステムは I2C スレーブデバイスに対して SW で人が指示を与えるものでした。この SW 押下の部分をさらに別のマイコンで外部から制御する、完全自動化を行ってみます。システム全体の最も上流に位置する、第三のマイコンが必要です。

図　277

第三のマイコンとして、WiFi 機能を持つマイコン ESP-WROOM-02 を使用します。このマイコンからの制御が成功すれば、WiFi 機能を用いてさらに WEB 越しの制御へと拡張ができます。

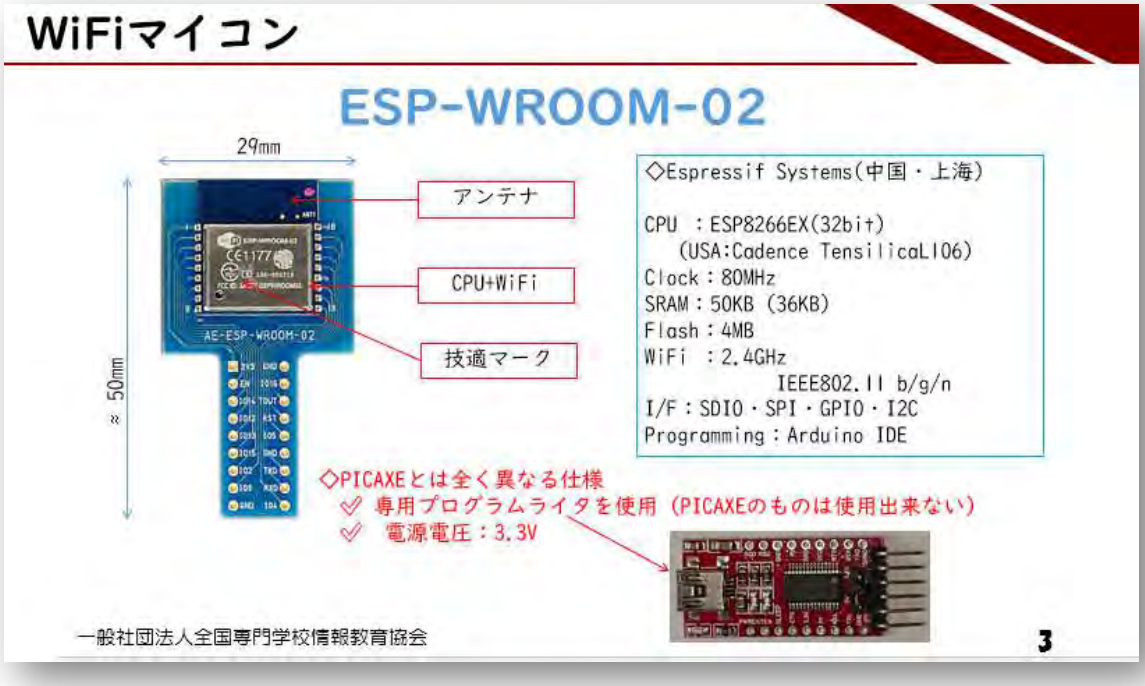


図 278

図は使用する第三のマイコンです。ESP-WROOM-02 はメーカ型番です。図左の基板の上部に配置されている金属ケースの内部にマイコンが入っています。使用されている CPU は ESP8266 です。このマイコンは中国の Espressif Systems 社製で WiFi モジュールが搭載されています。PC でプログラムを開発・書込みを行いますので、PICAXE と同様に USB ーシリアル I/F が必要ですが、通信の論理レベルが異なる為 PICAXE の物は使用できません。電源電圧 3.3V ですから USB の 5V をレギュレータで変換して 3.3V を作るための電源回路も必要になります。

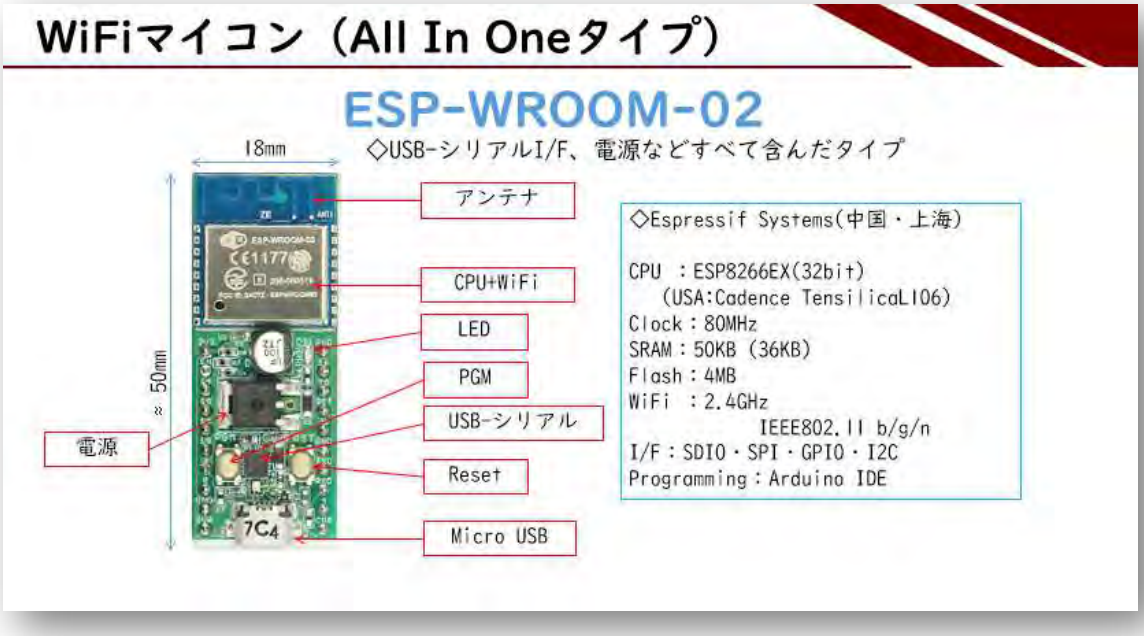
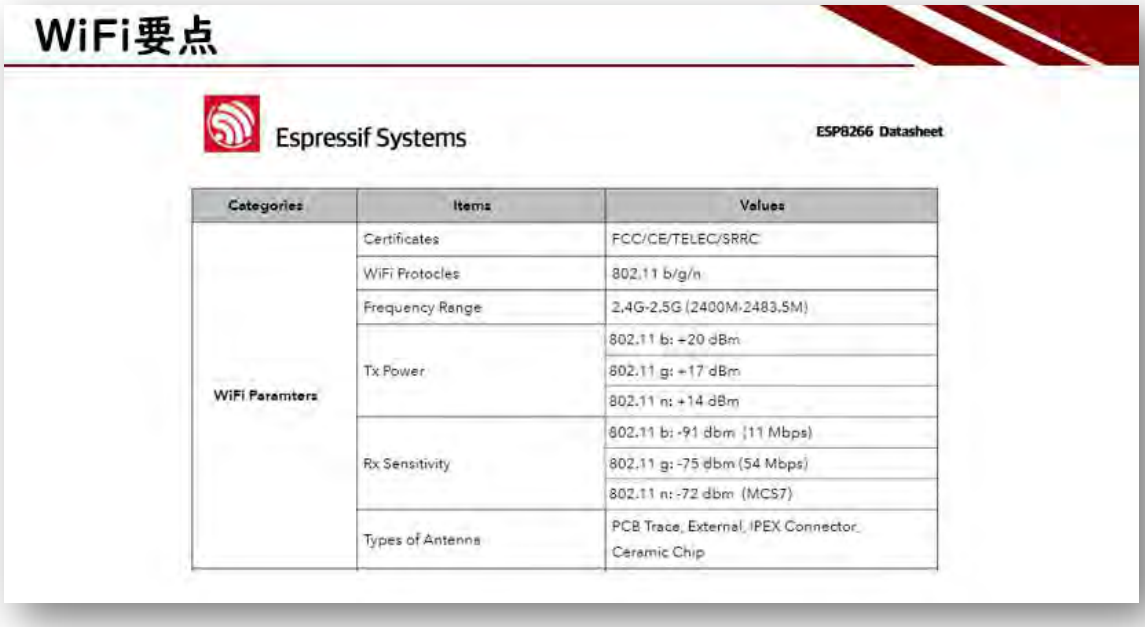


図 279

ESP-WROOM-02 は、電源回路、USB－シリアル I/F が搭載されているオールインワンタイプのマイコン基板もあります。このテキストを書き始めた頃、この基板の入手が難しかったので、マイコンだけの基板を使用して、電源回路、シリアル I/F は外付けした実験回路を設計しました。IoT システム開発実験を行うのであれば、図に示すマイコン基板を利用するのが便利です。



Categories	Items	Values
WiFi Paramters	Certificates	FCC/CE/TELEC/SRRC
	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4G-2.5G (2400M-2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Types of Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip

図 280

図は第三のマイコン ESP-WROOM-02 のデータシートの一部です。WiFi の使用電波は 2.4GHz 802.11 b/g/n に準拠しています。一般の WiFi アクセスポイントに接続できる規格です。

図 281

使用電源は 3～3.6V の範囲で、電流は平均 80mA となっているので、これが賄える電源 IC（レギュレータ）を使用します。

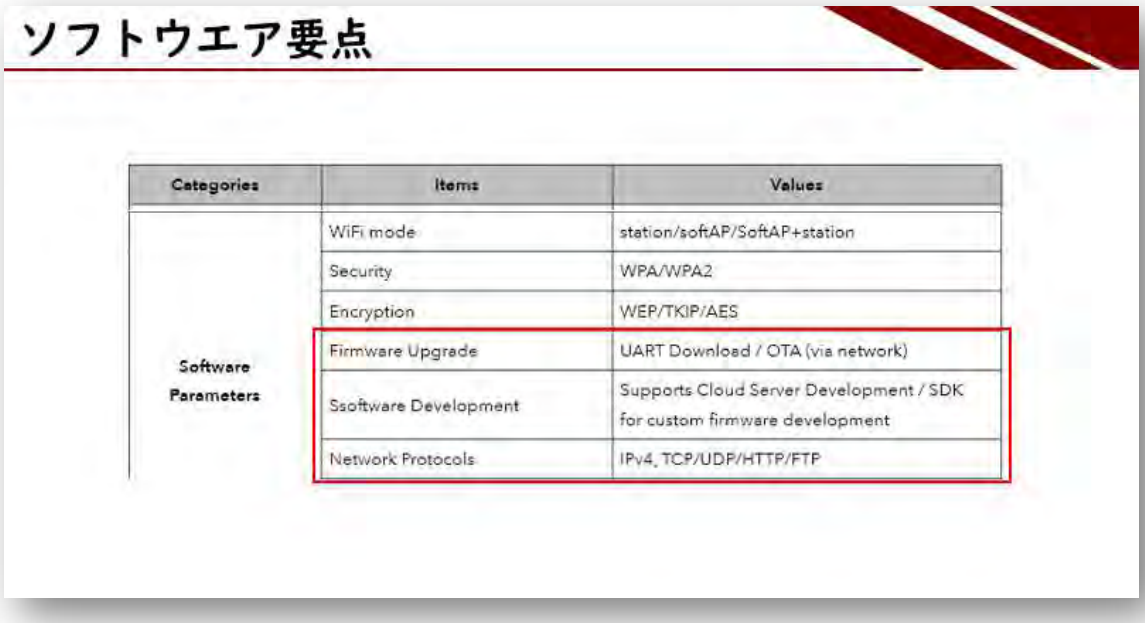


図 282

マイコンファームウエア（プログラム）の書換えは PICAXE と同様のシリアル通信によるか、OTA(Over The Air)という無線を使う方法がありますが、今回はシリアル通信による書換えを使用します。

図 283

第 13 回で開発したモータ制御ユニットを外部から制御するには、人が操作している SW の信号を、第三のマイコンで制御することで目的が達成できます。このマイコンで SW の信号を操作します。SW から右側は既に完成しているので、今回の開発対象は図の ESP8266 だけです。ESP8266 となっているのは ESP-WROOM-02 が使用するマイコンモジュールです。

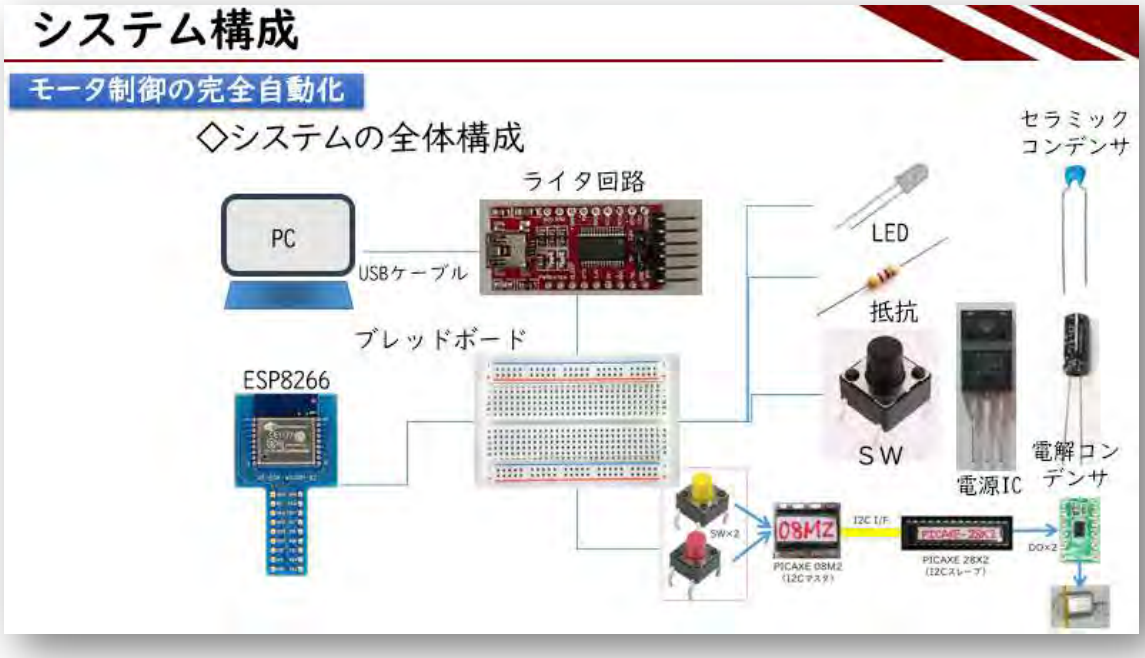


図 284

ESP8266 の電源 3.3V を 5V から作り出す電源 IC とコンデンサもパーツに含まれています。図に示す抵抗の絵は 1 つですが、実際は複数種類の抵抗値を使用します。多数のパーツを使うので回路図をよく見てパーツを揃えてください。

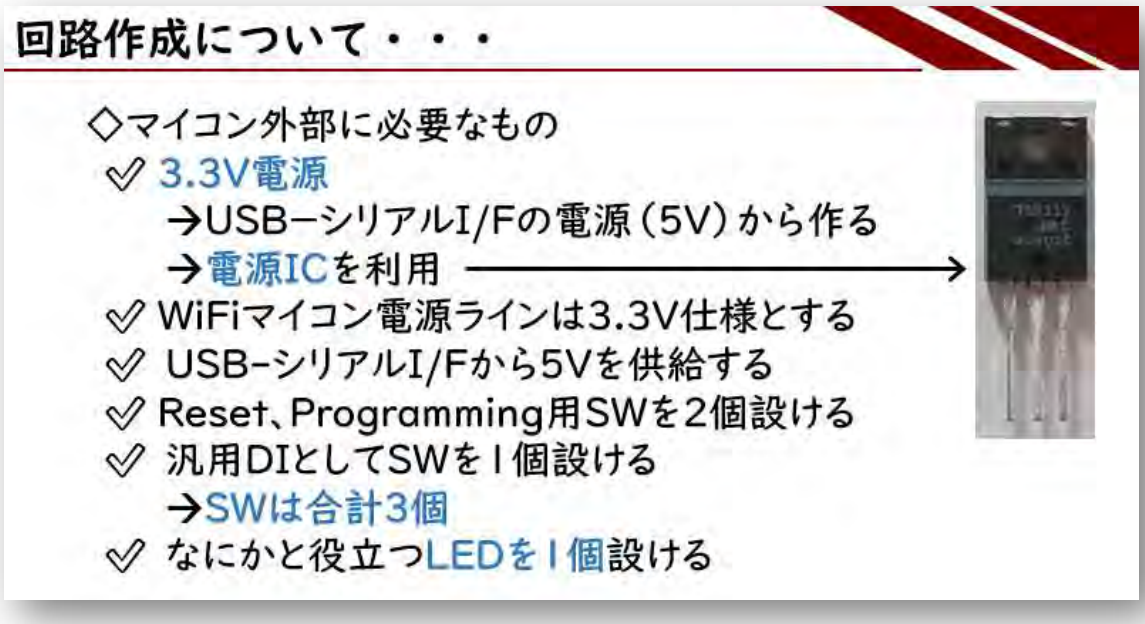


図　285

マイコン電源は電源 IC を使用して USB－シリアル I/F の電源（5V）から 3.3V を作ります。SW はマイコンの Reset 用、プログラム書込み操作用、汎用 DI として合計 3 個使います。何かと役立つ LED を 1 個設けておけば便利です。

図 286

図に電源 IC の仕様を示します。ESP8266 の WiFi 機能を使用すると電源消費が大きくなる事と、ここから外部デバイスの電源を取っても良いように、出力 500mA となっていて大きめです。元になる 5V はこのマイコン専用の USB－シリアル I/F から供給します。マイコン以外にデバイスを使用しないのであれば、PICAXE 用 USB－シリアル I/F から電源を取ってもよいでしょう。

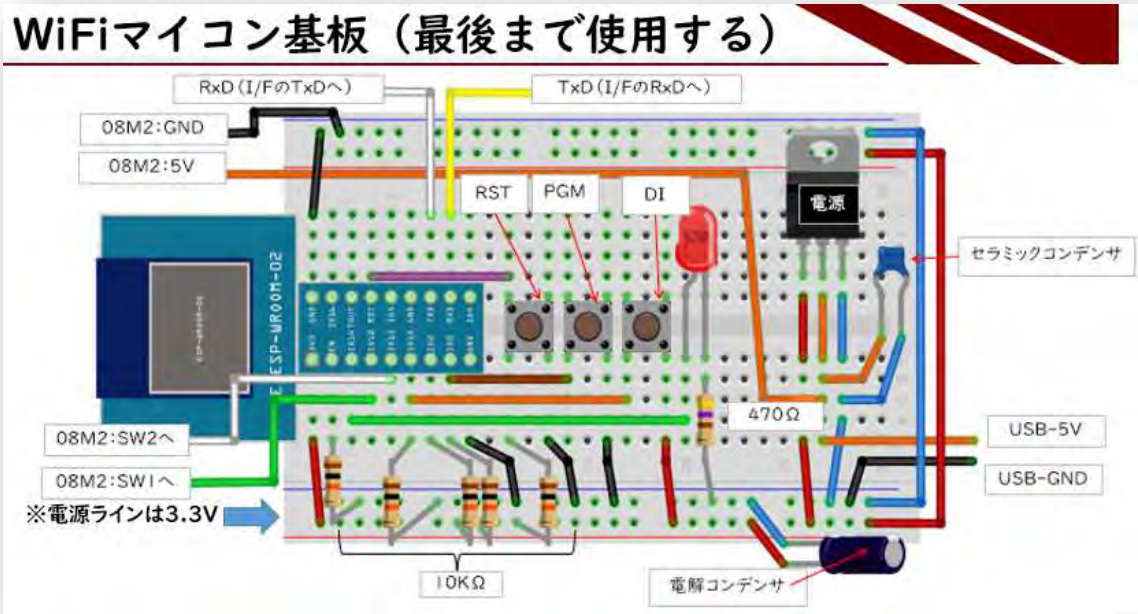


図 287

多くのパーツを搭載したWiFiマイコン用回路は、図の様になります。図左下の【08M2：SW1・SW2へ】は、全体を接続する際に配線します。ボード上が混雑しますので丁寧に作成してください。回路を恒久的に作成するのであれば、抵抗やコンデンサなどはリード線（脚）を必要な長さに切って配置します。電源ICへの入力5VとGNDはUSB－シリアルI/Fから分岐して使用します。このマイコン外部に他のデバイスを接続して電源を供給しようとする場合は、必用な電流の総量に注意しましょう。専用のアダプタ等を用意して、そこから電源ICに5Vを供給することも検討する必要があります。TxDとRxDはUSB－シリアルI/F基板のそれぞれRxDとTxDに接続することを忘れない様にしてください。WiFiマイコンの送信（TxD）はUSB－シリアルI/Fの受信（RxD）になります。

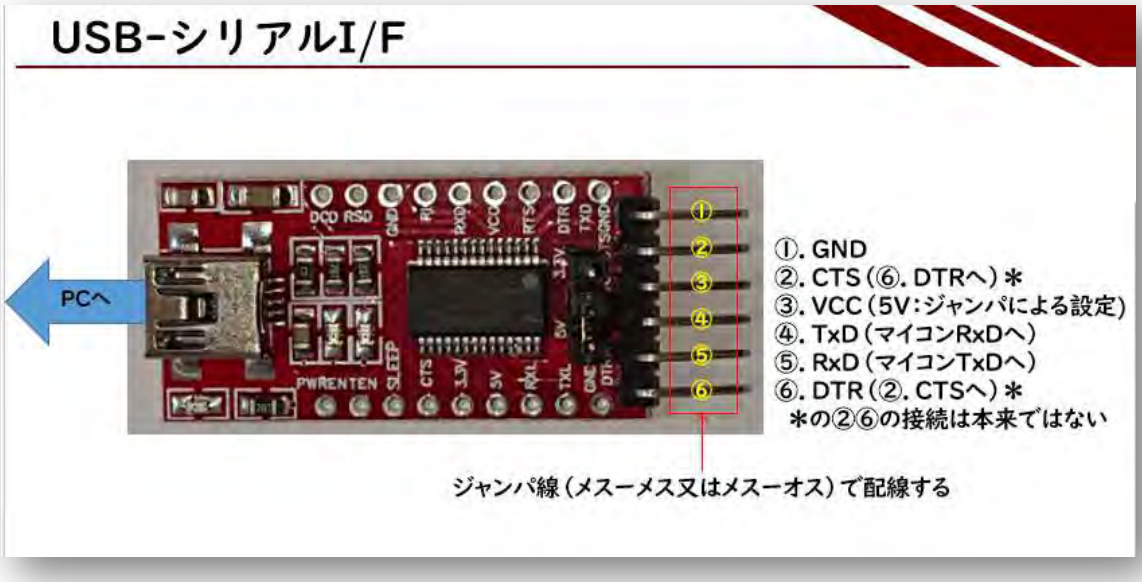


図 288

WiFi マイコン用 USB－シリアル I/F を示します。基板右側にピンがあります。③VCC は、基板上のショートピンで 5V に設定されています。この 5V を取り出して WiFi マイコンの電源として使用します。図で＊の付いている②⑥は正式なシリアル通信の使用方法ではありませんが、この信号自体は USB－シリアル通信上では未使用なので、この様に②⑥を接続します。本来は⑥DTR（Data Terminal Ready）は DSR（Data Set Ready）と対応し、②CTS（Clear To Send）は RTS（Request To Send）に対応します。これらは、シリアル通信を行う前のやりとりに使う信号で、【準備が出来ました→了解】というような意味合いです。

※実習キットに含まれるこの基板は既に②⑥は接続済みです。

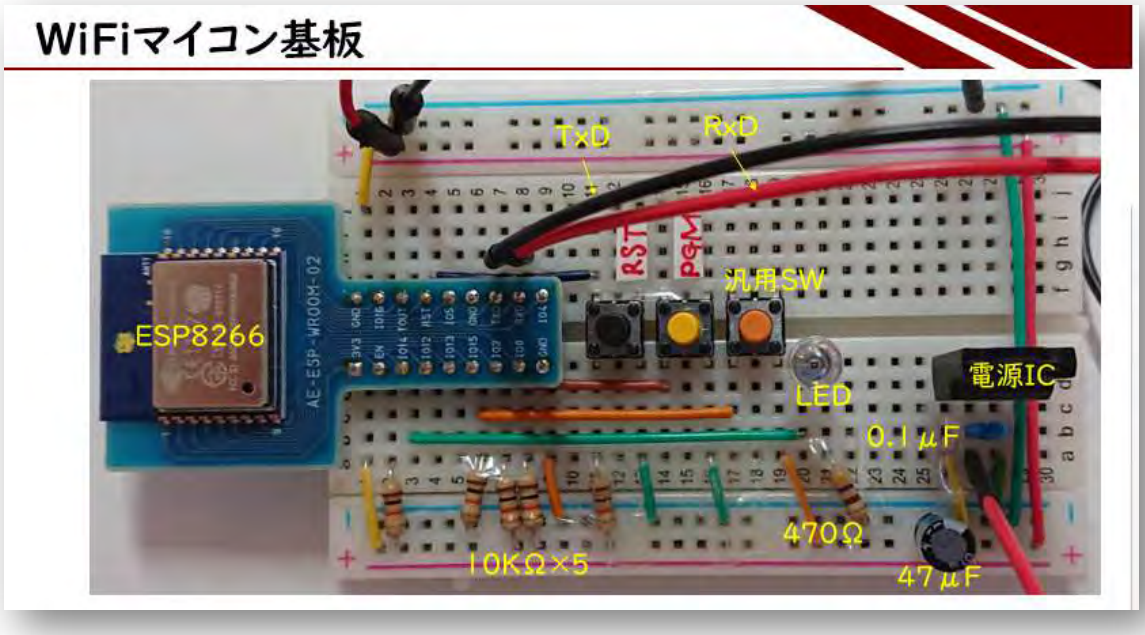


図 289

完成した WiFi マイコン基板を示します。抵抗の種類と接続先、電源 IC の入力、TxD と RxD の接続先に注意してください。

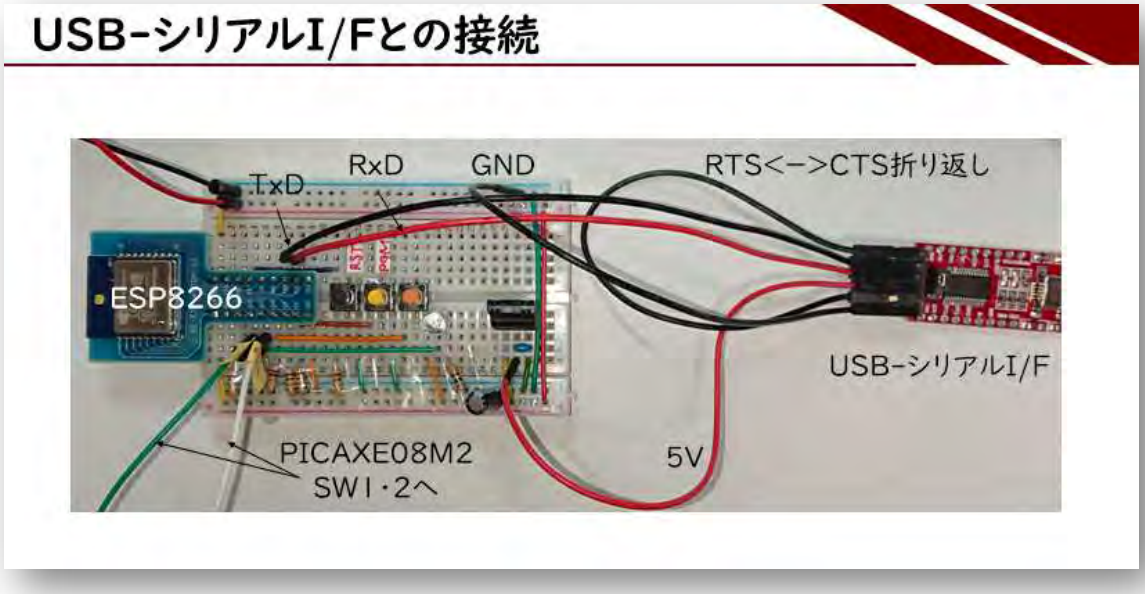


図 290

USB－シリアル I/F とマイコン基板を接続すると図の様になります。図の左下（PICAXE08M2 SW1・2 へ）は、全体を接続する際に、PICAXE08M2 基板の SW のマイコン入力側に接続して下さい。

図 291

WiFi マイコン用のプログラムを開発する環境 Arduino IDE を準備します。WEB で【Arduino】と検索してヒットするページの SOFTWARE を辿ります。

図 292

Download ページから Windows Installer をダウンロードしてください。図に示すバージョンは既に変わっているかもしれません。

図 293

ダウンロードしたファイルを開き、全て既定のままでインストールします。特にデバイスドライバなどは、全てインストールするようにしてください。終了するとデスクトップに図の様な眼鏡マークのショートカットができています。IDE を起動すれば図右のようなウインドウが開きます。ソースコードは IDE の白い部分に記述します。

