平成２２年度文部科学省
産学連携による実践型人材育成事業
―専門人材の基盤的教育推進プログラム―

農業を対象分野とするITコンサルタントを目指す人材の育成プログラムの開発と実施

指導書

平成２３年３月

学校法人三橋学園
船橋情報ビジネス専門学校

はじめに

今年のように、猛暑が続く夏を経験すると、来年からは他の人達と違った温度の記録をしてみようとする農業家が出てきても不思議ではありません。実際、私のところには、『前日の最高・最低気温を日付・時刻とともに記録して、パソコンにつなげば、その記録が取り込めて、過去１年間分くらい、自動で記録してくれるような温度計は作れないか』という相談がありました。ＳＤカードを記録媒体として使い、一定期間ごとに、ＳＤカードを自宅に持ち帰り、パソコンで記録を読み出せば良さそうです。温度センサーを一定時間間隔で駆動して温度を測る。記録に日付時刻を含めるとなると、時計ＩＣを使う必要があります。時計ＩＣを使うためには、高精度のクリスタル発信器が必要でしょう。温度記録装置は、自宅から離れたハウスなどに設置することから、マイコンを使って電池駆動し・・・、などと考えて、機能設計・回路設計・ソフト設計をし、パーツを集めて様々な加工をして組立後、ソフト開発を行い・・・ここまでできれば、記録だけでなく、温度によるファンのコントロールや、潅水制御などを行うこともできそうです。そのように思えるような人材を育てる事が、本書の目的の一つです。

そのために、センサーを使う方法を覚えて、いざとなったらＩＣを使って、目的とする情報を自動記録するシステムを作ってしまえるような技術を持っていれば、農業家は頼りにしてくれます。頼りにならない人に、何も相談してはくれません。

『農業家にとって便利な人材』となるために、ＩＴ機器の使い方や、作り方を学びましょう。農業家から様々な相談を受けて、解決してあげま

しょう。農業家が頼りにしたい人材となるために、いわゆるソフトだけでなく、ハードウエアの範疇も学習して、マイコンやパソコンを駆使して、活躍しましょう。

データベースの知識を活かして、情報収集を行い、まだ経験の浅い農業家（あるいは、これから農業に参入しようとしている人達）の補助をしてあげる事もできるのではないでしょうか。

データロガーなる機器は、１万円くらいから購入できます。でもそれを購入するだけなら農業ＩＴ技術者は不要です。農業ＩＴ技術者は、データロガーを欲しい農業家が居たら、その目的と費用などを勘案して最適な方法を提案する。もし、最適なものが無かったら、あなたが作って挙げるような、農業ＩＴ技術者になってください。もしあなたが、メーカーで仕事をする農業ＩＴ技術者なら、自社でそのニーズを取り込み、多くの農業家に使ってもらえそうなＩＴ機器を安く製造販売して、農業家の役に立ってください。

話がもどりますが、最高気温・最低気温の判断を、記録する側（マイコン側)で行うのが良いか､パソコンに取り込んだ後で行うのが良いか？これも、悩ましい問題です。あなたなら、どうしますか？

【テキスト編集にあたり】

これまで、筆者が IT エンジニアとして農業家の方々と接してきて感じ取れたことは、PC をはじめとする IT 機器・システムの取り扱いには意欲的な方々が多いということと、丁寧な説明・解説・対応を行うエンジニアを求めていると云うことです。

そのエンジニアには、IT に関する HELP を求めています。農業に関する HELP は、農業家の皆さんの知識と経験、あるいは、JA や農林振興センターの指導員の方々の知識と経験があるので、そちらにお任せすることがベストでしょう。私たち IT エンジニアがお届けできることは丁寧な IT 技術の導入指導です。それも難しい技術ではなく、基本的な事柄を理解して頂ければ、日々、農作業での改善・工夫をされている『頭脳を使う農業』を実践している農業家の方々は、ご自身で考えられて『こんな事ができるのではないか？』というアイデアを提供してくださいます。提供されたアイデアを実現して、農業家の方々に利便性を提供できる様にするために、私が行ってきたことは『基本の組み合わせ』です。自分が持っている技術を組み合わせて、できるだけの事を行ってきました。農業家の方々との交流が現在も続いているのは、その点が良かったのだと思っています。

私が組み合わせに使った『IT の基本』の主なものは、次のような事です。

1. PC の使い方（Office ソフトも含めた使用方法）
2. Office ソフトの利用方法の解説・指導（いわゆる会計処理がメイン）
3. 利用方法の応用として、農業家で必要な看板や旗・パッケージデ

ザインの仕方

4．農業家同士の情報交換の方法（インターネット・メール利用）

5．情報発信（ホームページ・新聞・文章作り）

6．日々の記録の効率化（農業日誌・携帯電話）

7．同、各種センサーユニットの使い方（使えるユニット作り）

この中で、1～5の内容は、情報ビジネス系の専門学校のカリキュラムには、すでに含まれているはずで、そこで勉強したことを実地で応用すれば、すぐにでも農業家の方々のお役に立てるはずです。6については、少し農業家の方とのおつきあいを深めて行くと、何が必要なのかが分かってくると思います。7については、情報ビジネス系の専門学校では、勉強する時間枠がないものと思います。

このテキストでは、特に6、7の部分の強化を図ることを目論見として、編集を行いました。ですから、最初にお読みになると、マイコンやセンサーや電源・・・と、電子専門学校のカリキュラムのように感じると思います。電子専門学校の学生諸君でも、1～5の項目を学べば、農業家の要望に応じて対応するエンジニアになると思います。7には、『制御』のセンスが不可欠ですので、その旨ご理解の上、勉強をすすめてください。

全般に、プログラミングによる柔軟な機能追加も含みますので、情報ビジネス系の専門学校の学生には大変魅力のある、実力の発揮し易い分野だろうと思います。

目次

１．農業と IT 技術

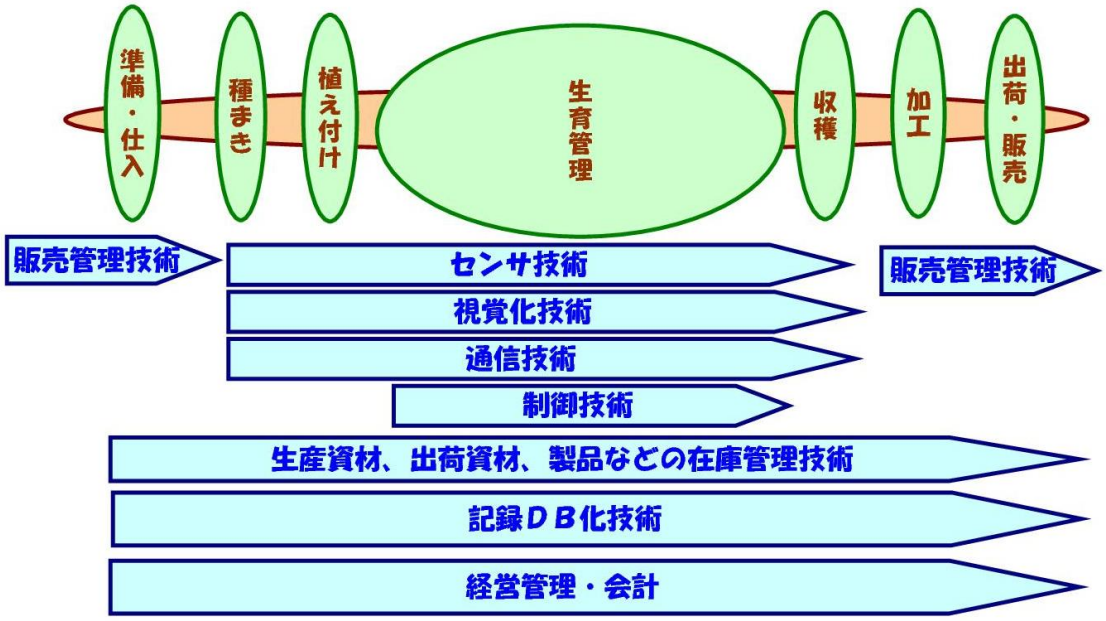


図 1.1　農業 IT マップ

農業には、さまざまな場面で IT 技術が活用される場面があります。

野菜栽培のフェーズと関連する IT 技術を概観してみたものが図１．１（農業 IT マップ）です。

１．１　農業の作業フェーズと、その内容

野菜栽培農家の作業フェーズを想定して、その内容を考えてみます。

１．１．１　準備・仕入

『何をどれくらい、いつごろから、どの畑で栽培するか』を決めて、そのために必要な畑の準備を

前もって行う必要があります。畑を畝って、種まきの準備や苗植え付けの準備を行います。

また、栽培品種によっては、柱を立てたり、『マルチシート』など、畝を覆うビニールシートをかけたり、

直射日光を和らげる屋根を付けたり、極端に冷え込まないように、トンネルを被せたり・・・と、行うことは、いっぱいあります。

また、肝心の栽培品種の種や苗の仕入れや、元肥の準備、水遣りが必要な場合には、水源の確保。水源が確保できない場合は、車に積んで水を運ぶタンクなどの用意が必要で

す。

準備段階では、栽培品種の収穫・出荷までを見込んだ、全体的な計画を立てることも必要です。

１．１．２　種まき

準備した畑、仕入れた種などを、予定数量種まきします。栽培品種によっては、直播ではなく、『種まき用ポット』や

『種まきシート』『種まきマット』・・・など、いろいろと便利なものが販売されていて、種まき後、しばらく育てて、苗の状態にしてから畑に定植することもあります。

このテキストを書いている２０１０年８月～９月は、猛暑が続き、雨が降らず大根や白菜の種まきを行ってはみたものの、発芽が思うようにゆかず、まき直しをしたり、どうしてもうまくゆかなくて諦めてしまい、他の作物に畑を転用する農家も出ています。

自然相手の農業は、これで安心ということがないので大変ですが、その分収穫時の喜びは、大きいと思います。

１．１．３　植え付け

定植ともいい、種から育てた苗を、畑に地植えします。一定の大きさに育った苗は、定植の際に、柱にくくりつけて

成長によって倒れてしまうことを避けたり、台風時期の強い風や雨から植物を守るように付帯の作業を行います。

農家によっては『木枠で囲んだ中に牛乳パックを輪切りにしたものを並べて、その上から種と土を混ぜたものを入れて育苗し、そのまま畑の上に置くだけ』というような方法をとっている人もいて、作業をいかにして楽にするか、工夫をしている様子がわかります。

１．１．４　生育管理

農業の醍醐味といったら、この『生育管理』でしょう。　日々のお世話と、日々の記録で、作物が毎日ぐんぐん育つ様子がわかります。　季節が、春から夏、夏から秋へと変化する中での、気象条件や、成長具合にあわせたお世話が、後の収穫に大きく結びつきます。　本書の主題である『農業 IT』も、この生育管理期間が一番効果を発揮する時期です。

不要な芽を取り除く『わき芽かき』、植物の茎や枝を支柱などにくくりつけて、目的の方向に育つようにする『誘引』、『水やり』、『雑草取り』。『追肥』や、『間引き』。ハウス内の温度を管理するための『換気』。全体の様子を見極めるための『日誌・記録作成』等々。できることは数多く、どこまで細かく管理できるかが、ポイントです。

栽培品種によっても、いろいろと細かな管理ノウハウがあるので、それに応じた IT 技術があります。

１．１．５　収穫

一定の大きさになったり、実が熟したり、根菜類は葉っぱの色などで地下の様子を推測して、期待が持てそうならいよいよ収穫です。収穫時は、一度に収穫時期が来てしまうと収穫作業が大変です。ある程度まとまった出荷を目指すのであれば、種まき・定植時期を少しずらした作業を行って、一定の期間で収穫が終えられるように、生産計画を立てます。

また、水やりの調整や追肥の時期、日々の育成管理で出荷の時期を微妙に調整したりできるようになると、栽培品種をコントロールできる感じが実感できて充実感が湧くでしょう。

１．１．６　加工

収穫した野菜は、そのままでは出荷できません。　土を洗い落とし、大きさをそろえ、一定量をまとめてパッケージに入れて生産者のラベルを貼る等の作業があります。

１．１．７　出荷・販売

JA などに出荷する際は、指定のパッケージに入れて集荷場に決められた時刻までに持ち込みます。当然、搬送は自力です。軽トラで出荷する農家は多いものです。出荷量の多い農家は、一気に大きなトラックで出荷したり、出荷先を分散して直売場に出したりもしています。最近では各地の直売場に大変人気が集まり、朝の早い時間帯（6 時頃）に直売所の前で開店を待つ農家の行列ができているのを見かけます。

JA や市場に出荷すると必ず売り上げになりますが、競りにかけられますので、値段はその日の周りの状況と出荷した品物の善し悪しや全体の出荷量などにより変動します。その点、直売場は自由な価格設定ができるので、自信のある農家は直売場に並べたり、自分で直売所を開設したりして、売り上げを上げています。

最終的には、生産したものがお金になって帰ってくることが重要なので、みなさん出荷先には当然神経をとがらせています。

競りにかかると実際の消費者の声は聞こえにくいのですが、直接販売なら喜んでいただいた声も届くので、その分張り合いがあるのは当然です。でも、クレームも直接届きますが・・・

直売が軌道に乗ると、自分の育てた作物が一つのブランドになってゆくので、その点での充実感はとても言葉では言い尽くせない気分でしょう。

１．２　関連するＩＴ技術

１．２．１　センサー技術

農業は、自然とのつきあいですから、いろいろな計測が必要になります。計測するに

はセンサーが必要です。そこで、農業に関連する、センサーを考えてみましょう。

誰でもまず思いつくのは、温度を計測することではないでしょうか。温度といっても、いろんな温度があります。

□ 気温--------->畑の上に被さる空気の温度。これは、時々刻々、風の影響や雲の影響、雨の影響で大きく変わります。天気予報では、最高気温・最低気温などが報じられますが、農業に必要な気温はどのようなものでしょうか？

IT 技術者を目指すみなさんが、スキルとして身につけなくてはいけないのは、

【どのようなレンジで計測するのか】

たとえば、低温側は－10℃からでよいのかそれとも－20℃か？

同じように高温側は、40℃か、それともももう少し高くて 45℃か？

ハウス内でしたら、45℃もあり得ますね。その代わり、零下は不要かもしれません。でも、露地の畑なら霜や雪、日照りなどいろいろとありますから、それなりの計測レンジを考えなくてはいけません。

【どのような頻度で計測するのか】

たとえば、日に 1 回？

1 時間ごと？　30 分ごとか、それとも毎分？

あまり間隔が狭くなると、計測装置の A/D 変換時間も検討する必要がでてきます。

【どの程度のスパンで記録するのか】

つまり、1 日分でよいのか、1 週間分か、1 ヶ月あるいは 3 ヶ月にもわたるのか？

それにより、記録に使うメモリのサイズが算定されます。

最近は、いろいろなメモリーカードが安価に提供されていますが、メモリーカードを使うのか、それとも IC メモリを使うのか？

IC メモリはカードに比べて容量は小さいですが、低消費電力で駆動できて、取り外すことがない（基板に取り付けてしまいます）ので比較的丈夫です。電源が切れても記憶内容が消えることはありません。

メモリーカードは、取り外せば PC で読み書きができ、長期間にわたり多くのデータを記憶できる利点がありますが、インターフェイスがやや複雑でメモリーカードに計測結果を記録するプログラムは、多少複雑になります。

温度センサー一つとっても、多くのノウハウが必要であり、理屈だけでなく、実際にセンサーを使ったマイコンシステムを作成して、温度を記録してみるとよい勉強になります。皆さんは、是非、勇猛果敢にチャレンジしてください。

※ このテキストでは、後の章で SD カードに温度データをロギングするシステムを作ります。

□ 水温--------->灌水の温度や、水路の温度などを計測することになるでしょう
この際、対象が『水』ということが、問題です。ＩＴ技術で使用するセンサーは、必ず電気を使いますから、絶縁対策が必要になります。また、レンジで考えると、水温は、零下は氷ですから、対象の最低水温は、0℃より大きいと考えられます。

□ 土壌温度-------->畑の土の中の温度。実際は、土に含まれる水分や空気、これらが入り交じった水蒸気や土壌成分の温度が計測されると考えられます。水分が多い場合は、露地の畑では、冬場に凍っていることもあり得ます。これに対してハウス内の土壌は、少なくとも凍ることはありませんので、それなりの計測レンジを考えてセンサーを使います。

気温はセンサーむき出しでもかまいませんが、水温は絶縁が大事です。土壌も水分を含むことから、水と同じように絶縁が必要でしょう。独自にセンサーユニットを開発するには、オールマイティに使えるよう、熱伝導のよいケースにセンサーを絶縁して封入し、棒状にして水・土壌に差し込む、または気温を計測したい場所（高さ）に、固定する等の設置が必要です。

センサーは電気を必要としますが、大きな電流を流すようなものはありませんので、電源は、周辺回路と併せて電池駆動なども検討できます。

センサーで検出した値は、そのままでは比較検討したり、統計を取ったりするのに都合が悪いので、取り扱い易い形に変換して、記録します。その際、センサーと周辺回路で、検出値を電圧に変換して、その電圧を、マイコンの A/D 変換ポートに入力します。　読み込んだ電圧値を、0～255 (8bit)や 0～65535 (16bit)数値にして記録します。

他にもいろいろなセンサーがありますが、詳細は後の章で詳解します。

１．２．２　視覚化技術

センサーで、検出した値は、最高精度が 1bit 表現ですから、一番簡単な視覚化は、その bit に対応する ON/OFF 表示です。　つまり『ランプが点灯しているか、していないか』になります。　ある閾値より高いか低いか等を表す場合は、『L・H』等の文字表示を利用することもありますが、ON/OFF 表示と同じです。

また、検出値をそのまま数値として読みたい場合は、10 進表示が必要ですから、7 セグメント LED や液晶表示器を使って数値として表示します。この際、8bit または 16bit の 16 進数値を 10 進表現に変換するのはソフトウエアで行います。

『視覚化』というと、画像のことを思い浮かべる方が多いかもしれませんが、安価に簡単に実現できて用途の多いものの方が、ほとんどの場合便利です。

もちろん、畑やハウス内の全体の様子や少し離れた水源の様子をカメラで監視したりすることもありますが、カメラ監視や画像記録は何か起こったときの保険のようなもので、撮影した画像を元に、何か自動で判断し、制御を行うような使い方は、まだあまり応用例

がありません。結局、圃場を見に行かなくても、作物や土の状態を観察できるので、カメラがあれば、広い農場での作業計画を立てることには役立ちます。ただ、この場合は、解像度が大きく、ズーミングやパン・チルト動作がリモートでできるようなものが必要になります。

変わった応用例では、ハウス内の空気をブロワーで吸い込んで細いホースに導き、ホース内部をカメラで撮影して画像処理を施し、害虫の発生とその数を把握する。などという方法も開発されています。基本的には、管理している農業従事者の目で判断して、仮にカメラを害虫検出に使ったとしても、あくまで補助的なものに過ぎません。　センサーを設置した場所と離れたところで害虫が発生していても気づかないこともあり得るわけですから、一つで万全といえるセンサーは、なかなかありません。

１．２．３　通信技術

通信というと、一般的には電話や無線、ネットワークなどによる、少し離れたところとの、情報交換を思い浮かべると思います。

農業に関連するＩＴでは、どのような通信が行われているでしょうか。また、どのような通信技術が必要でしょうか。

温度などの情報をセンサーで検出しても、そのままでは使えないので、人に分かり易いように数値化するためにマイコンに送ったり、または記録しておくために外部の不揮発性メモリに保存したりします。センサーにつながっているマイコンは、センサーから受け取った情報をパソコンに送り、後に統計などで使用したりします。

最初に情報を受け取ったパソコンは、全体を管理する別のコンピュータシステムにその情報を転送することもあります。

また、農業従事者が手入力した情報を、別のコンピュータシステムや、他の農家に送ることもあるでしょう。

このように、情報を伝えあう通信は、至る所で使われていて、その実現方法も様々です。今後も新しい通信技術が開発されてゆくでしょう。農業ＩＴ技術者としては、適材適所の方法・手法を選択、組み合わせて、農業の手助けをする役割があります。

このテキストでは、農業にまつわる通信をつぎのフェーズに分けて、後の章で詳解します。また、開発例も紹介します。

１　センサー・マイコン間通信
２　マイコン・メモリ間通信
３　マイコン・ＰＣ間通信
４　ＰＣ・ＰＣ間通信

さて、センサーにて『検出したデータを、マイコンに送る』または『記録、統計などのためにパソコンに送る』などの処理は、通信技術を利用しておこなわれます。　また、

マイコンで数値化したデータを、後にパソコンで利用するために、不揮発性メモリに書き込んでおくことも通信技術を用いて行われます。

センサー自体からの出力はアナログ出力のものとデジタル出力のものがあり、デジタル出力のものはシリアル通信を使います。このシリアル通信もいろいろな規格が作られていて、IIC 方式(I2C 方式)，SPI 方式、CAN、1Wire 方式などがあります。

規格の名前でいうと難しいように感じますが、センサーの足 PIN に付けられた名前と、マイコンの足 PIN の名前を合わせて接続して、その規格に対応する通信ライブラリを用いれば、難しい通信の規格・仕様書を読まなくても、実現することができます。

本書では、後の章で液晶表示器（LCD）や SD カードをマイコンに接続していますが、この接続が、IIC 方式であったり SPI 方式であったりします。プログラムも、掲載していますので、参考になると思います。

１．２．４　制御技術

農業ＩＴで必要となる、制御技術を考えます。単純に考えて、次のような事例です。

１．温度が上がりすぎたら冷やす、冷えすぎたら暖める。
２．暗くなったら、照明を点ける。
３．時間になったら水を出す。
４．一定時間経過したら水を止める。
５．一定時間経過したら照明を消す。

この中で、１と２、３、４と５は、それぞれ同じ内容であることを理解してください。

１と２：　センサーからの検出値が、一定の値（あらかじめ決めておいた『閾値』）を超えたら、『どうするか』

３：時計を持っていて、一定の時刻（あらかじめ決めておいた『閾値』）を超えたら（または、一致したら）『どうするか』

４と５：ある動作を開始してから、一定時間（あらかじめ決めておいた『閾値』）を超えたら『どうするか』

『どうするか』もまた、あらかじめ決めておいた『動作』になるように制御を行えば、自動温度管理システムや、自動電照管理システムや、自動灌水管理システムが構築できるわけです。急激な温度上昇などには、それまでの温度変化を記録しておいて換気扇の ON/OFF と併せて、灌水のコントロールも組み合わせるなどすれば、より柔軟性の高いシステムが構築できます。