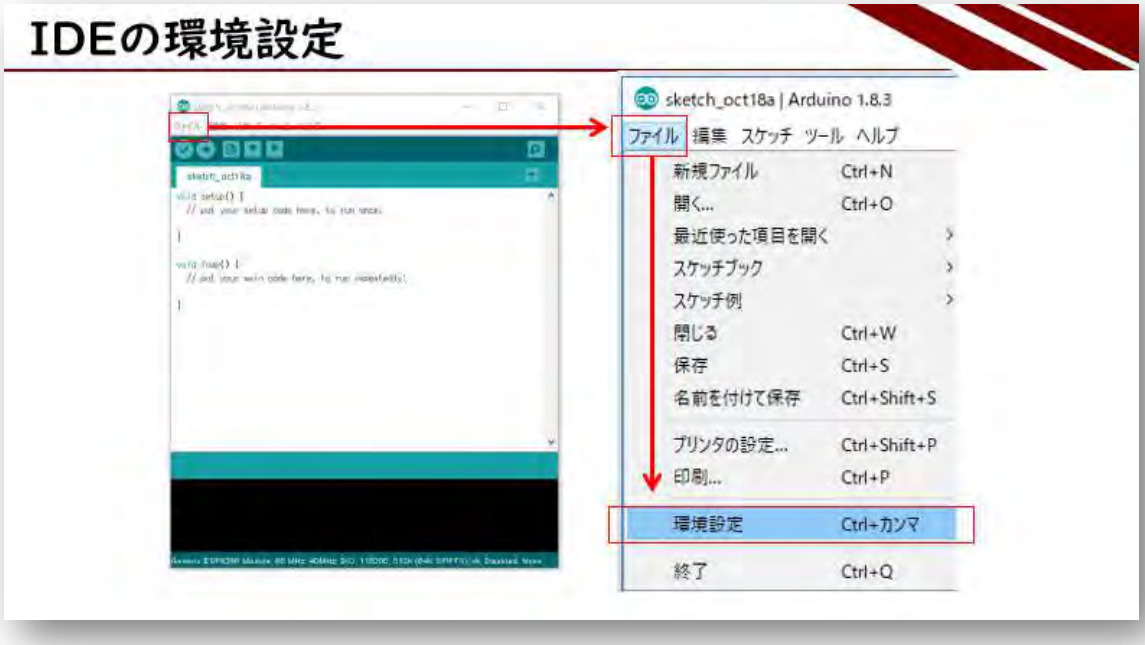


図 294

この IDE を使用するには、環境設定が必用です。図の様にして環境設定を開きます。

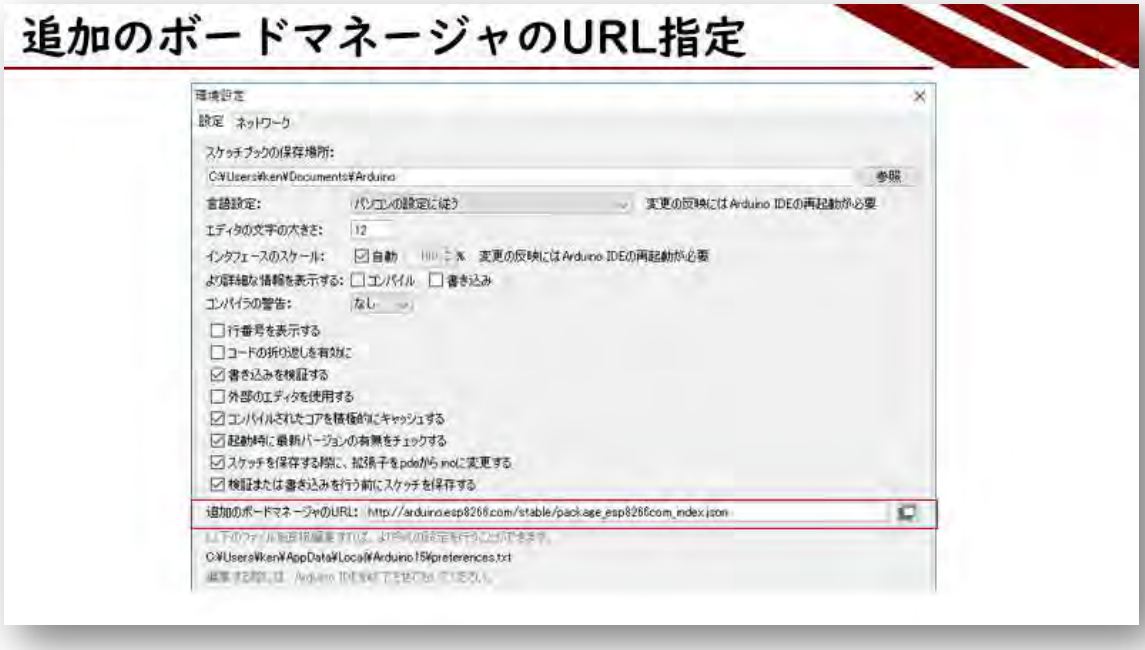


図 295

環境設定の追加のボードマネージャの URL に次頁の URL を記述します。

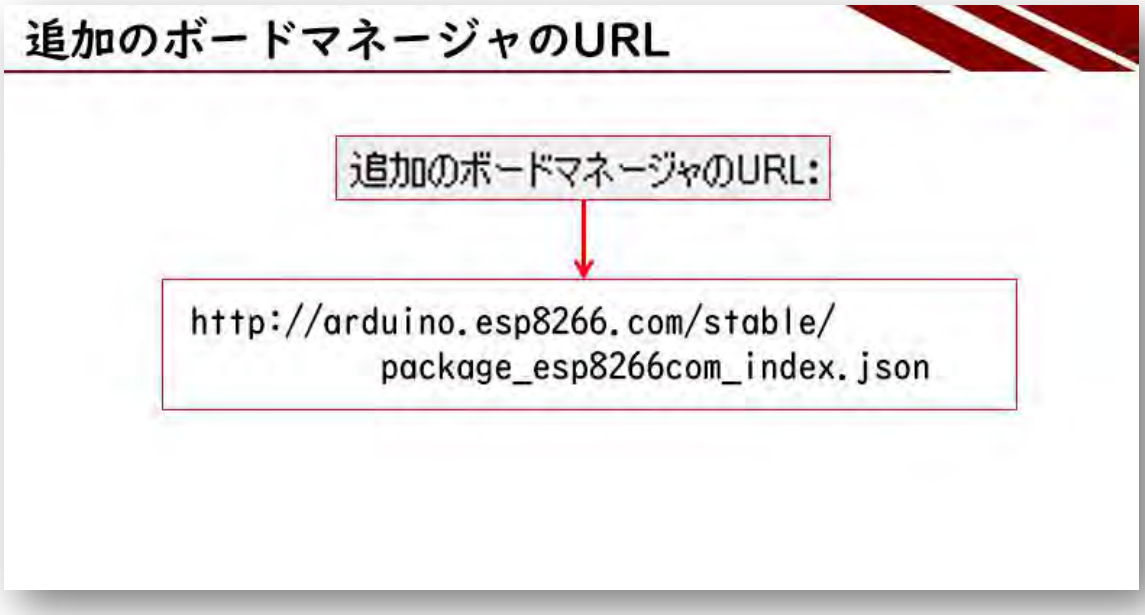


図 296

この URL は WiFi マイコン（ESP8266）のライブラリ等を IDE から取得するために必要な設定です。正しく入力してください。

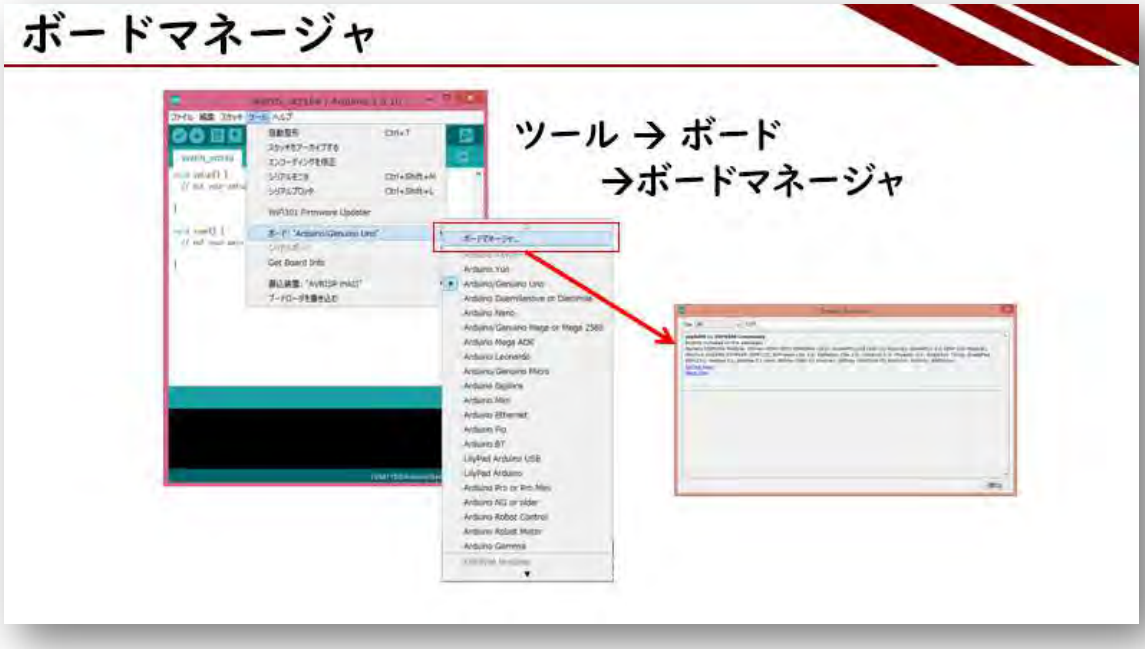


図　297

WiFi マイコン用にボードを追加します。図の様にボードマネージャを開きます。

図 298

開いたウインドウの上部にある検索窓に ESP と入力してしばらくすると、該当するマイコンボードがヒットします。

図 299

図に示す【esp8266 by ESP8266 Community】を選択して Install ボタンを押下します。終了するとボード名の隣に INSTALLED と表示されます。

図　300

このボード用のコンパイルを指定するために、図の様にして【Generic ESP8266 Module】を選択します。これで環境の準備は整いました。

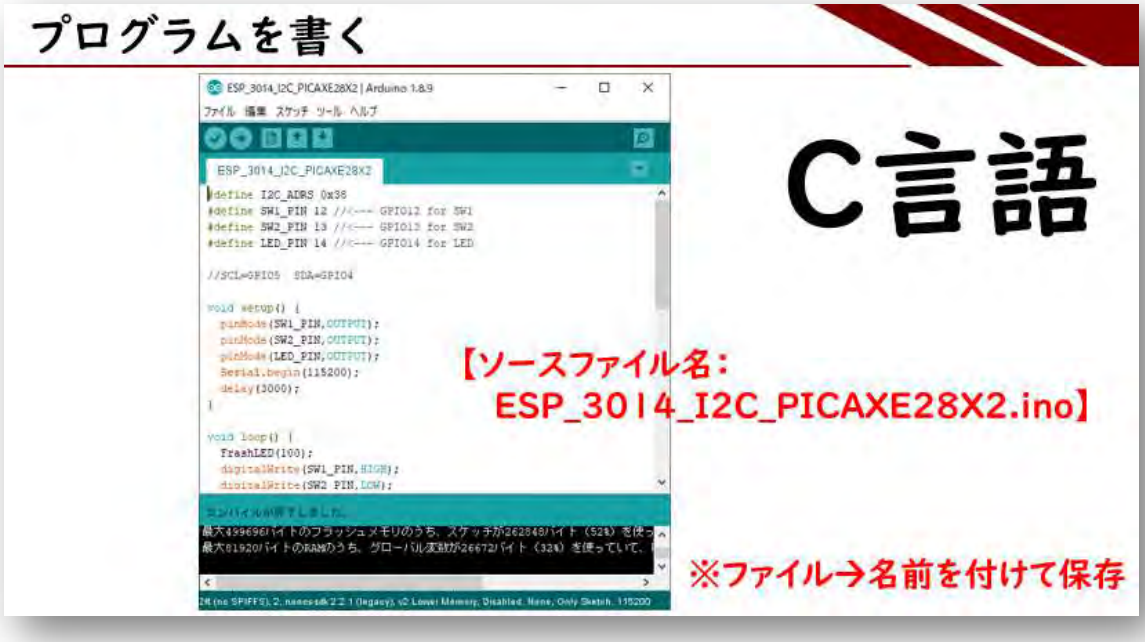


図　301

図のようにIDEにソースコードを記述します。実習キット付属のCDには、図のファイル名でソースファイルが含まれています。

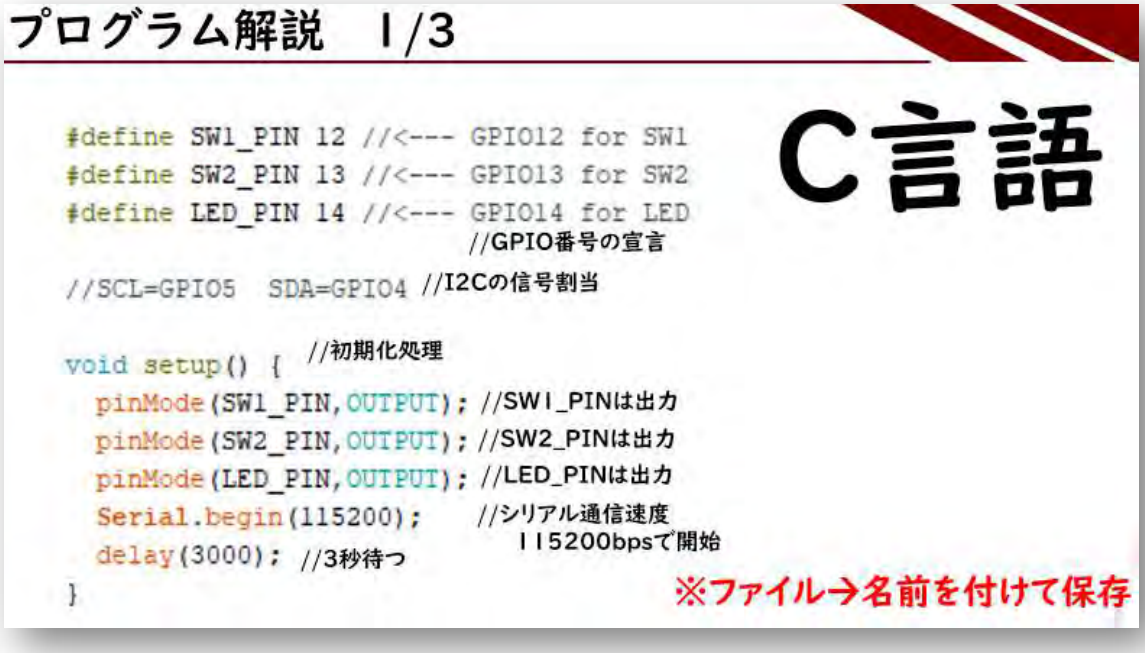


図　302

プログラムはC言語で記述します。Arduino IDEでは、プログラムをスケッチと呼びます。**ファイル→新しいスケッチ** で新規プログラムを作成すると、中身が空のsetup()とloop()という関数が作られています。先頭の4行（#define）は、シンボル（名称）の定義です。I2Cのスレーブアドレスも定義しておきます。残り3行は汎用I/Oの番号です。なおI2C I/FのSCLとSDAはそれぞれGPIO5とGPIO4を使うことがコメントに書かれていますが、プログラムに出てくることはありません。setup() は初期化関数で、GPIOの用途（DIかDOか）とシリアル通信速度を指定します。ここに記述したGPIOの番号はマイコン基板のピン番号とは異なります。

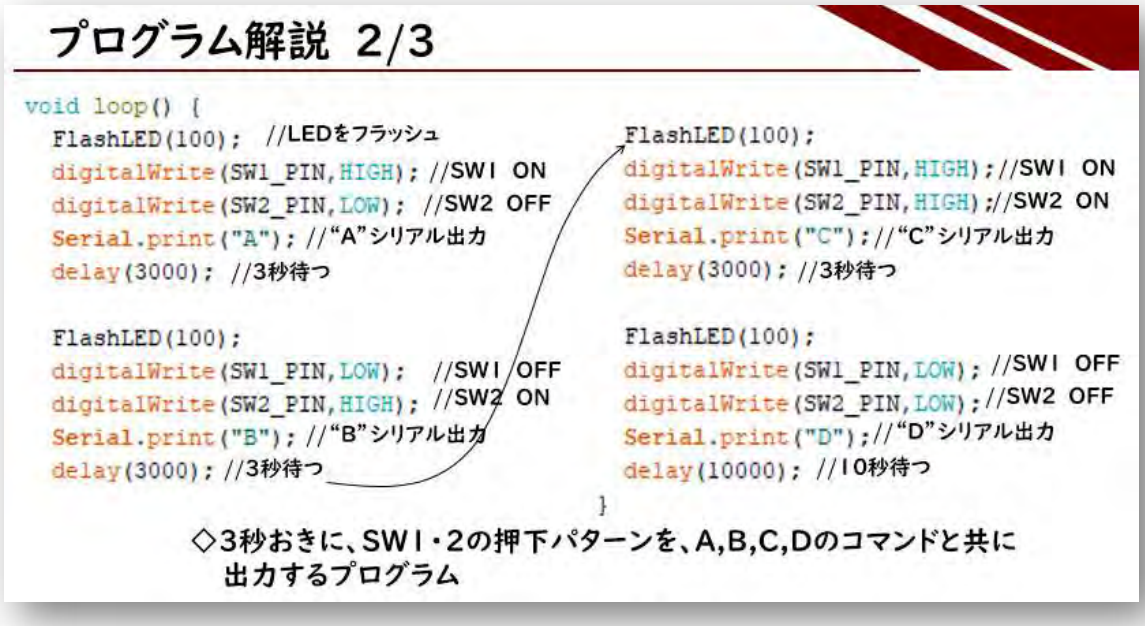


図 303

loop()は通常の処理を行う関数で、繰り返し実行されます。プログラムは、一定間隔で SW の押下パターンを切り替えて出力する処理を行います。一組の SW 押下パターン出力が終えたら 10 秒間待ちます。FlashLED()は LED を一瞬光らせる関数です。

図　304

FlashLED()では、一度 LED を点灯して、パラメータ t で渡された時間（ms 単位）待って LED を消灯します。

プログラムが完成したら名前を付けて保存しましょう。

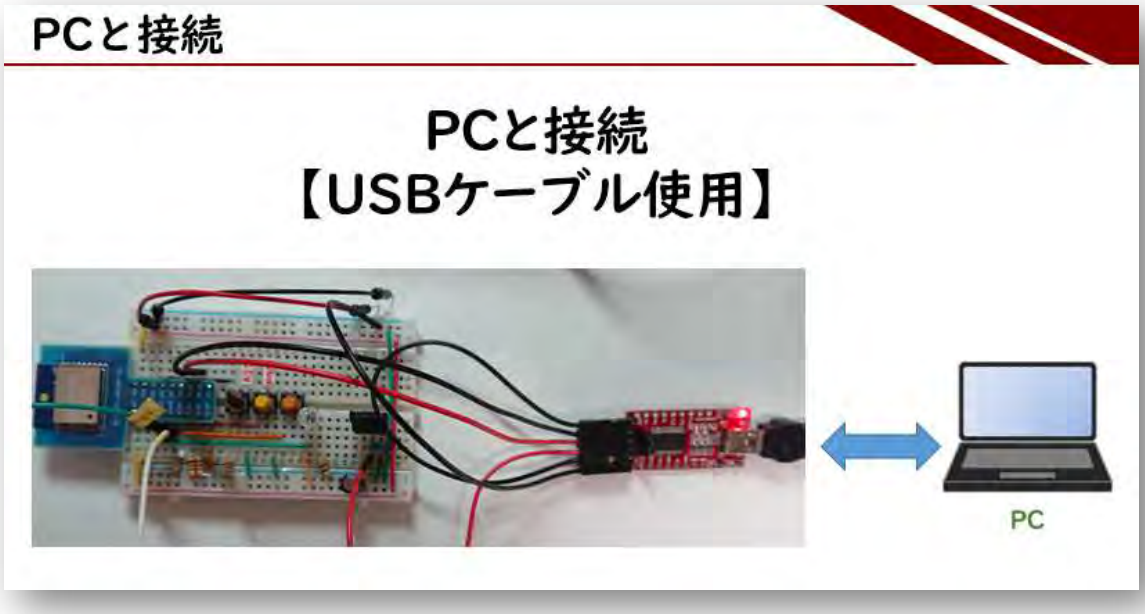


図 305

プログラムのコンパイルとマイコンへの書込みは続けて行われます。システムを PC と接続します。USB ケーブルで電源が供給されると USB－シリアル I/F 基板の LED が点灯します。初めて PC と接続すると、USB－シリアルドライバがインストールされます。メッセージなどが表示されるかもしれません。その際は、全て【Yes・OK・はい】で進めてください。

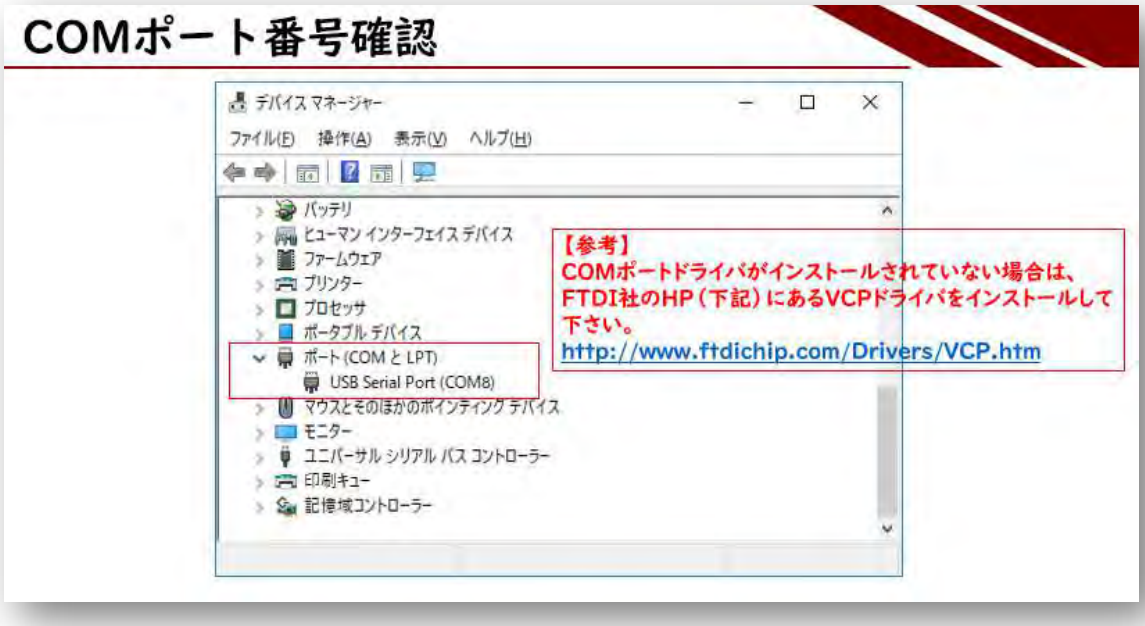


図 306

Windows デバイスマネージャで COM ポート番号を確認します。もし COM ポートが現れないようでしたら、図に示す URL から VCP ドライバを取得してインストールして下さい。

図 307

図の様にして IDE で、該当の COM ポートにチェックを入れます。

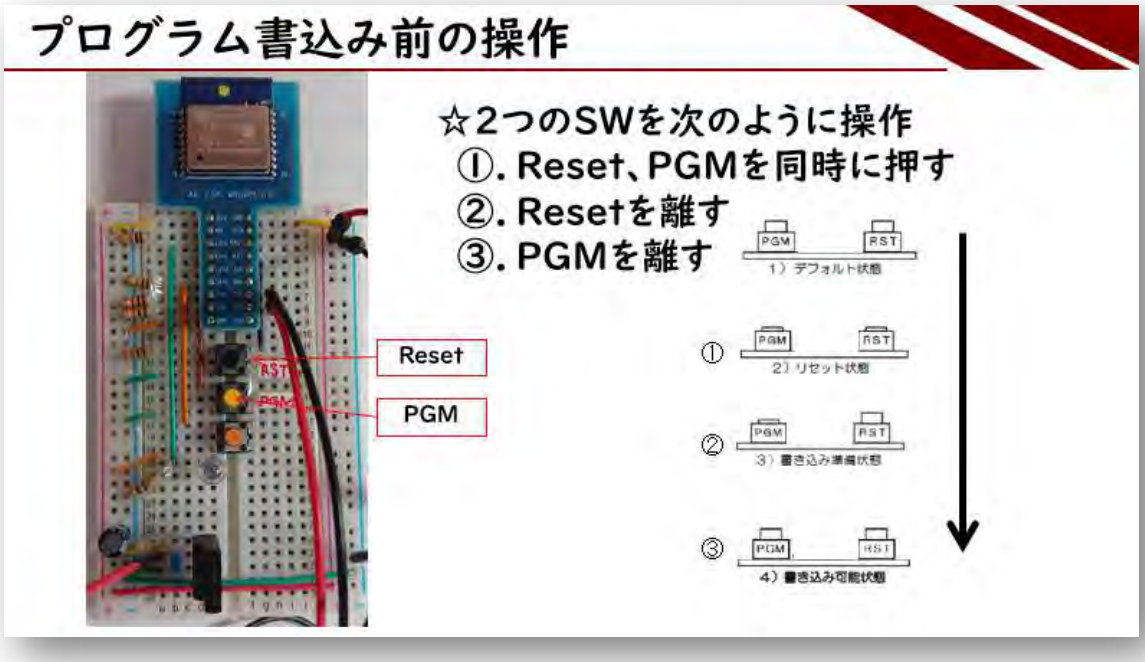


図 308

WiFiマイコンには書込みの準備をさせる必要があります。図の様にボタンを操作して書込み準備をして下さい。この操作が正しく行われていない場合は、書き込みでエラーが発生します。

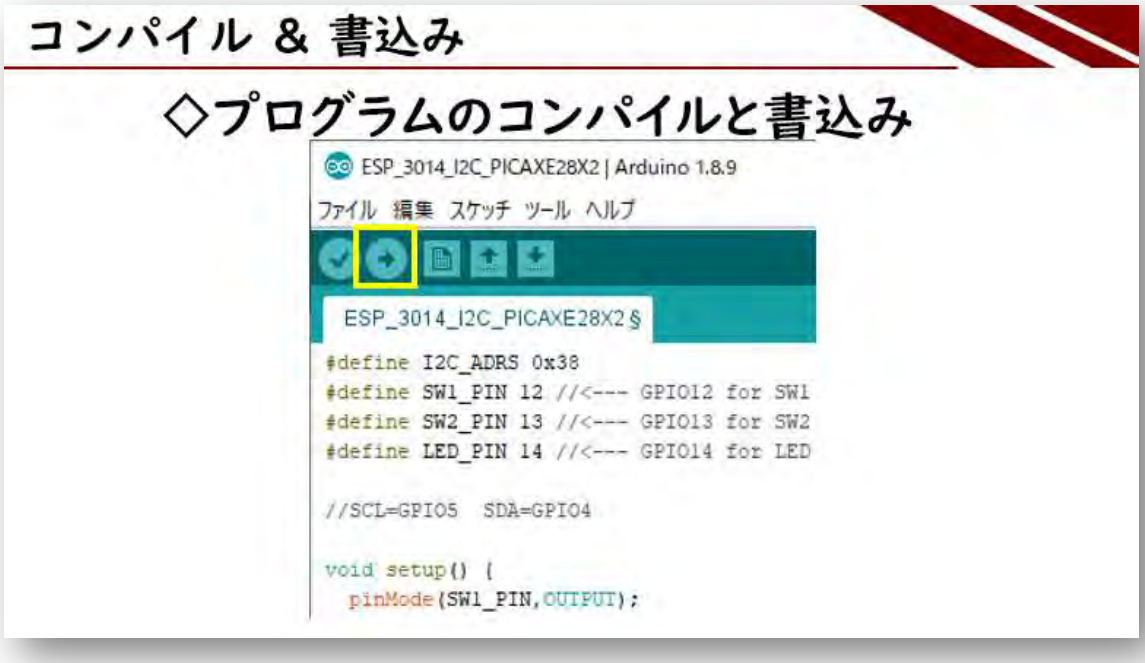


図 309

コンパイルと書込みは、IDE 上部にある右向き矢印ボタンを押下して行います。

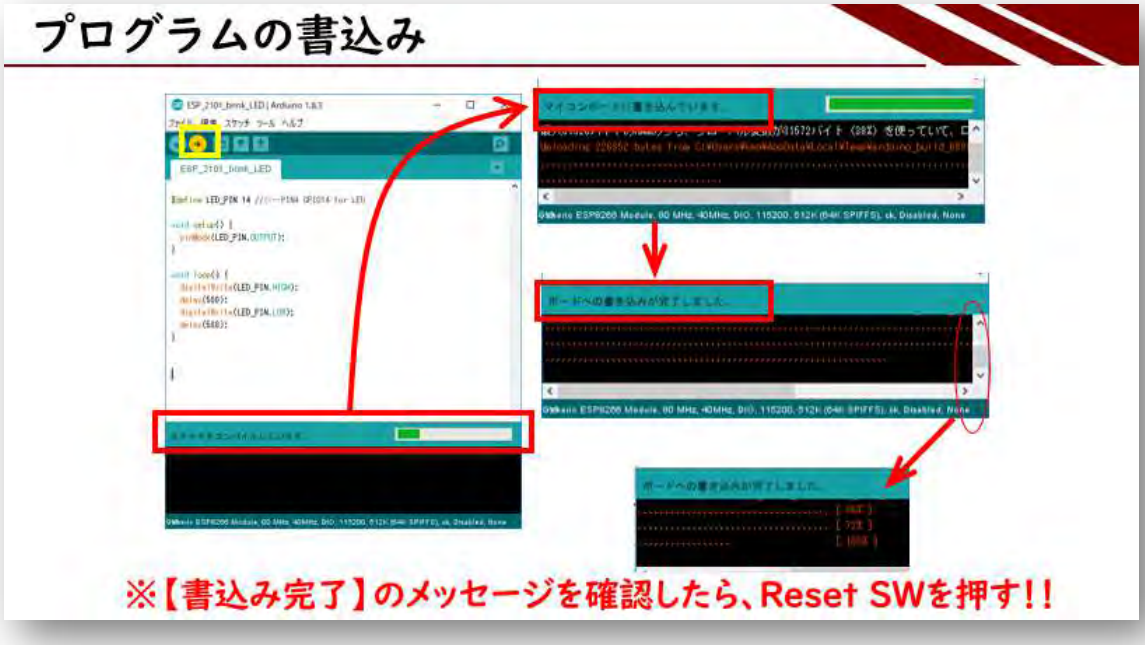


図 310

コンパイルでエラーが無ければ、書き込みに進み、終了するとその旨メッセージが表示されます。書込み完了のメッセージを確認したら、マイコンの Reset SW を押下してください。