温度や、周りの明るさなどはセンサーで検出すればよく、時刻はマイコンやパソコン内部の時計を使えば実現できます。また、カレンダータイマーＩＣというデバイスがあるの

で、これをマイコンと接続して、高精度の時刻・カレンダ（日程）まで管理できるようにもできます。

この場合の精度は、温度は1℃刻み、明るさについいては、用いる場所により、かなりラフな検出でかまわないと思われます。

また、時刻や、タイマーは、通常は、マイコン内部のタイマー機能で十分な精度が得られますが、さらに精度を求める場合は、水晶発振子をカレンダＩＣ用に別途取り付ければ、高精度になります。

１．２．５　記録・DB 化技術

記録技術

農業従事者は、必ず日記なりメモなり、いろんな方法で毎日の記録をとっています。短期では、役立つことは少ないかもしれませんが、長く農業に従事していると、統計的な情報が必要になる場合が多く、また自分の耕作地に特有の性質・傾向などがわかってくるから、長い間記録をとり続ける必要があるのです。

我々が子供の頃には、知り合いのおじさん達は、夕方暗くなって帰ってきて、お風呂に入り、ご飯を食べながら一杯やり、さてそこから一日の作業を振り返る・・・となると、大まかな記憶に頼る記録になってしまい、詳細な記述はできないことが多かったのが現実です。時には、数日まとめて記録するなど、後回しになっていたこともあるでしょう。

こんな時、現在ならば『携帯電話』があります。操作は少しやっかいですが、その場でメモをとることができます。また、携帯電話のカメラ機能はとても高機能で、いまやハイビジョンムービーも撮影できるようになっています。

携帯電話のメモ機能も有効ですが、どおせなら、その後が楽になるように携帯の電子メール機能を使って、写真やビデオを添付した記録をメールで送信すると時系列に整理保存されるようなシステムがあると、とても便利です。電子メール機能を使うことによって、その時、その場で記録ができて、画像も一緒に保存でき、たとえその時の作業場所が携帯電話の電波の弱いところであっても、日誌メールを作成してしまえば、後からまとめて送信もできます。

ただし、このシステムは、携帯電話以外にパソコンが必要なのはいうまでもありません。また、メールアカウントも使用しますので ISP（Internet Service Provider）との契約で、メールアドレスを取得しておく必要があります。

そこで、このような機能を、普通のパソコンなら簡単に実現できて、後から記録を検索したり、統計を取ったりすることも楽に行える農業日誌システムが２００９年に実用化されました。『アグリー』がそれです。

このシステムは、Excel に組み込まれたプログラムによって動作していて、携帯電話から送ったメールをワークシートに時系列に保存するようになっています。添付の画像は、枚

数などに制限がなく携帯電話の可能な添付画像数・ムービーの撮影時間内であれば、いくらでも添付記録ができます。この、ムービー画像を利用すると直売場などでは、簡易コマーシャルを作成することが可能です。出荷されている農家の特徴などを生産者自信が作成できて、直売場のお客様に生産者の生の声と、画像を届けることができます。

また、記録は Excel のワークシートに行われるので、後で特定の項目や生産物などで絞り込んで経過を見たり、統計を取ったりすることが、オートフィルタ機能を使って容易に実現できます。

よいことばかりのシステムですが、難点があります。それは、最近、Excel のバージョンアップによって、マンマシンインターフェイスが変わってしまい、Excel 自体が使いにくくなってしまったことと、OS のバージョンアップにより、Excel の動作が遅くなった事です。

携帯電話の操作が困難な方には、このシステムは残念ながら使えないでしょう。ただ、農業 IT を活用しようとする生産者は、そんなことはないでしょうから、安心しています。

DB 化技術

記録ができてしまえば、DB 化は容易です。　しかし、DB を作成する際には、注意が要ります。

まず、何を目的として DB 化するかです。

単に記録を後に検索するためだけでしたら、とりあえず、メール機能を利用した日誌の全文と画像を DB 化してしまい、後の検索では全文・フリーワード検索を行ってヒットした結果をみればよく、特定目的の統計などを行おうとするときは、全文 DB をさらに処理して、キーワードで分類したり、数値化をあらかじめおこなって別の DB を作っておくこともできます。

キーワードの設計は、開発時、その時に最善と考えられていても、長いスパンで DB 化を続けて蓄積された情報が多岐にわたると、違う言葉で検索してみたくなるものです。

もとになる情報が同じでも、キーワードの振り方だけで目的に応じて、使いやすい DB とすることができます。

大切なことは、記録を漏らさないことです。漏らさず何でも記録することです。記録されていないことは、どのようにしても調べることはできません。

２．センサー

２．１　センサーの種類

２．１．１　画像センサー

WEB で『画像センサー』を検索すると、いろんな用途があることがわかります。

ある産業用制御機器メーカーの WEB には、『画像センサーとは、CCD カメラでとらえた対象物の画像をデジタル信号に変換し、種々の演算処理を行なうことで、対象物の面積、長さ、個数、位置などの特徴を抽出し、設定された基準をもとに判定結果を出力するものです。』と説明があります。

『文字認識画像センサー』OCR 機能を使って、カメラで撮影した（刻印や印刷の）文字部分を認識して、辞書と比較して一致しているか、不一致か、どの部分が違っているか等を判断して結果を出力するセンサー。

『カラーセンサー』
測定する対象の色が，所望の色と合致するのか，合致しないのかを判断するといった，主として色を判別するために使うセンサー

『CC イメージセンサー』

農業ＩＴで用いる画像センサーとは、単に圃場や栽培している野菜の生育状態を、現場で見るのと同じように、画像として提供する、いわゆるカメラ機能を指すことがほとんどです。

ネットワークインフラができてさえいれば、いわゆる『WEB カメラ』で目的はほぼ達成できます。　解像度が問題ですが、これは光学系（レンズなど）を工夫するか、または、CCD の大きなカメラを使えばクリアできます。

リアルタイムな画像情報が必要な場合は、LAN 等を使って、圃場と管理する場所を結ぶことができれば、今現在の生育・栽培の様子が見て取れます。

LAN などを引くことが難しい場所には、携帯電話のテレビ電話機能などを使ったシステムを組むことができます。携帯電話に自動着信・応答装置を取り付けて、必要な都度電話をかけると、現地の様子を映像で確認できるようにするものもあります。

最近、携帯電話の様々な割引サービスがありますので、これを利用して低料金で上記シ

ステムを使うことができます。ただし、双方ともテレビ電話機能を持った携帯電話に限られますので、機種選定には注意が必要です。

２．１．２　温度センサー

温度センサーは、文字通り温度を測る物ですが、システムの中でセンサーは、温度を直接測っているのではなく温度に対応したセンサーの出力電圧を計測しています。

ある温度に対応する電圧が分かっていれば、変換表を利用したり比例計算などで温度変換ができます。これをマイコンなどでソフト的あるいは、ハード的に行い、温度計測をしています。

良く使われる安価な温度センサーとしては次のようなものがあります。

１．LM35DZ

計測範囲：０℃～１００℃

※マイナス側は測定できませんので、注意。

精度：±1℃

温度係数：10.0mV/℃

価格：約１００円

２．LM61CIZ

計測範囲：－３０℃～１００℃

精度：±1℃

温度係数：＋１０ｍＶ／℃

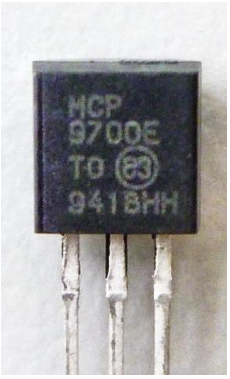
価格：約５０円

３．MCP9700-E/TO

計測範囲：－４０℃～＋１２５℃

精度：精度：±４℃（最大）

価格：約３０円



４．S-8100B

計測範囲：－４０℃～＋１００℃

・リニア出力電圧：１度あたり－８ｍＶ（負の傾き）

－２０℃：１．９０８Ｖ

＋３０℃：１．５０８Ｖ

＋８０℃：１．０９５Ｖ

・１個２００円

上記は、温度に対応する電圧を出力するセンサーですが、電圧値をコンピュータ処理するためには温度換算しなければいけませんが、A/D 変換過程・温度換算過程で誤差が生じます。これを解消するために、直接温度数値を出力してくれるデジタル温度センサーもあります。LM75BD がそれで、価格も安く（５０円くらい）I2C インターフェイスをもつマイコンならば簡単に接続ができて、温度を直読する事ができます。

LM75BD

ここに紹介したようなセンサーをマイコンとつなげば、安く簡単に温度測定器が作れます。もちろん PC との連動で長期にわたり温度の変化を記録するような事も可能です。しかし、いちいち作ることが困難な場合には、温度データロガーなる機器がありますので、これを用いれば、買ってくるだけで一定時間ごとに温度の計測を行ってくれて、PC に計測データを取り出したりすることもできます。

どちらを選ぶかは、そのときの状況次第です。もし、あなたがメーカー技術者であれば、農業家のニーズを理解して、安価なロギングユニットを製造販売すれば、喜ばれるでしょう。また、あなたが農業生産者側の技術者でしたら、自分達のニーズにあうユニットをこしらえて、IT を応用した農業家を育ててください。

２．１．３　湿度センサー

湿度センサーも、基本的に温度センサーと同じように、ある湿度に対応する電圧を出力するので、それをマイコンの A/D 変換ポートに接続して、変換表や換算計算のプログラムで湿度を求めます。（写真は、GE 製 HS-15P、約５００円）

HS-15P

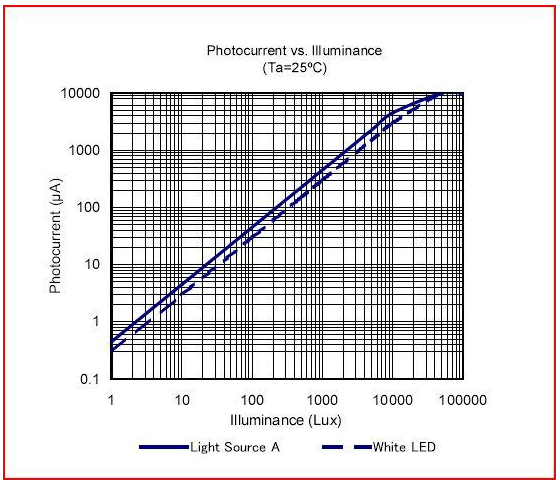
温度センサーは、リニアに更正された電圧を出力するものが多いのですが、湿度センサーは、その出力電圧が湿度に対して指数関数的に出力される物が多く、また温度補正をかけないと正確な湿度が把握できないので、外部に回路を組むかマイコンに接続して使用する前提であれば、それなりのプログラムが必要になります。また上記湿度センサーは1KHz の交流駆動が必要なのでドライブの工夫が必要です。他の湿度センサーもほぼ同様の仕様で交流駆動は必須です。湿度センサーは実際の湿度との対応を行うために、更正を行う必要があり、温度センサーほど簡単には使えません。

下記に、このセンサーHS-15P について、大変参考になる WEB ページがありますので、是非一度ごらんになってください。

参考 URL：http://homepage3.nifty.com/sudamiyako/zk/hs15/hs15.htm

２．１．４　照度センサー

明るさを記録するには、特性が人間の目の感度に近いフォト・トランジスタが使えます。

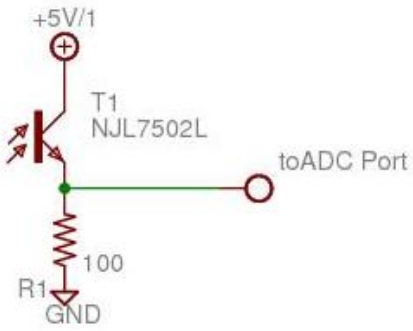


照度センサー(NJL7502L)(左)と特性(右)

5V の電源に抵抗と直列に接続して、光電流を電圧に変換します。あとは、この電圧をA/D 変換すれば、照度が測定できます。

このセンサーは光電流：33μA　(100Lux)なので、照度換算は次式でおこないます。

照度 ＝V＋((33μA × 100Ω) ÷ 100Lux)



照度センサー利用回路(100Ω抵抗の電圧を A/D 変換する)

２．２　センサー検出値の数値化（マイコンを活用した簡易温度計作成）

センサーの検出値は、そのまま直読できるのであれば便利ですが、たとえば温度センサーは温度に対応する電圧（アナログ値）が出力されて、それをＡ／Ｄ変換して、私たちが理解しやすい、または使いやすい形の数値に変換して、何らかの形で表示する事が必要です。このように書くと複雑そうですが、実際にやってみると案外単純です。

この章では、温度センサー（S-8100B）をマイコンのＡ／Ｄポートにつないで、簡易温度計を作成した例を解説します。この温度センサーは出力電圧が、リニアに更正されていて、１℃あたり－８mV の電圧変化を得ることができます。

温度センサーをつなぐことのできるマイコンには、Ａ／Ｄ変換機能が必要です。最近は、とても使いやすいマイコンが、多く出回っていて、使う側もいろいろと選択肢があり、便利になっています。マイコンを使う目的は、温度に対応した電圧をＡ／Ｄ変換することと、その電圧を温度に変換して、数値として出力・記録・表示することです。(時には、ある閾値を持たせて、その値を基準にして、ＯＮ・ＯＦＦ制御したりする事も必要ですが、ここでは温度表示に徹します。)

今回は、最低限の機能で温度を表示する、簡易温度計を作成してみます。

２．２．１　マイコンの選定と主要パーツ

マイコンは、マイクロチップ社の 8PIN マイコン（PIC12F675-I/P）を使いましょう。ピン数は少ないですが、汎用 IO ポートで LED 駆動ができ、１０bit のＡ／Ｄ変換入力

を４ｃｈ持っています。クロックは、２０ＭＨｚまで可能ですが、今回は内部発信を使い４ＭＨｚ動作として、部品点数を減らしています。プログラムメモリは、１Ｋワードのフラッシュメモリを内蔵しています。今回のような用途にはちょうど良いサイズです。当然フラッシュメモリですから、専用ライターで何度でもプログラムの書き換えができます。

主な仕様は、

◆クロック２０ＭＨｚ動作

◆動作電圧：２．０Ｖ（MAX4MHz）～５．５Ｖ（MAX20MHz）

◆１０ビット４ｃｈＡ／Ｄコンバータ内蔵

◆フラッシュメモリ：１Ｋワード

◆ＲＡＭ：６４バイト

◆データ用ＥＥＰＲＯＭ：１２８バイト

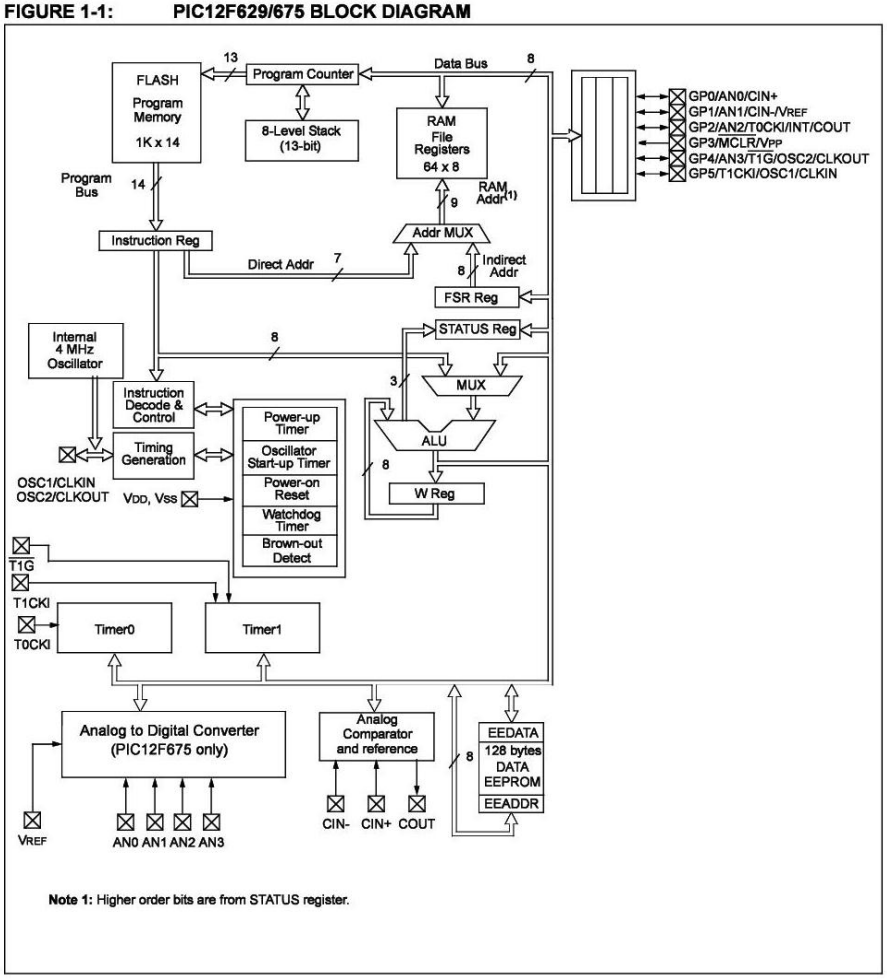
内部発信を選択すれば４ＭＨｚ動作し、外部に発信器は不要です。

以下に、マイクロチップ社が公開している PIC12F675 のデータシートの一部を掲載します。Pin Diagrams をみると、各ピンが複数の機能を持っていることが分かります。現在主流のマイコンは、そのほとんどが１チップで、ピンアサインは、プログラムで書き換えることにより、複数の機能を切り替えて使えるようになっています。

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)				
PIC12F629	1024	64	128	6	–	1	1/1
PIC12F675	1024	64	128	6	4	1	1/1

* 8-bit, 8-pin devices protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

FIGURE 1-1: PIC12F629/675 BLOCK DIAGRAM

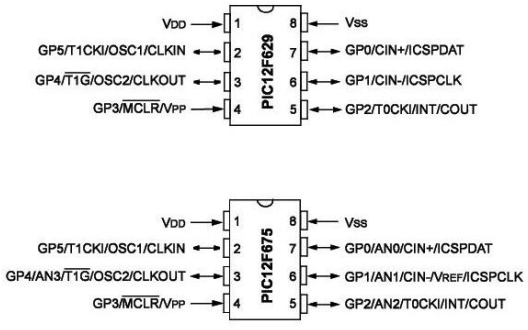


Note 1: Higher order bits are from STATUS register.

PIC12F629/675

Pin Diagrams

8-pin PDIP, SOIC, DFN-S



そのほかのパーツの選定、

①．ＰＩＣマイコン（PIC12F675-I/P：８ピン）

ピンの割り当ては、

・電源　　×１

・グランド　　×１

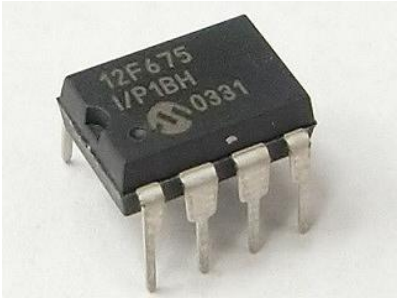
・入力専用ピン　×１（今回未使用）

・アナログ入力　×１（温度センサーと接続）

・ＬＥＤ表示　　×４

合　計　８ピン

・１個１２０円



②．温度センサー

（S-8100B：セイコー電子工業）

・－４０℃～＋１００℃の計測範囲

・リニア出力電圧：１度あたり－８ｍＶ（負の傾き）

－２０℃：１．９０８Ｖ

＋３０℃：１．５０８Ｖ

＋８０℃：１．０９５Ｖ

・１個２００円

③．LED×４個

・温度表示用

4bit でどう表示するか、工夫のしどころです。

・１個２０円

④．４７０Ω（1/6W）カーボン皮膜抵抗×４個

・LED 電流制限用

・１個１０円

⑤．三端子レギュレータ

（５Ｖ用１５０ｍＡ　XC6202P502TB）

・マイコン、センサー、LED 点灯に使う５ＶＤＣ電圧を乾電池から作り出します。

・５０円

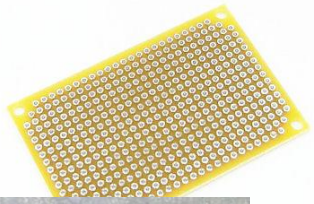
⑥．セラミックコンデンサ（０．１μＦ）×２個

・出力電圧の安定化を行います。

・１個１０円

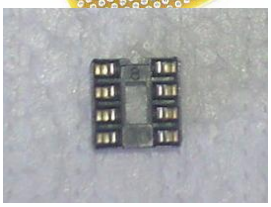
⑦．名刺サイズ汎用基板　×１枚

・１００円



⑧．IC ソケット

・８ピン用１個１０円



⑨．乾電池と電池ホルダー

- ６Ｖ以上の乾電池が必要です。
- 今回は、単３電池が４本入る電池ホルダーを用意しましたが、００６ｐ乾電池（９Ｖ）も使えます。これでしたら、電池ホルダーも不要です。

・１個５０円でした。

⑩．バッテリスナップ

・１個２０円

．コネクタ

バッテリスナップを基板に接続するために使いますが、

無くても構いません。ない場合は、基板に直に半田付けしましょう。

⑫．そのほか、半田、配線用線材少々

今回使う部品を改めて、リストにしました。

パーツリスト：

１．PIC マイコン　PIC12F675-I/P　×1 個	120 円
２．温度センサー（S-8100B）×1 個	200 円
３．LED×4 個	80 円
４．470Ω（1/6W）カーボン皮膜抵抗×4 個	40 円
５．三端子レギュレータ（5VXC6202P502TB）×1 個	50 円
６．セラミックコンデンサ（0.1μF）×2 個	20 円
７．名刺サイズ汎用基板　×1 枚	100 円
８．IC ソケット　×1 個	10 円
９．電池ホルダー　×1 個	50 円
１０．バッテリスナップ　×1 個	20 円
合　　計	690 円

２．２．３　開発言語と環境

PIC は、プログラムを内蔵フラッシュメモリに書き込んで実行します。　そのためには、プログラムを開発する必要があります。

開発言語は、アセンブラ、C、PASCAL、Basic 等が選択できます。このテキストでの学習目的は、プログラミングではないので、言語は簡単な Basic を使うこととします。

『mikroBasic for PIC』という開発環境が MikroElektronika 社から販売されています。この『mikroBasic for PIC』は、一定の制限範囲内（プログラムサイズ 2K ワードまで）であれば、無償のフリーソフトとして、使用することができます。開発事例も豊富にありますので、今回の開発環境は『mikroBasic for PIC』とします。　『ｍｉｃｒｏ』ではなく『ｍｉｋｒｏ』ですので、間違わないようにしてください。

つぎのＵＲＬにて開発環境とマニュアル類のダウンロードができます。

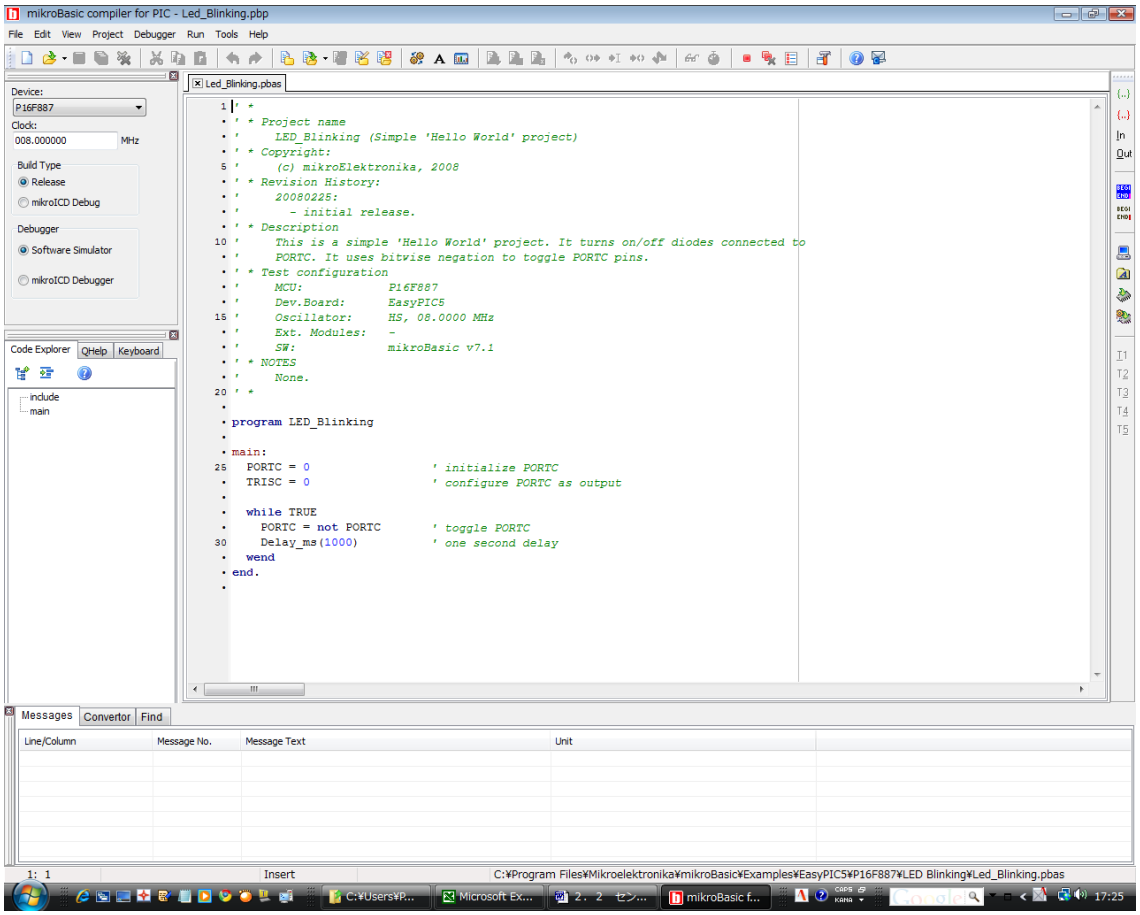
http://www.mikroe.com/eng/products/view/409/mikrobasic-for-pic/

今回使用する PIC は、１Ｋワードのプログラムメモリしかありませんので、フリーソフトとして使用しても、なんら不便はありません。

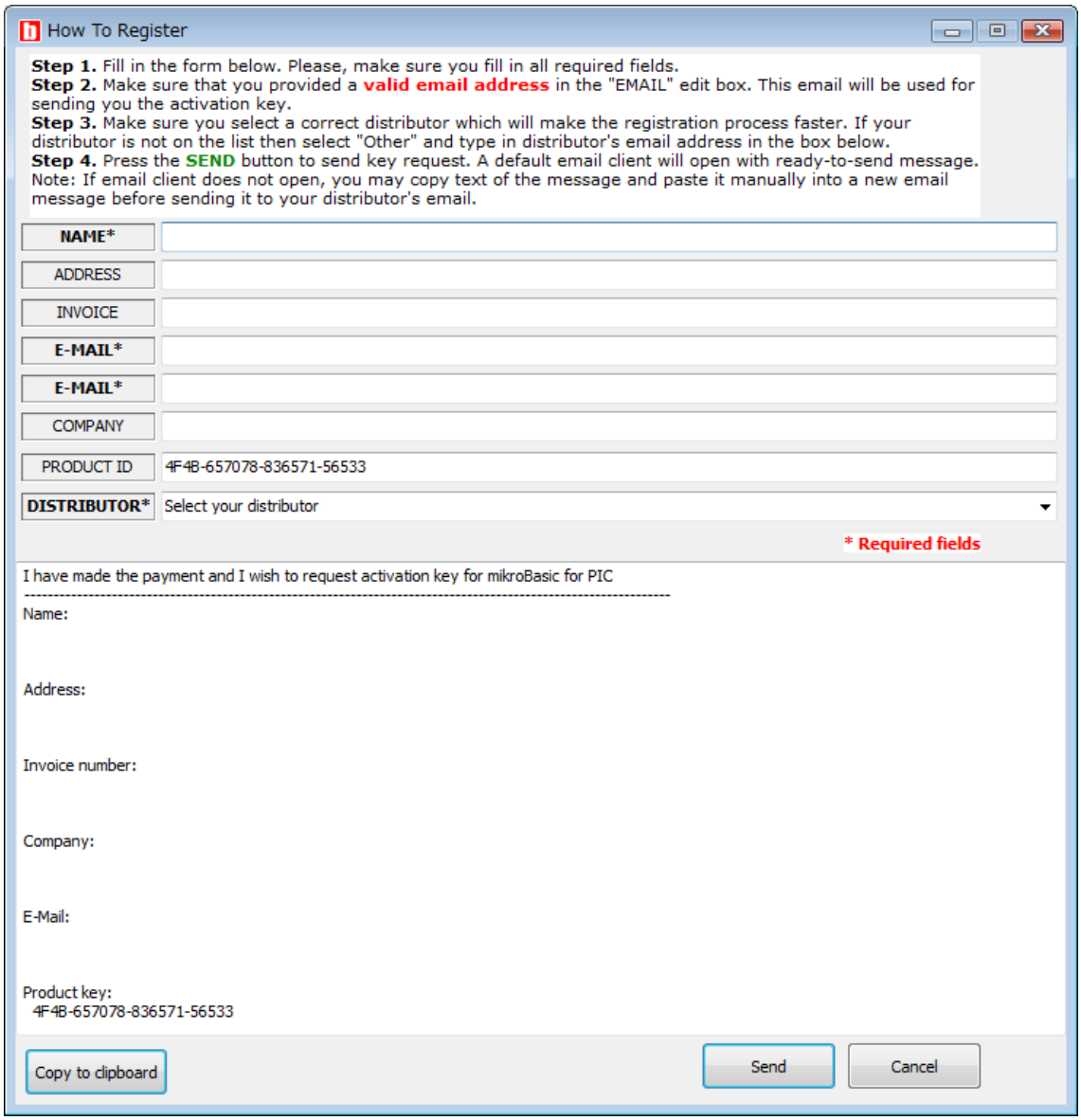
この原稿執筆時点２０１０年１０月では、最新のものが version v7.0.0.2 となっていました。

このファイルをダウンロードして解凍します。できあがったＥＸＥファイルを起動して、簡単な質問にいくつか答えると mikroBasic for PIC がインストールされます。Windows Vista でも問題無く動作します。

旨くインストールができると、次のような画面が開きます。表示されているのは、マイコン開発の『Ｈｅｌｌｏ　ｗｏｒｌｄ！！』とも言うべき、LED 点滅プログラムです。



インストール直後のままでは、ビルド（コンパイル～リンク）ができませんので、次にアクティベーションを行います。HELP の『How To Register』（下記）の指示に従ってメールを送ると、ライセンスキーが送られてきます。



このライセンスキーを入力することで、アクティベーションが完了して、プログラム作成～ビルドまで行う事ができるようになります。操作はすべて英語ですが、難しいことはありませんので、じっくりと取り組んで、環境を整えてください。

（注意）環境のフォルダー名に日本語を使ったフォルダーがあると、コンパイルなどでエラーが出る場合がありますので、適宜パスを変更して使用してください。詳細は、ヘルプとマニュアルが充実していますので、そちらをご覧ください。

また、アクティベーションで、メールを送受信する必要がありますから、メール環境の整ったＰＣに『mikroBasic for PIC』

をインストールする方がよいでしょう。

この環境で、Basic のプログラムを開発する手順については、後ほど説明します。

２．２．４　ライター

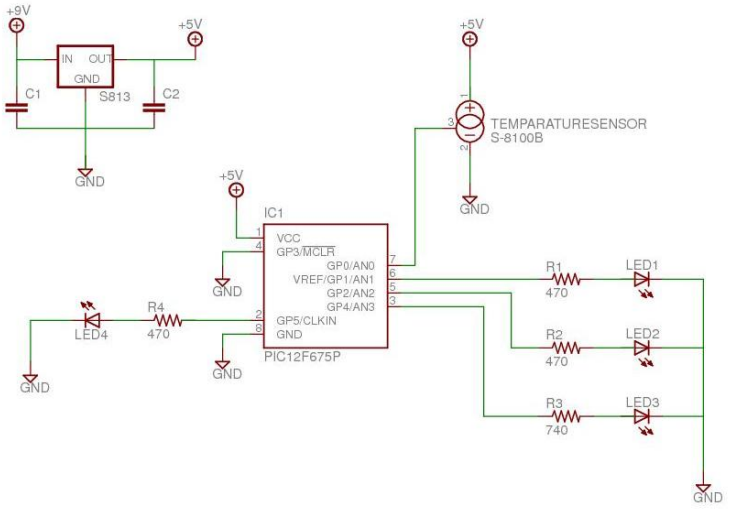
PIC マイコンに限らず、一般にマイコンは、プログラムをビルドすると HEX ファイルができあがり、それをライターや ISP でプログラムメモリや ROM に書き込む必要があります。

PIC マイコン用には、メーカー純正のライターや、多くのサードパーティーから、ライターが販売されています。

私のところには、１０年ほど前から使用しているライターがありますので、これを使います。これは、秋葉原で購入してきたキットを組み立てたものです。PIC マイコンのラインナップが更新されるたびに、書き込みソフトもバージョンアップする必要があるので、純正品を準備しても良いと思います。純正品もキットも 5000 円ほどで入手できます。

２．２．４　回路設計

今回設計した、簡易温度計の回路図がこれです。（プリント基板 CAD ソフト『EAGLE』にて設計）



簡易温度計の回路図

一次電源からレギュレータ（S813）の手前にあるセラミックコンデンサは、入力側のノイズ低減の役割です。ただ、乾電池駆動ですから不要かもしれません、一応安心のために入れてある程度です。５Ｖ出力側にあるコンデンサは、出力電圧を安定化する役割があり、必須です。

温度センサーは、マイコンのアナログポート（AN0）に直結します。また、センサー駆

動電源は 5V ですので、これもレギュレータで作った電源に直結します。

このマイコンは 3.3V でも動作可能なので他のセンサーを選択して、3.3V 駆動とすれば、より省電力が図れます。

PIC マイコンの GP3 は、入力にしか使えないため、GND に接地して信号レベルがふらつかないようにしています。

LED は、抵抗 470Ω（1/6W）を介してマイコンの出力ポートに接続しています。この抵抗で、PIC から大きな電流が流れ出すのを防ぐとともに、LED の明るさを調整しています。LED に流れる電流は、数 mA です。

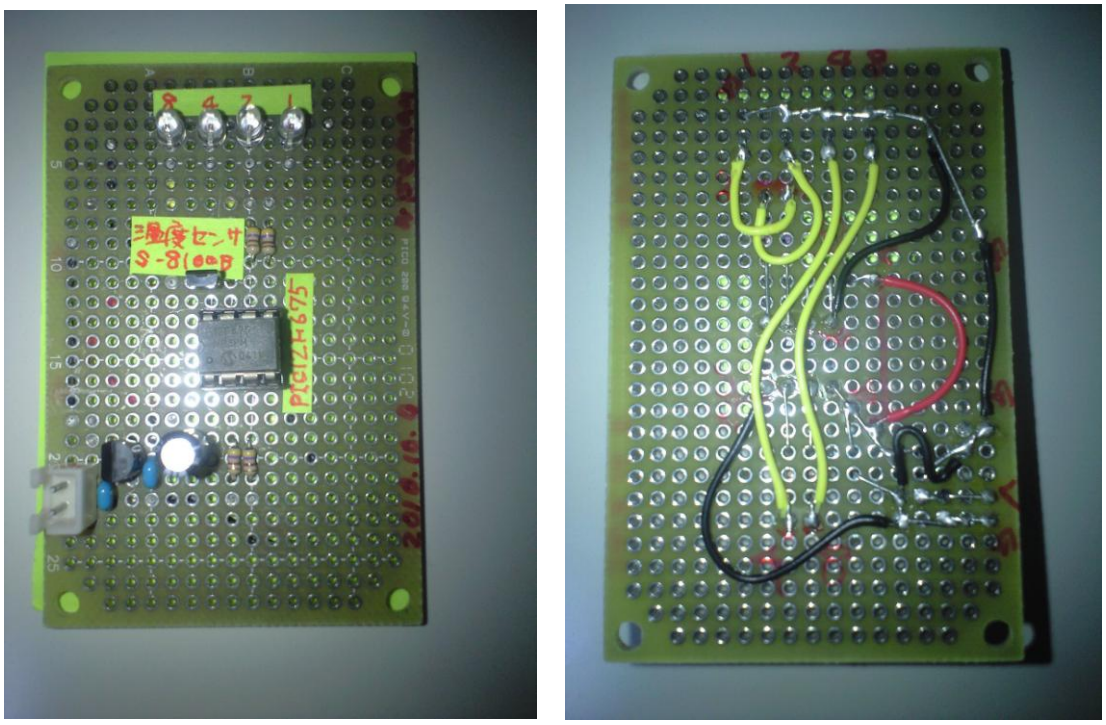
温度センサーの消費電流は、25℃で 10μA ですので、150mA 流せる電源 IC は、もったいなかったかもしれません。同じスペックで 75mA の電源 IC がありますので、こちらにすれば省エネになります。

電源用 IC（S813）は、これで無くてはいけないという IC ではありませんので、スペック・機能が同等品であれば、それで構いません。

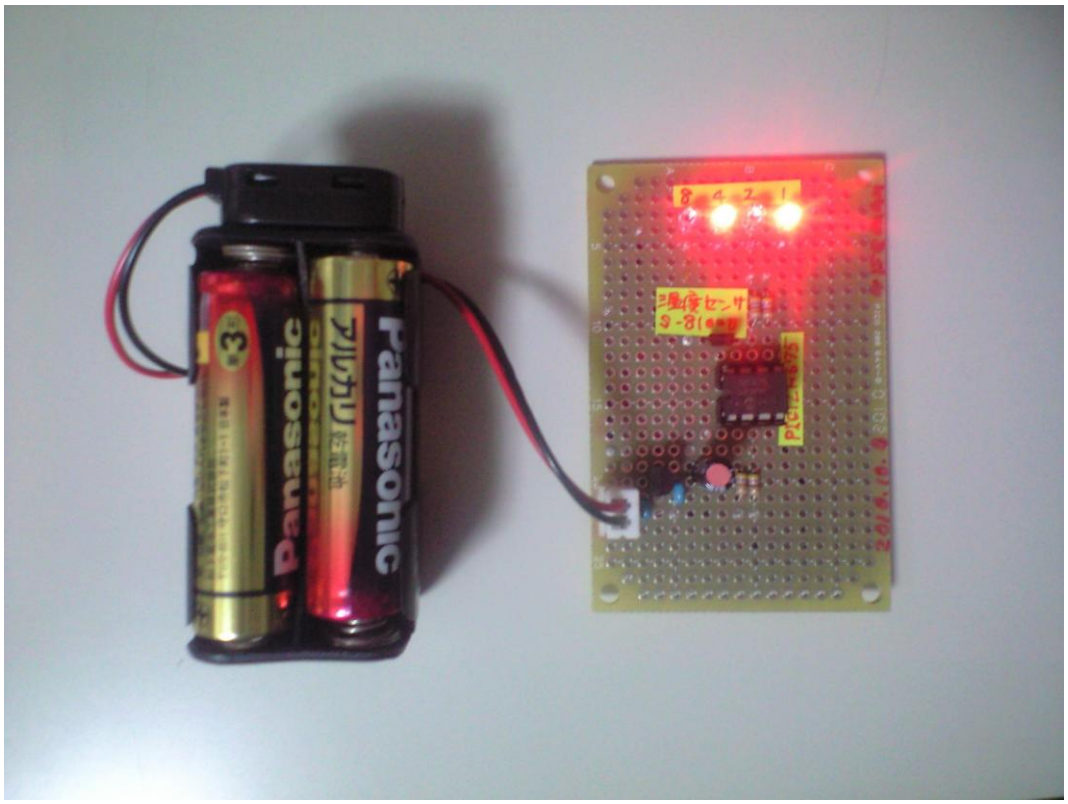
TA78L05 や LM78L05A という品番の物が入手しやすいかもしれません。

また、基板には、電源 IC 付近に電解コンデンサが載っていますが、これも必須ではありません。

できあがった基板の表・裏を見ていただけると分かるとおり、半田付けの箇所も少なく、実にシンプルで初めての方でも、30 分ほどで容易に作成できます。



簡易温度計の基板（表・裏）



簡易温度計動作の様子

２．２．５　プログラム

この簡易温度計のプログラムは、表示に少し工夫が必要です。
というのも、わざわざ IO ピン数の少ない PIC を選択したからです。
4 bit で温度をわかり安く表現しなければいけないので、次のように表現を行います。

①．温度計測のループに入ったら、ループの先頭で、４つの LED を２回点滅します。

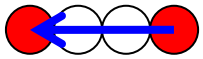
１回目　　２回目

②．Ａ／Ｄ変換を行い、温度計測を終えたら、温度を１０の位と１の位に分けます。

③．ＬＥＤの下位 bit から上位 bit に向けて、フラッシュします。

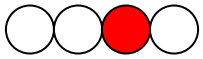
※上位桁の目印です。

右（下位）から左（上位）へフラッシュ



④．１０の位の温度を BCD で表示します。

上位温度 BCD 表示（例２０）



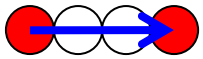
⑤．一度、全消灯します。

一度、全消灯

⑥．LED の上位 bit から下位 bit に向けて、フラッシュします。

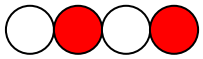
※下位桁の目印です。

左から右へフラッシュ



⑦． 1 の位の温度を BCD で表示します。

下位温度 BCD 表示（例５）



⑧．一度、全消灯します。

一度、全消灯

⑨．ループを①から繰り返します。

今回作成したプログラムは、電源投入後、PIC のレジスタを設定して、A/D 変換を含むループを行うようにしています。また、温度センサーの特性、(―20℃：1.908V、＋30℃：1.508V、＋80℃：1.095V）とマイコンの A/D 変換が 10bit なので、0V のとき、A/D 変換値は 0。5V のとき、A/D 変換値は 1023 となります。

―20℃の時の電圧 1.908 の A/D 変換値は 391 になるので、温度センサーからの A/D 変換値を temp とすると、次の式で電圧に対応する温度が求められます。

温度=(391－temp）＊ 100/167.－20.