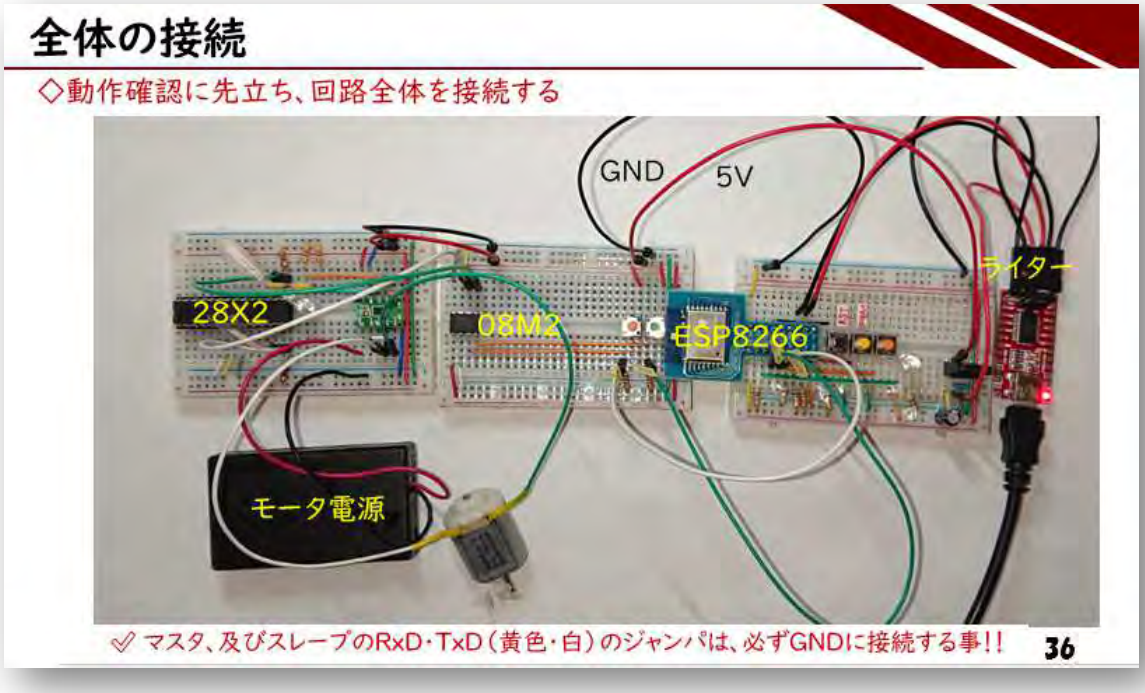


図 311

動作確認に先立ち、システム全体を接続します。2 枚の PICAXE 基板の TxD、RxD は共に GND に接続して下さい。図 288 左下を確認して ESP8266 から PICAXE08M2 の SW1・2 への配線を行って下さい。この信号が SW1・2 をシミュレーションしています。全ての確認を終えたら最後にモータ電源を ON にします。動作確認が終ったらモータ電源を OFF にすることを忘れない様にしてください。

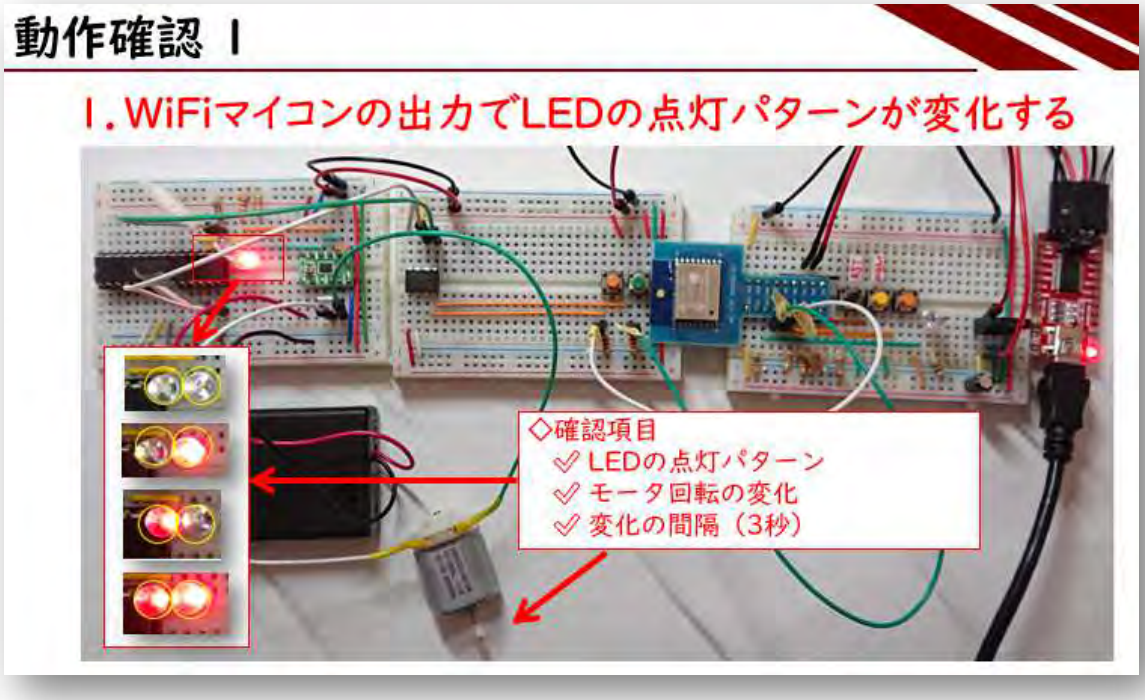


図 312

【PICAXE 基板は前回までに動作確認が取れている】のを思い出して動作確認します。USB ケーブルを PC に接続すると動作が始まります。そのまましばらく様子を見ます。LED の点灯パターンが一定間隔で繰返されることを確認します。LED が点灯しない場合は ESP8266 と PICAXE08M2 間の配線を確認してください。その後 LED の点灯パターンとモータ回転の様子がどうなっているか確認します。

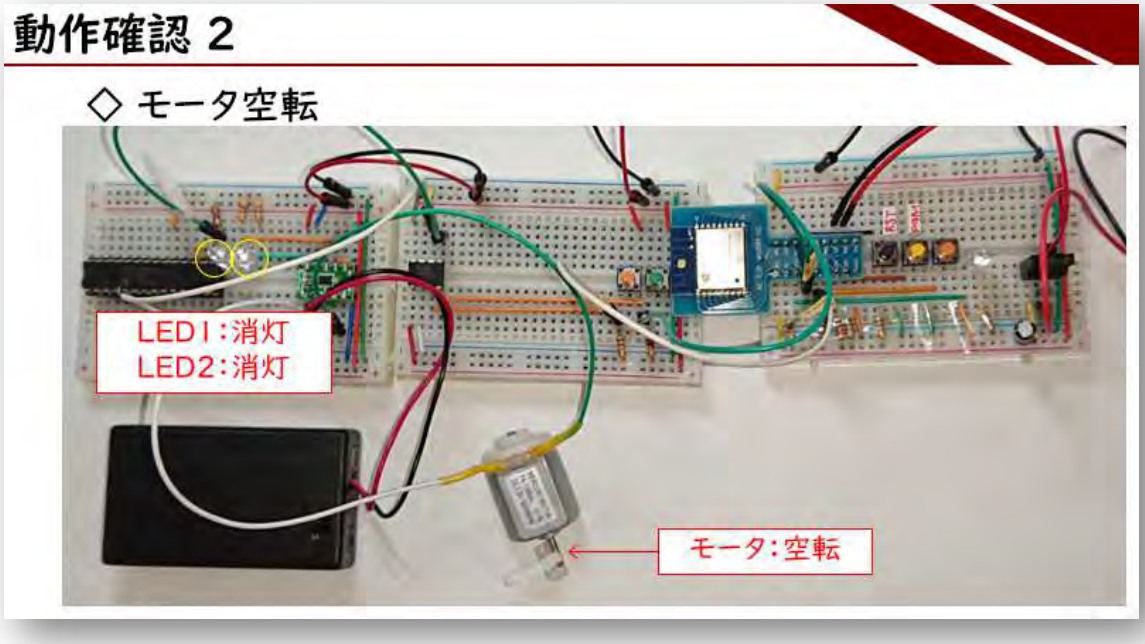


図 313

LED が消灯している時モータは空転状態です。

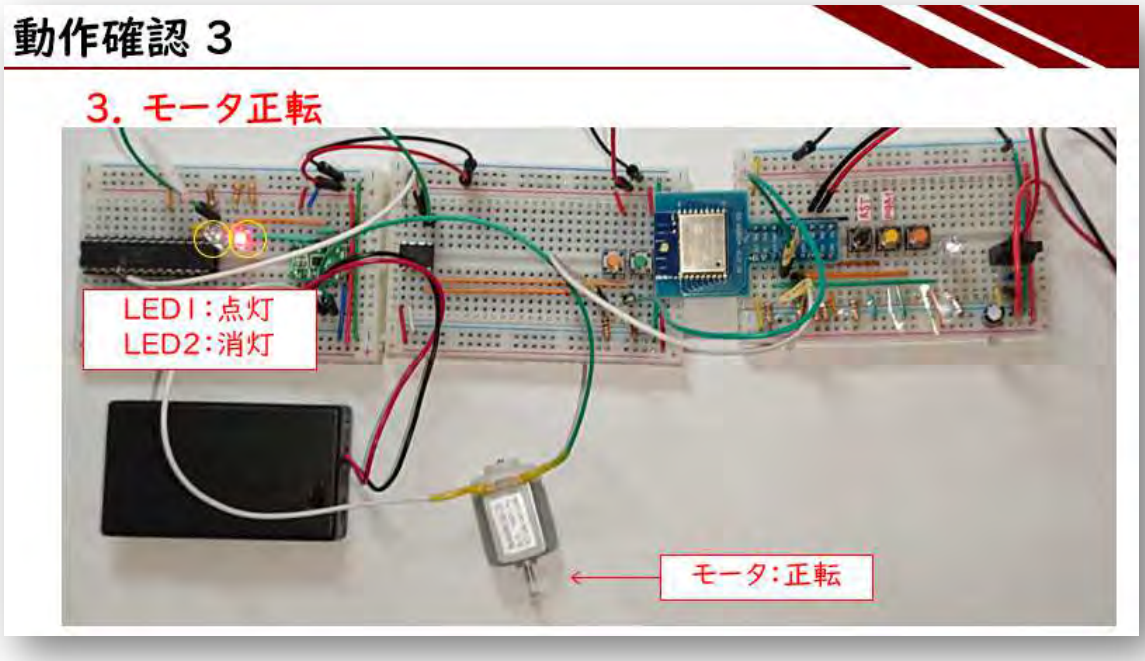


図 314

LED1：点灯、LED2：消灯の場合、モータは正転します。

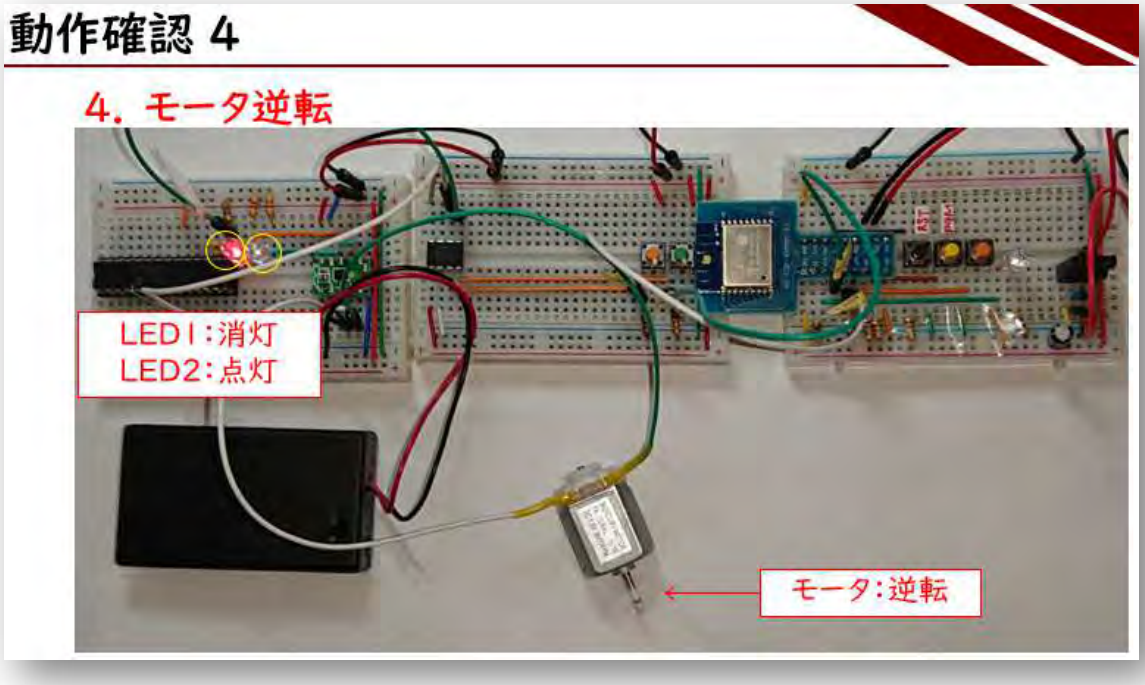


図 315

LED1：消灯、LED2：点灯の場合、モータは逆転します。

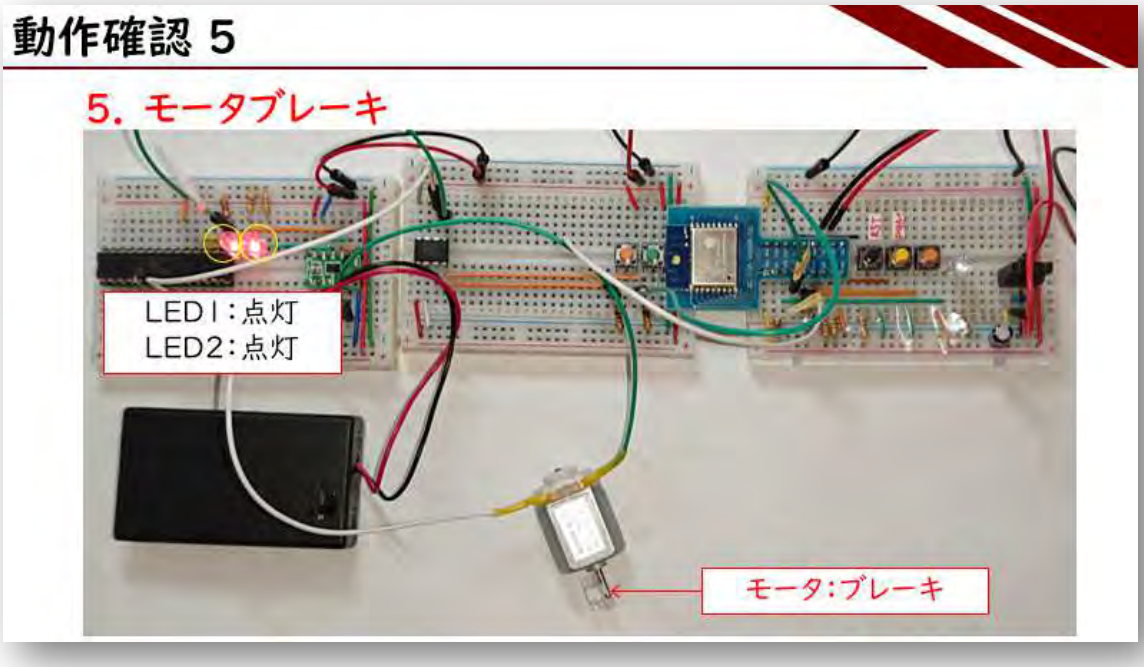


図 316

LED1・LED2 共に点灯すると、モータはブレーキ状態です。

これらの動きを一定時間ごとに繰り返します。この動作の最上流工程は、ESP8266（WiFi マイコン）からの出力信号（DO）です。このマイコンが持つ WiFi 機能を活用して WEB 経由通信を行えば、遠隔地からのモータコントロールができます。

最後にモータ電源を OFF します。

第15回　遠隔モータ制御

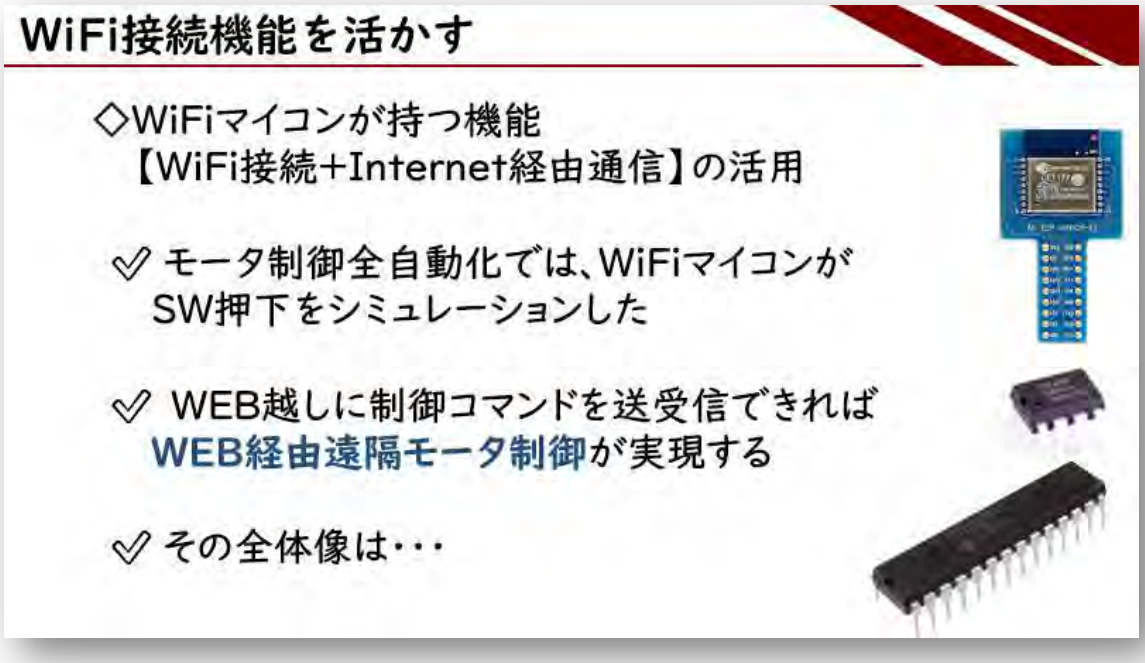


図 317

ESP8266 の WiFi 機能で WEB 経由通信を行います。モータ制御コマンドを WEB 経由で送信・受信すれば、前回開発したモータの自動制御を遠隔制御に展開できます。

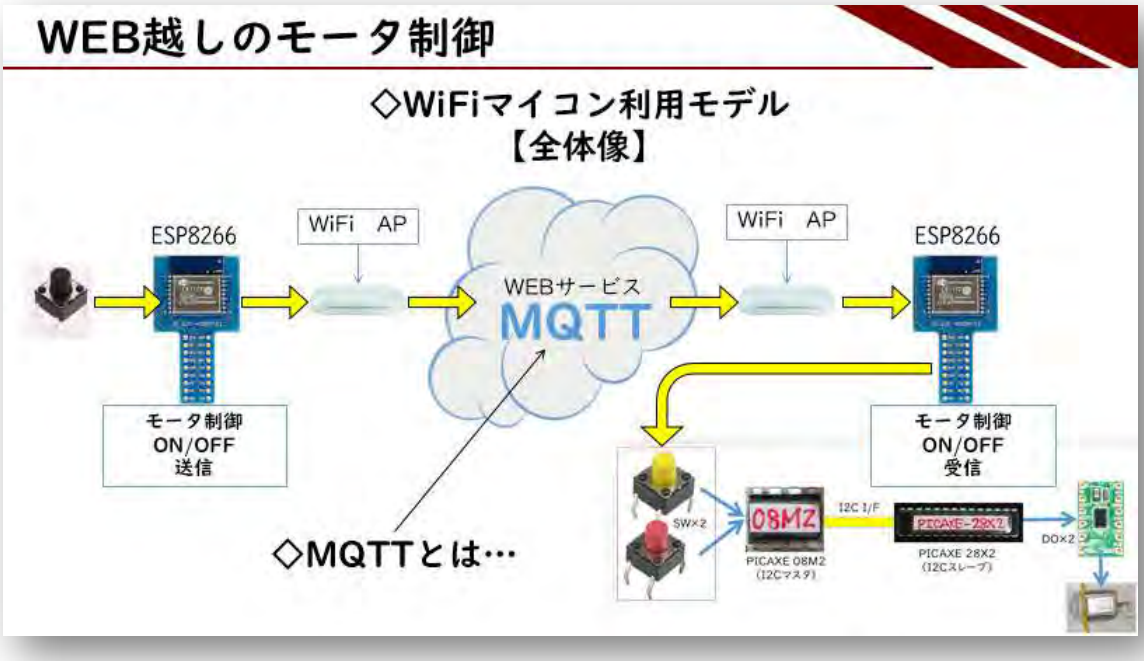


図 318

図は全体の機能ブロックです。図左の ESP8266 がモータ制御の ON/OFF コマンド(0/1)を送信します。コマンドは WiFi Access Point 通り、MQTT という WEB サービスを経て反対側の WiFi Access Point を通って図右の ESP8266 に受信されます。受信した制御コマンドを解釈して SW の ON/OF 信号に変換します。SW の ON/OFF 信号は I2C マスタで再びコマンドに変換されて I2C 通信でスレーブに通知されます。I2C スレーブは受信したコマンドに対応するモータ制御信号をドライバに出力します。その結果モータが回転・停止します。

図 319

前頁で説明の WEB サービス MQTT というのは Message Queue Telemetry Transport のことです。利用開始時に ID 取得や登録などが不要で、WEB を経由してどこにでもメッセージを送ることができ、送信したメッセージを複数個所で受取ることができます。短いメッセージに特化した通信サービスです。このサービスはモータ制御コマンドにピッタリの通信です。

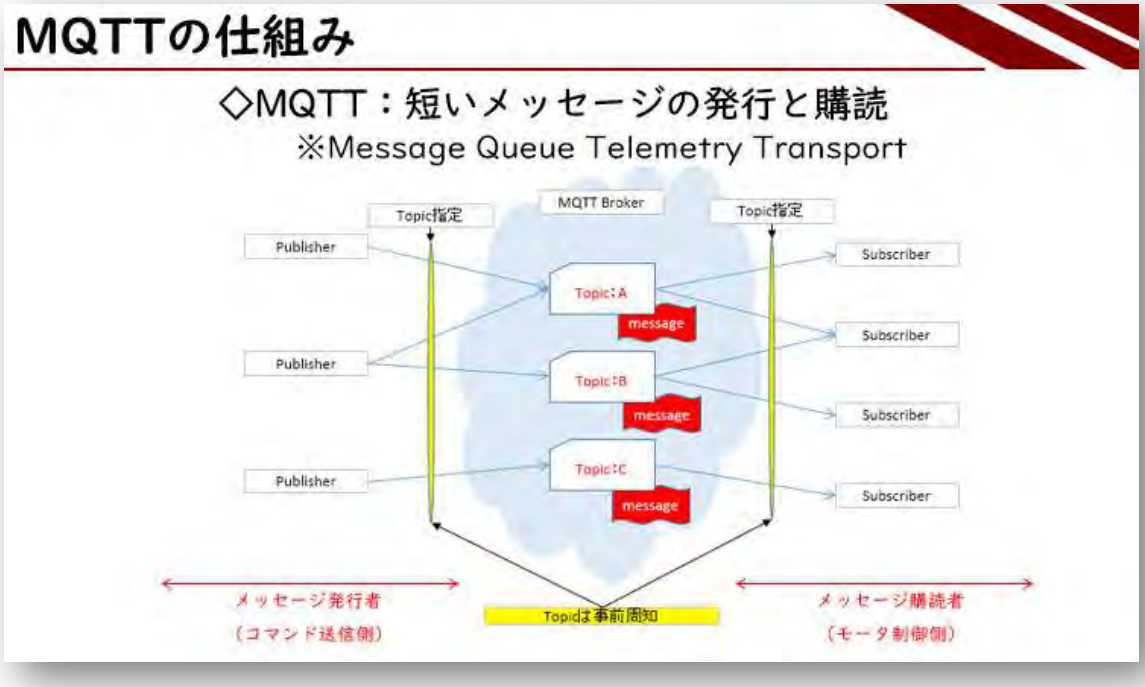


図　320

図は MQTT の仕組みです。メッセージ発行者を Publisher、購読者を Subscriber と言います。左側が Publisher、右側が Subscriber です。メッセージ発行の際、双方で予め決めておいた Topic 名をメッセージに付けて発行します。購読者は読みたいメッセージの Topic 名を指定して講読します。発行者と購読者は、複数の Topic 名を使い分けることができるので、メッセージを読みたい複数の相手に届けられます。購読者は複数の発行者からのメッセージを読むことができます。

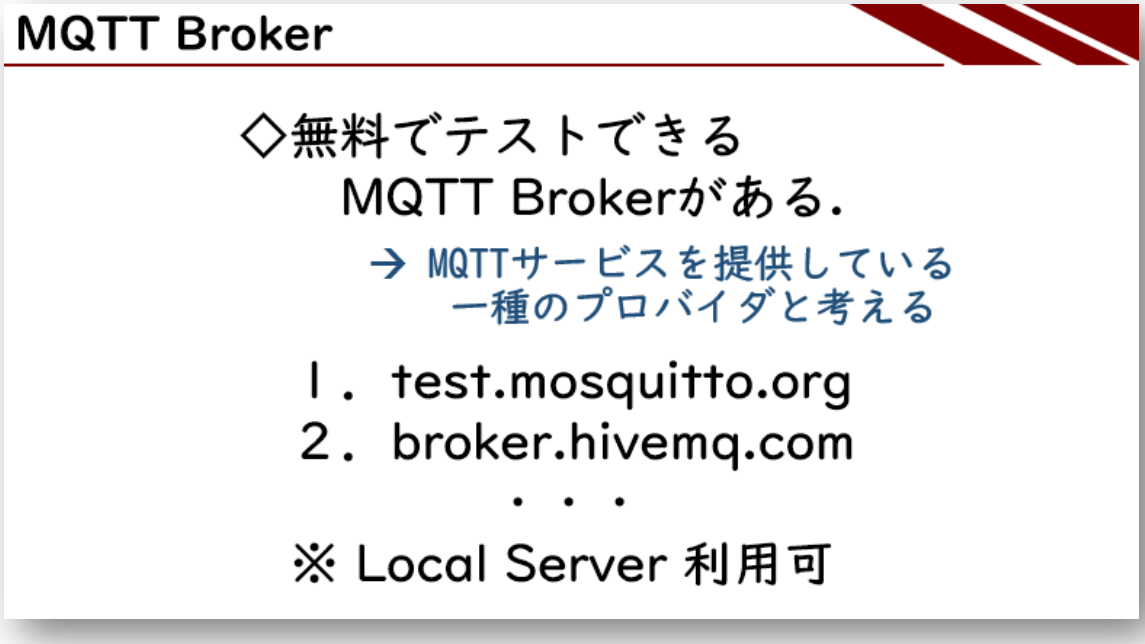


図 321

MQTT サービスの提供者は MQTT Broker と言われています。世界中に多くの Broker が存在していて、WEB で検索すると沢山見つかります。図に 2 つ示したのは代表的な MQTT Broker です。最近日本にも Broker が現れて、有料のサービスを展開しています。本講座では無料で使用できる MQTT Broker を利用して実験を行います。

【参考】

https://blogs.akamai.com/jp/2019/06/iot---iot-egge-connect--.html

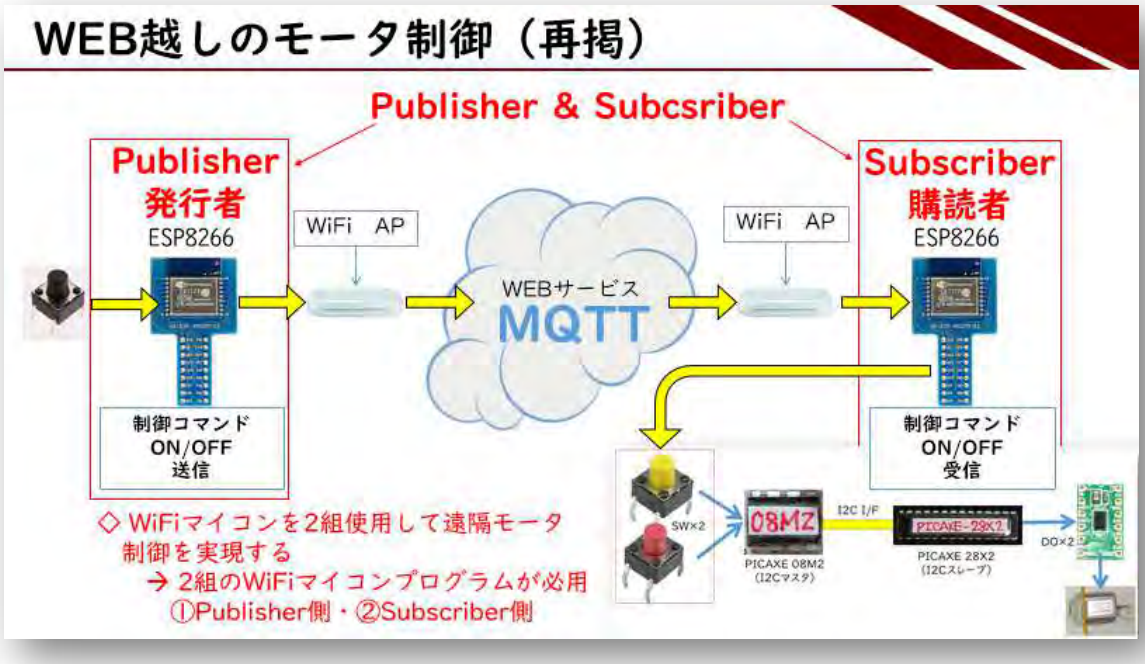


図 322

既に説明した図に Publisher と Subscriber を重ねてみると、今回の実験内容が分かり易いでしょう。WiFi マイコンを 2 組使用し、Publisher と Subscriber の役割を分担してシステムを開発します。実際の開発では、複数の企業が役割分担して開発を進めるのはよくある事なので、実習としてちょうど良いテーマです。