この変換計算を含めた、全プログラムが次のリストです。

```
program Thermometer20101012
'4bit LED displey Thermometer
```

```
 dim temp as word
 dim temp_H as byte
 dim temp_L as byte

main:
 OSCCAL=OSCCAL      'PIC マイコンの内部発信を選択したので、
 Asm            '発信器のキャリブレーションを行うためのコードです。
   bsf           status,rp0
   call          0x3ff
   movwf          OSCCAL
 end asm

 gpio=0                              'GPIO をリセットします。
 cmcon =%00000111                    'コンパレータを停止
 adcon0=%10000001                    'ＡＤコンバータ電源ＯＮ
                                     ‘Ａ／Ｄ変換結果右詰め
 ansel =%00010001                    'Ａ／Ｄ変換クロック設定・AN0 ピンアナログ入力
 trisio=%11001001            'GP0、GP3 を入力に、他は出力に設定
 gpio=0                             'GPIO をリセット
 Delay_ms(100)                '１００ｍｓ待ちます。
 while 1          'for ever loop      '永久ループです。
   gpio=$36                           'LED 全点灯
   Delay_ms(500)                     '０．５秒待ちます
   gpio=$00                           'LED 全消灯
   Delay_ms(500)                     '０．５秒待ちます
   gpio=$36                           '以下、繰り返し。
   Delay_ms(500)
   gpio=$00
   Delay_ms(500)
   'Flash L->H       'LED の Low から High に向けてフラッシュします。
   gpio=$02
   Delay_ms(200)
   gpio=$04
   Delay_ms(200)
   gpio=$10
   Delay_ms(200)
```

```
    gpio=$20
    Delay_ms(1000)
    gpio=$00                    '一度、全消灯します。
    Delay_ms(1000)
    temp=Adc_Read(0)            'ＡＤ変換した値を読み込みます。
    temp=temp and $03FF         '１０bit 残して、フィルタリング
    temp=(391-temp)*100/167.-20.    'ＡＤ値に対応する温度を計算
    temp_L=temp mod 10          '温度１の位を求める
    temp_L=((temp_L<<2)and $30) or ((temp_L<<1) and $06)
    temp_H=temp/10              '温度１０の位を求める
    temp_H=((temp_H<<2)and $30) or ((temp_H<<1) and $06)
    gpio=temp_H                 '１０の位、LED に BCD 表示
    Delay_ms(1000)
    gpio=$00                    '１秒間、全消灯します。
    Delay_ms(1000)
    'Flash H->L                'High から Low にフラッシュします。
    gpio=$20
    Delay_ms(200)
    gpio=$10
    Delay_ms(200)
    gpio=$04
    Delay_ms(200)
    gpio=$02
    Delay_ms(1000)
    gpio=$00
    Delay_ms(1000)
    gpio=temp_L                 '温度、１の位を BCD 表示します。
    Delay_ms(1000)
    gpio=$00                    'LED 全消灯
    Delay_ms(1000)
  wend
end.
```

２．２．６　『mikroBasic for PIC』によるプログラム開発

プログラムを作る手順は、次のとおりです。

①．プロジェクトを作る

②．PIC マイコンを選択する

③．PIC マイコンのコンフィギュレーション bit を設定する

④．プログラムを入力する

⑤．ビルドする

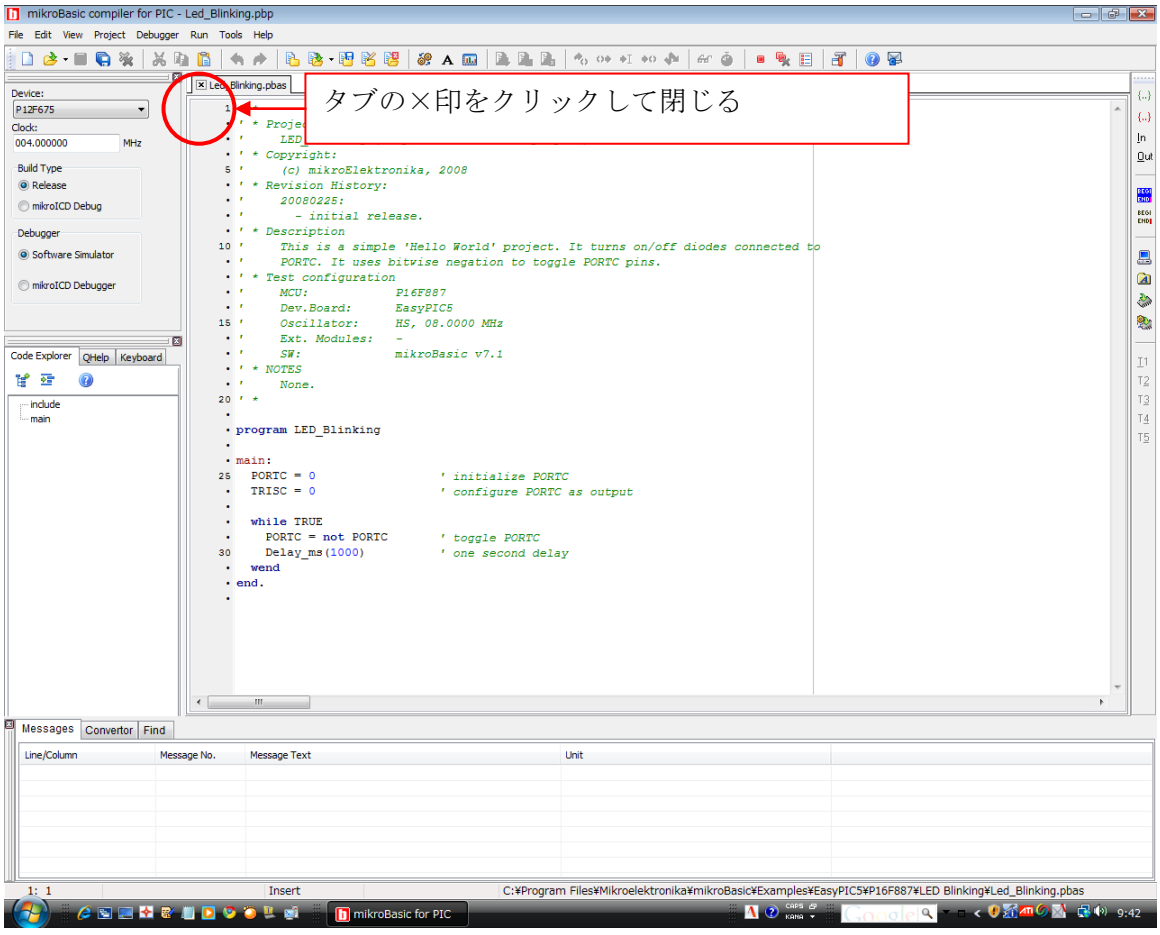
※ 動作テストをして不具合があれば、④⑤を繰り返してプログラムを完成させます。今回の簡易温度計は、プログラムのサイズも大きくないので、このような手順で開発・テストを繰り返すことができますが、規模が大きくなるとテストも大変ですので、Debugger を使用して開発します。

Debugger の使用法に関しては、マニュアルを参照してください。

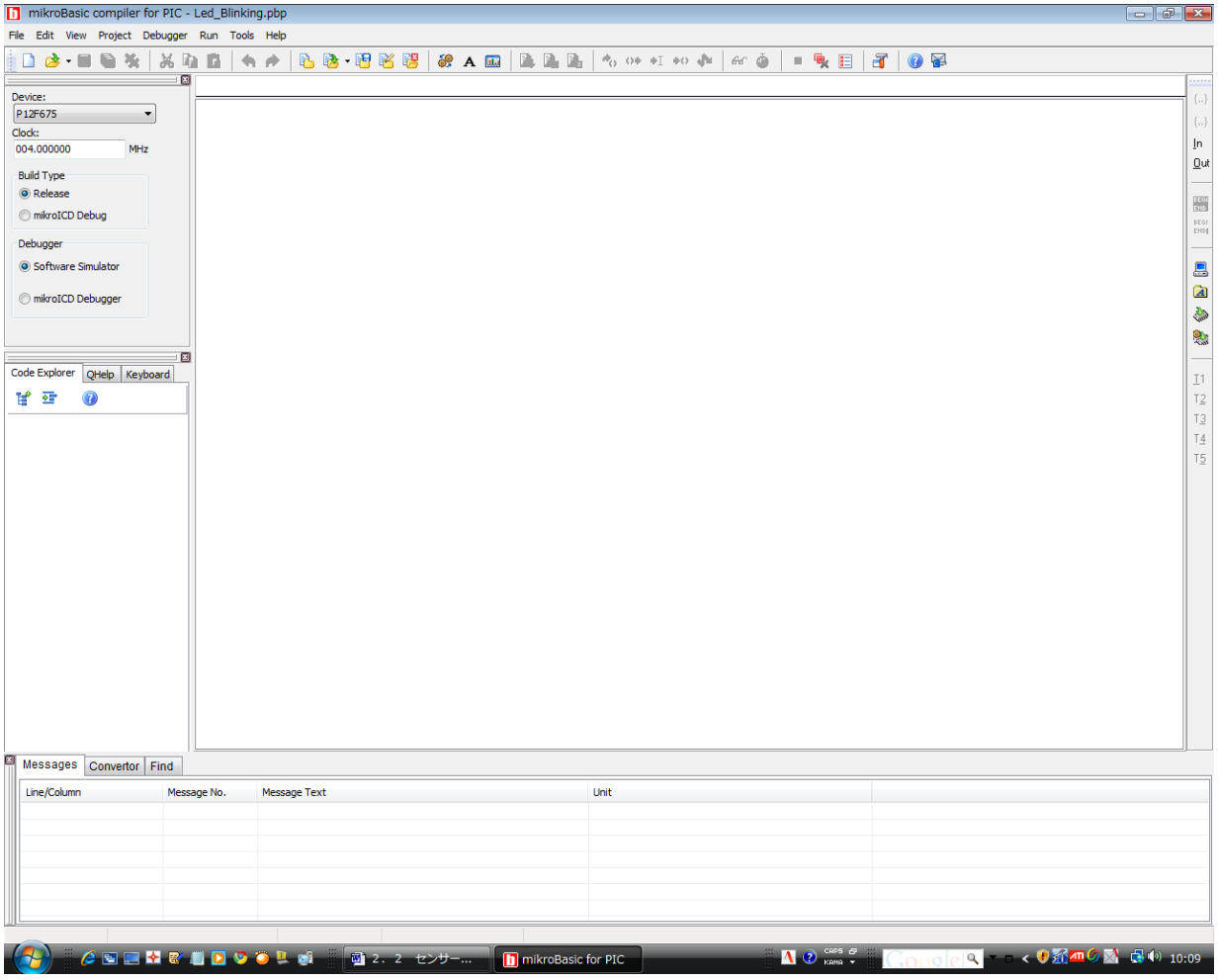
以下、上記手順に従って、操作した画面を示します。

まず、　mikroBasic for PIC　を起動すると、次の画面が現れます。

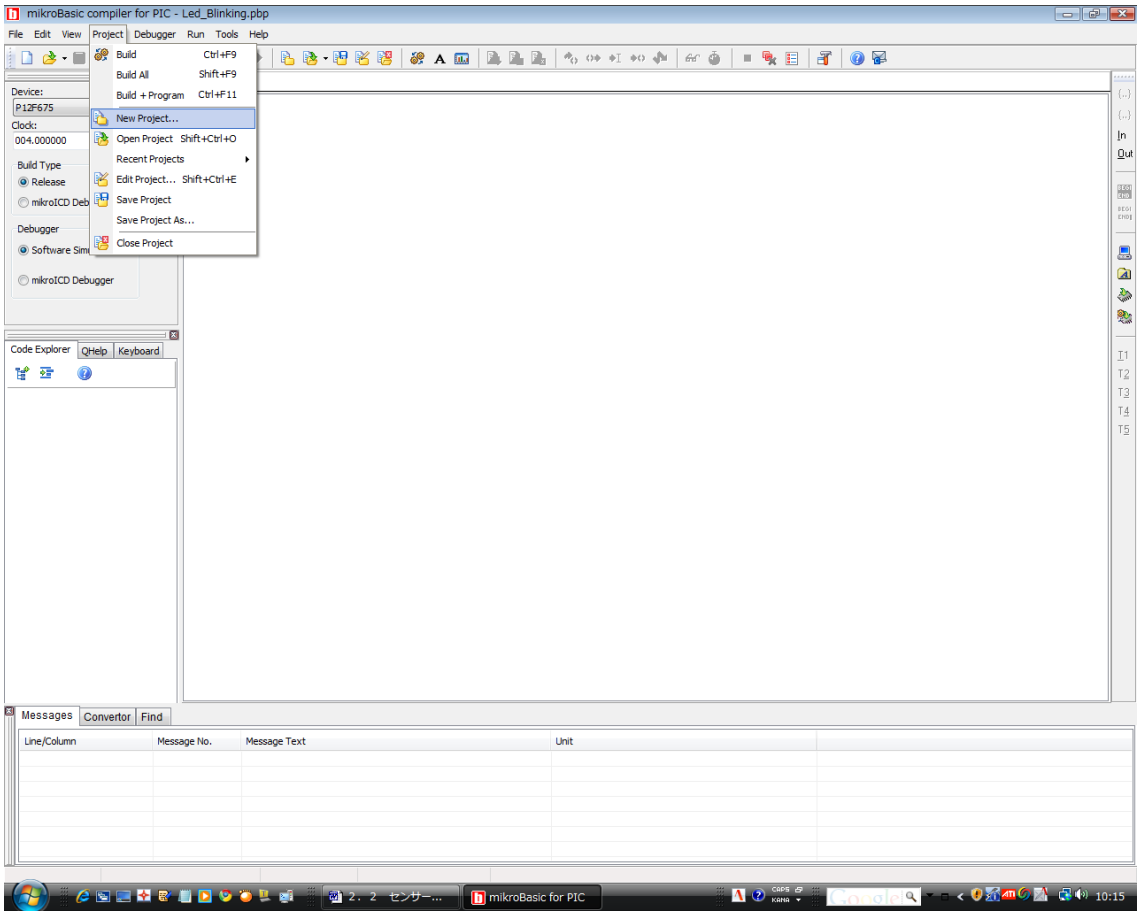
このときに表示されているのは、『LED_Blinking』というサンプルプロジェクトのソースファイルです。



このファイルは、必要ありませんので、ソースファイルの１行目のさらに上にある、タブの×印をクリックして、このファイルを閉じます。

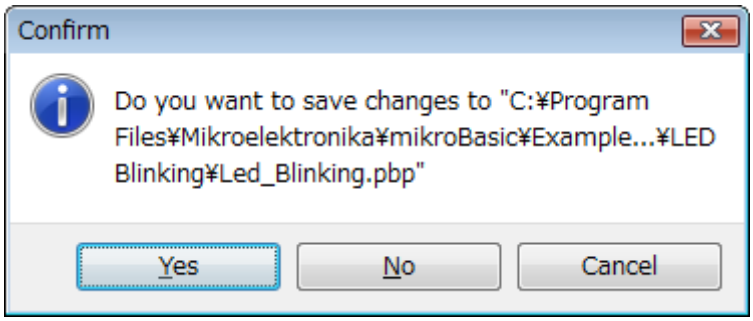


①．プロジェクトを作る



メニューから『Project』→『New Project』と選択します。

次のようなメッセージが表示されるかもしれませんが、先に閉じたサンプルプロジェクトを変更する必要はありません。



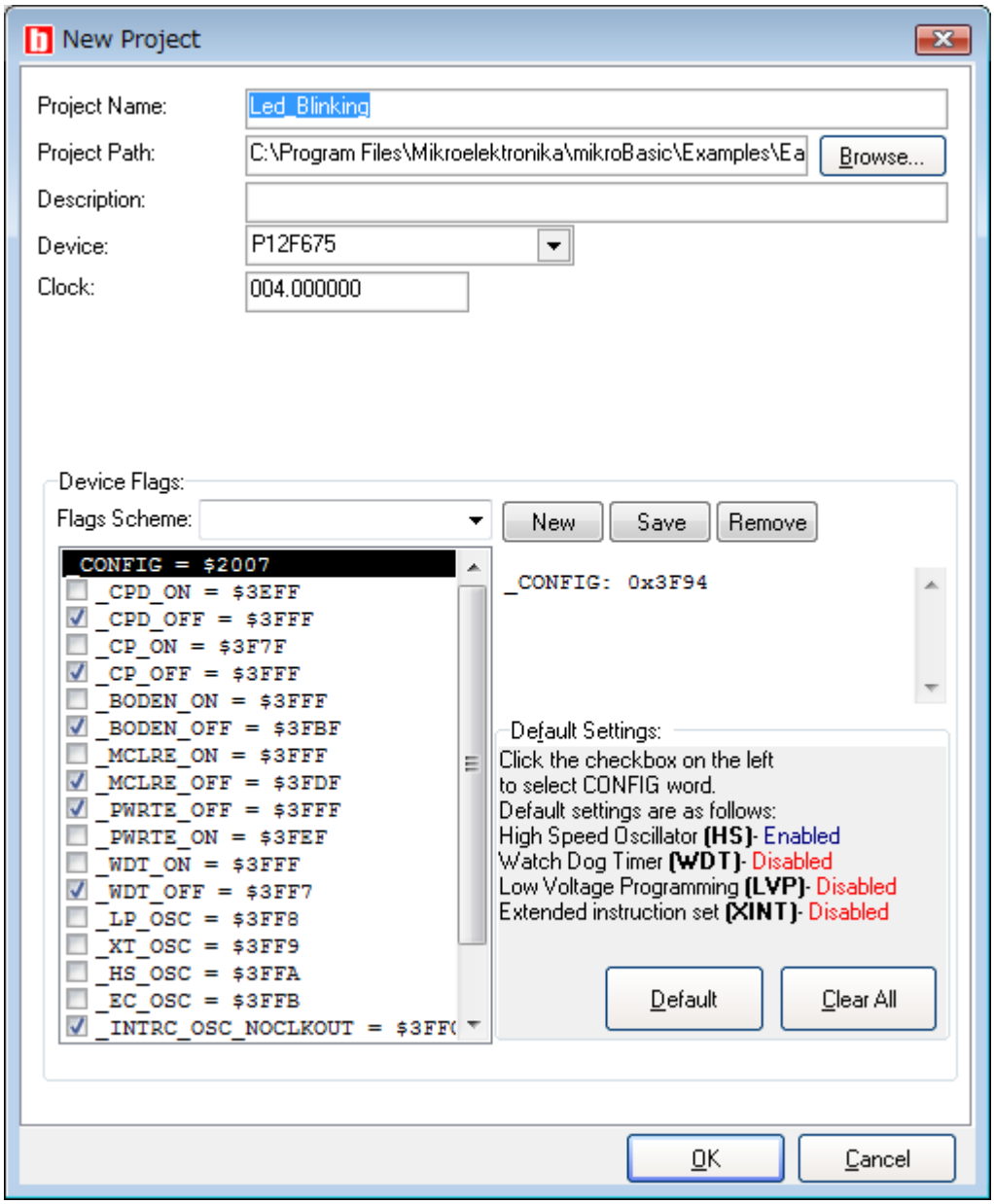
『No』を選択して次に進みます。

『NewProject』ウインドウが現れます。

ここで、次の設定を行います。

②．PIC マイコンを選択する

③．PIC マイコンのコンフィギュレーション bit を設定する



Project Name、Project Path、Device、Clock を入力または、選択します。Description は、入力しなくてもよいです。

今回は、内部発信を利用して４MH z 動作させることを考えていますので、Clock は 4.000000 を入力します。

ウインドウの下部には、PIC のコンフィギュレーション bit が並んでいますので次の項目をチェックします。

◆_CPD_OFF：データ EEPROM をプロテクトしない

◆_CP_OFF：プログラムメモリをプロテクトしない

◆_BOODEN_OFF：ブラウンアウト・リセットを使わない

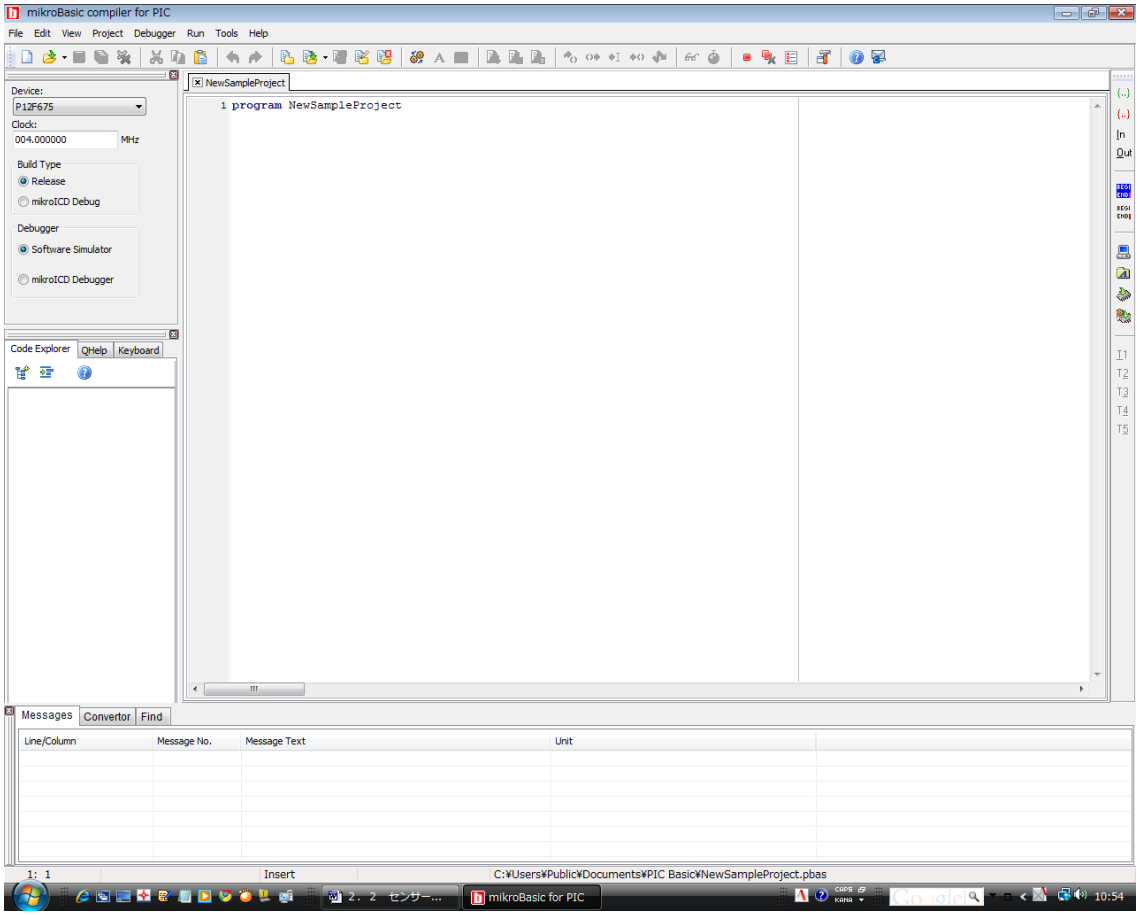
◆_MCLR_OFF：GP3/MCLR ピンを GP3 として使用する

◆_PWRTE_OFF：パワーアップ・タイマーを使わない

◆_WDT_OFF：ウォッチドッグ・タイマーを使わない

◆_INTREC_OSC_NOCLKOUT：内部発信でクロック外部出力をしない

上記設定を行ったあと、OK ボタンをクリックします。



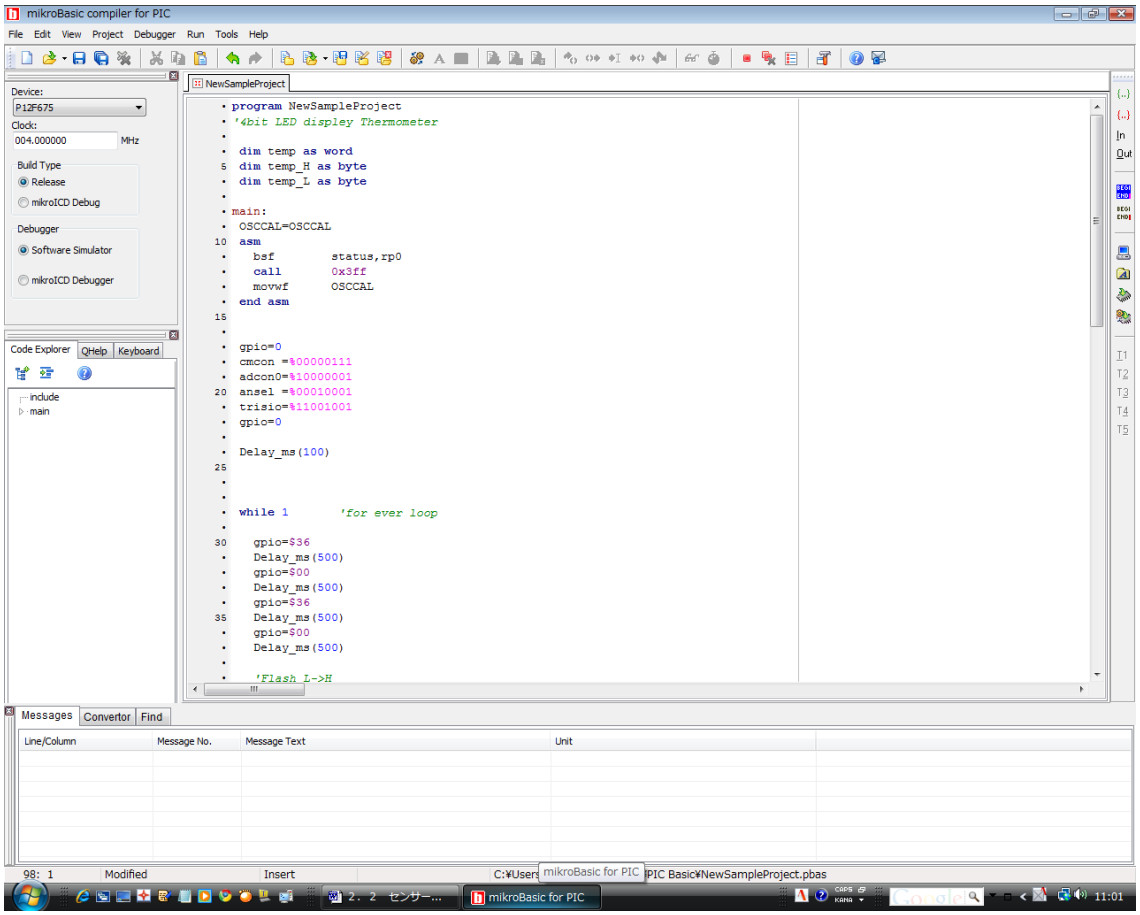
上記は、『NewSampleProject』というプロジェクトを作った例です。

ウインドウ左上に Device と Clock が設定どおり表示されているか確認してください。

④．プログラムを入力する

中央に表示された『program NewSampleProject』の下に２．２．６で説明したプログラムを入力します。

こんな感じです。

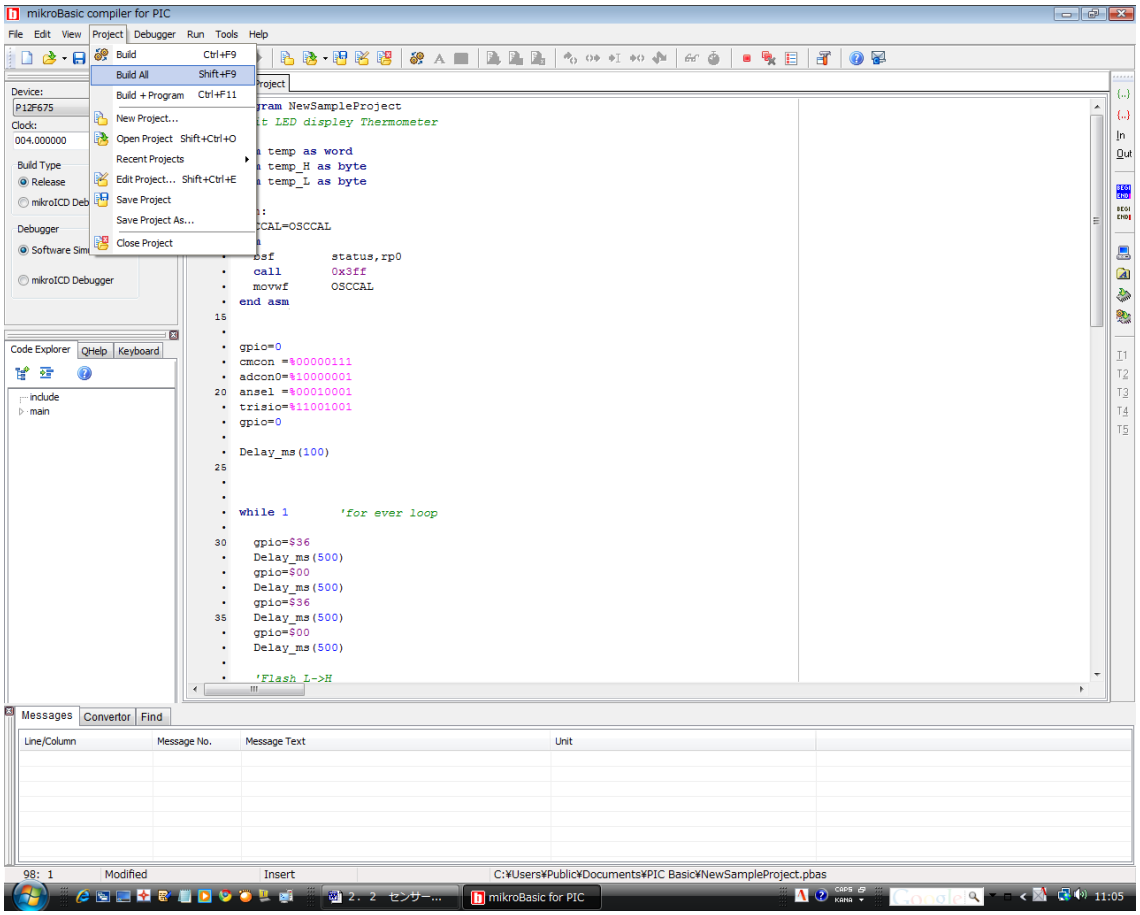


入力したプログラムに誤りが無いか、良く確認します。

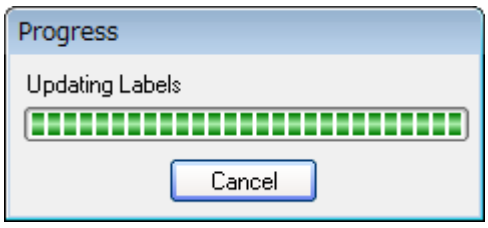
次は、いよいよビルドです。

⑤．ビルドする

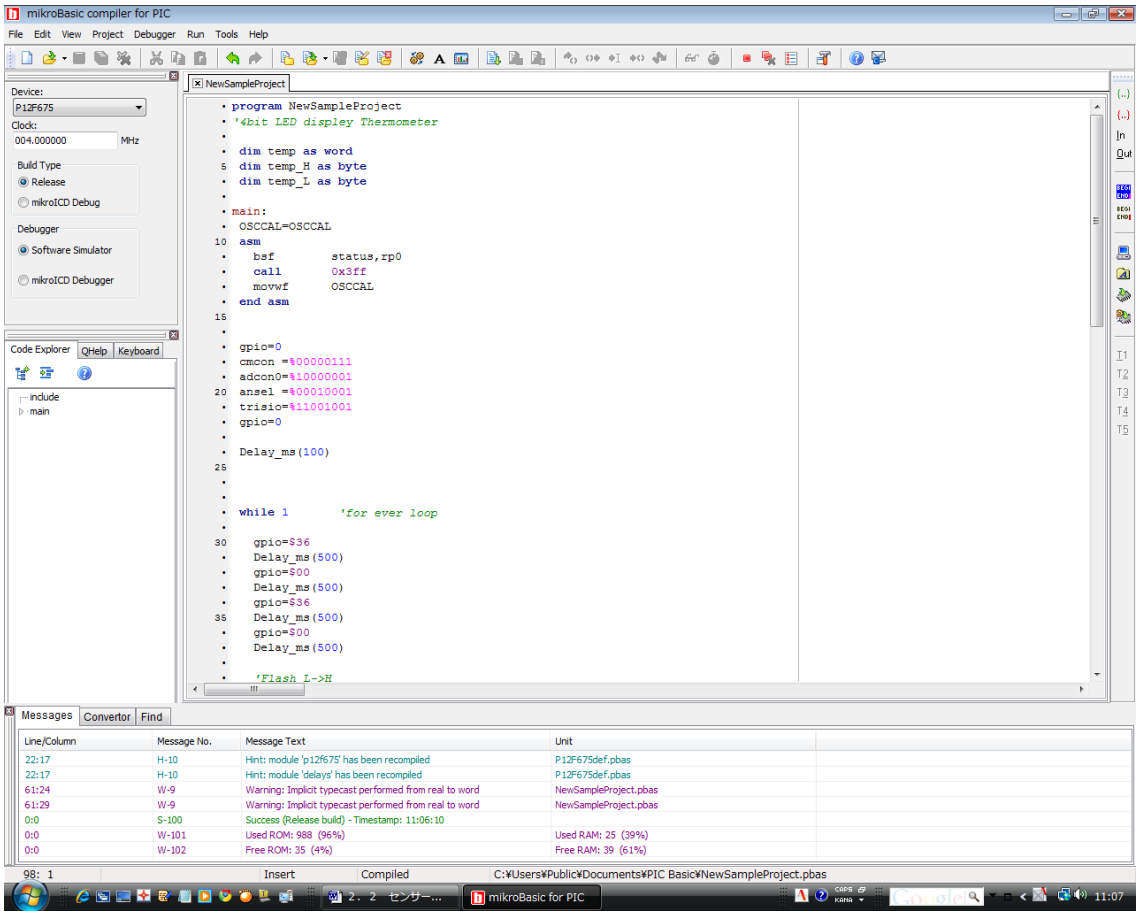
メニューの『Project』→『Build All』を選択します。



すると、このような画面が現れて、緑色のインジケータが表示されます。

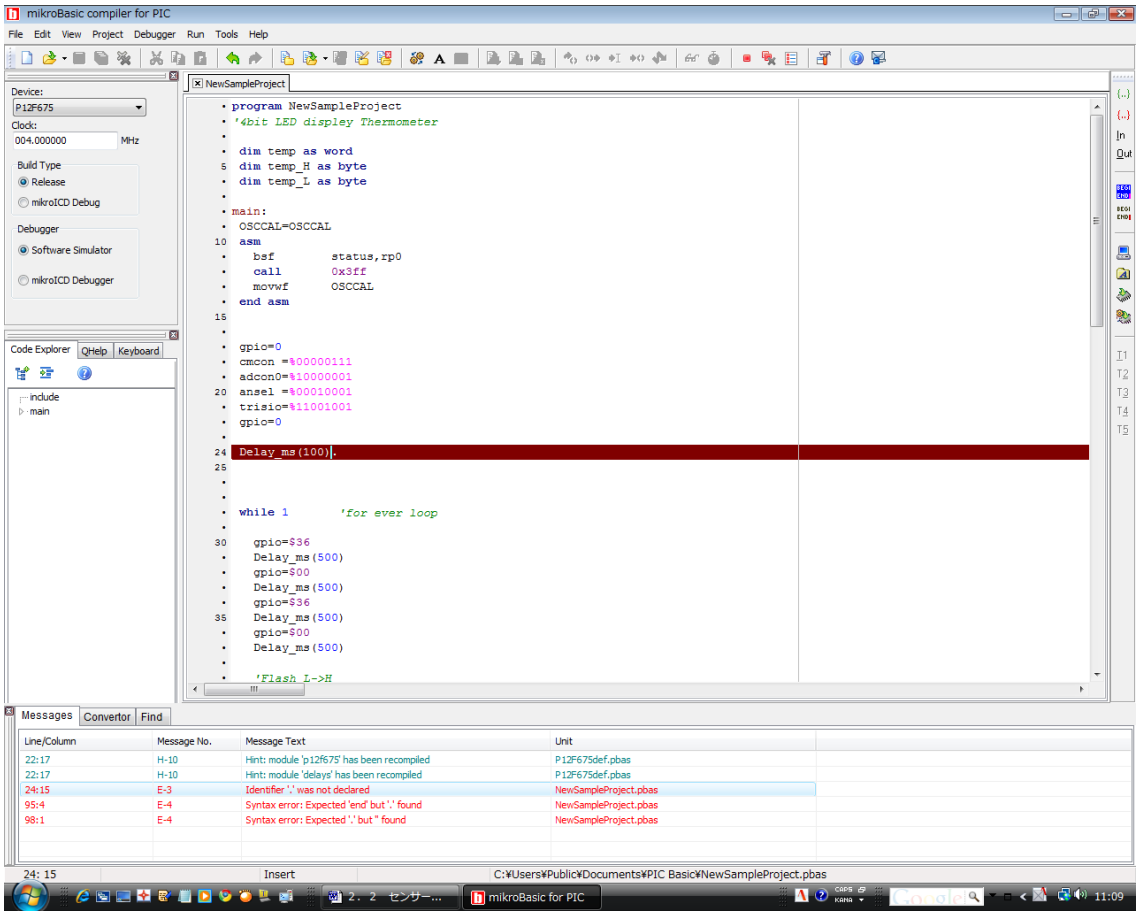


これと同期して、画面下部にメッセージが表示されます。



緑色の『Success』のメッセージがあればビルド終了です。
このメッセージの中に、『Used ROM』という項目があって、ROM 領域を何%消費したかがわかります。このプログラムは 96%となっていて、ぎりぎり 1KWord のメモリに収まるサイズでビルドできました。大変効率の良い、マイコンとプログラムの組み合わせになりました。

もしエラーがあった場合は、次のように赤い色で表示されエラー箇所を通知してくれます。

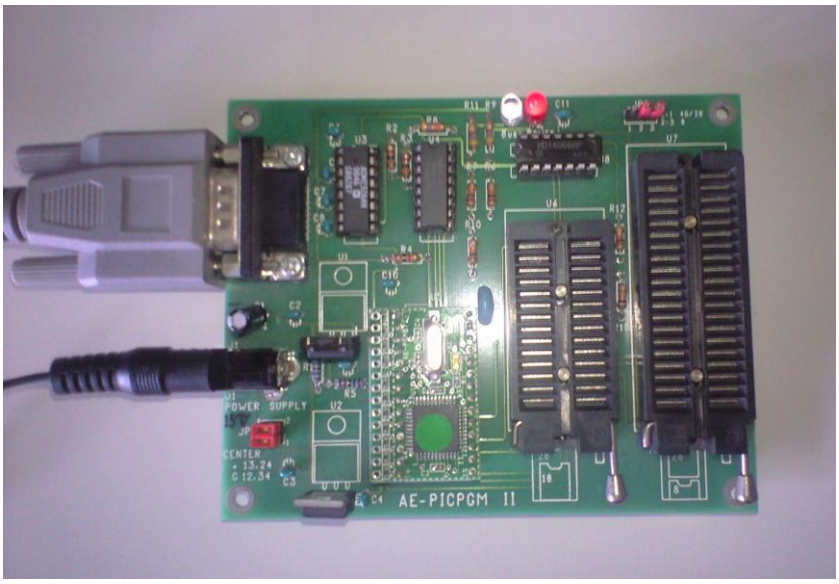


表示された箇所を訂正して、もう一度ビルドすれば、『Success』のメッセージが現れます。

２．２．７　プログラムの書き込み

ビルドしたプロジェクトフォルダには、『Project 名.hex』というファイルができています。このファイルを、ライターで PIC のプログラムメモリ（フラッシュ）に書き込みます。

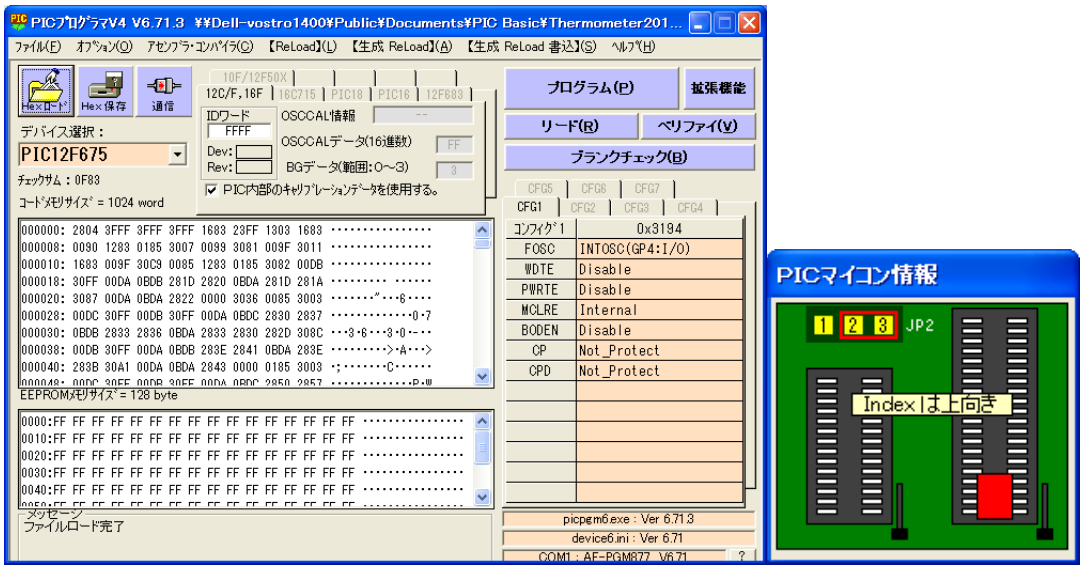
書き込みに使用したのは、秋月電子の『AKI-PIC プログラマー』です。



AKI-PIC プログラマー

このライター基板と PC をシリアルケーブルで接続して、基板上のゼロプレッシャーソケットに、PIC セットします。

書き込みソフトを起動すると、次のようなウインドウが現れます。また、ソケットが２つあり、どちらを使うべきかもサブウインドウに表示されるようになっています。ウインドウ左上の『Hex ロード』ボタンでビルドした Hex ファイルをロードし、右上の『プログラム』ボタンをクリックすると、書き込みを行ってくれます。



AKI-PIC プログラマー起動画面

私は、１０年以上このライターで、マイコンの書き込みを行っていますが、不自由は全くありません。

書き込みが終わった PIC は、作成した基板上のソケットに差し込みます。電池をつないで、思惑通り LED が光ってくれれば大成功！！

いかがでしょうか。最低限度のスペックで温度計をつくりましたが、LED を光らせる代わりに、IO ピンを 1 本使って、調歩同期のシリアル通信で、PC に温度データを送信するようにすることもできます。
また、IO ピンの出力をある閾値基準で ON・OFF して、出力をソリッドステートリレーに接続すれば、AC 電源で 20～40A 級の設備の制御も可能です。

マイコンの選択をもっとグレードの高いものにすれば、7SEG LED を利用したり、LCD 表示器を使用して、数字で温度を表示することもできますし、SD カードなどと接続して、単体で温度を記録できるデータロガーもつくれます。このような場合は、ここで紹介した『mikroBasic for PIC』に含まれる、豊富なライブラリ関数が使えますので、プログラム開発の負担も相当軽減されています。

小さな簡易温度計ですが、応用範囲は大変広く、農業 IT 技術者としては、習得しておくべきものだと思います。

３．視覚化技術

最近、『見える化』という言葉を耳にします。

WEB で検索すると、**『・・・現場の問題等の早期発見や効率化や改善に役立てることを目的とする。業種等により適用方法は異なるが、一般的には問題や課題の認識に利用される。図やグラフにして可視化する場合もあれば、音や光による体感認識を用いる場合もある。・・・IT インフラの整備により、電子データ化された各種業務内容を有効活用するために、蓄積されたデータの抽出・加工を行うことにより、見える化を行う場合がある。・・・（引用：Wikipedia）』**と出てきました。他にもたくさんヒットしました。

『農業 IT としての見える化』は、何でしょうか？　現場で仕事をされている農業家の方々は、何に『見える化』が必要でしょうか？

少なくとも、センサーで検出した結果は『見える化』しやすそうです。
この章では、センサーの検出結果を見やすい表現に変えて実現する方法をあれこれ考えてみます。

３．１　表現の種類

センサーにて捉えた値を私たちが利用しやすい形にするには・・・要するに『見える化』するための変換先は、数表やグラフの基になる数値が一番でしょう。数値化すれば、基準値や過去の値と比較して大きい・小さい等の判断ができます。大きければ『大』、小さければ『小』と表現できます。数値化したものを、さらに『数字化（または文字化）』してあげれば、私たちはわかり安くなります。纏めると次のようになります。

３．１．１　数値表現

『数字化・文字化』といいかえても良いと思います。検出した値を、１０進数での表現に変換して表すことです。一般的に、センサーの検出値が１０進数で直読できることは希なので、必ずこの『数値表現』を経て通知・表示または記録されます。

３．１．２　閾値表現

ある基準値と比較して、『大きい』のか『小さい』のか、または『等しい』のかを表すものです。僅か 1bit で表現できるので、LED を１個取り付けることで表現可能です。これができれば一定の温度範囲を保つような制御の仕組みに使えます。電気こたつやホットカーペットは、温度によってスイッチが ON/OFF するような仕組みが組み込まれています。これがハウスなどで用いられる、温度センサーを利用した換気ファン制御や、ヒーター制御につながります。この ON/OFF 制御は、後の章で実際に開発した例を説明します。

３．１．３　推移表現

『推移』と書きましたが、『経過』としても良いかもしれません。『経過』が分かるような表現をするためには、過去の記録が必要です。そのためには、検出した値を数値化して、システムのどこかに記憶させておく必要があります。また、電源が落ちてしまう、いわゆる停電の状態になっても、それ以前の記録が消えてしまわない場所に記録する技術が必要です。それには、マイコンの EEPROM 領域や外部メモリ（外付けの EEPROM や SD カードなど）に記録することで、対処できます。
そして必要に応じて記録を取りだし、数表にしたり後に述べるグラフィカル表現（グラフ化）をしたりすることで、推移（経過）が分かります。

３．１．４　相対値表現

これは、検出した値が、『ある基準点からどれほどのところにあるか』を表す事です。たとえば、２章で作成した簡易温度計は、更正機能がありませんので、実際に棒温度計などと比べてみると、全体に温度が高めに現れたり、低めに現れたりします。棒温度計の温度を基準とすると、そこからの温度の隔たりが、相対値表現になります。

この相対値を簡易温度計の PIC マイコンの EEPROM 領域に書き込んでおいて、その温度を検出した温度に加算すること（更正）で、より正確な温度を表示できるようになります。

３．１．５　絶対値表現

『相対値』に対する『絶対値』表現で、得られた数値そのものが、その状況をそのまま表す、つまり、３０℃と検出したら、それがそのときの温度であると取り扱う事です。普通は、センサー検出値を更正するか、センサー回路での電気的な更正をおこなうので、本当にセンサーの検出値そのものずばりを使うことは希かもしれません。
ちなみに、WEB では・・・