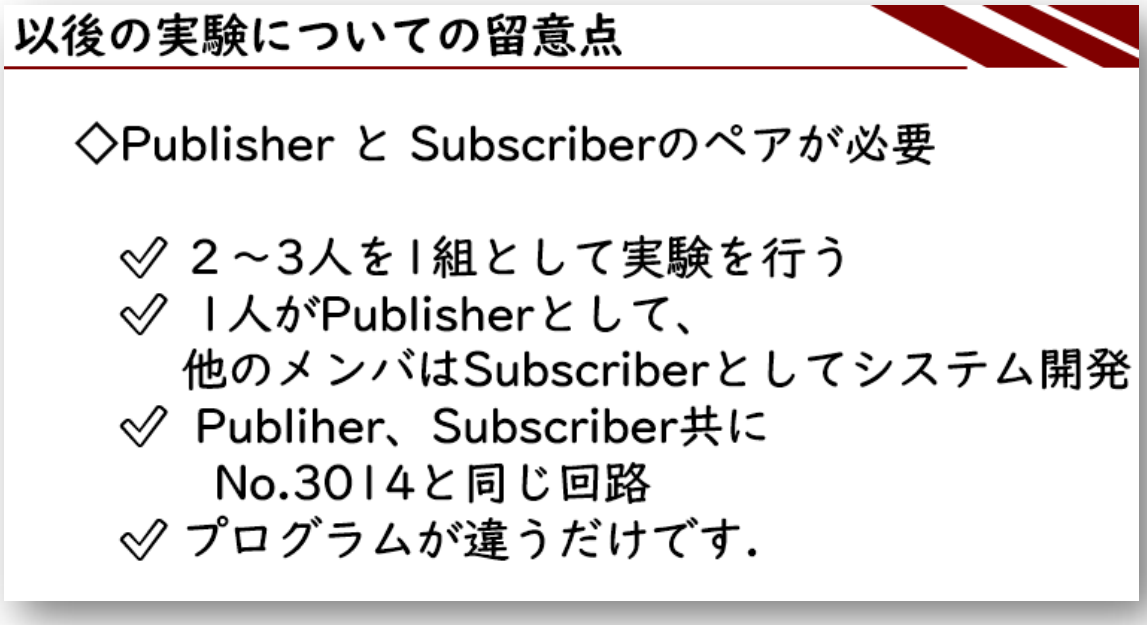


図 323

この実験では WiFi マイコンが 2 個以上必用なので、2～3 人を 1 組として実験を進めます。グループ中 1 人が Publisher、他のメンバが Subscriber という役割でシステムを開発すれば、動作確認が容易でしょう。回路は第 14 回と同じ回路を使用します。WiFi マイコンのプログラムで役割を変えています。

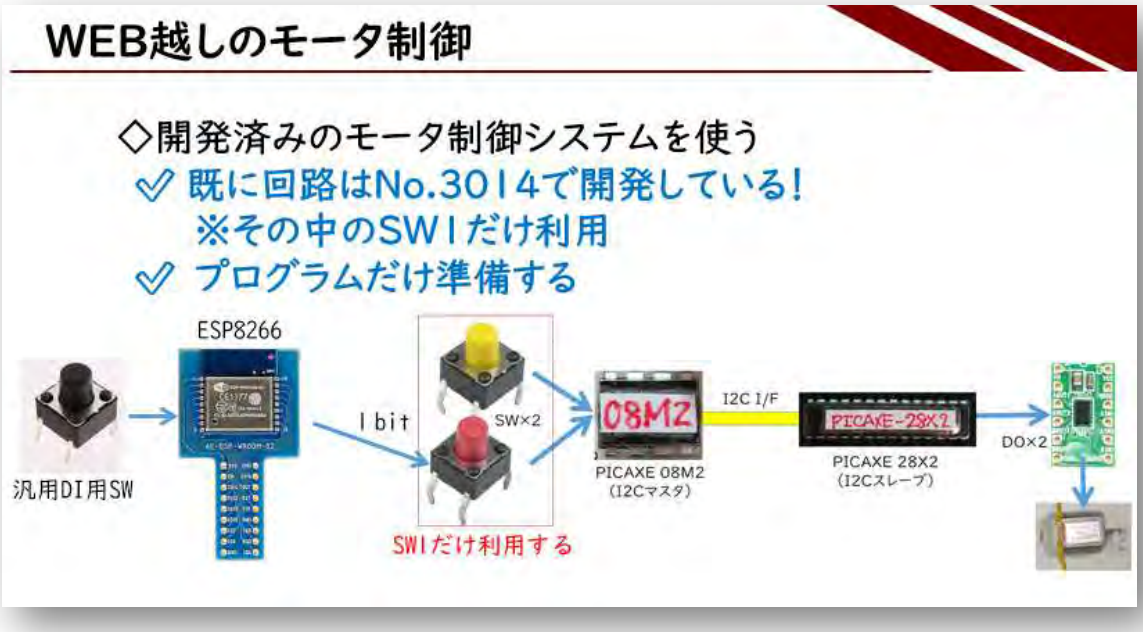


図 324

行うことを整理すると、既に開発済みで動作確認が取れているモータ制御システムを使います。今回は最上流の ESP8266 基板に取り付けた汎用 SW を用いてモータの ON/OFF 制御を行います。ON/OFF なのでモータ制御デバイス側に 1bit の信号が出力されれば SW1 の制御ができます。回路は既にできていますから、ESP8266 の MQTT 対応プログラムを開発すれば良いことになります。

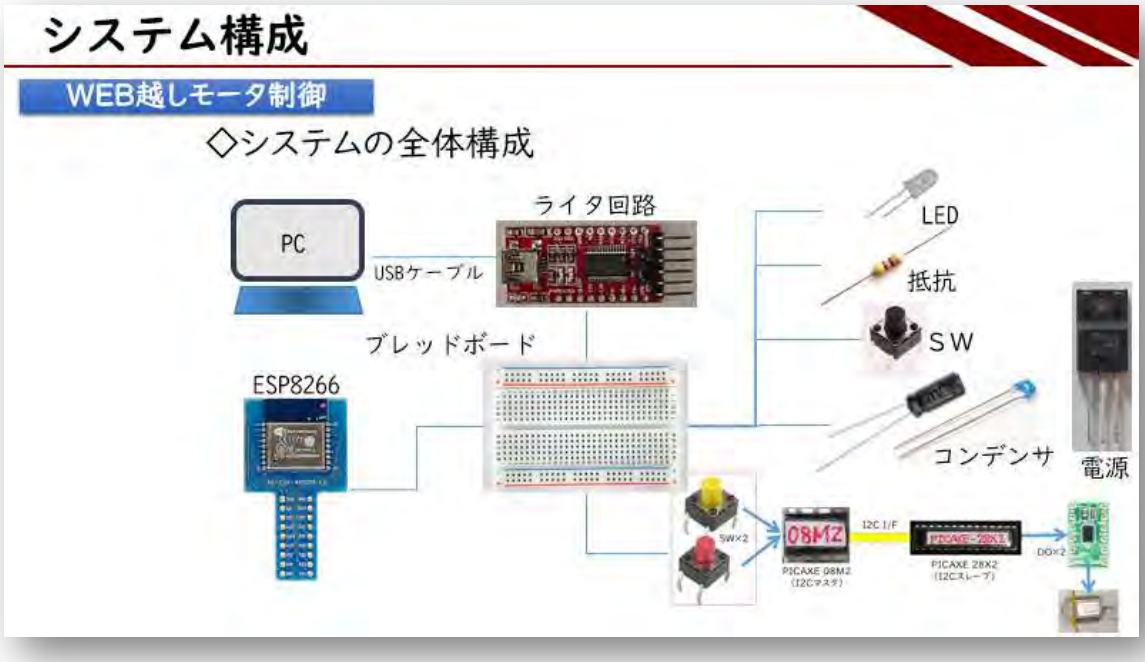


図　325

使用するパーツはこれまでと同じですが、これらを 2 組以上 WEB で接続して使います。

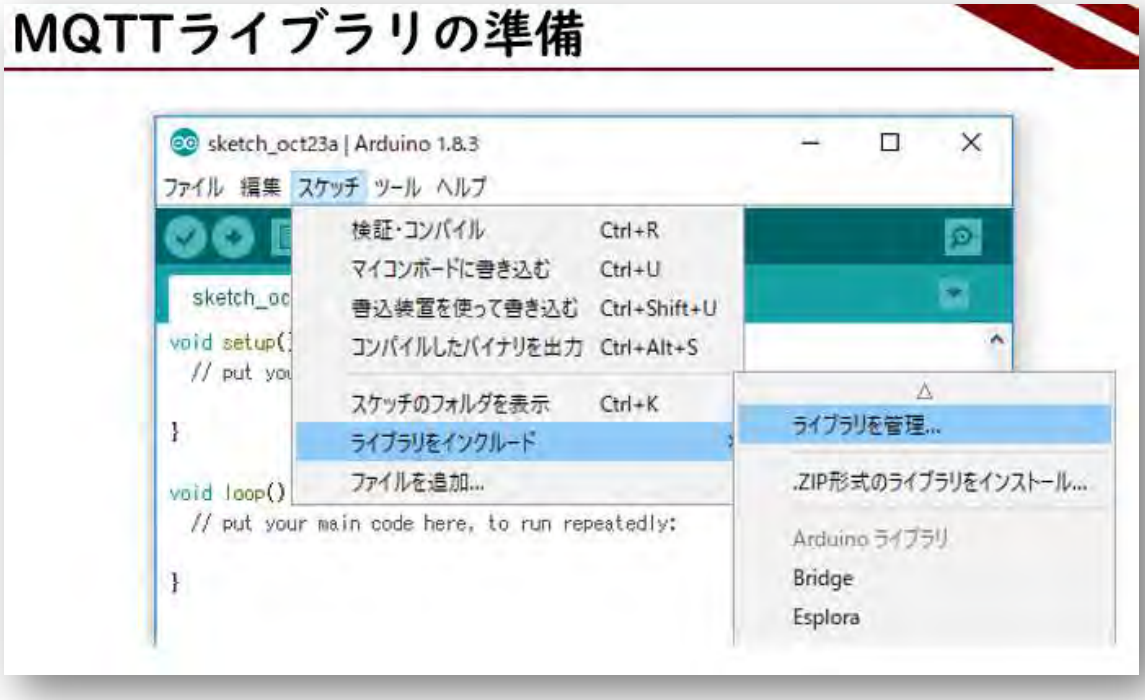


図 326

プログラムを作成する前に、MQTT サービスの利用を容易にしてくれる MQTT ライブラリを準備します。Arduino IDE のメニューからスケッチ→ライブラリをインクルード→ライブラリを管理 と辿ります。

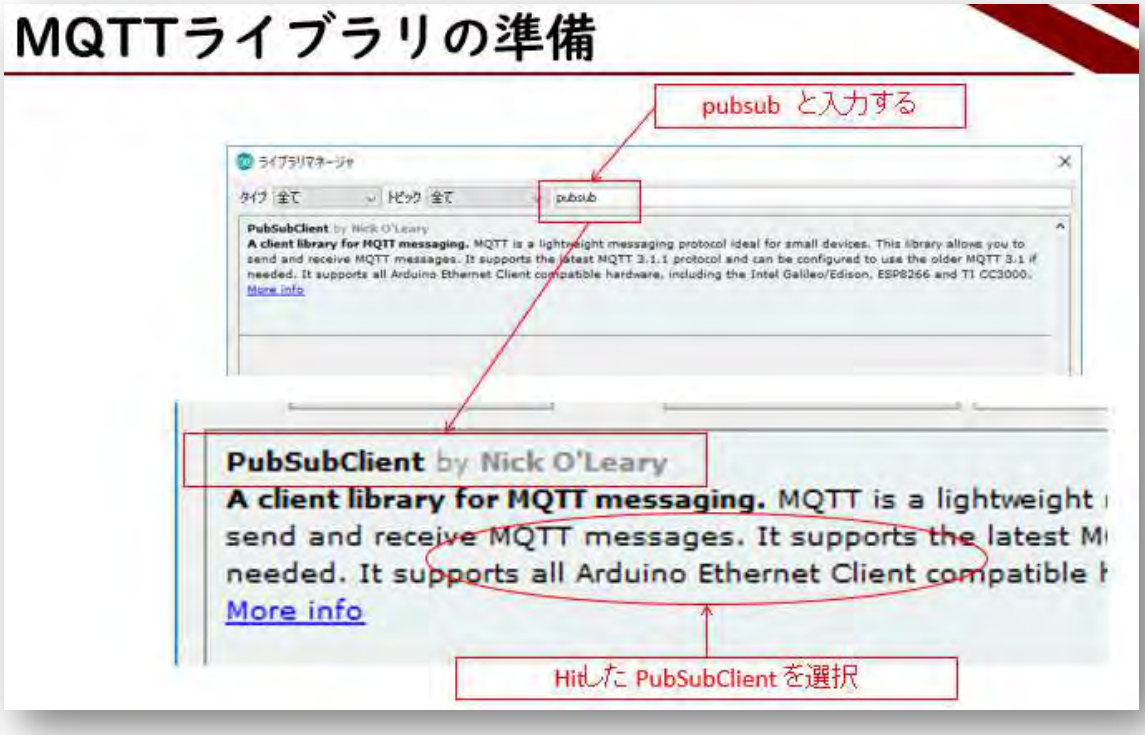


図 327

ライブラリマネージャが開きます。その運動の右上にある検索窓に pubsub と入力して少し待つと、その下に PubSubClient というライブラリがヒットします。これをクリックして選択てください。

図 328

選択するとそのライブラリの右下にインストールボタンが現れるので、それをクリックします。WEB から該当のライブラリをダウンロードして IDE の中に組み込んでくれます。インストールが終了すると、ライブラリ名の右側に INSTALLED と青い文字で表示されます。

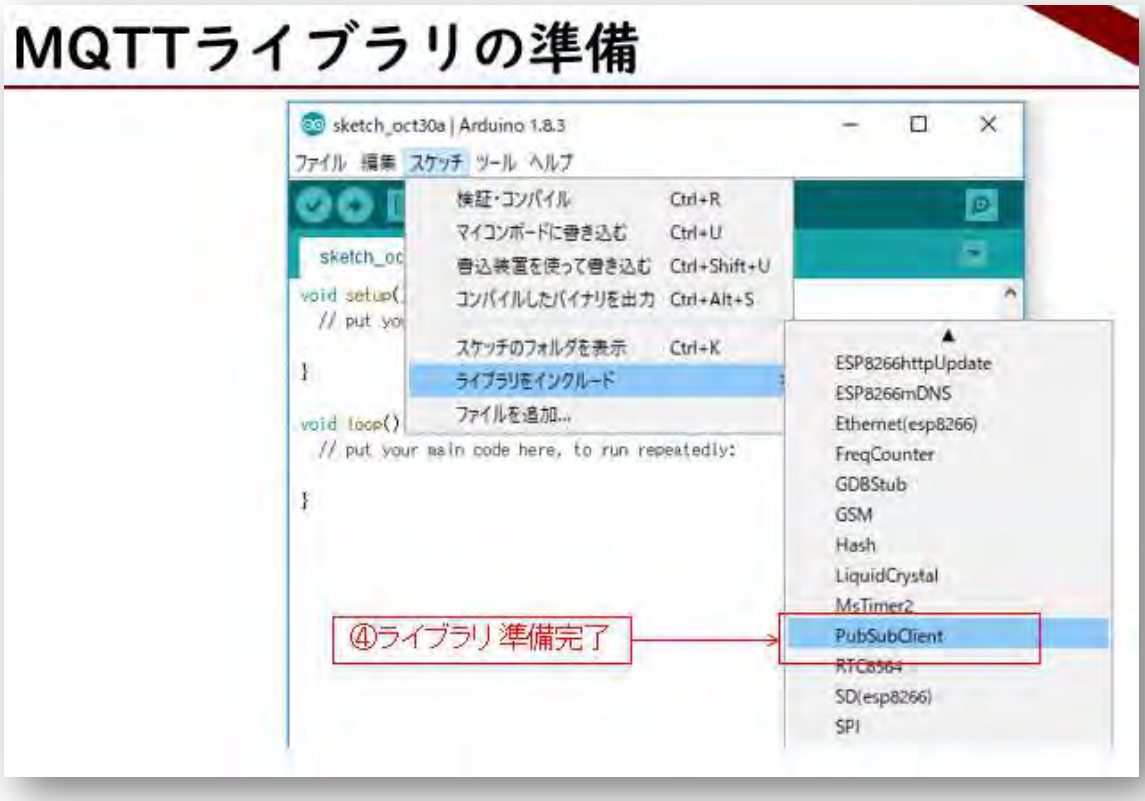


図 329

次に、このライブラリが使用可能になっているかを確認します。スケッチ→ライブラリをインクルードと辿り、現れるプルダウンの中に PubSubClient があれば、準備完了です。


参考にするプログラム
◇ファイル →スケッチ例 →
PubSubClient
→ mqtt_esp8266
mqtt_esp8266

図　330

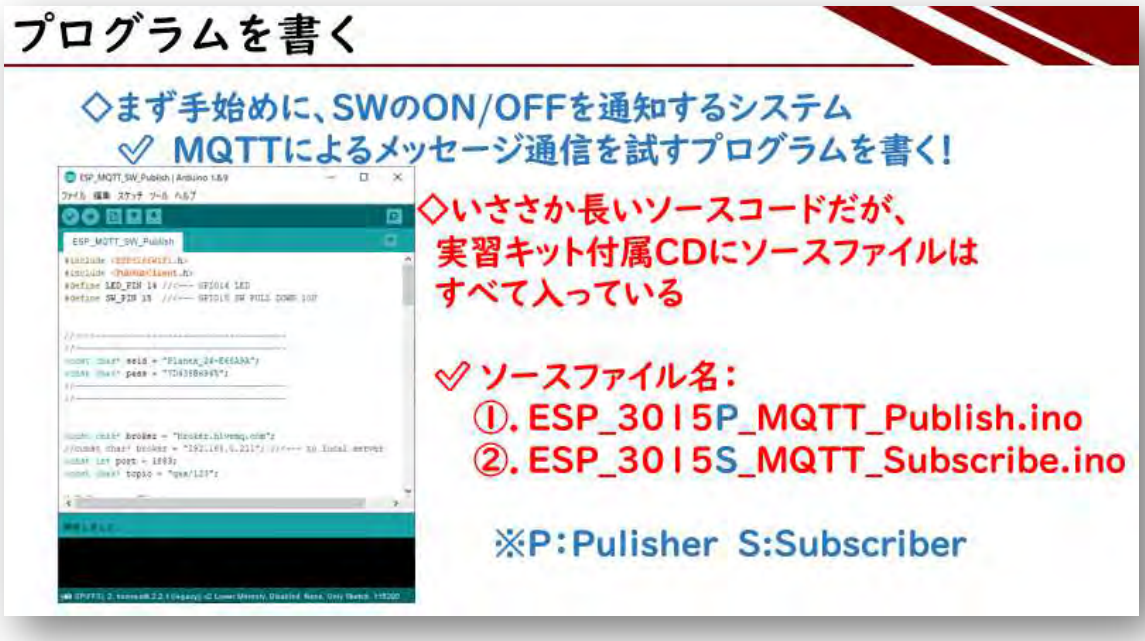


図 331

今回は 1bit の制御を行うプログラムを作成します。Publisher・Subscriber 共に長いプログラムになっています。

※ソースファイルは実習キット付属 CD に図のファイル名で入っています。

図 332

Publisher プログラムです。冒頭の 2 行は各々WiFi 機能と MQTT のライブラリを使用するためのヘッダ取込みです。次の 2 行はこれまでにも出てきた GPIO 番号割り当てです。次に使用するアクセスポイントの ssid と pass をしていします。実際に使用する MQTT Broker の URL を broker に設定します。Local LAN を使用する際は、アクセスポイントの IP アドレスを指定します。MQTT ポート番号は固定で 1883 と決まっています。Publisher、Subscriber で取り決めた Topic 名を topic で指定します。

図 333

WiFi 接続と MQTT 接続のために各々オブジェクトを作ります。変数はコメントで用途が分かるでしょう。setup()では LED、SW のピン設定とシリアル通信の初期化をした後、init_wifi()で WiFi 接続を行い、クライアントが MQTT に接続するためにブローカー情報を設定します。

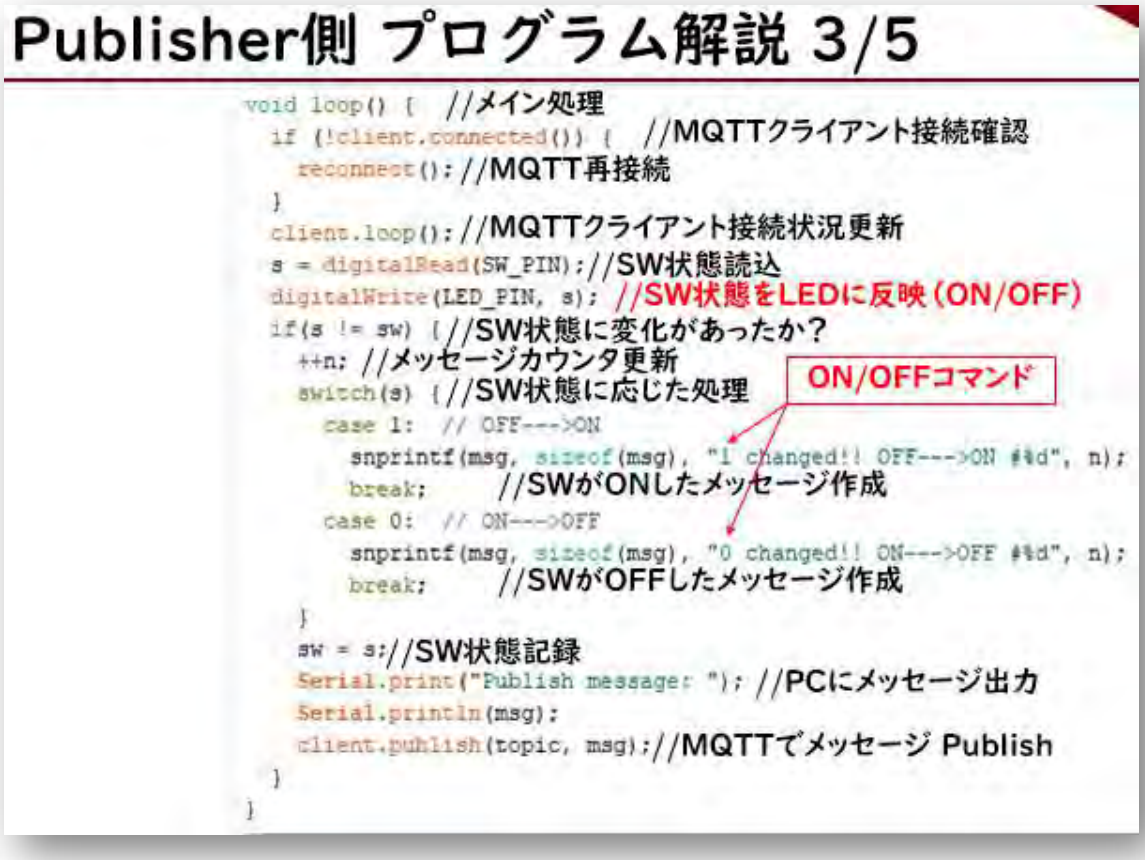


図 334

loop()では、まず reconnect()（後に説明）で MQTT 接続を行い client.loop()メソッドで送受信メッセージバッファの処理を行います（※）。SW 状態を LED に反映させた後、SW 状態に変化が有れば対応するメッセージを作製しシリアル通信で出力後、MQTT でメッセージを発行します。

※loop()メソッドの内容は下記 URL に解説があります。実際のメッセージは loop()メソッドが実行されるタイミングで LAN に流れて行きます。参考 URL を下記に示します。

http://www.steves-internet-guide.com/loop-python-mqtt-client/

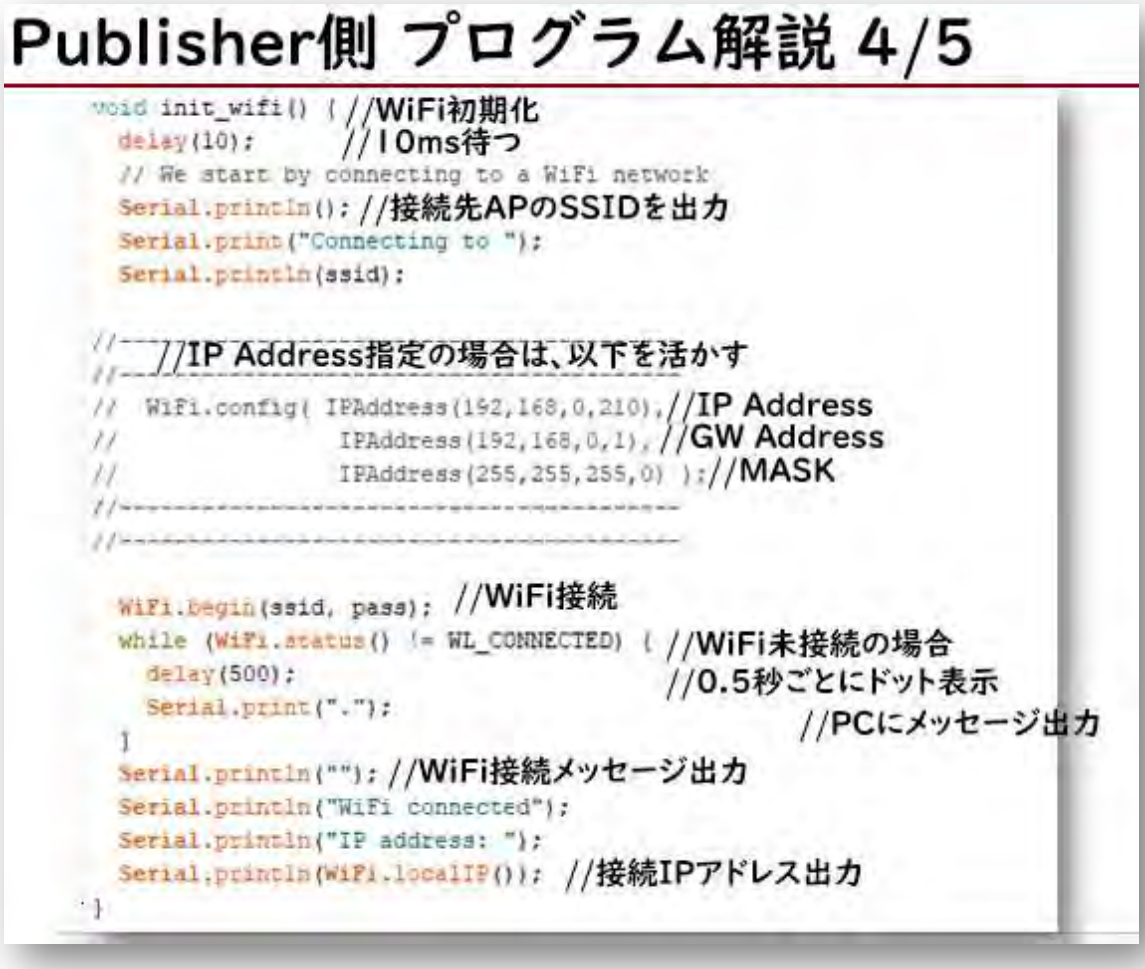


図 335

init_wifi()は、WiFiアクセスポイントに接続を行う関数です。接続はWiFi.begin()メソッドで行われ、接続状態をWiFi.status()で判断しています。接続が確立できない場合は、0.5秒毎にドット【.】を出力しながら再接続を繰り返します。