『数学において、絶対値は数の「大きさ」の概念を与える規準の一つである。その数が 0 からどれだけ離れているかを知ることができる。（引用：Wikipedia）』 WEB では、数学の絶対値がヒットしますが、ここでいう絶対値表現は、少し意味合いが違いますね。

３．１．６　グラフィカル表現

『推移表現』で説明したように、一般的には『グラフ化』だと考えていただければよいと思います。グラフにするためには、もとの数値が必要ですから、『数値表現』を経ています。

グラフにするために必要なデータが不揮発性メモリへ記録されるような機能をシステム

が持っていることが必須条件になります。当然ですが、記録されたデータを後ほど、読み出す機能も必須です。

表示環境は、簡易温度計の LED の列配置などが最低限のものになると思います。

もし可能であれば、簡易温度計の改良をして、温度データをシリアル通信でパソコンに送り、パソコン側で記録して、後ほど全データを読み出して表計算ソフトなどでグラフ化する方法は、実現しやすい方法です。

『グラフィカル』な表現には、他にも方法があります。それは、ドットマトリクス LED のような、ある画素を持つ表示装置を使い、『閾値表現』で説明したような『大』『小』、『高』『低』をそのものずばり、文字で表すのも『グラフィカル』の例です。
そのときの景気状況をニコニコマークのような『顔』で表現するのも、良い例です。

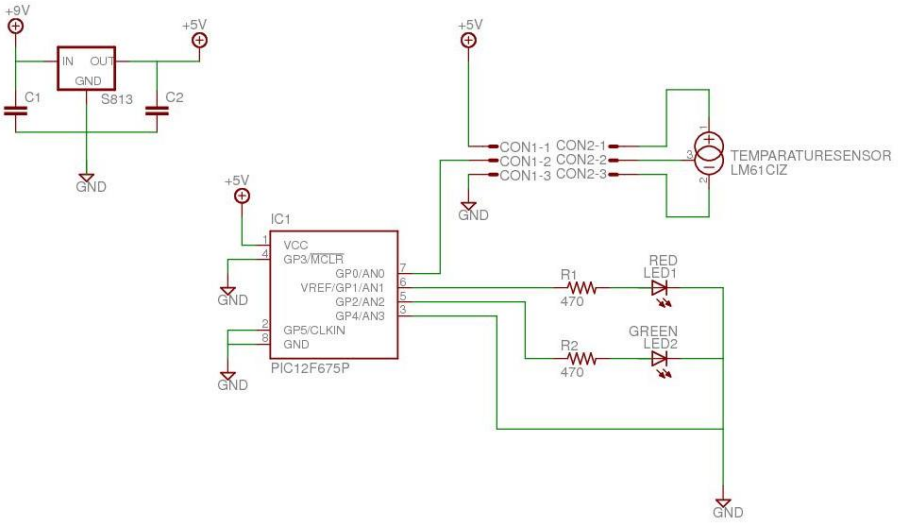
３．２　いろいろな温度計の製作

３．２．１　LED による閾値表現

温度センサーを使って、計測した温度が一定値よりも高い場合は、LED を点灯し、低い場合は消灯するという、ユニットを作ってみましょう。
２．２で作成した簡易温度計を応用すれば、容易に作成することができます。温度センサーに LM61CIZ を使って、システムを作ってみましょう。

温度センサーが変わっても、回路を変更する必要はありません。
計測範囲：－３０℃～１００℃
温度係数：＋１０mV／℃
出力電圧は、次のようになっています。
　―３０℃：　３００mV
＋１００℃：１６００mV
この間が温度係数に比例してリニアに変化するので、PIC の A/D 変換器の入力に直結することで、簡易温度計の回路を何ら変更せずに、澄みます。もし、もっと精度を求めたい場合は、温度センサー出力電圧を OP アンプ回路で N 倍してやることで、精度が上がります。この回路も後で説明します。

回路図は、次の通りです。



簡易温度計のセンサーを取り替えるだけで澄みますから、温度センサーをソケット式で取り付けるように工夫すれば、いろいろなセンサーを試すことができますし、センサーをユニットから離して土中に埋めたりすることも可能になります。

LED は２つ取り付けて、指定温度（プログラム埋め込み、固定）より低い場合・高い場合の２つのパターンで点灯するようにプログラムを作成します。余った I/O ピンは入力に設定して GND に接続し、常に LOW(=0)入力に安定させておきます。

A/D 変換値の温度への換算式は、次の通りです。

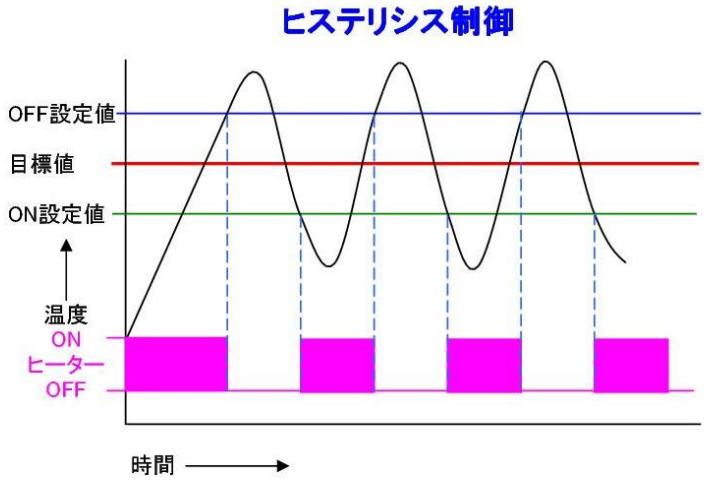
温度＝（(A/D 変換値× 1 bit の分解能－300mV）÷10mV）－30℃

ここで、もう一つ工夫しなければいけないことがあります。

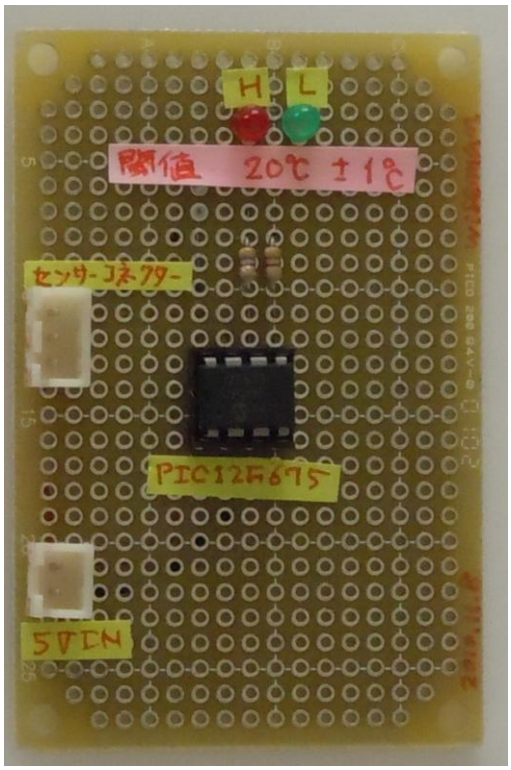
それは、温度変化がゆっくりで指定温度付近をふらつくと、２つの LED がパラパラと点灯・消灯を繰り返してしまうことがあります。

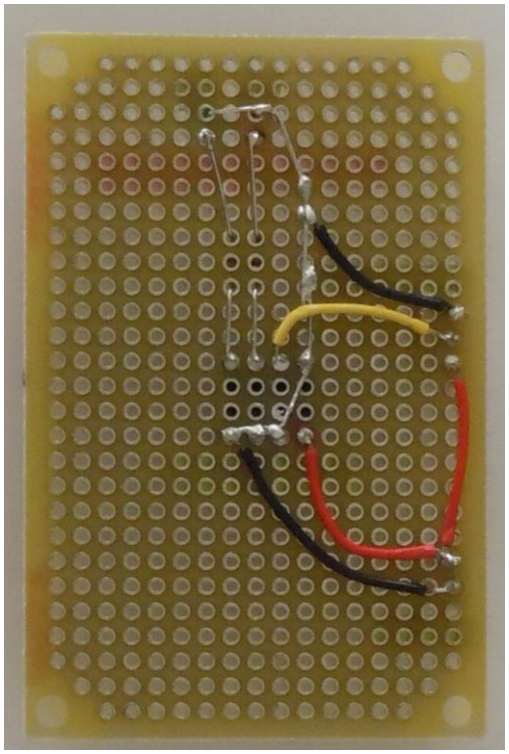
もし、このシステムを応用して外部機器の ON/OFF を行うような場合は、

この現象で頻繁に機器の電源が ON/OFF を繰り返して、最悪の場合機器の故障を招くことがあります。このような場合には『ヒステリシス制御』を行うことで対応します。これは、指定温度の上下にいくらかの余裕を持たせ、たとえば、20℃を指定温度とすれば、18℃と 22℃を閾値の範囲として、温度が 22℃を超えたとき OFF とし、温度が 18℃未満になったとき ON するように制御をおこないます。すると、ON/OFF の出力が安定して得られます。今回作成したプログラムでは、目標値に対して±1℃で ON/OFF 制御するようにプログラムを組みました。

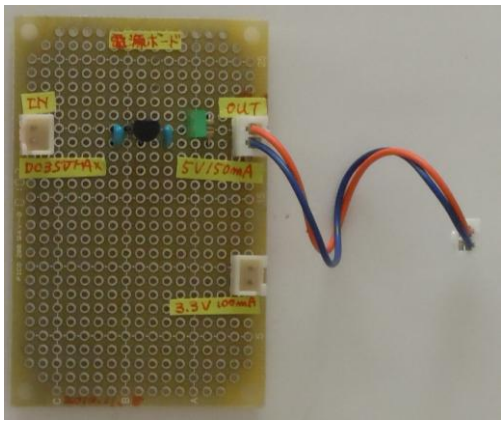


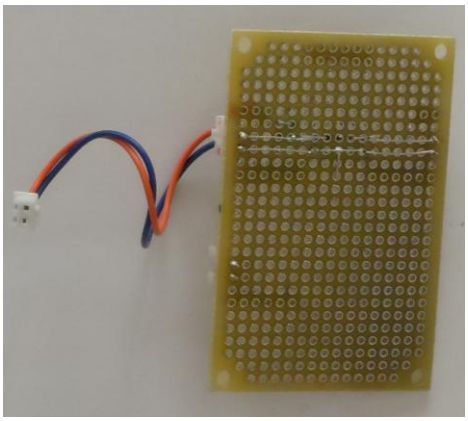
作成した基板（写真）





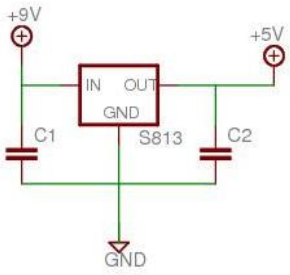
これから、沢山の基板を作るので、電源ユニットを共通化して、別基板としましたので、そちらの写真も掲載しておきます。





写真には、3.3V 用のコネクタも付いていますが、実装していません。

電源回路の回路図は、簡単です。



作成したプログラム（shikiichi_ondokei）を下記に示します。
この温度計は、閾値を２０℃に設定してあります。

```
program shikiichi_ondokei
' Declarations section
' 2bit LED displey Thermometer
  const CON_deg = 20              '←閾値を２０℃としてあります。
  const CON_his = 1 '←閾値の上下１℃の範囲がヒステリシス範囲です。
  dim temp as word
  dim temp_C as integer
  dim led as byte

main:
'     Main program
```

```
'   Oscillate Block regulation    ‘マイコン内部の発信器を調整しています。
  OSCCAL=OSCCAL
  asm
     bsf            status,rp0
     call           0x3ff
     movwf          OSCCAL
  end asm
gpio=0
  cmcon =%00000111       'コンパレータは使いません。
  adcon0=%10000001       'for A/D Power ON and Right to LSB
  ansel =%00010001       'アナログ CH0 を使用します。
  trisio=%11111001
  gpio=0                 ‘LED を消灯します。
Delay_ms(100)            '100ms 待ちます。
   ‘ここから、温度計測のループが始まります。
  while 1            'for ever loop
     gpio=$06                    ‘LED を 2 つ点灯します。
     Delay_ms(500)    ‘500ms 待ちます。
     gpio=$00                    ‘LED を消灯します。
     Delay_ms(500)
     gpio=$06
     Delay_ms(500)
     gpio=$00
     Delay_ms(1000)
     temp=Adc_Read(0)            ‘A/D 変換したセンサー電圧を読みます。
     temp=temp and $03FF   ‘10bitA/D 変換なので、不要なビットを落とします。

     ‘ここから、A/D 値を温度換算します。
     'temp_C=temp_C*488/1000              'A/D to Degree convert
     temp_C=temp-123                      '123=600mV
     temp_C=temp_C*49                      'A/D to Degree convert
     temp_C=temp_C/100                     'A/D to Degree convert

  ‘ヒステリシスも含めて、温度範囲の上限をこえたか？
     if temp_C > (CON_deg + CON_his) then
        ‘超えたので、赤い LED を点滅。
```

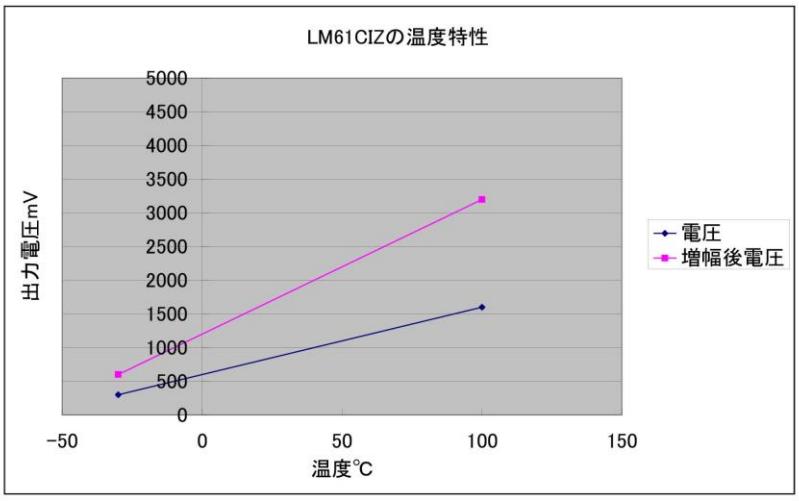
```
    gpio=$02     'RED LED ON
    Delay_ms(200)
    gpio=$00     'RED LED OFF
    Delay_ms(200)
    gpio=$02     'RED LED ON
    Delay_ms(200)
    gpio=$00     'RED LED OFF
    Delay_ms(200)
    gpio=$02     'RED LED ON
    Delay_ms(200)
    gpio=$00     'RED LED OFF
    Delay_ms(200)

else
    if temp_C < (CON_deg - CON_his) then
        ‘下限範囲より下がったので、緑 LED を点滅。
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$00    'GREEN LED OFF
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$00    'GREEN LED OFF
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$00    'GREEN LED OFF
      Delay_ms(200)

    Else
      ‘ヒステリシスの範囲内なので、赤・緑とも点滅。
      gpio=$02     'RED LED ON
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$02     'RED LED ON
```

```
            Delay_ms(200)
            gpio=$04    'GREEN LED ON
            Delay_ms(200)
            gpio=$02     'RED LED ON
            Delay_ms(200)
            gpio=$04    'GREEN LED ON
            Delay_ms(200)

        end if
    end if
    gpio=$00        ‘LED 消灯
    Delay_ms(1000)    ‘1 秒待ちます。
Wend        ‘最初に戻って繰り返します。
end.
```

◆OP アンプ利用による高精度化　について

　今回作成した回路に、OP アンプを使いセンサー出力電圧を増幅することで、精度を上げることができます。

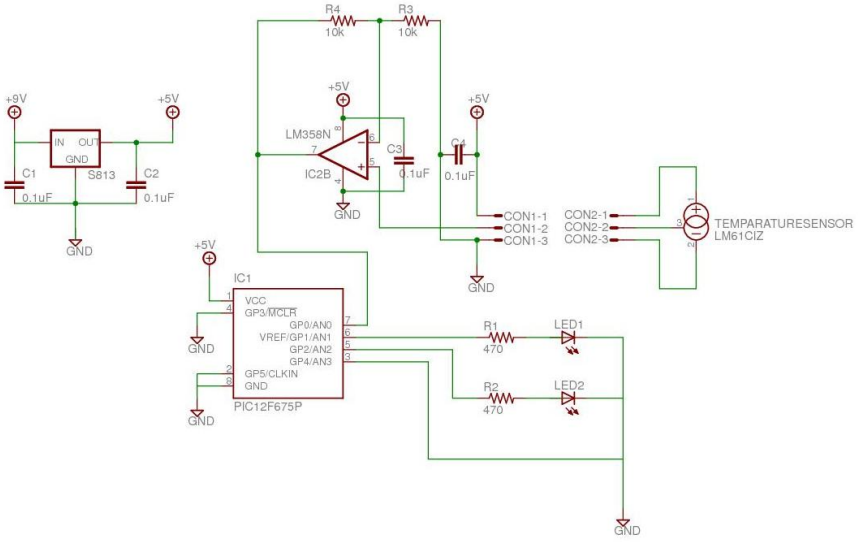
まず、次のグラフをご覧ください。



青い線で描いたのが LM61CIZ のそのままの出力電圧です。この電圧を OP アンプ（LM358）を使うことで、２倍に増幅することができます。

元々、A/D 変換の分解能は、5V÷1024（10bit）≒4.9mV です。LM61CIZ の出力電圧は、1℃あたり 10mV なので、4.9mV÷10mV≒0.5（℃）となり、±0.5℃の誤差がでます。これを上記 OP アンプ（LM358）で２倍の電圧にしてやることで、±0.25℃に精度を高めることができます。

使用する OP アンプと回路図を下記に示します。温度センサーの出力を OP アンプ（LM358）に入力して、電圧を２倍にしています。なお、使用しているコンデンサは、いずれも 0.1μF です。

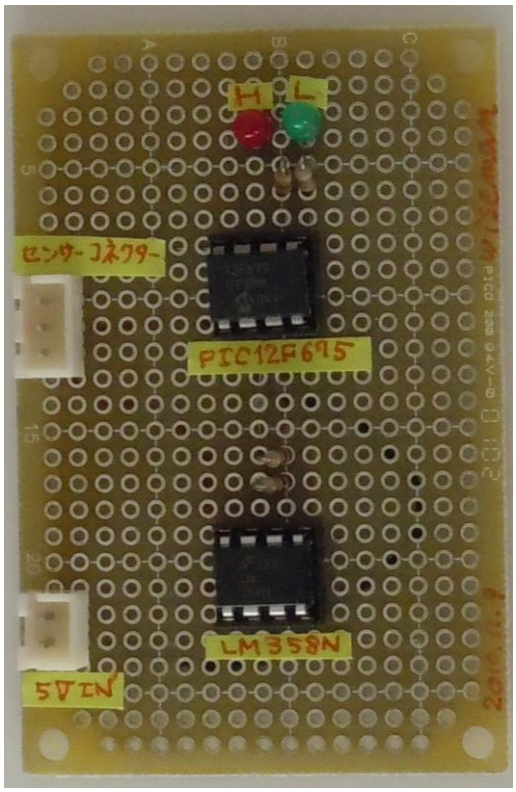


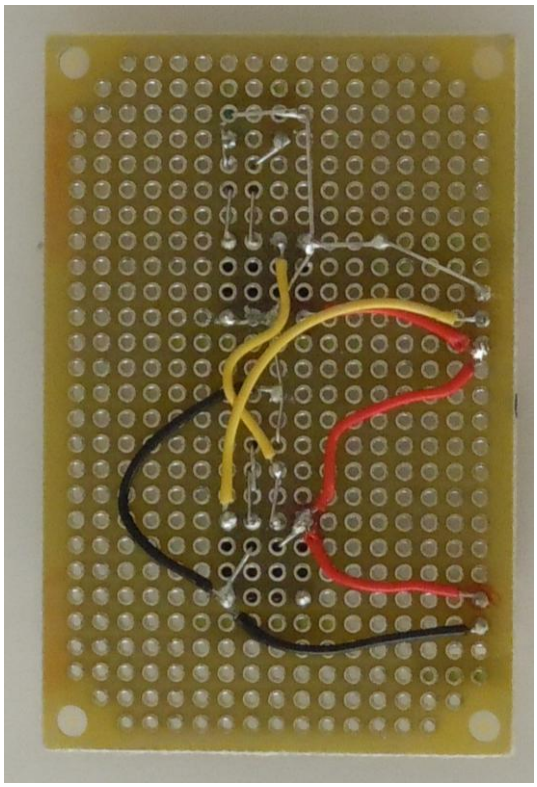
また、プログラム上の A/D 変換値を温度に換算する式は、

温度＝（(A/D 変換値－600mV）÷20mV）－30℃

となります。

できあがった温度計は、(写真)





全体のプログラムは、次の通りです。

```
program OPamp_ondokei
' Declarations section
' 2bit LED displey Thermometer

   const CON_deg = 20                  'kijun ondo
  const CON_his = 1                    'kyoyo hani(+/-)
  dim temp as word
  dim temp_C as integer
  dim led as byte
main:
'     Main program
'   Oscillate Block regulation
  OSCCAL=OSCCAL
  asm
     bsf            status,rp0
     call           0x3ff
     movwf          OSCCAL
```

```
 end asm
gpio=0
 cmcon =%00000111       'for Comperater OFF
 adcon0=%10000001       'for A/D Power ON and Right to LSB
 ansel =%00010001
 trisio=%11111001
 gpio=0
Delay_ms(100)

   ‘ここから温度計測のメインループです。
 while 1            'for ever loop
   gpio=$06
   Delay_ms(500)
   gpio=$00
   Delay_ms(500)
   gpio=$06
   Delay_ms(500)
   gpio=$00
   Delay_ms(1000)
   temp=Adc_Read(0)                'A/D Start & Read
   temp=temp and $03FF

   ‘ここから４行が、オペアンプを使った時の、温度換算です。
   ‘ここから４行が、オペアンプを使った時の、温度換算です。
   'temp_C=temp_C*488/1000              'A/D to Degree convert
   temp_C=temp-246                      '246=1200mV
   temp_C=temp_C*49                      'A/D to Degree convert
   temp_C=temp_C/200                     'A/D to Degree convert

   ‘他の部分は、前記のプログラムと同じです。
   if temp_C > (CON_deg + CON_his) then
     gpio=$02     'RED LED ON
     Delay_ms(200)
     gpio=$00     'RED LED OFF
     Delay_ms(200)
     gpio=$02     'RED LED ON
```