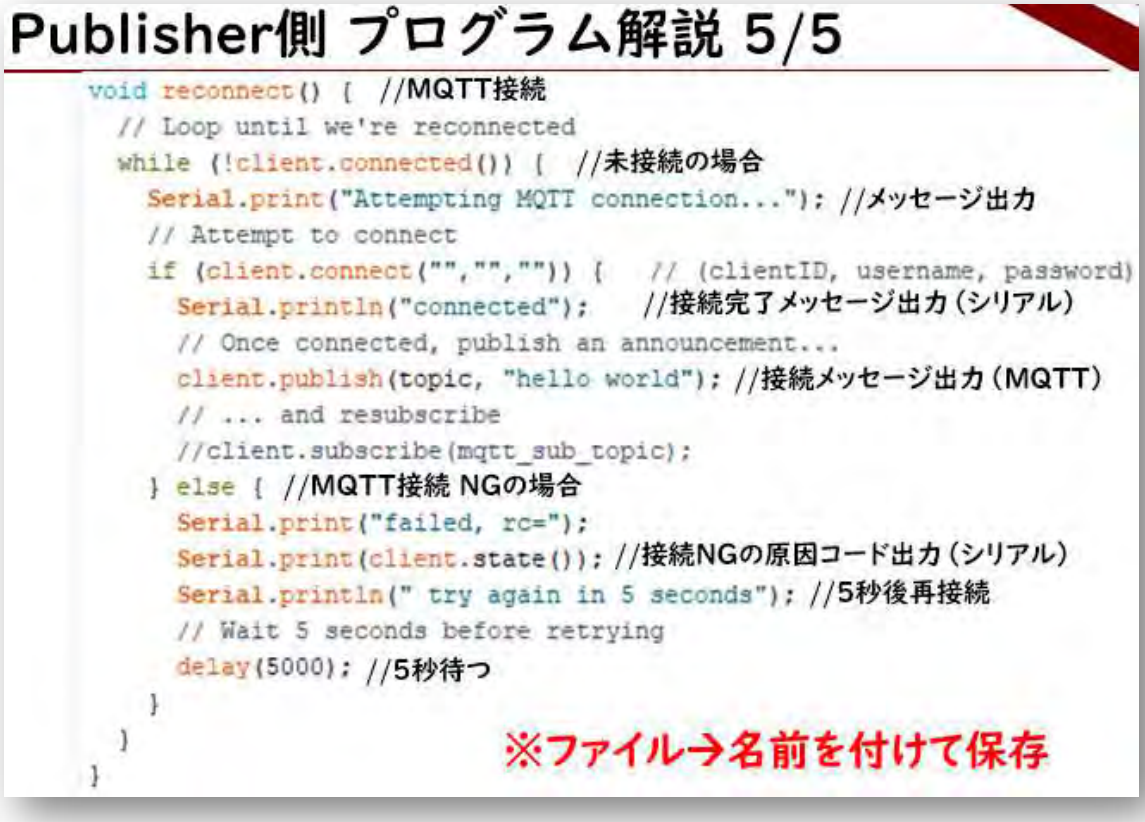


図 336

reconnect()は MQTT Broker への接続状態を調べ、未接続の場合 client.connet()で接続を行います。接続ができない場合は、原因コードをシリアル出力して 5 秒後に再度接続を行います。

図 337

Subscriber の冒頭は Publisher と同一です。最後の qos と言うのは、Quality Of Service のことで、3 段回のレベルがあり最上位レベルを指定すると、メッセージが届いたことを発行側に返す必要があります。発行側ではメッセ―が届いた通知が来なければ再送する必要があり、双方のプログラムが複雑になるので、この実験では最も低位のレベル 0 を指定します。下記 URL に詳しい説明があります。

https://sakiot.com/what-is-qos-of-mqtt/

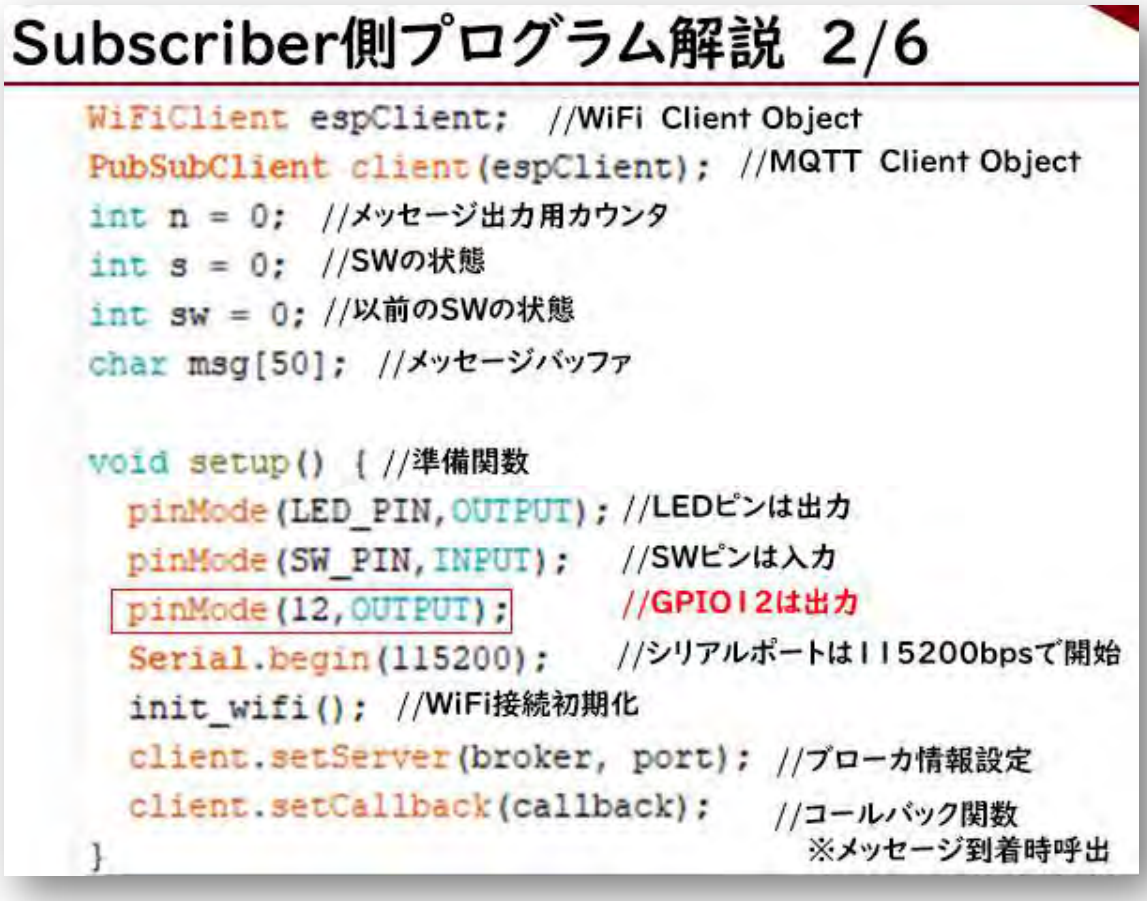


図 338

setup()で最後に setCallback()で指定しているのは、メッセージが到着した際の処理関数名です。受信したメッセージの処理は callback()で処理します。

図 339

MQTT の Subscriber 側はメインの処理はいたってシンプルです。メッセージを受信した場合の処理は、ソースコードの最後で説明する callback()で処理されます。

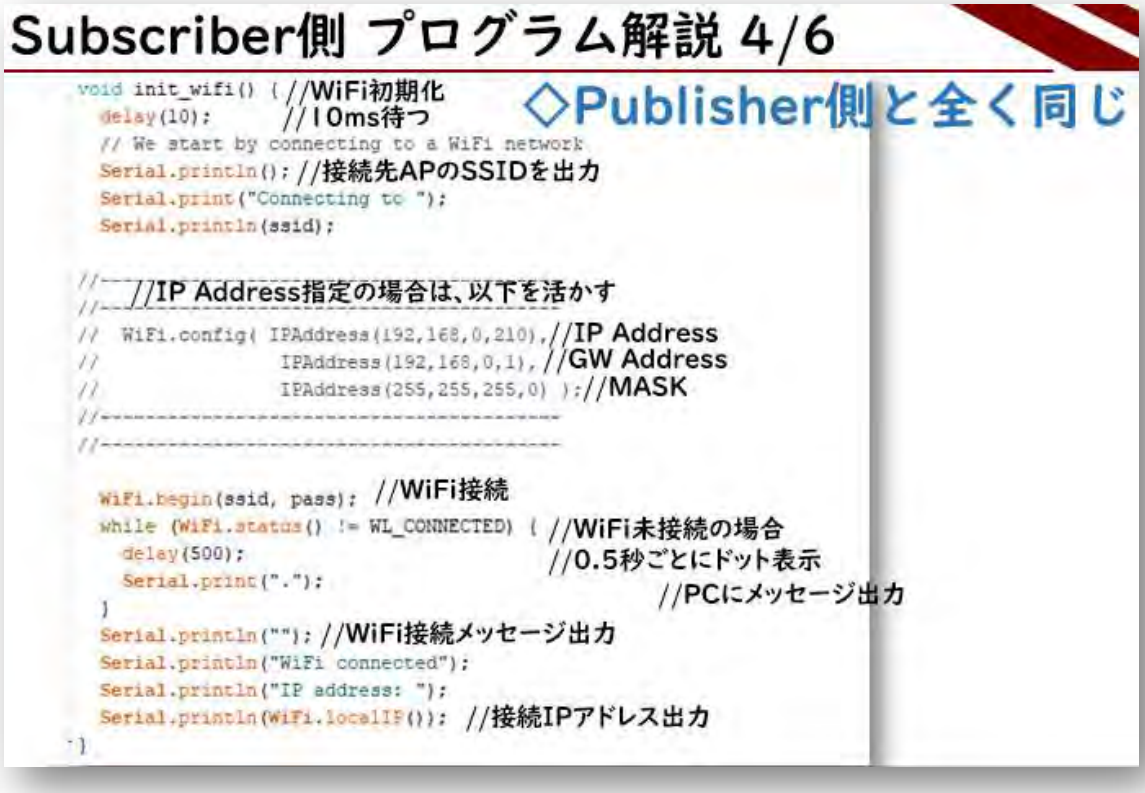


図 340

この部分は、Publisher 側と全く同じです。


```
void reconnect() {  //MQTT接続
  // Loop until we're reconnected
  while (!client.connected()) {  //未接続の場合
    Serial.print("Attempting MQTT connection..."); //メッセージ出力
    // Attempt to connect
    if (client.connect("","","")) {   // (clientID, username, password)
      Serial.println("connected");    //接続完了メッセージ出力(シリアル)
      // Once connected, publish an announcement...
      client.publish(topic, "hello world"); //接続メッセージ出力(MQTT)
      // ... and resubscribe
      //client.subscribe(mqtt_sub_topic);
    } else { //WiFi接続 NGの場合
      Serial.print("failed, rc=");
      Serial.print(client.state()); //接続NGの原因コード出力(シリアル)
      Serial.println(" try again in 5 seconds"); //5秒後再接続
      // Wait 5 seconds before retrying
      delay(5000); //5秒待つ
    }
  }
}
```

図 341

この部分は、Publisher 側と全く同じです。

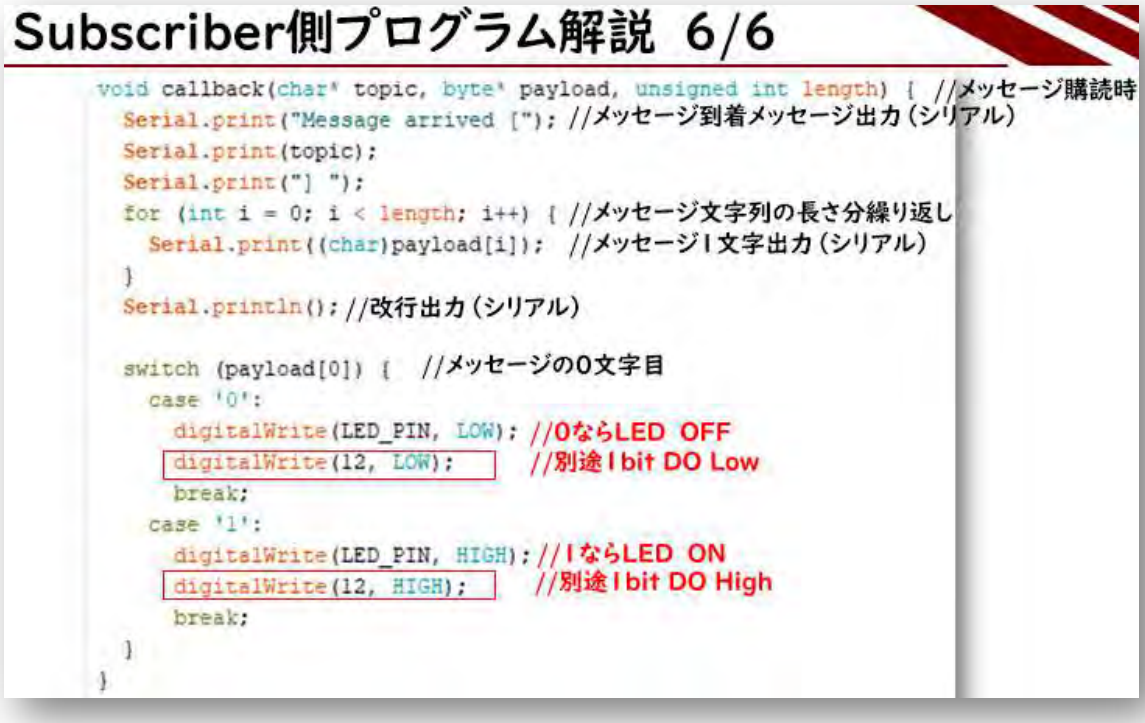


図 342

callback()は、メッセージを受信する処理本体です。メッセージはpayload[] 配列で渡されます。この配列の 0 文字目が ON/OFF コマンドになっています。この関数でコマンド内容に応じて LED の点灯・消灯とPCAXE08M2 の SW1 への ON/OFF 信号を出力します。

プログラムが出来たら、保存することを忘れない様にしてください。

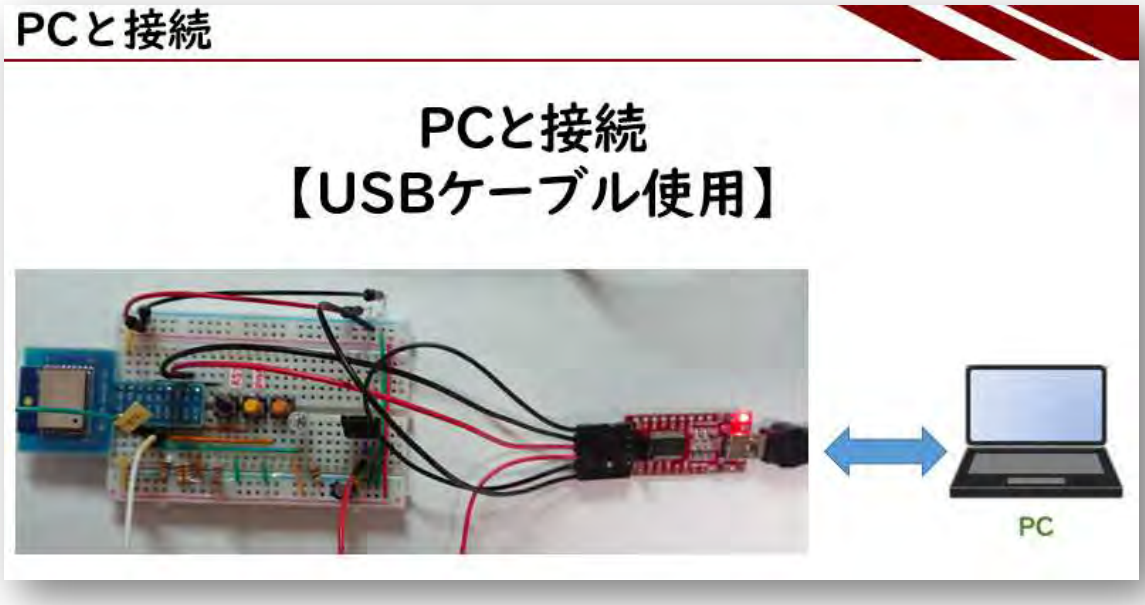


図 343

コンパイルと書込みを行います。各自の担当が Publisher なのか Subscriber なのかを再確認して、USB ケーブルで PC と USB－シリアル I/F を接続します。

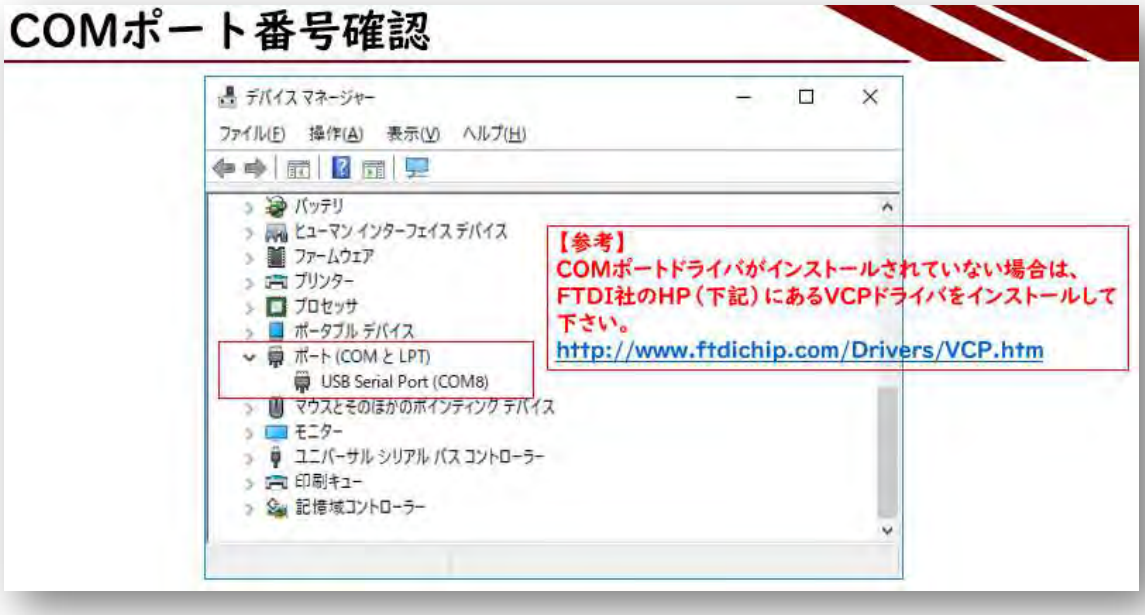


図 344

COM ポート番号を確認してください。

図 345

COM ポート番号を設定します。

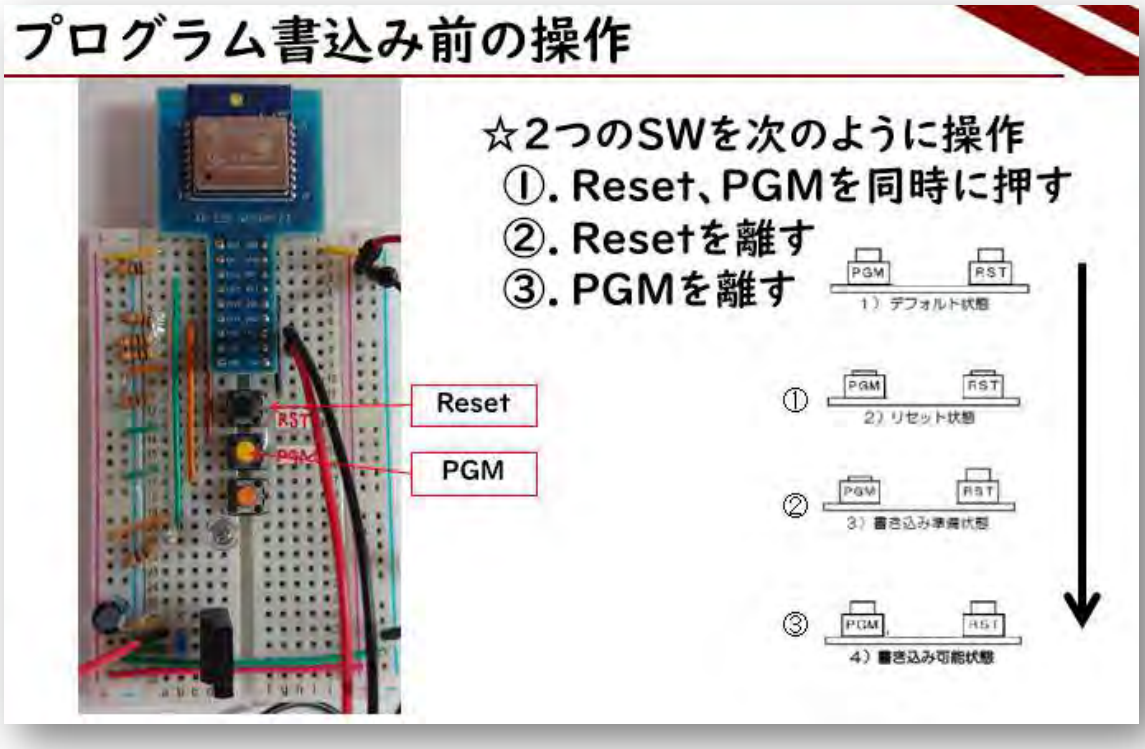


図　346

図に従いSWを操作してマイコン側の書込み準備を行います。

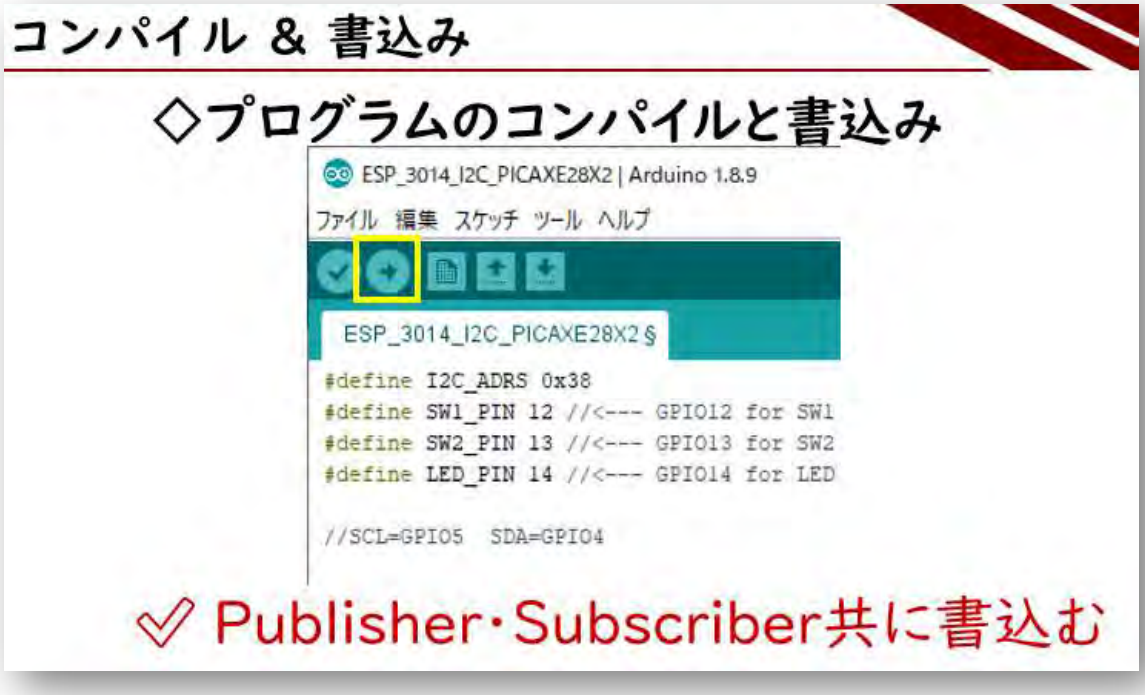


図 347

IDE 左上の右矢印ボタンを押下します。

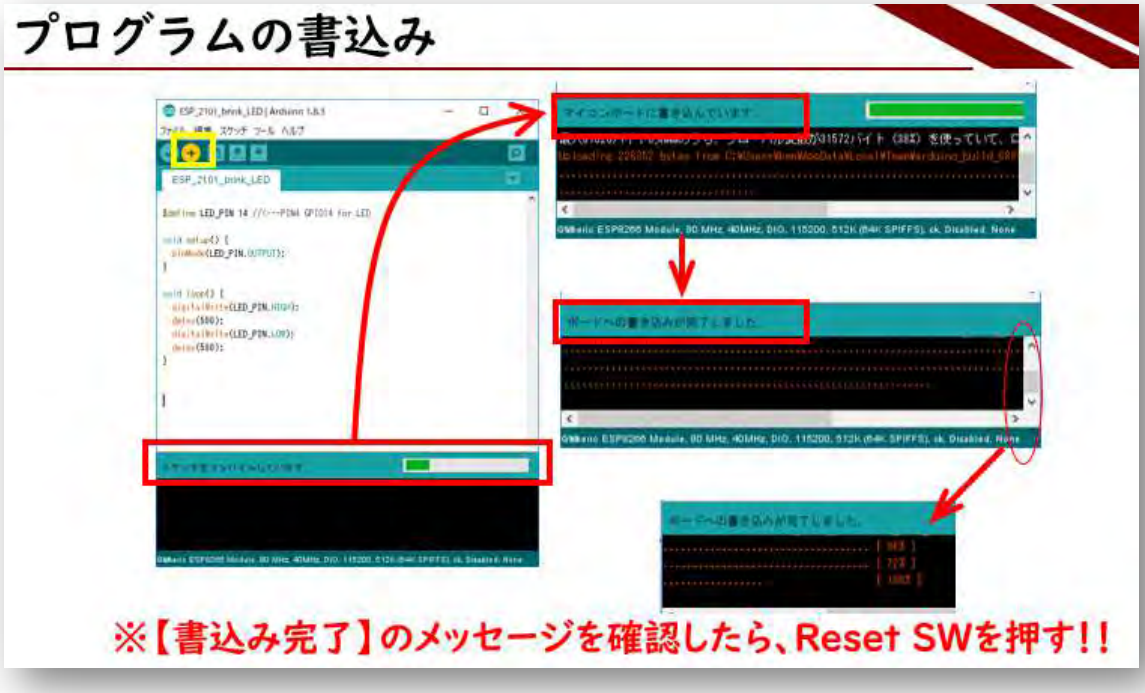


図　348

コンパイルでエラーが無ければ、書き込みに進み、終了するとその旨メッセージが表示されます。書込み完了のメッセージを確認したら、マイコンの Reset SW を押下してください。

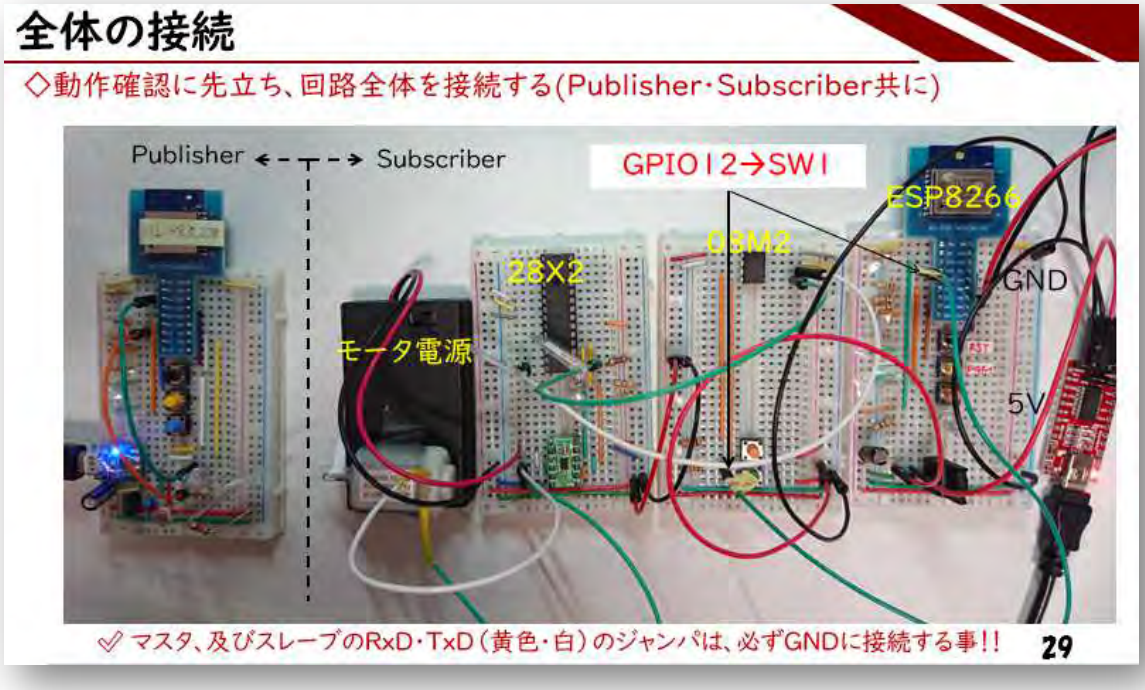


図 349

複数の開発者で構築したシステムの初稼働です。Publisher と Subscriber 間は物理的には離れていますが Internet で繋がることになります。PICAXE の TxD、RxD は GND に接続して下さい。

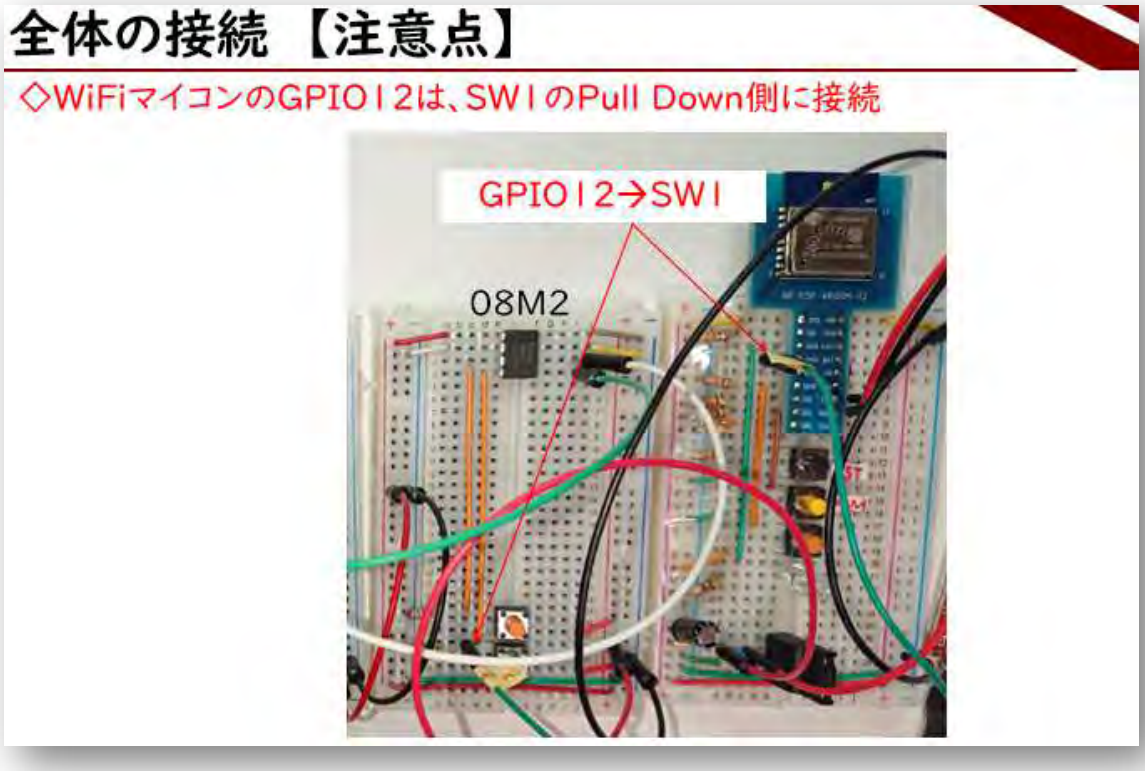


図 350

ESP8266 の GPIO12 が PICAXE08M2 の SW1 Pull Down 側に配線されているか確認します。（SW2 は未使用）

全体の確認が出来たら、動作確認直前にモータ電源を ON します。

図 351

まず Subscriber の LED が消灯していることを確認します。次に Publisher の SW を押下すると LED が点灯します。この通知（メッセージ）が Subscriber に届き LED が点灯します。SW から指を離すと LED は消灯します。

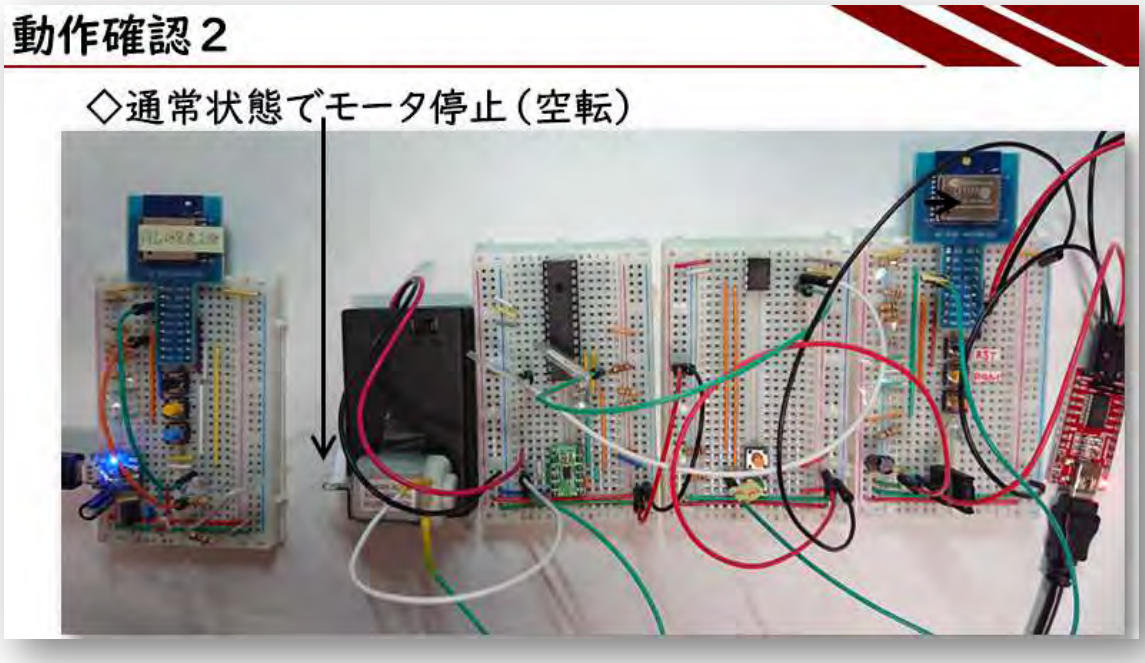


図　352

次にモータの動作を確認します。通常ではモータは停止（空転）状態です。

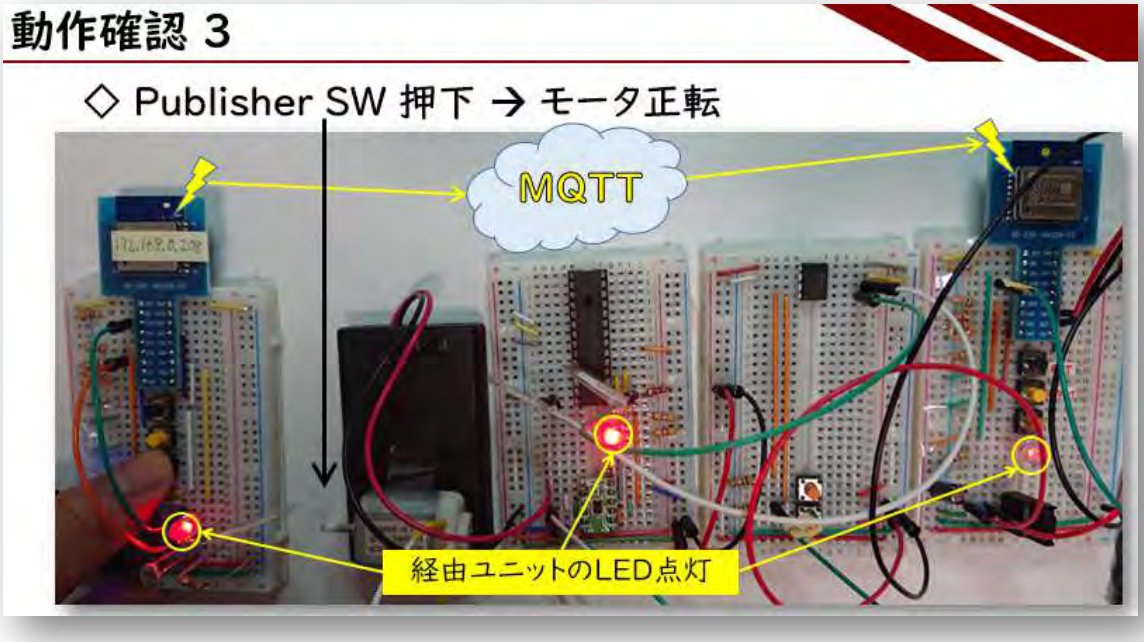


図 353

Publisher の SW を押下するとメッセージが MQTT で届いて経由するユニットの LED が点灯し、モータが正転します。SW から指を離せば、LED は消灯しモータは停止します。

最後にモータ電源を OFF します。

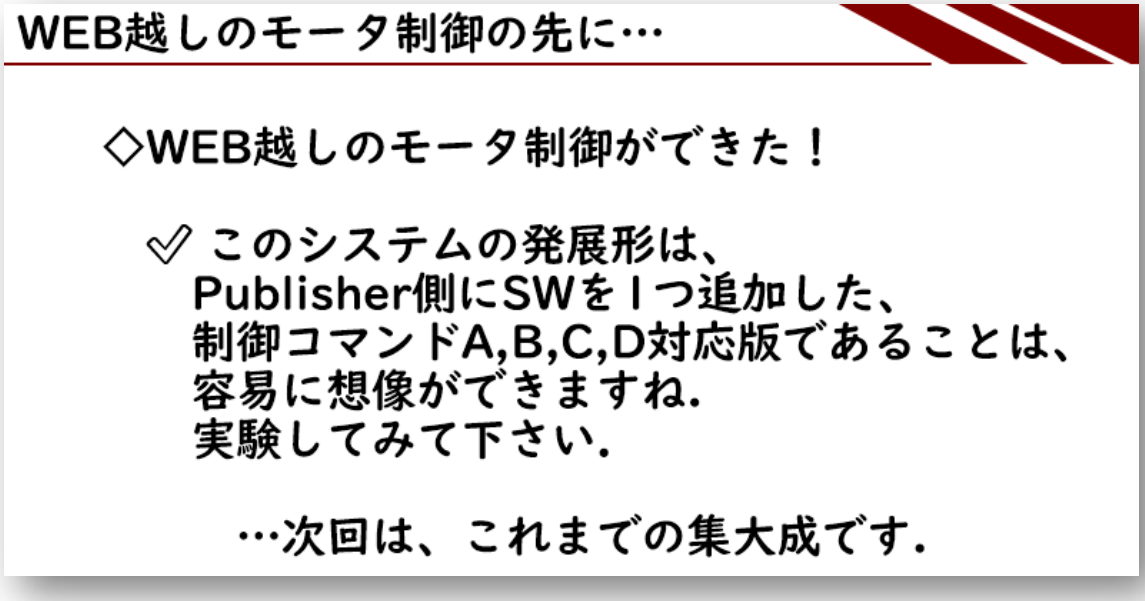


図 354

複数の開発者による WEB 越しのモータ制御システムが開発できました。このシステムは単にモータ正転・停止だけですが、既に経験済みの A、B、C、D のコマンド対応版に拡張するのは容易です。システム開発の実際は新規開発だけでなく、リフォームやリノベーションの様な既存システムの改造や発展がよくあります。実習も終盤戦です。ここで、これまでの成果を発展形開発に注いでみて下さい。

第16回　応用開発

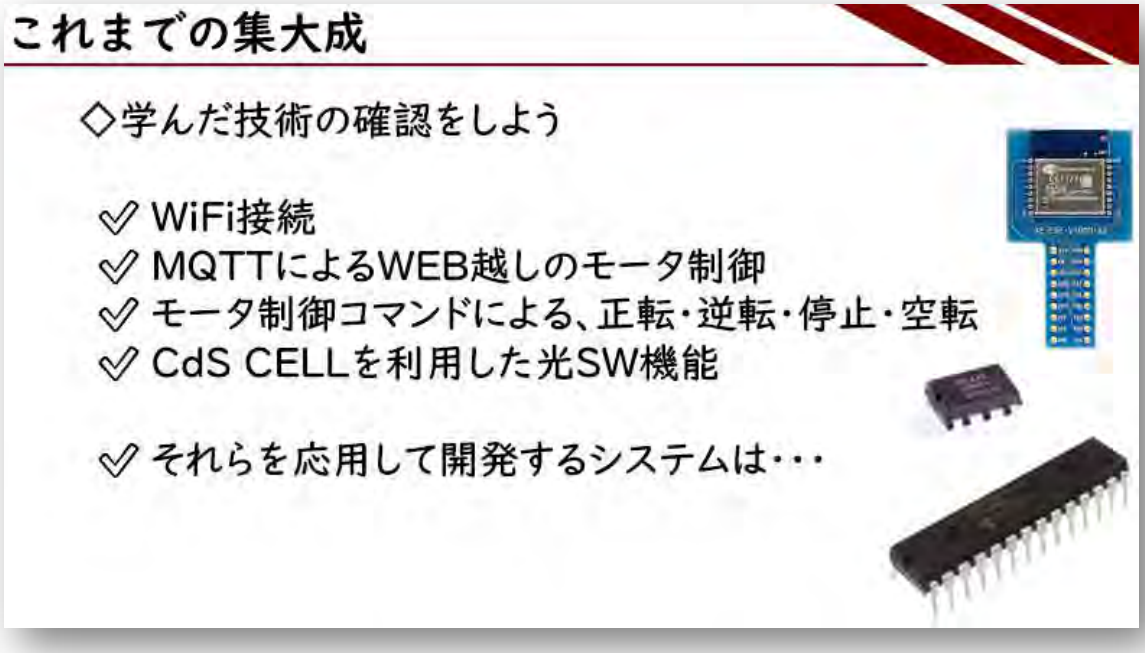


図 355

最期の実習は、これまでの実践で身に付けた技術を用いた応用開発を行います。

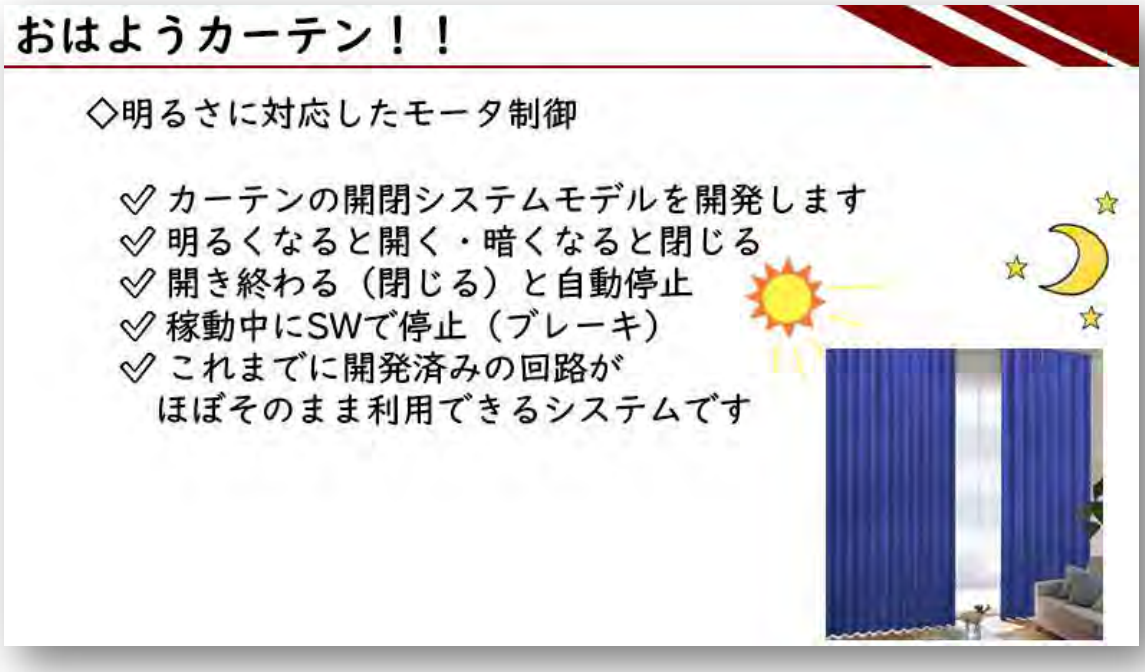


図 356

タイトルの通り、明るくなると開き暗くなると閉じるカーテン開閉システムを開発します。開閉するのでモータは正転・逆転します。開閉完了時にモータは自動停止します。また駆動中に SW を押せばブレーキが掛かる仕組みです。これまでの開発内容をほぼ全て応用したシステム開発です。

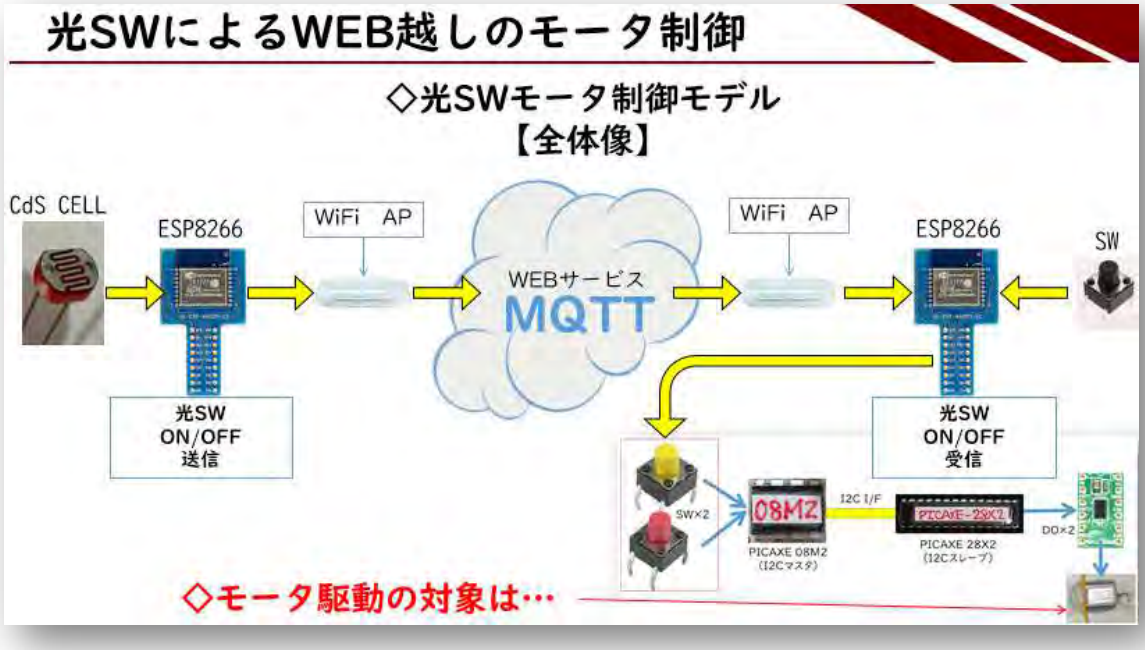


図　357

光 SW によるモータ制御モデルを図に示します。図を見て第 15 回（前回）との違いを見出せるでしょうか。異なる点が 3 つあります。どこでしょう？

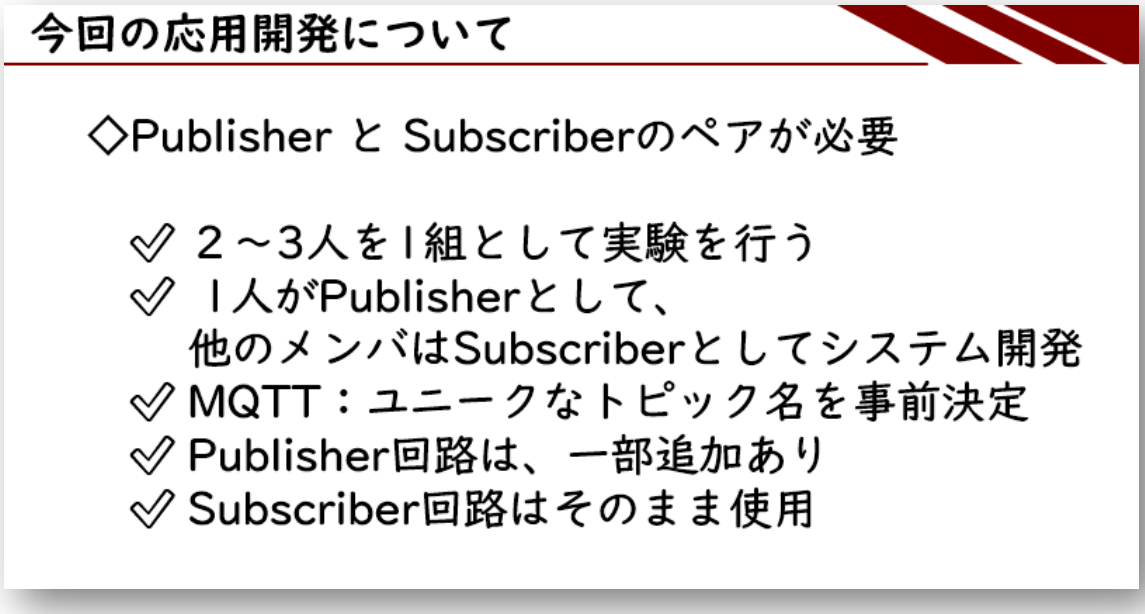


図 358

今回も MQTT を利用するので、予めグループで一意のトピック名を決めてください。トピック名は他のグループとも打ち合わせて、競合しない様に決めます。Publisher の回路には変更があります。

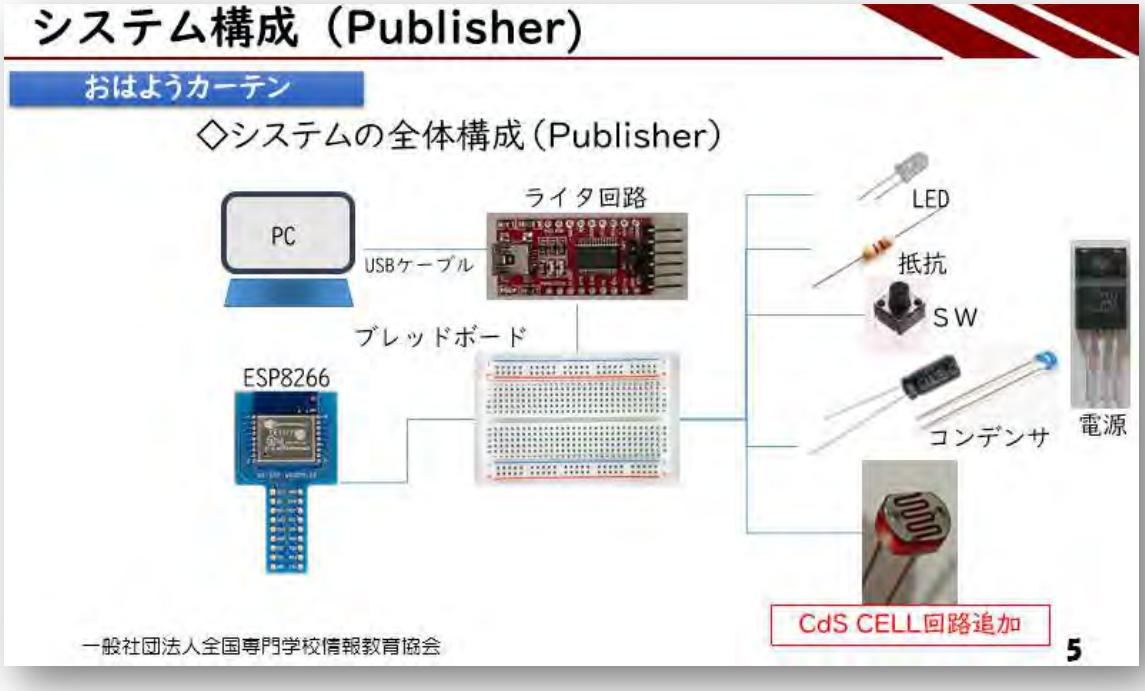


図 359

Publisher の回路には、第 6 回で使用した CdS CELL を追加します。第 6 回では明るさにより LED の点灯制御を行う光 SW の開発を行いました。モータ制御の最上流にこの技術を用います。

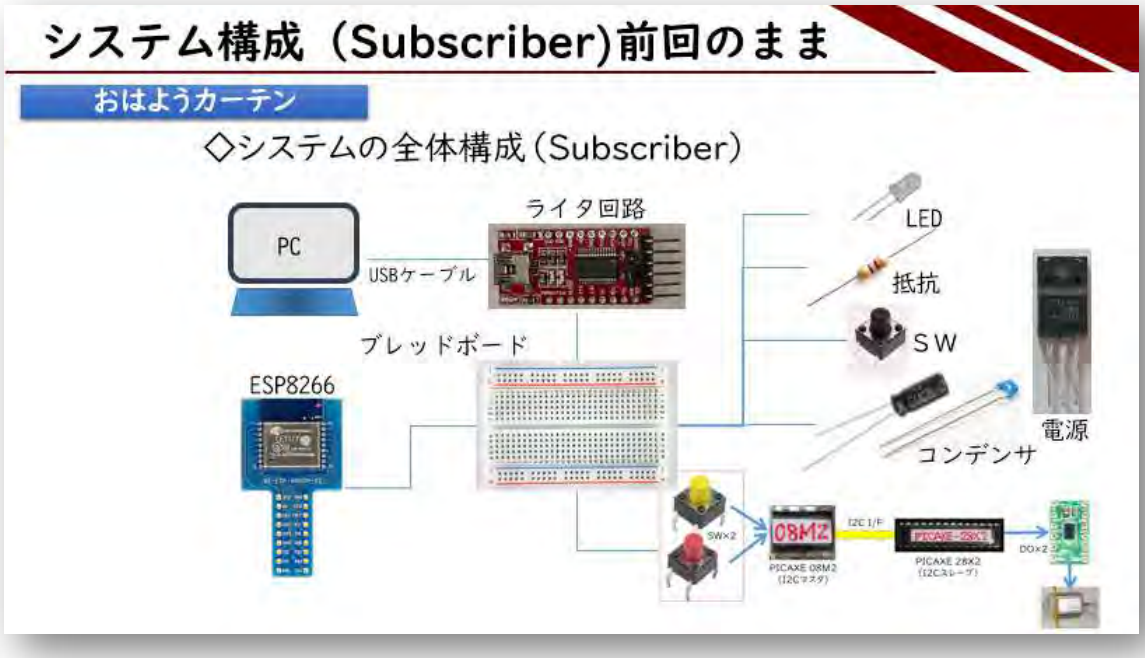


図　360

Subscriber 側は第 14 回の図 284 と図 15 回の図 325 と同じ構成です。確認して下さい。

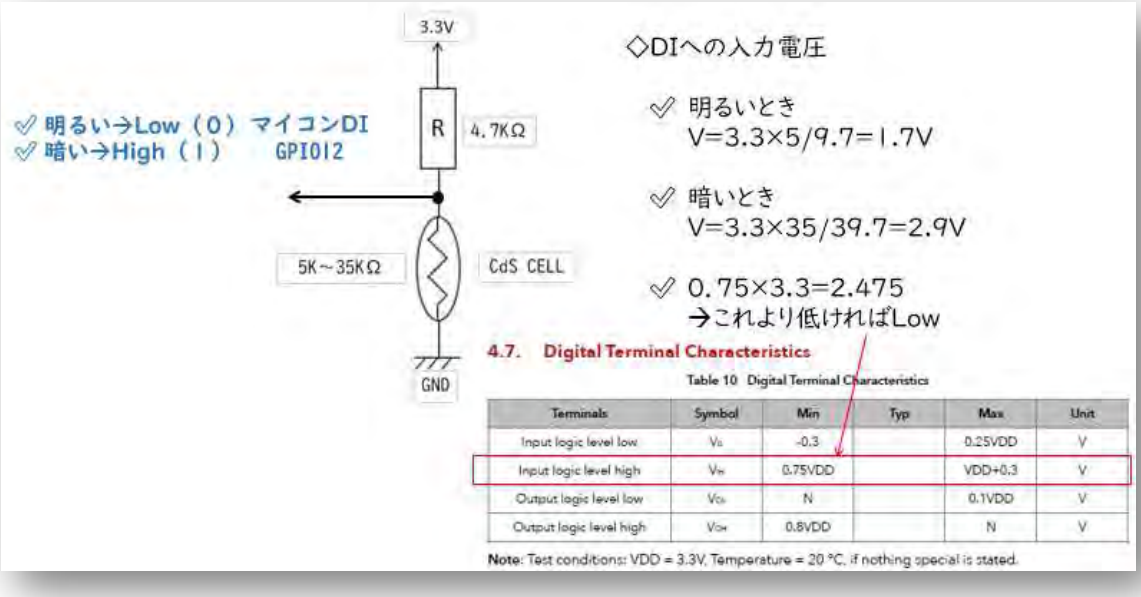


図　361

最上流の SW 部分となる CdS CELL 回路を図に示します。第 6 回でも示したように CdS CELL の光による抵抗値変化を利用しています。CdS CELL の抵抗値は明るいとき 5kΩ、暗いとき 35kΩでした。図のように 4.7kΩを直列接続すれば、全体の抵抗値は明るいとき 9.7kΩ、暗いとき 39.7kΩです。この時の CdS CELL と抵抗器の接続点での電圧を計算すると明るいとき 1.7V、暗いとき 2.9V です。ESP8266 のデータシート（図右下の表）によると、Input logic level high は 0.75VDD（VDD=3.3V で計算すると 2.475V）。これより低い電圧なら ESP8266 の DI では Low と判断されますので、SW の代わりにこれを利用すれば、明るいとき DI=0、暗いとき DI=1 になります。この点を十分理解してください。面倒そうな計算と判断ですが、オームの法則（比例計算）は色々な場面で役立つことが分かります。