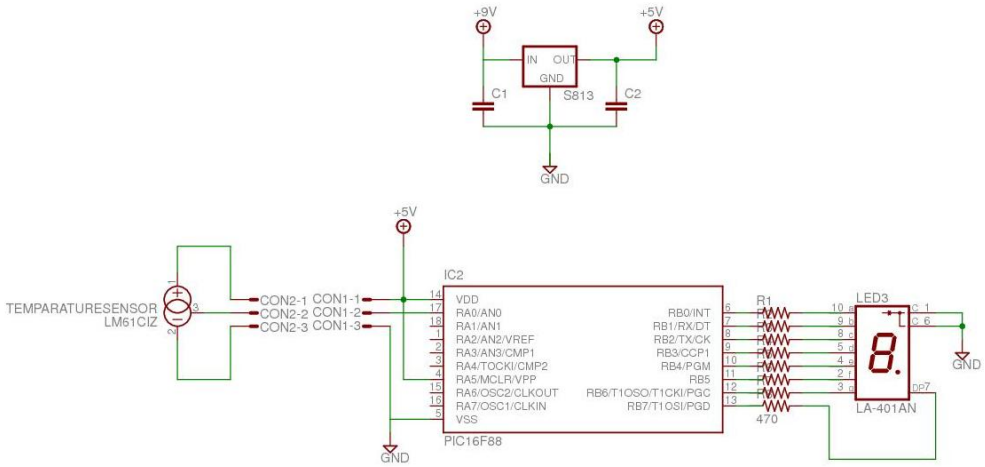
```
    Delay_ms(200)
    gpio=$00     'RED LED OFF
    Delay_ms(200)
    gpio=$02     'RED LED ON
    Delay_ms(200)
    gpio=$00     'RED LED OFF
    Delay_ms(200)
  else
    if temp_C < (CON_deg - CON_his) then
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$00    'GREEN LED OFF
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$00    'GREEN LED OFF
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$00    'GREEN LED OFF
      Delay_ms(200)
    else
      gpio=$02     'RED LED ON
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$02     'RED LED ON
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
      gpio=$02     'RED LED ON
      Delay_ms(200)
      gpio=$04    'GREEN LED ON
      Delay_ms(200)
    end if
  end if
```

```
    gpio=$00
    Delay_ms(1000)
wend
end.
```

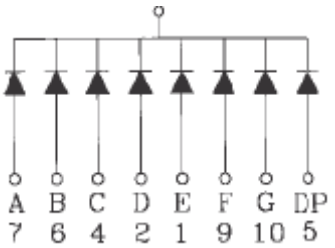
３．２．２　7SegLED による、数値表現

7SegLED は、７つの LED が埋め込まれて０～９と小数点を表す表示器です。LED の方向により、アノードコモンとカソードコモンの製品があります。アノードコモンとは、マイコン I/O の出力そのままに、Hi レベルを出力すると点灯します。カソードコモンはその逆で、Low レベルを出力すると、電気を引き込んで LED が点灯するので、負論理になります。

これもまた、小規模 PIC マイコンで表示する事ができます。
回路図を示します。



カソードコモン 7SegLED の中は、次のようになっています。LED のカソードが内部で接続されていて、Common 端子を GND に接続すれば良いようになっています。A～DP の各 LED 端子は、PIC の I/O に電流制限抵抗を介して接続します。

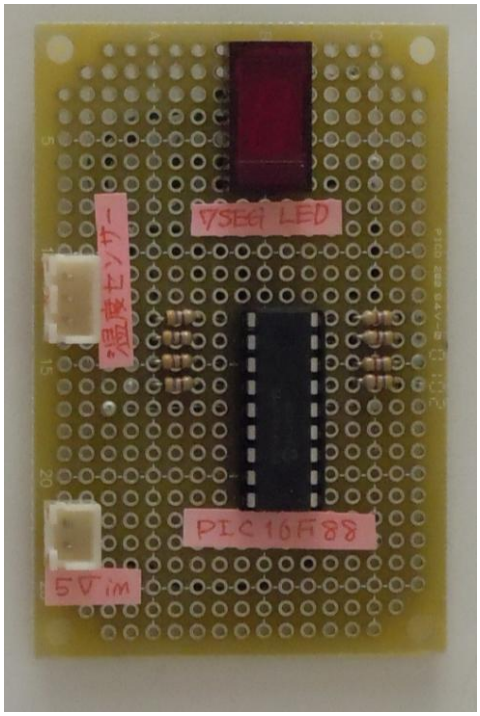


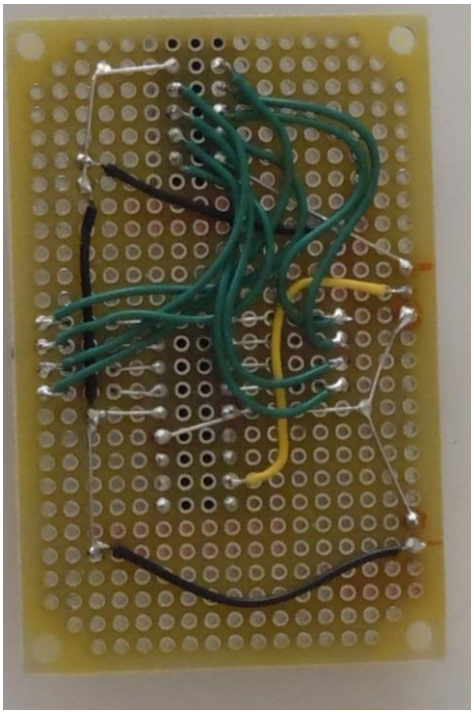
温度センサーは、LM61CIZ を使いそのまま直結します。
ここでも簡易温度計と同じように、少しプログラムに工夫をして、次のような表現のシーケンスにします。

①．７SEG の外周をセグメントごとに光らせ、２回回ります。
②.『H』の文字表示
③．10 の位の温度表示
④．消灯
⑤.『L』の文字表示
⑥．1 の位の温度表示
⑦．消灯　以後、①から繰り返し。

※このシーケンスの間に A/D 変換を行い、適当な時間間隔で、表示を繰り返し行います。

できあがった基板をお見せしましょう。
（写真）





プログラムは、0～9 の数字を LED のセグメントに対応させるように、変換を行っていますが、それ以外は前のプログラムと同じですね。
（プログラム）

```
program Seven_Segment_ondokei
```

```
' Declarations section
' 7SEG LED displey Thermometer
const CON_time = 200     ‘LED 表示時間を設定しています。
dim temp as word
 dim temp_C as integer
 dim temp_H as byte
 dim temp_L as byte
```

下記の部分は、7 セグメントで数字を表すために、数値に対応するセグメントを光らせています。

```
sub procedure Disp_7SEG(dim n as byte)
  select case n
    case 0
          portb=$3F
    case 1
          portb=$06
    case 2
          portb=$5B
    case 3
          portb=$4F
    case 4
          portb=$66
    case 5
          portb=$6D
    case 6
          portb=$7D
    case 7
          portb=$27
    case 8
          portb=$7F
    case 9
          portb=$6F
  end select
end sub
```

数字の 0 を表示するセグメントを右回りに、順に光らせています。

```
sub procedure Rolling()
```

```
    portb=$01
    Delay_ms(CON_time)
    portb=$02
    Delay_ms(CON_time)
    portb=$04
    Delay_ms(CON_time)
    portb=$08
    Delay_ms(CON_time)
    portb=$10
    Delay_ms(CON_time)
    portb=$20
    Delay_ms(CON_time)
    'portb=$40
    'Delay_ms(CON_time)
    'portb=$80
    'Delay_ms(CON_time)
    'portb=$00
    'Delay_ms(CON_time)
end sub

main:
'     Main program
'   Oscillate Block regulation
' OSCCAL=OSCCAL
' asm
'     bsf             status,rp0
'     call            0x3ff
'     movwf           OSCCAL
' end asm

  porta=0
  portb=0
  osccon=%01110110
  cmcon =%00000111       'for Comperater OFF
  ansel =%00000001
  adcon1=%01000000
```

```
 adcon0=%01000000
 trisa =%11111111       'Port A is All input (RA0 is Analog input)
 trisb =%00000000       'Port B is All output
Delay_ms(100)

 while 1            'for ever loop
   Rolling()
   Rolling()
   portb=$00
   Delay_ms(1000)

   ‘A/D 変換と取り込み、温度換算は、他の表示器を使った時と同じプログラムです。
   temp=Adc_Read(0)                'A/D Start & Read
   temp=temp and $03FF

   'temp_C=temp_C*488/1000          'A/D to Degree convert
   temp_C=temp-123                  '123=600mV
   temp_C=temp_C*49                 'A/D to Degree convert
   temp_C=temp_C/100                'A/D to Degree convert

   ‘温度を 10 の位(H)と 1(L)の位に分けます。
   temp_H=byte(temp_C/10)
   temp_L=byte(temp_C mod 10)

  'UPPER Number of temp.
   portb=$76        'H character
   Delay_ms(500)
   portb=$00
   Delay_ms(500)
   portb=$76        'H character
   Delay_ms(500)
   portb=$00
   Delay_ms(500)

   Disp_7SEG(temp_H)    ‘←10 の位の温度表示
   Delay_ms(1000)
```

```
    portb=$00
    Delay_ms(1000)


  'LOWER Number of temp.
    portb=$38       'L character
    Delay_ms(500)
    portb=$00
    Delay_ms(500)
    portb=$38       'L character
    Delay_ms(500)
    portb=$00
    Delay_ms(500)

    Disp_7SEG(temp_L)    '←1 の位の温度表示
    Delay_ms(1000)
    portb=$00
    Delay_ms(1000)
wend
end.
```

３．２．３　ＬＣＤによる、数値表現

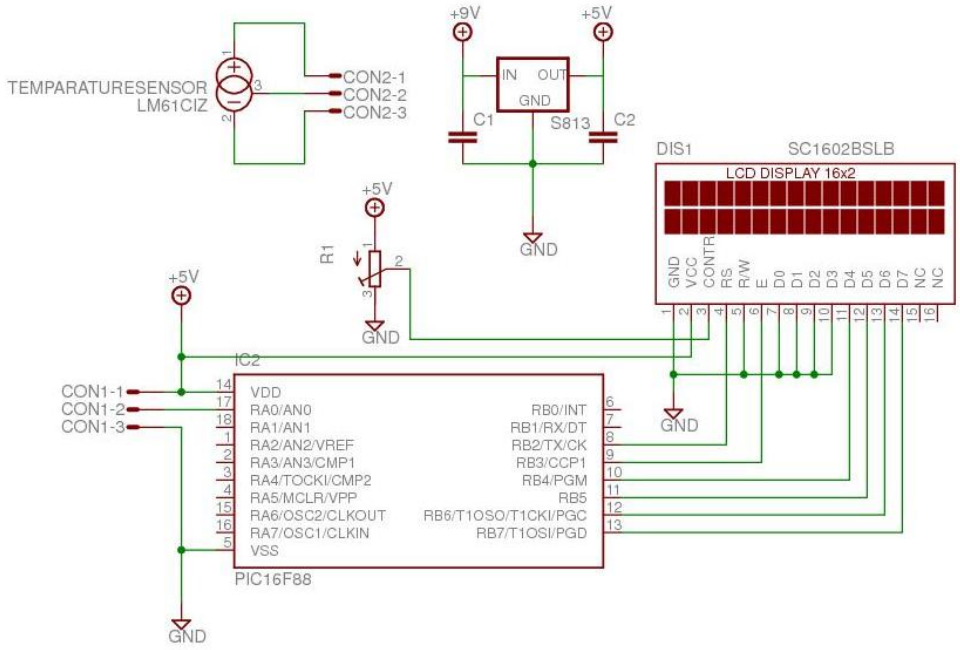
安価でわかり安い表示ができる液晶表示器があります。1000 円前後で、半角英数カタカナ 16 文字×2 行の表示ができます。

LCD コントローラを内蔵している PIC を使えば、容易に LCD 表示が実現します。コントローラが無くてもそれほど難しいことではありません。LCD とマイコンの接続 I/F は、8bit と 4bit があります。ここでは接続の簡単な 4bitI/F で LCD の表示を制御してみましょう。

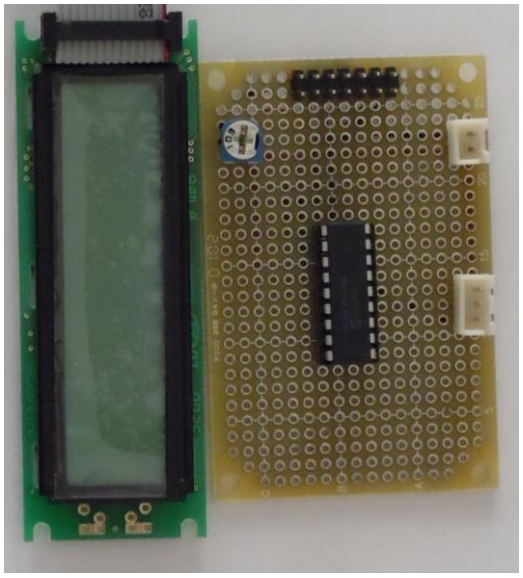
制御用に I/O が 4bit あれば良いので、簡易温度計で使用した、8pin マイコン（PIC12F675）

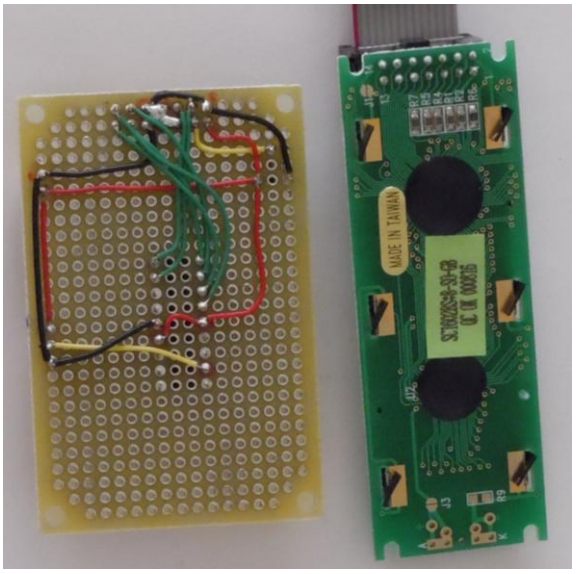
でも制御できそうですが、プログラムメモリ容量が 1KW しかなく、簡易温度計でもその95%以上を使っていましたので、今回はプログラムメモリが4KWのPIC16F88を使います。

回路図を図××に示します。入手するLCDによりGNDとVCCピンが入れ替わっている場合がありますので、使用するLCDのデータシートにより確認してください。



できあがった基板は次のようなものです。





プログラムは、次のようなものですが、これも、全然難しい部分はありませんね。
　A/D 変換したセンサー電圧を、温度数値に変換後、LCD 表示のために数字列にする mikroBasic の内蔵関数を使っています。この辺りは、C 言語によく似ています。

（プログラム）

```
program LCD_Ondokei
```

下記は、LCD のポートを、マイコンのどの I/O に割り当てるかを指示しています。この割り付けによって、後に、簡単なプログラムで LCD 表示が可能になります。割り付けの状態は、回路図と照らしあせてみてください。

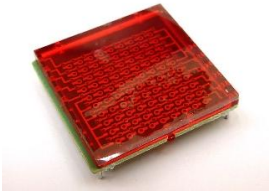
```
' Lcd module connections
dim
   LCD_RS as sbit at RB2_bit
   LCD_EN as sbit at RB3_bit
   LCD_D7 as sbit at RB7_bit
   LCD_D6 as sbit at RB6_bit
   LCD_D5 as sbit at RB5_bit
   LCD_D4 as sbit at RB4_bit

dim
   LCD_RS_Direction as sbit at TRISB2_bit
   LCD_EN_Direction as sbit at TRISB3_bit
   LCD_D7_Direction as sbit at TRISB7_bit
   LCD_D6_Direction as sbit at TRISB6_bit
   LCD_D5_Direction as sbit at TRISB5_bit
   LCD_D4_Direction as sbit at TRISB4_bit
' End Lcd module connections

' Declarations section
' LCD displey Thermometer
   dim temp as word
   dim temp_C as integer

   dim text6 as string[6]‘表示器に出力する文字列用のバッファです。
   dim text20 as char[20] ‘表示器に出力する文字列用のバッファです。
```

```
main:
'    Main program
    ‘IO ポートなどの設定を最初に行います。
   porta=0
   portb=0
   osccon=%01110110
   cmcon =%00000111       'for Comperater OFF
   ansel =%00000001
   adcon1=%01000000
   adcon0=%01000000
   trisa =%11111111       'Port A is All input (RA0 is Analog input)
   trisb =%00000000       'Port B is All output

   ' LCD Initialized and Initial LOGO display
   Lcd_Init()                                          ‘LCD の初期化です。
   Lcd_Cmd(_LCD_CLEAR)                 ' 表示を消します。
   Lcd_Cmd(_LCD_CURSOR_OFF)
   text20="LCD Thermometer"            ' タイトル表示です。
   Lcd_OUT(1,1,text20)                      ‘←1 行目の 1 桁目から表示します。
   text20=" by wiseman (^^)"
   Lcd_OUT(2,1,text20)                      ‘←2 行目の 1 桁目から表示します。
   Delay_ms(2000)                       ' 2 秒待ちます。

   while True            'for ever loop
      ‘このブロックも、これまでのプログラムと同じです。
     temp=Adc_Read(0)                'A/D Start & Read
     temp=temp and $03FF

     'temp_C=temp_C*488/1000              'A/D to Degree convert
     temp_C=temp-123                      '123=600mV
     temp_C=temp_C*49                      'A/D to Degree convert
     temp_C=temp_C/100                     'A/D to Degree convert
      ‘ここで、temp_C に温度が入っています。
     IntToStr(temp_C,text6)     ‘温度の数値を数字列に変換します
```

```
    'text20="LCD Thermometer"          ' Title display
    text20="Measurement      "         ' Title display
    Lcd_OUT(1,1,text20)
    text20="Now Temp: "
    Lcd_OUT(2,1,text20)
    Lcd_OUT(2,11,text6)      '温度を 2 行目、11 桁目から表示します
    Delay_ms(1000)
  wend
end.
```

３．２．４　ドットマトリクス

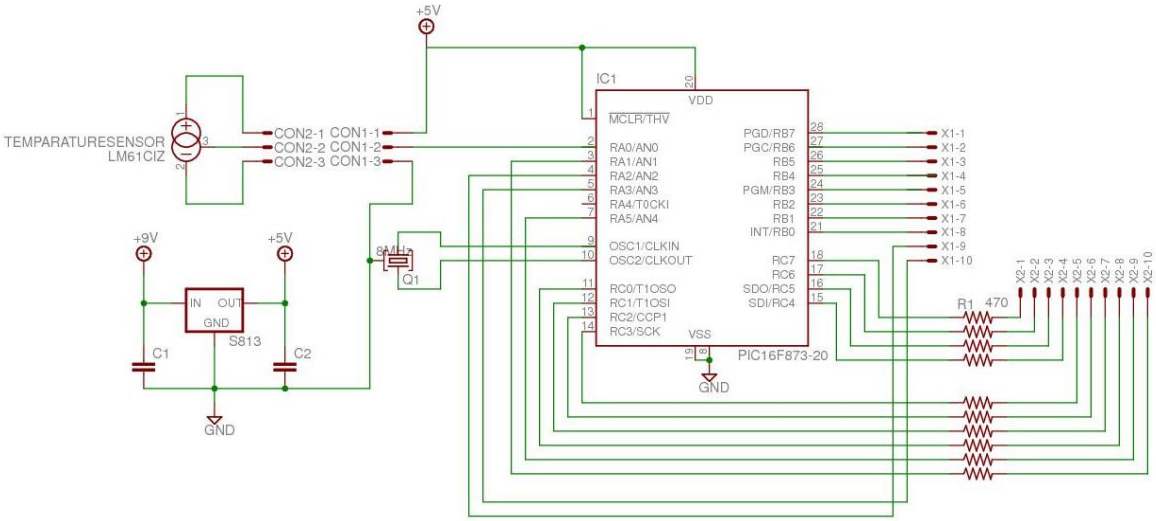
手持ちのドットマトリクス表示器は LED を縦横 10 個の配列に並べた物です。カラードットマトリクス LED もありますが、制御が面倒なので今回は、この赤色 10×10 ドット LED を使ってみましょう。



この表示器は、１５年以上も前から安定的に供給されているパーツで、１個 200 円と安価です。

10×10 ドットなので、マイコンの I/O で直接制御するためには、20bit の I/O が必要です。このため、ピン数の多い PIC16F873 を使いました。

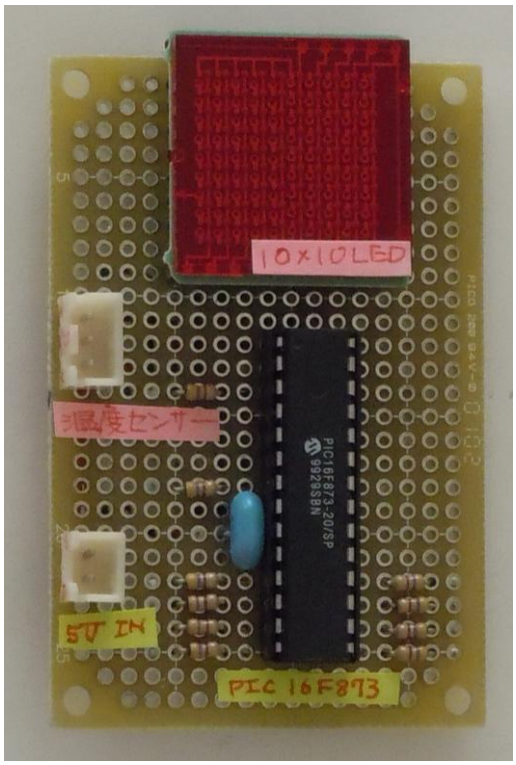
回路図を示します。PIC の右側にある部分が、ドットマトリクス LED の脚ピンになります。