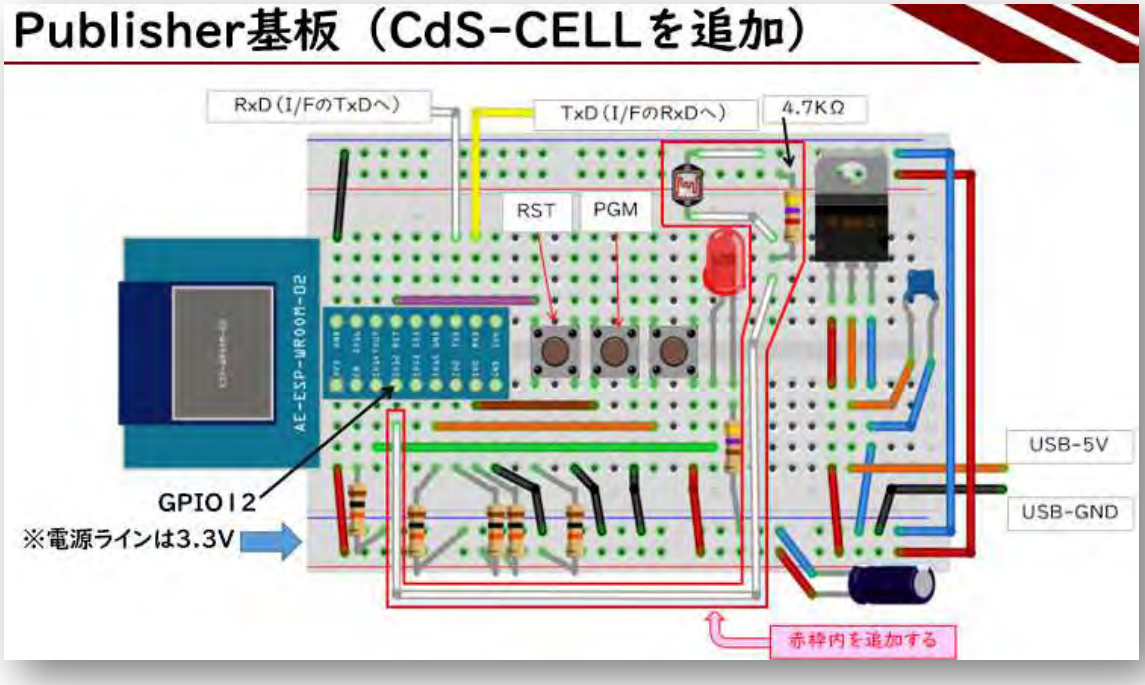


図 362

回路作成は簡単です。Publisher 基板には CdS CELL 回路を追加するだけです。その他の回路に変更はありません。

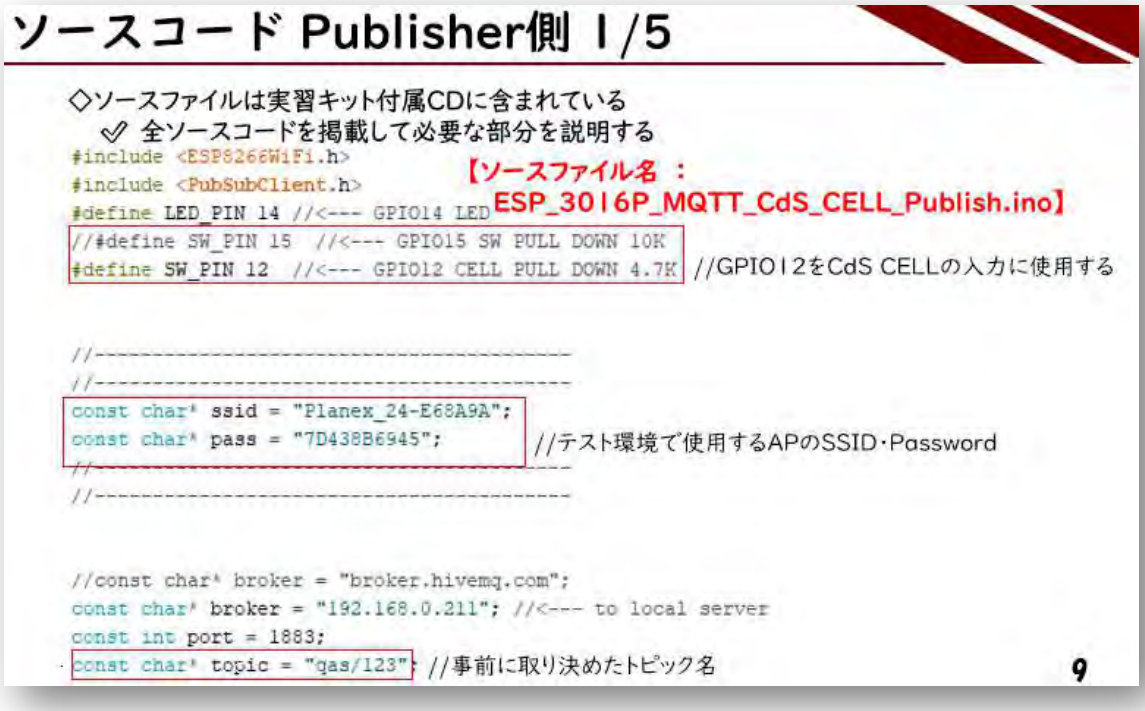


図 363

ソースコードの必要な部分について説明します。SW を利用していた GPIO を CdS CELL 用に変更します（PULL DOWN 不要であるため）。アクセスポイントの ssid と pass は使用する環境に応じて適宜変更します。topic はグループで決めたものに変更します。

図 364

図の部分に変更はありませんが、SW の代わりに CdS CELL が繋がっています。

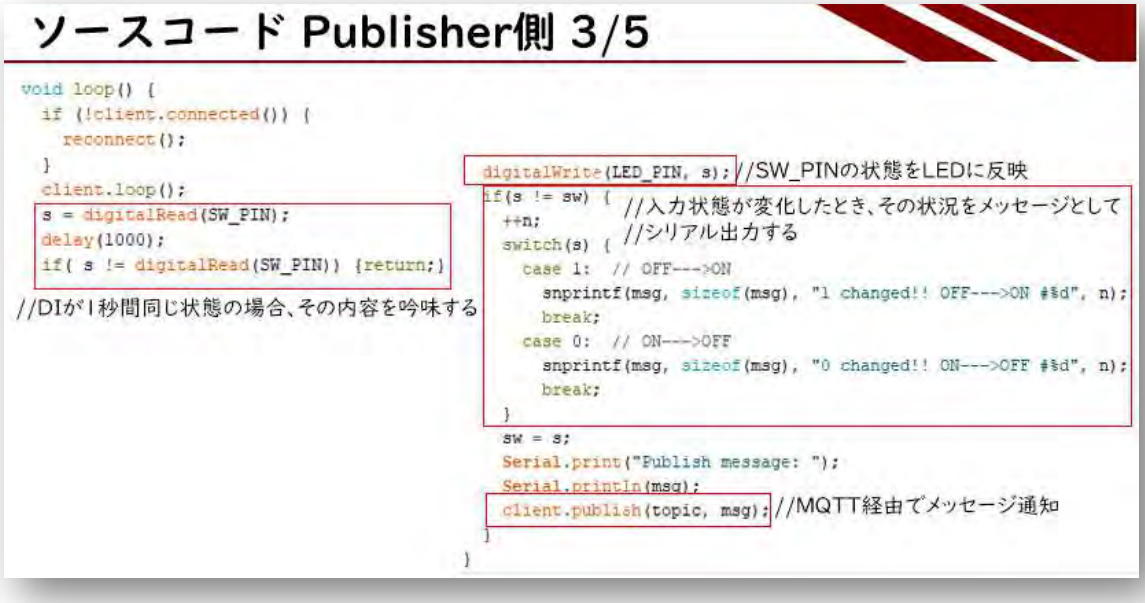


図 365

CdS CELL の明るさに対する抵抗値変化は遅いので、1 秒間隔で DI を 2 回読み込んで、同じだったとき DI の状態が前回と変化したか判断することにします。この時間の目的は、光 SW の動作にヒステリシスを持たせるためです。この時間はシステム完成後にもっと長くする調整が必用かもしれません。

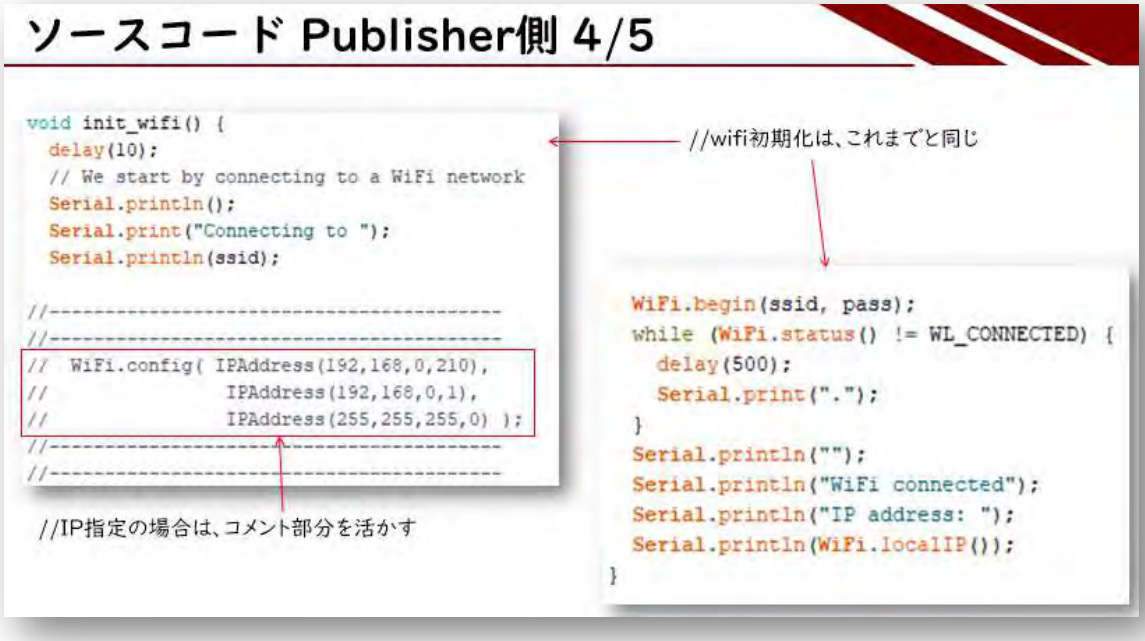


図 366

図の部分は前回と同じです。

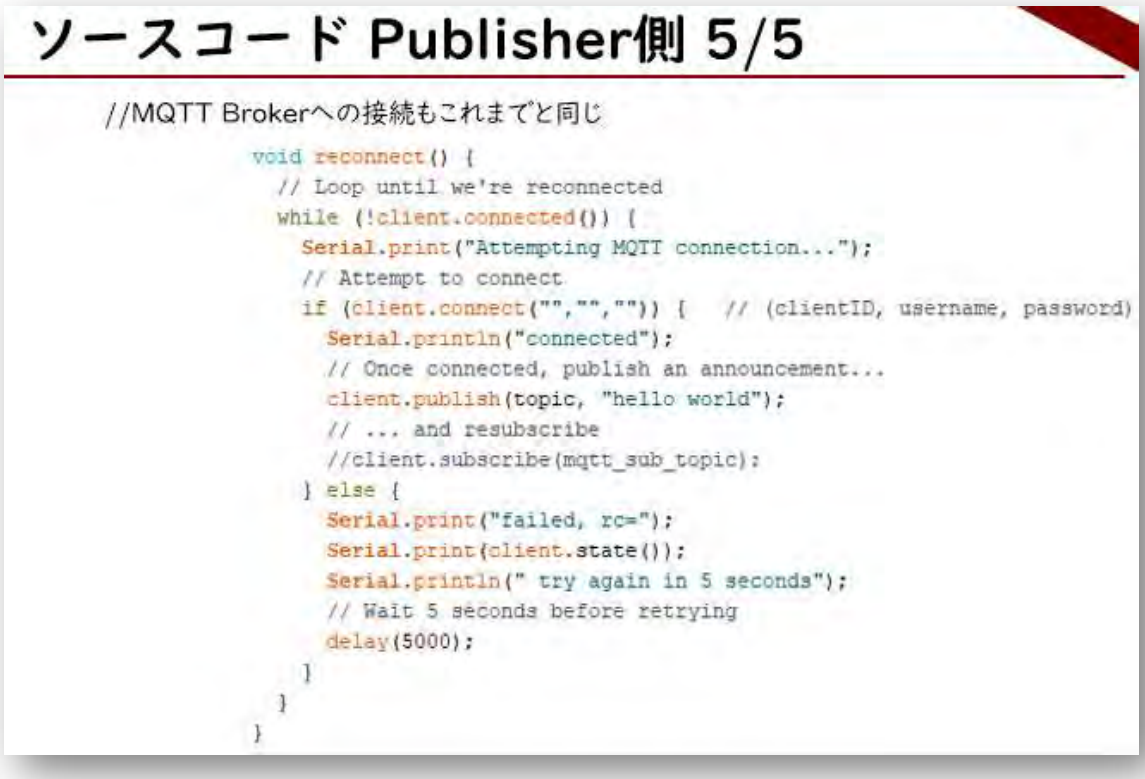


図 367

図の部分は前回と同一です。

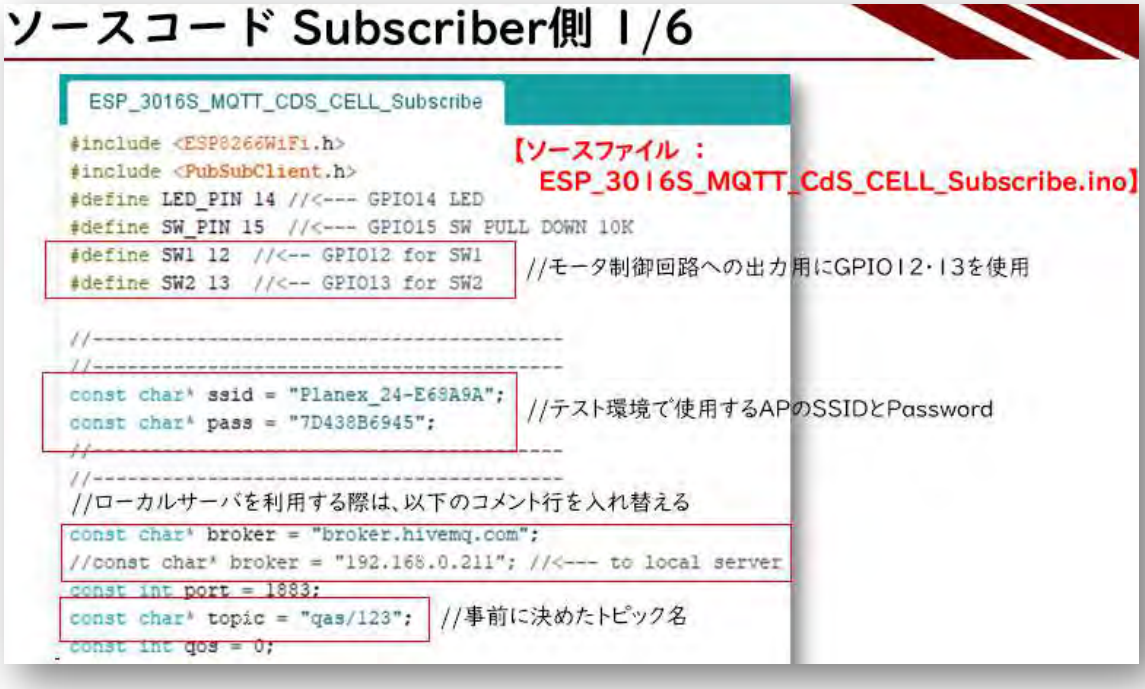


図 368

Subscriber 側には、モータの正転・逆転制御に対応したプログラムです。SW1、SW2 用に DO を使用します。

図 369

SW1、SW2 への出力を設定します。

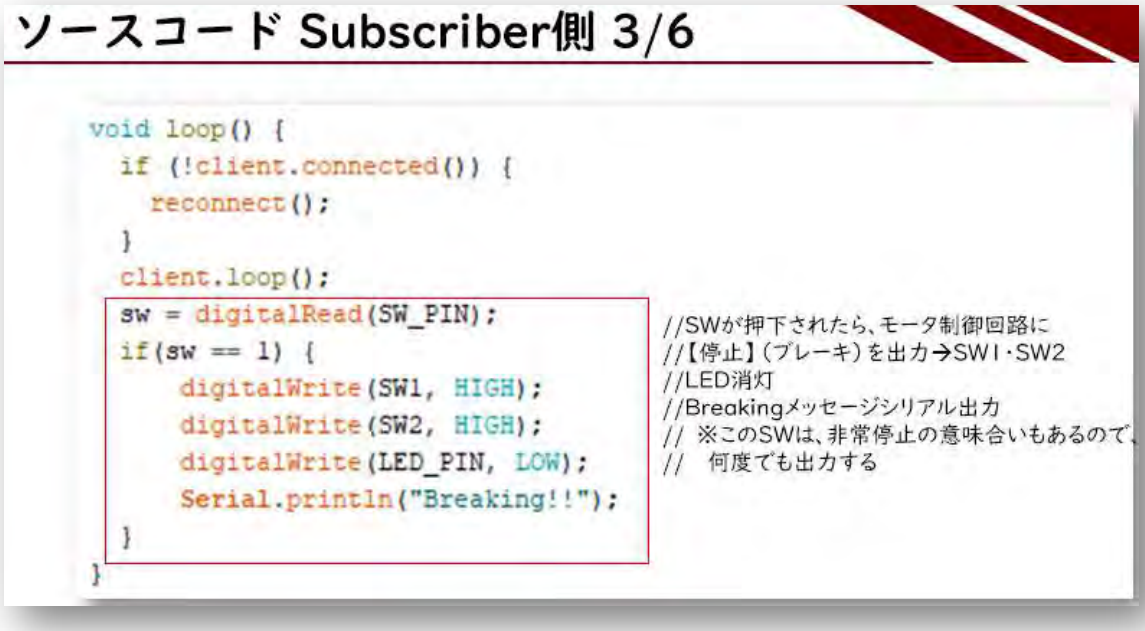


図 370

MQTT によるメッセージ受信時の処理は callback()で行っているので、loop()で行うべき処理は、停止 SW が押下された際の処理です。この SW 操作は非常停止の意味もあるので押下されていれば何度でもモータにブレーキをかけます。

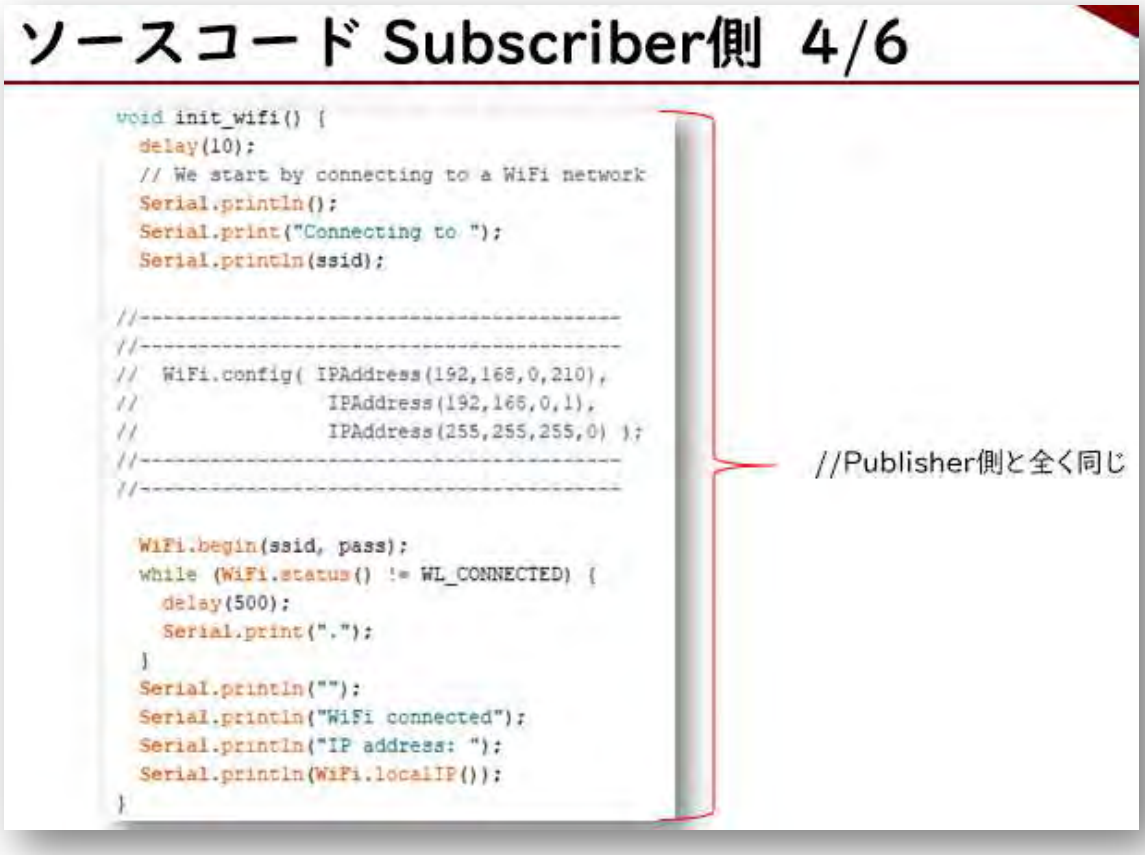


図 371

図の部分は Publisher と同一です。

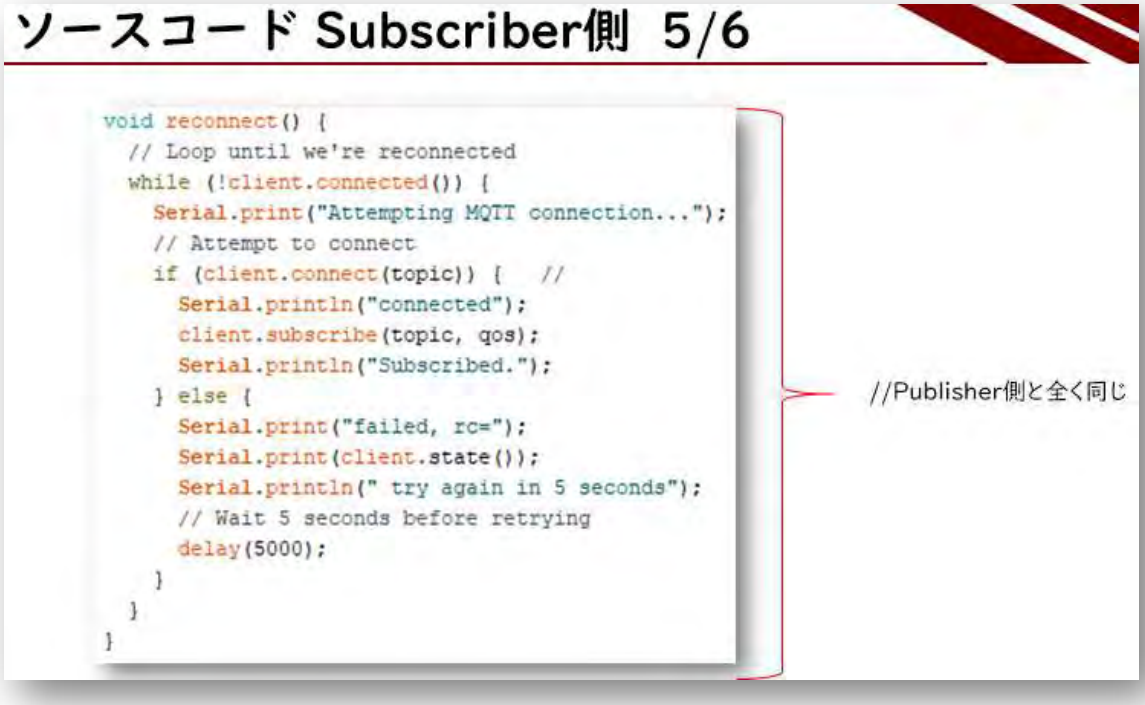


図 372

図の部分は Publisher と同一です。

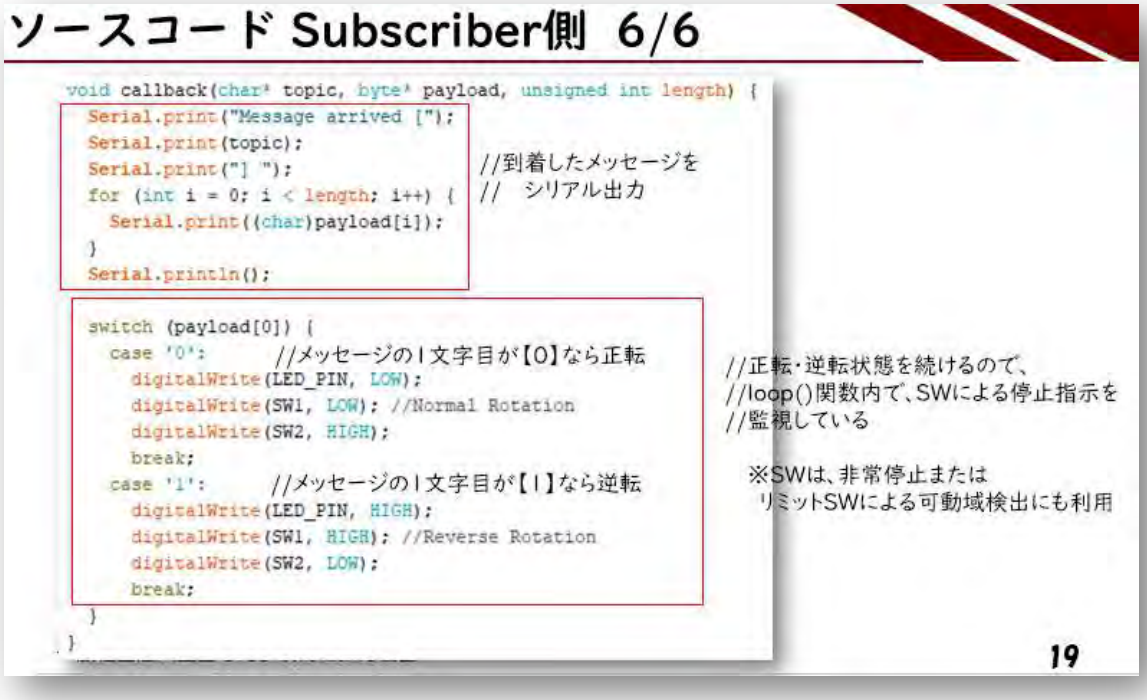


図 373

メッセージ受信時の処理を行う callback()です。メッセージが到着したらモニタ用にシリアル通信で出力します。payload[] 部に含まれている正転、逆転のコマンドを判断してモータ制御ユニットの SW1 と SW2 のコントロールを行います。この処理はメッセージが到着したとき、1 回だけ行われる制御で、モータ正転・逆転信号が出力されます。カーテンの開閉が完了すると、図の※で説明しているリミット SW が ON します。loop()内でこれを検出してブレーキが掛かります。このリミット SW は、Subscriber 基板の汎用 SW と並列に接続することで上記機能を実現できます。汎用 SW が押下されると、モータ制御ユニットの SW1 と SW2 への出力が OFF されるので、モータにはブレーキが掛かったまま（カーテン停止）になります。

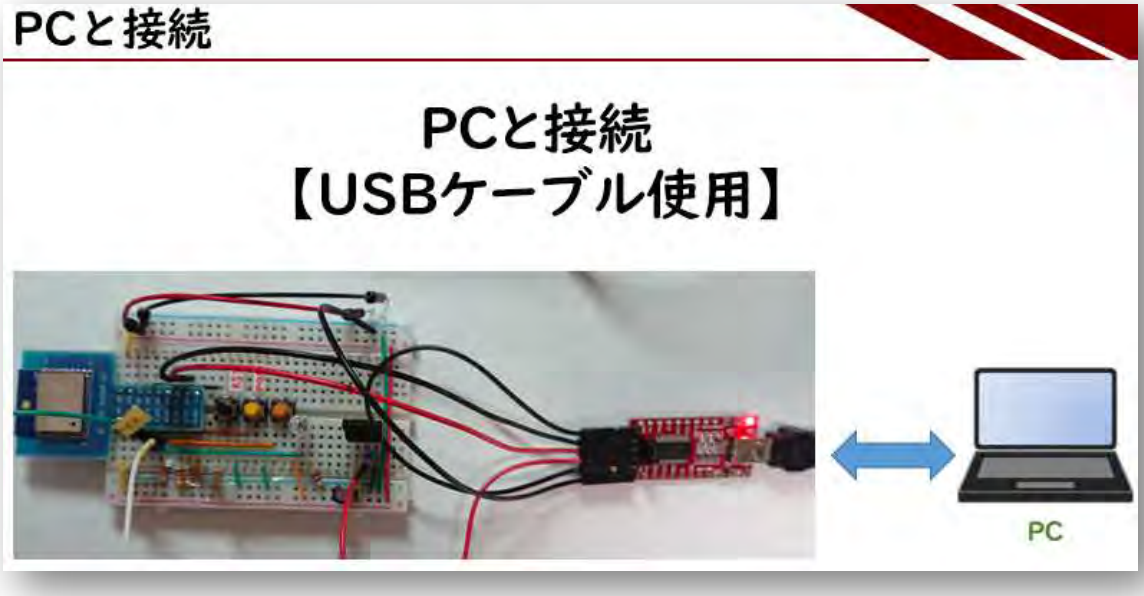


図 374

以下 Publisher と Subscriber 各々で行う作業の目的を再確認しながら進めましょう。

PC と USB－シリアル I/F を接続します。

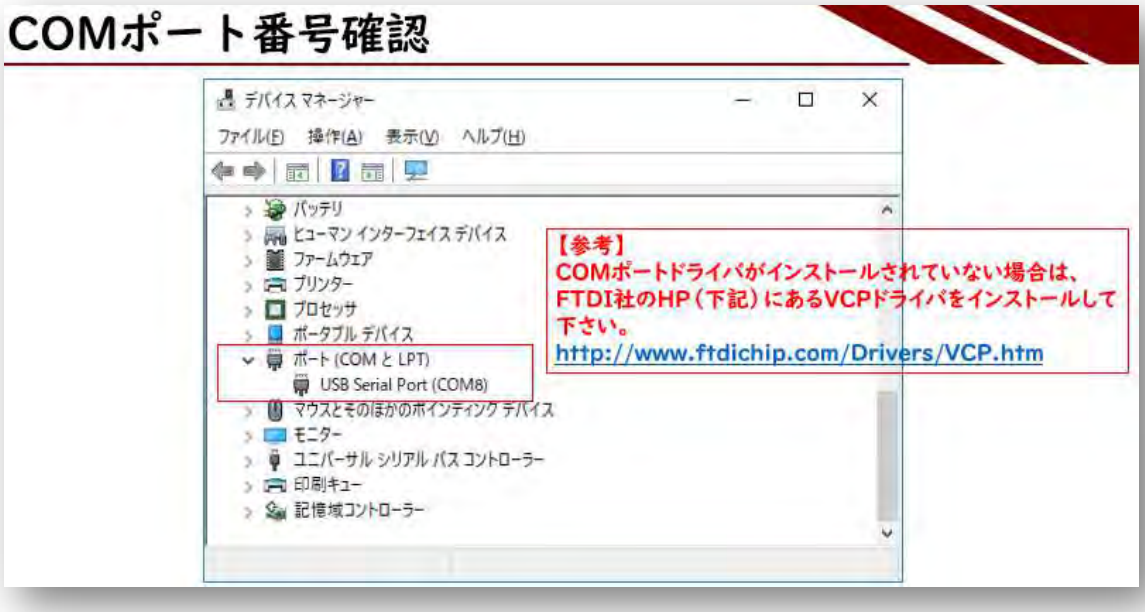


図　375

COM ポート番号の確認をします。

図 376

IDE に COM ポート番号を設定します。

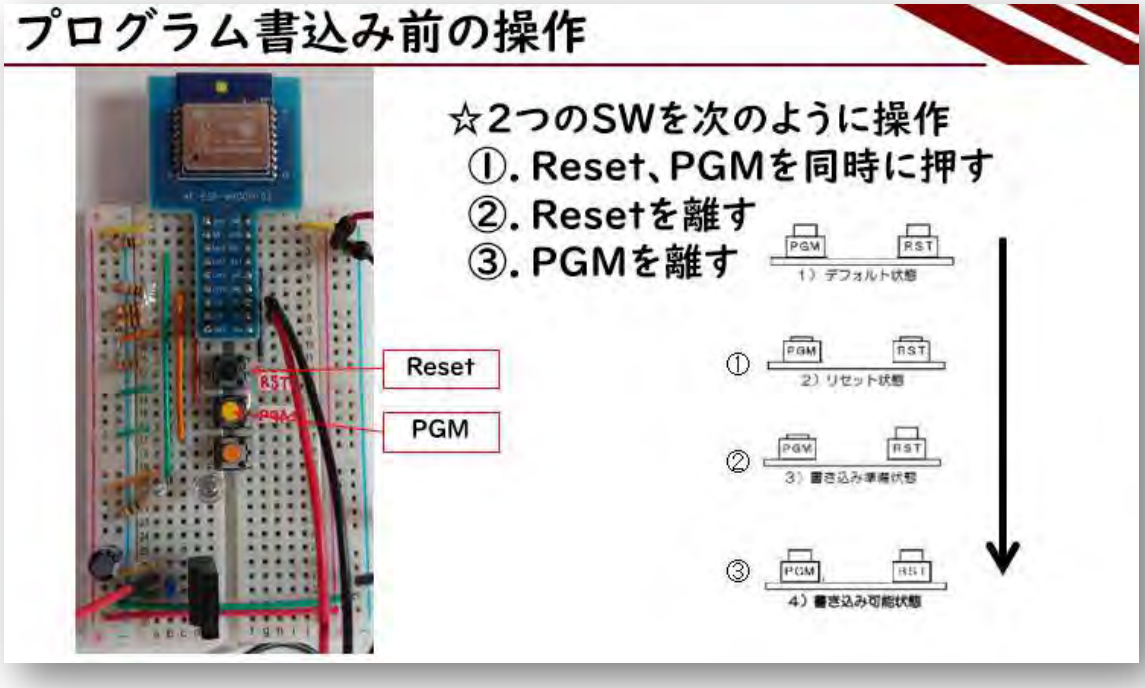


図　377

マイコンプログラム書込みの準備操作を行います。

図 378

IDE 左上の右向き矢印ボタンを押下します。

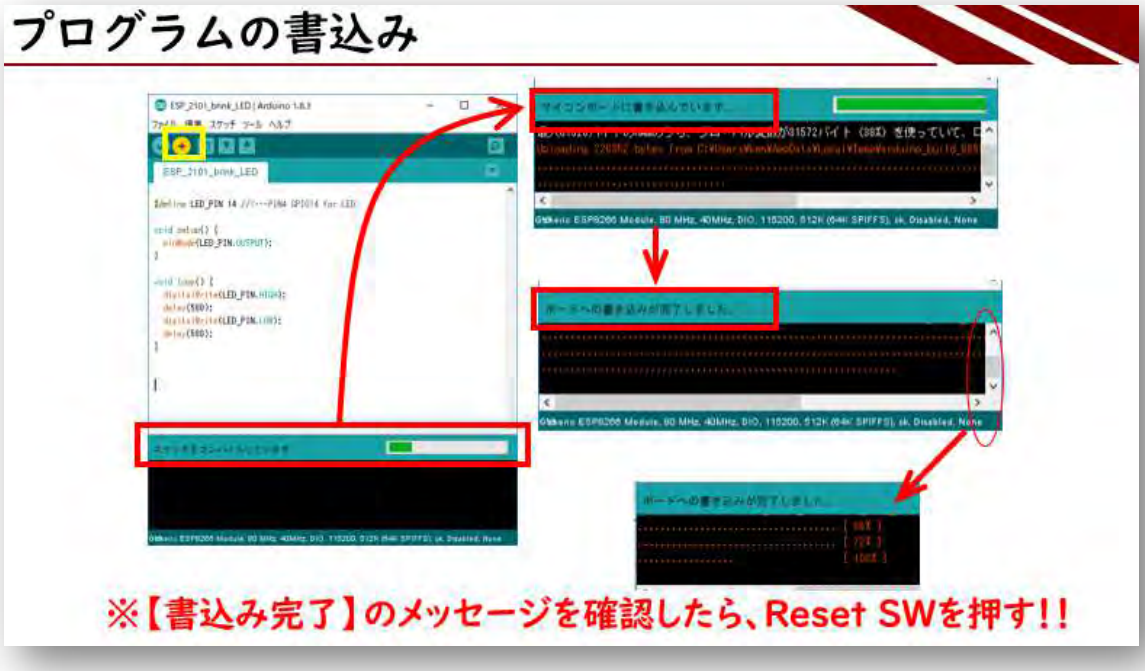


図 379

書込み完了のメッセージを確認後、マイコンを Reset します。

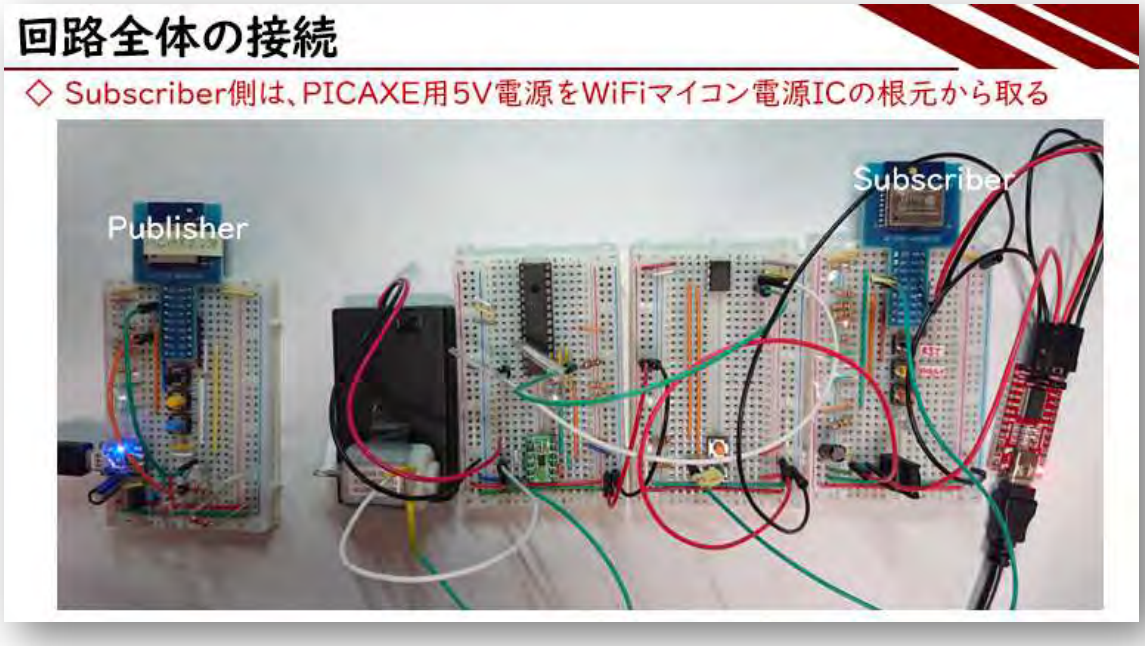


図 380

全体の接続確認を行って下さい。I2C マスタ・スレーブの TxD・RxD が GND に接続されていることを確認してください。

図 381

Publisher 側は、CdS CELL と抵抗が追加されました。

※基板の都合で LED と CdS CELL を近い位置に配置しましたが、本格的に回路を作成する際は、両者は離しておくべきです。

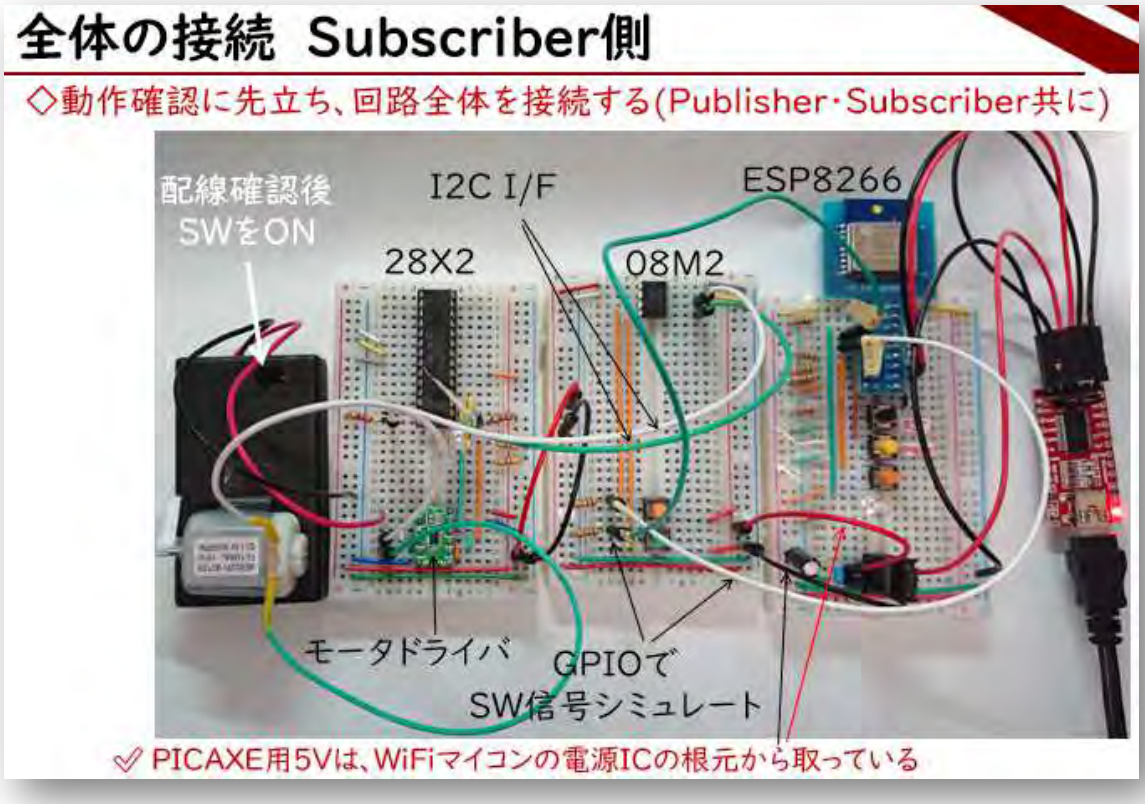


図 382

最後の動作確認です。十分確認をして一回で稼動させましょう！

説明図の通りに配線されている事を確認することが大切でが、**【なぜそこに配線があるのか】**を考えながら確認するのが良い開発者です。確認ができたら、動作確認直前にモータ用電池ボックスの SW を ON にスライドします。

動作確認はシリアルモニタも使用します。IDE の右上にある虫眼鏡マークのボタンを押下してシリアルモニタを起動し、開いたウインドウの下部にある通信速度を、setup()で指定した 115200bps に設定します。

図　383

まず、通常状態で何も起こらないことが大切です。モータは空転状態になっています。

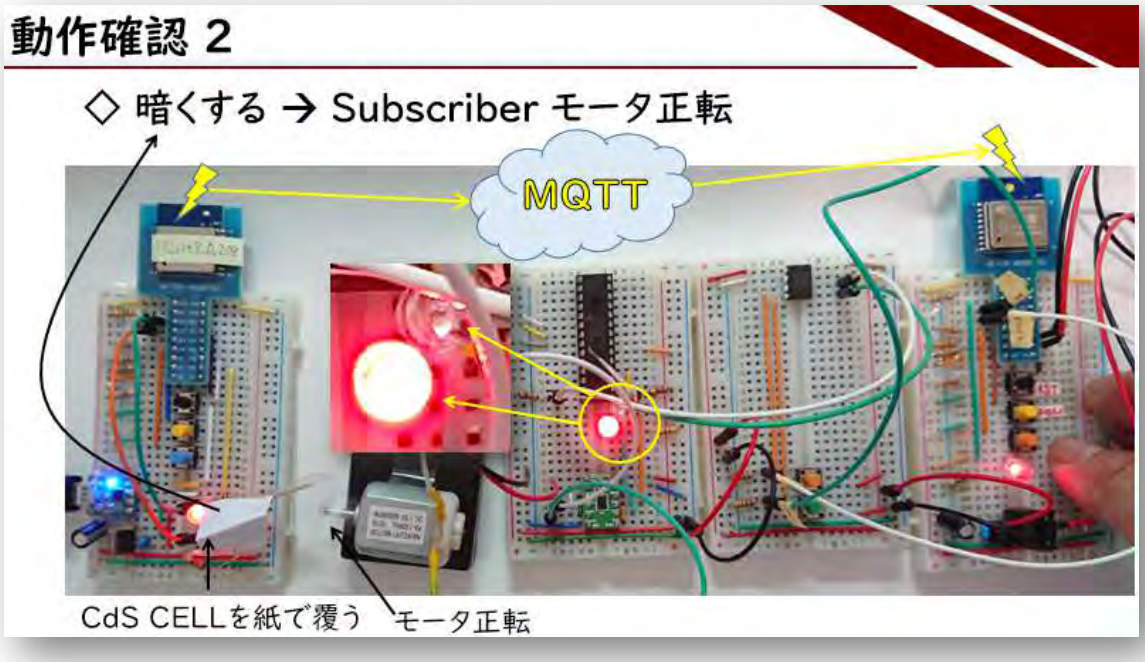


図 384

次に紙片で CdS CELL を覆います。暗くなると情報伝達経路の LED が順次点灯。LED1 が点灯、LED2 が消灯します。モータは正転状態になります。カーテン開き動作です。

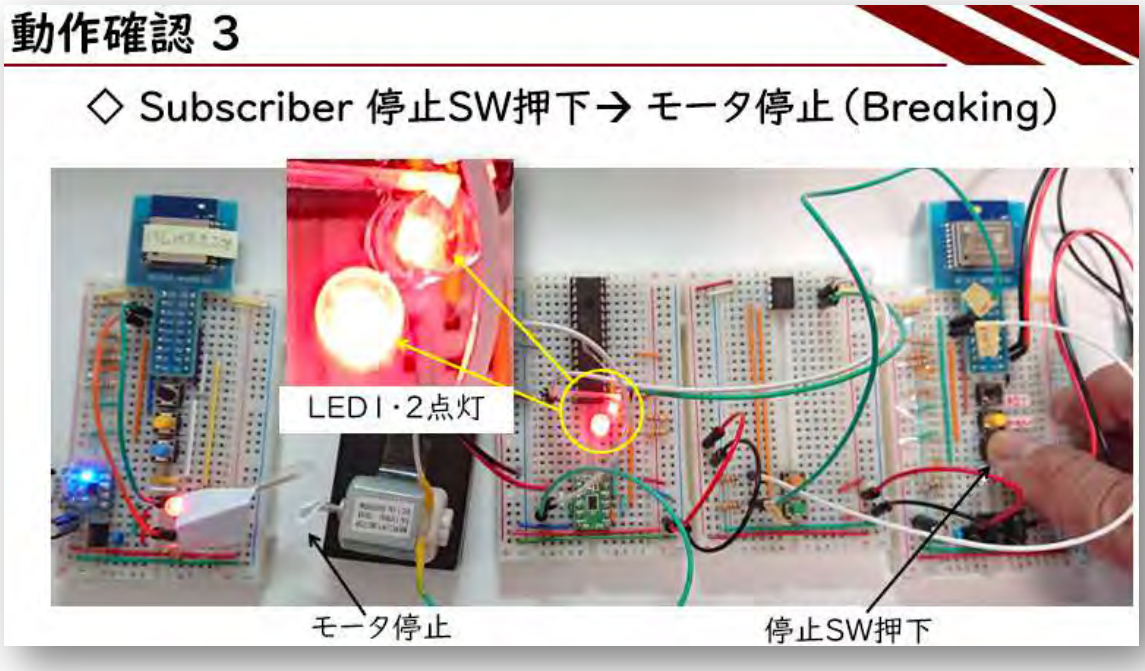


図　385

リミット SW と非常停止ボタンを兼ねた、Subscriber の汎用 SW を押下すると、LED1・2 が点灯し、モータにブレーキが掛かります。この状態はカーテン開き動作完了または、非常停止です。

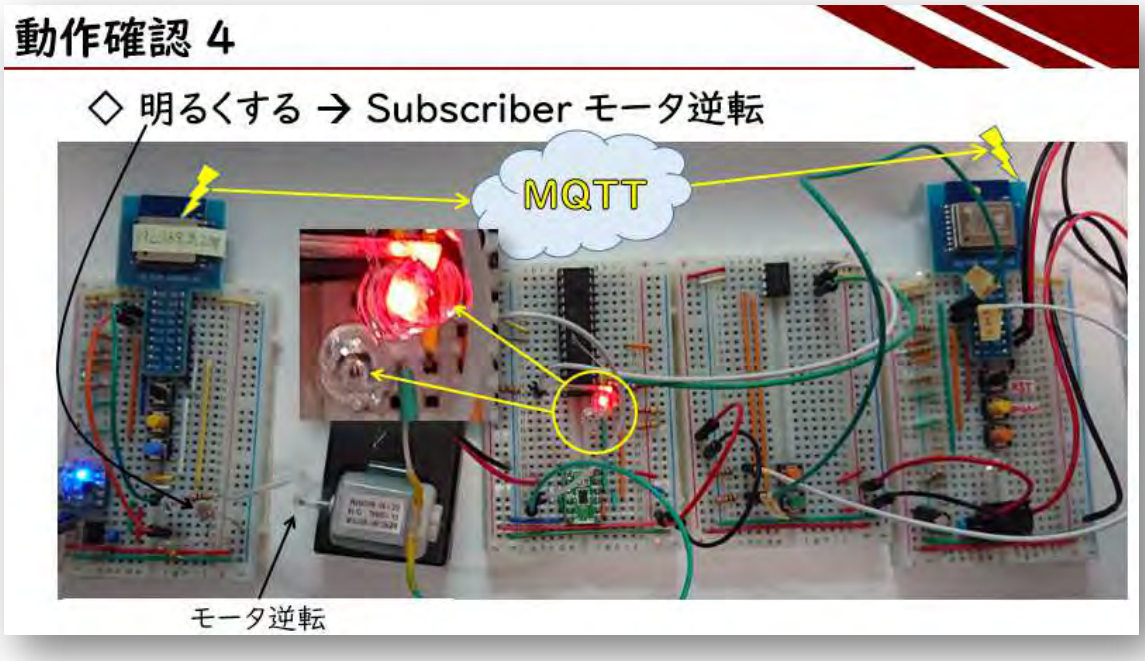


図 386

次に CdS CELL を覆っている紙片を外します。明るくなると LED1 は消灯し、LED2 が点灯します。モータは逆転状態になります。カーテン閉じ動作です。

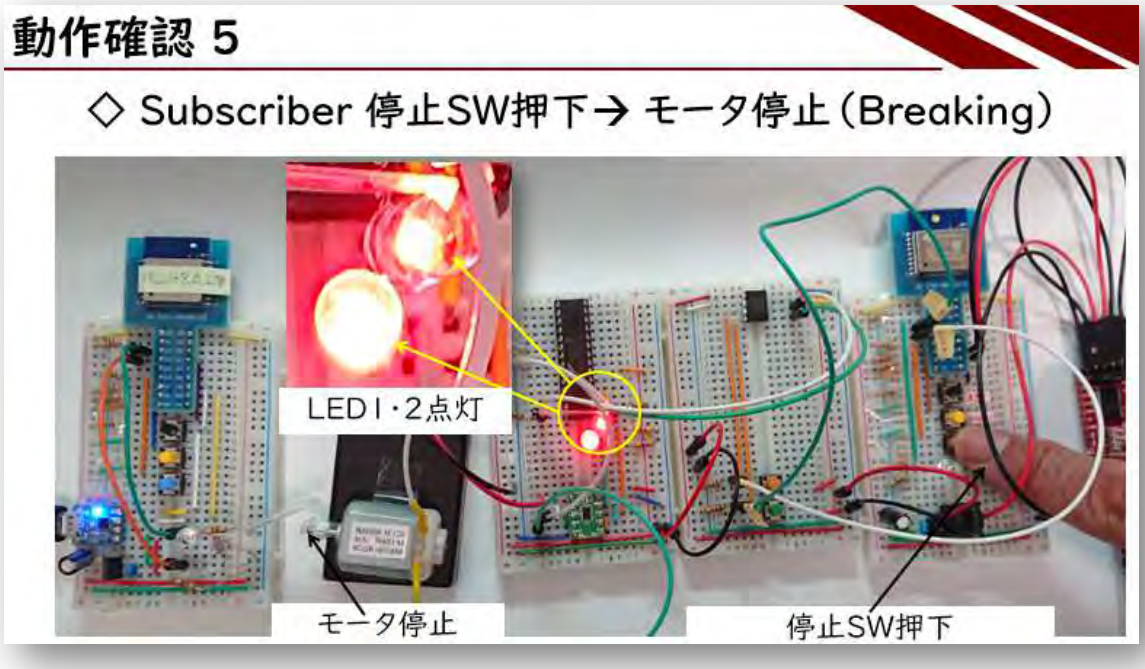


図　387

Subscriber の汎用 SW を押下すると、LED1・2 共に点灯し、モータにブレーキが掛かります。

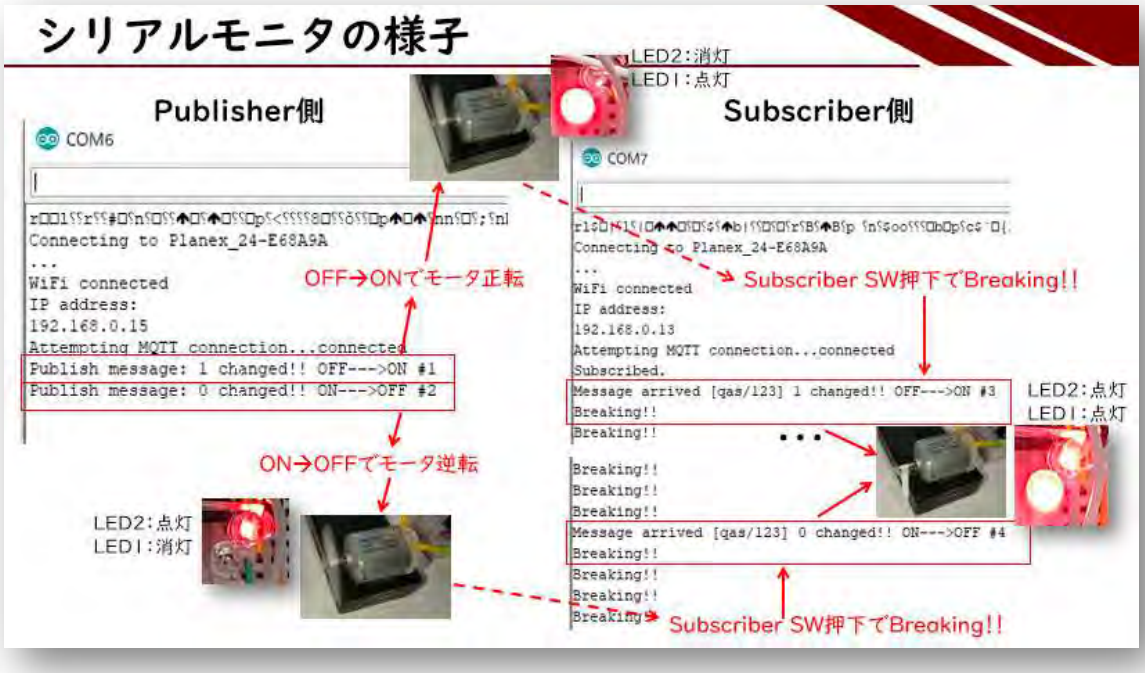


図 388

この動作の様子をシリアルモニタのメッセージと合わせて観察すると、図の様に変化していることが分かります。

動作確認では、設計した操作を何度も行い、設計通りに動くことを確認することが大切ですが、異なる状況になった場合はどうなるかを見ておくことも必要です。例えば、カーテン開き動作中に明るくする、等。このような事を動作確認時に観察しておくと、故障でシステムが異常動作した場合に、故障個所を素早く判断できます。また、次の開発に活かせるノウハウが得られます。

最後にモータの電源を OFF にします。

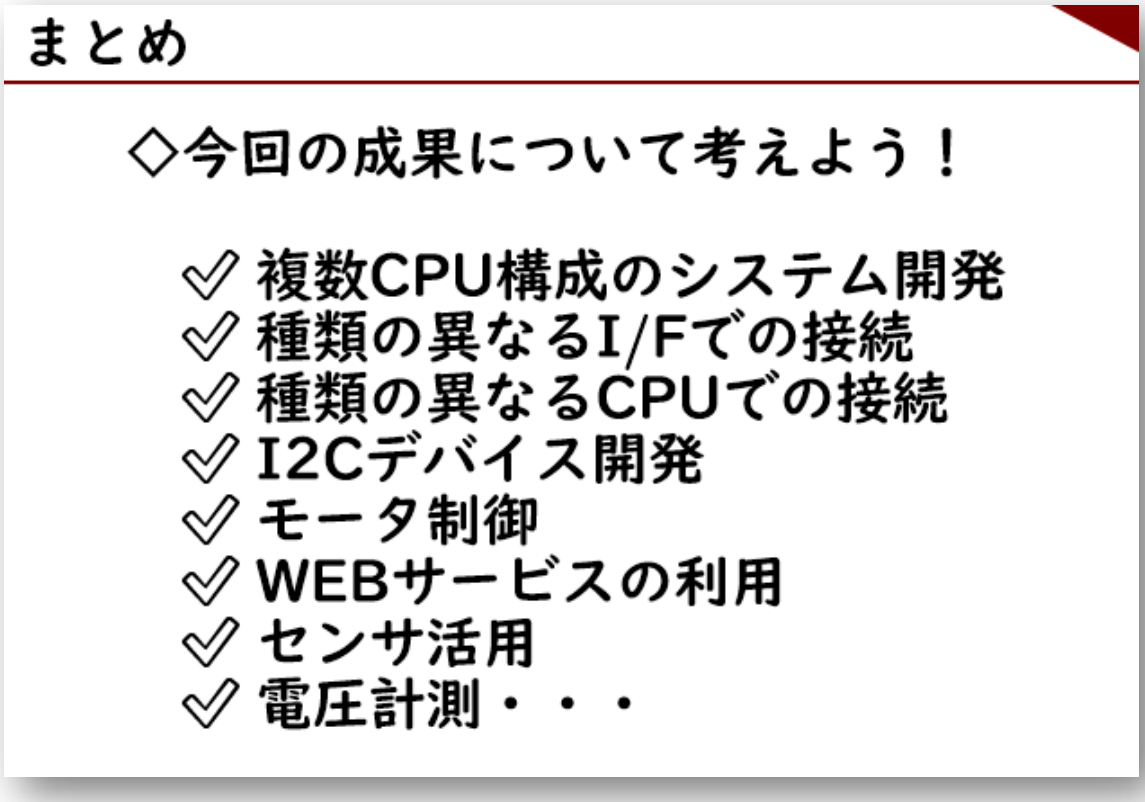


図 389

全 16 回にわたる実験はこれで完了です。この講座を振り返り、成果を確認する完成報告会を開催しましょう。

最後に【IoT の種】とは何か？ という質問をして講座を終わります。

最後に

最後まで読み進めていただき、ありがとうございます。教育用マイコンの超入門から、複数マイコンが絡んだ WEB 連携を含む実践開発まで無事終了しました。最終回を終えたときの感想はいかがだったでしょうか。全工程を終えた直後に各チームの完成報告会を行うと、記憶の底にこの【IoT の種】がやんわりと着床するのではないかと思います。

ところで、テキストの中で繰り返し同じような説明が出てくる点に、「また同じ説明をしているな」と思われた方も多いと思います。実は、それは私の思う壺です。「またか！」と感じるというのは、【経験済みのことを覚えている】つまり、以前出てきたことが記憶に残っているという事です。そうなっていただければ、時間をかけた甲斐があります。

終盤のチーム開発まで経験された皆さんに敬意を表します。

たいへんお疲れ様でした。

2019 年度「専修学校による地域産業中核的人材養成事業」
情報通信技術に対応した組込みシステム開発技術者育成のモデルカリキュラム開発と実証事業

IoT（組込み）統合システム開発教材

【IoT の種】

令和 2 年 2 月

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

●本書の内容を無断で転記、掲載することは禁じます。