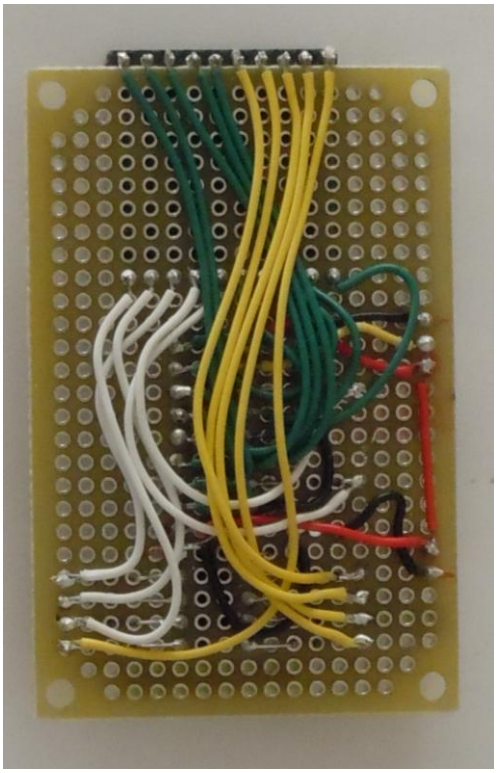


表示は、次のようにします。

①.『A/D』を２回、表示します。
②. 10 の位の温度を表示します。
③. 消灯します。
④. 1 の位の温度を表示します。
⑤.『℃』を表示します。
⑥. ①から繰り返します。

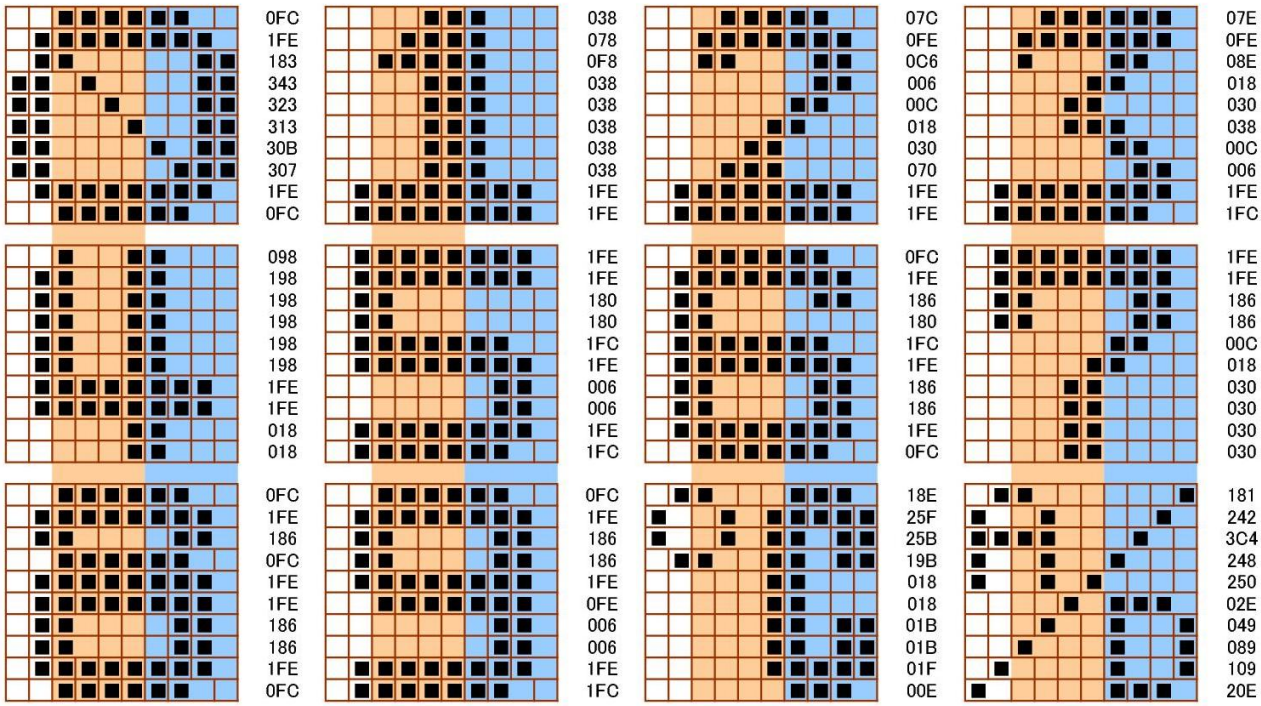
基板は、次のようになりました。やはり、LED 制御のために縦・横のコントロールが必要ですから、配線も多くなりました。





ドットマトリクス表示は自由な表現ができる反面、文字のフォントを作らなければいけないので、少しソフト開発作業に手間がかかりますが、これも一度作ってしまえば、後々使えるデータですので、是非一度チャレンジしてください。フォントを作成するときは、Excel を使って設計すると楽に作れます。元になったワークシートの一部を掲載しておきます。

（フォント）



制作したプログラムとは、次の通りです。

　このプログラムは、LED のダイナミック表示の原理が分かり安いように、わざと点滅をゆっくりにしてあります。実際に使用する際は、プログラムを少し変更して、表示間隔を短くした方が、文字がちらつかず見やすい表示になります。

（プログラム）

```
program DotMatrix_Ondokei
' Declarations section
' DotMatrix LED display Thermometer

'font data　ここから、フォントのデータ定義ブロックです.
const CON_A as　word[10] =　($030,$048,$084,$102,$201,$3FF,$201,$201,$201,$201)
const CON_B as word[10] = ($3FC,$302,$301,$302,$3FC,$3FC,$302,$301,$302,$3FC)
const CON_C as word[10] = ($18E,$25F,$25B,$19B,$018,$018,$01B,$01B,$01F,$00E)
const CON_D as word[10] = ($181,$242,$3C4,$248,$250,$02E,$049,$089,$109,$20E)
const CON_0 as word[10] = ($0FC,$1FE,$383,$343,$323,$313,$30B,$307,$1FE,$0FC)
const CON_1 as word[10] = ($038,$078,$0F8,$038,$038,$038,$038,$038,$1FE,$1FE)
const CON_2 as word[10] = ($07C,$0FE,$0C6,$006,$00C,$018,$030,$070,$1FE,$1FE)
```

```
const CON_3 as word[10] = ($07E,$0FE,$08E,$018,$030,$038,$00C,$006,$1FE,$1FC)
const CON_4 as word[10] = ($098,$198,$198,$198,$198,$198,$1FE,$1FE,$018,$018)
const CON_5 as word[10] = ($1FE,$1FE,$180,$180,$1FC,$1FE,$006,$006,$1FE,$1FC)
const CON_6 as word[10] = ($0FC,$1FE,$186,$180,$1FC,$1FE,$186,$186,$1FE,$0FC)
const CON_7 as word[10] = ($1FE,$1FE,$186,$186,$00C,$018,$030,$030,$030,$030)
const CON_8 as word[10] = ($0FC,$1FE,$186,$0FC,$1FE,$1FE,$186,$186,$1FE,$0FC)
const CON_9 as word[10] = ($0FC,$1FE,$186,$186,$1FE,$0FE,$006,$006,$1FE,$1FC)
'Fonts Data Buffer --- CON_X--->cgen by procedure FontSetting()
dim cgen as word[10]

' FontSetting は ch に対応するフォントを cgen にコピーします.
sub procedure FontSetting(dim ch as byte)
select case ch
  case $0A
    cgen=CON_A
  case $0B
    cgen=CON_B
  case $0C
    cgen=CON_C
  case $0D
    cgen=CON_D
  case $00
    cgen=CON_0
  case $01
    cgen=CON_1
  case $02
    cgen=CON_2
  case $03
    cgen=CON_3
  case $04
    cgen=CON_4
  case $05
    cgen=CON_5
  case $06
    cgen=CON_6
  case $07
```

```
      cgen=CON_7
   case $08
      cgen=CON_8
   case $09
      cgen=CON_9
   end select
end sub

'Display character n times
sub procedure Display(dim n as integer)
   dim buf_L as byte
   dim buf_H as byte
   dim i as integer
   dim j as integer
   dim font as word

   for j=0 to n
      for i=0 to 9

      font=cgen[i]
      buf_L=not byte(font)          ' for portb
      buf_H=byte(font>>8)           ' for porta
      buf_H=buf_H and $03
      buf_H=(not buf_H<<2) and $0C

      select case i
         case 0
            portc=$01
         case 1
            portc=$02
         case 2
            portc=$04
         case 3
            portc=$08
         case 4
            portc=$10
```

```
        case 5
          portc=$20
        case 6
          portc=$40
        case 7
          portc=$80
        case 8
          portc=$00
          buf_H=buf_H or $02
        case 9
          portc=$00
          buf_H=buf_H or $20
      end select
   porta=buf_H
   portb=buf_L
   Delay_ms(10)
   next i
next j
end sub

sub Procedure ClearPort()
   porta=0
   portb=0
   portc=0
   end sub

   dim temp as word             'real A/D value
   dim temp_C as integer     'for Temperature (DEG)
   dim temp_H as byte          'for 10^1
   dim temp_L as byte          'for 10^0

main:
'    Main program
dim ptr as ^word

ClearPort()
```

```
adcon0=%00000000
adcon1=%10001110
trisa =%00000001       'Port A : RA0 is Analog input.
trisb =%00000000       'Port B is All output for X
trisc =%00000000       'Port C is All output for Y

Delay_ms(100)

while 1            'for ever loop
ClearPort()
FontSetting($0D) 'A/D
Display(8)
ClearPort()
Delay_ms(200)

Display(8)           'A/D
ClearPort()
Delay_ms(500)
temp=Adc_Read(0)                'A/D Start & Read
temp=temp and $03FF
'temp_C=temp_C*488/1000     'A/D to Degree convert
temp_C=temp-123                 '123=600mV
temp_C=temp_C*49              'A/D to Degree convert
temp_C=temp_C/100             'A/D to Degree convert

temp_H=byte(temp_C/10)
temp_L=byte(temp_C mod 10)

'UPPER Number of temp.
FontSetting(temp_H)
Display(10)
ClearPort()
Delay_ms(300)

'LOWER Number of temp.
FontSetting(temp_L)
```

```
Display(10)
ClearPort()
Delay_ms(300)

'Deg
FontSetting($0C)
Display(20)
ClearPort()
Delay_ms(500)
wend
end.
```

いかがでしょうか。いろいろと表示器をつないだマイコンを活用した実例を説明しましたが、どうやら比較的自由に標示ができるのは、LCD かドットマトリクス LED であることがおわかりでしょうか。また、小さなマイコンでも表現さえ工夫すれば、たった 4bit でも十分な表現能力があることも分かりました。

３．２．５　SD カードロガー作成

さて、この章の纏めとして、これまでに掲げたプログラムと表示装置などを使い、もう少し実用的なシステムの開発事例を紹介します。

現在のマイコンは、A/D 変換も去ることながら、外部デバイスとの通信もたやすく実現出来るようになっています。この機能と mikroBasic のライブラリを合わせて使用した、SD カードに温度をロギングするユニットを作成してみましょう。

温度センサーは、これまでに使用したものと同じで、A/D 変換後の温度換算が簡単です。また、表示器は LCD を使うことで、メッセージを表示できるようにします。

ロギングする媒体は、SD カードを使います。SD カードに温度データが記録できれば、後で PC に SD カードを挿して Excel などで、データの分析などができるので、データの応用価値が大幅に広がります。また、マイコンを利用したシステムの可用性も大きくなります。

SD カードのコントロールは、mikroBasic の MMC（マルチメディアカード）ライブラリを使うことで、実現出来ます。

使用するマイコンは、MMC カードとの I/F が容易な MSSP（Master Synchronous Serial Port）モジュールを内蔵している PIC18F2550 を使うことにしましょう。MSSP モジュー

ルとは、外部デバイスとの２大 I/F である、SPI（Serial Peripheral Interface）モードと EEPROM などの不揮発性メモリとのインターフェースである I2C（Inter-Integrated Circuit）モードをサポートしているので、応用範囲がとても広いマイコンです。

このマイコンは 28pin で、LCD を接続しても A/D 変換ポートが使用でき、さらに SD カードとのインターフェースも確保できなおかつ、1 個 400 円ほどで入手可能なものです。他の型番でも同様の機能を持ったものが数多く出回っているので、ここでその利用方法を学んでしまえば計測技術の実践手法が身につきます。

SD カードロギングユニットの回路図を示します。SD カードとの詳しい I/F は後の章で解説することして、回路図上の接続を確認してください。ロギングスタートと終了を指示する SW が必要になります。

また、ロギングの周期を後で設定出来るようにするために、2Bit の DipSW を取り付けてあります。ここでは、ロギング周期は固定とします。

SD カードは、電源 ON 時にはすでに装着されていることを前提とします。電源 ON のまま、抜き差しするには、『活線挿抜』という技術が必要で、プログラムも込み入ります。また、旨く行かないとデバイスを壊してしまうので、今回は行いません。興味のある方は研究してみてください。

LCD コントロールは PORTB に、SD カードは PORTC に接続します。

SD カードとマイコンの接続するポート名を眺めておいてください。新しいロギングユニット開発にこの接続が大きく役立ちます。

しい部分はありませんが、SD カードは 3.3V 電源なので、電池駆動でなく、AC アダプターからの電源供給として、レギュレータで 5V とそこからさらに 3.3V を作ります。

全体の動作は、

①. 電源 ON すると、内部の初期化を行い、LED を点滅、LCD にメッセージを表示して、スタート SW が押下されるのを待ちます。

②. スタート SW は、基板上の黒い SW です。この SW が押下されるまでは、そのときの温度を計測して LCD に表示を繰り返し行います。

③. スタート SW が押されると、SD カードのフォーマットと初期化を行います。このときエラーが発生したり SD カード未挿入の時は、LCD にメッセージを表示します。
★このような状態になったら、電源を OFF して、SD カードを確認します。

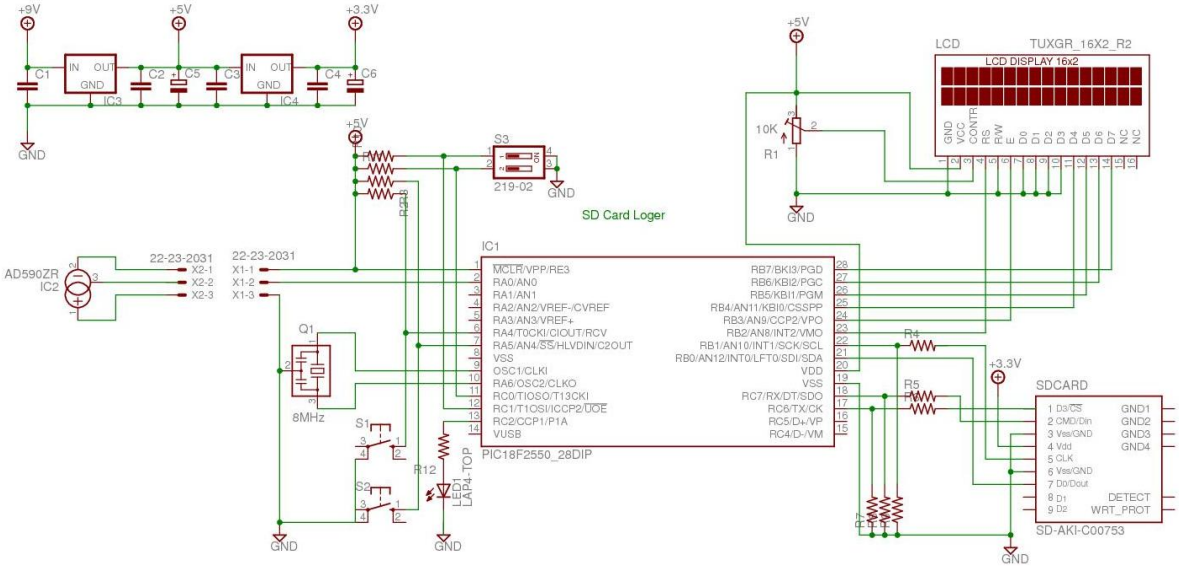
④. スタート SW が押下されたら、温度計測を行い、結果を SD カードに書き込みます。ストップ SW（基板上の赤い SW）が押されるまで、これを繰り返します。

⑤. ストップ SW が押されると、ロギングを停止して、LCD 二メッセージを表示します。

⑥. 電源を OFF して SD カードを取り出し、PC でその内容を開きます。SD カードには、LOG.txt というファイルができていますので、そのファイルを Excel などで開

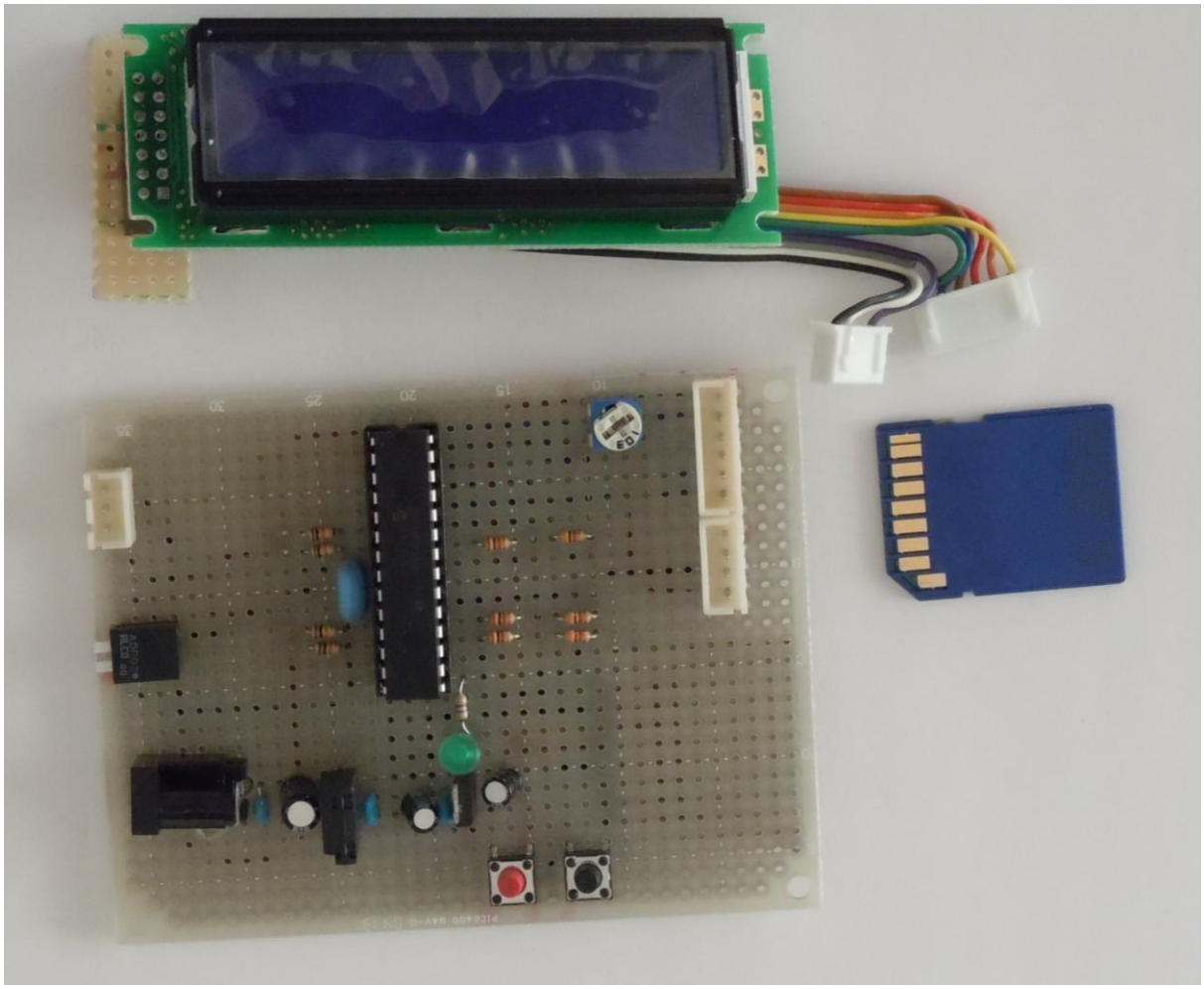
くと、温度変化の経過が分かります。SD カードは、毎回フォーマットされるので、必要なデータは Excel で開いて確認した後、PC に保存しておきます。

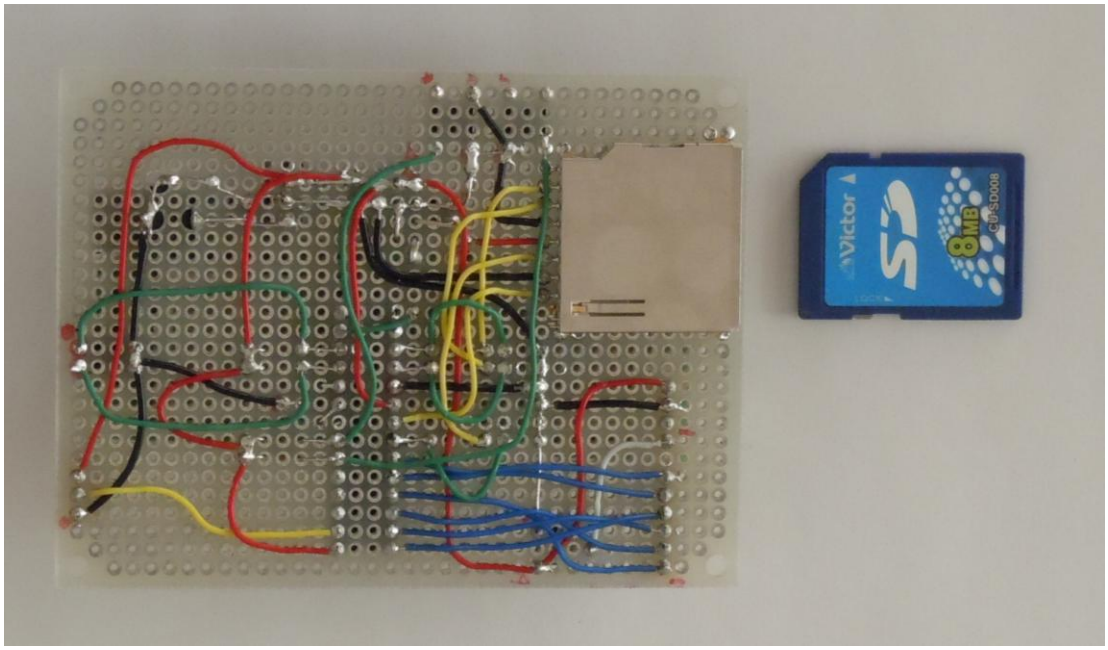
（回路図）



パーツで目新しいものは、SD カードソケットと SW くらいです。
SD カードソケット（写真）
タクト SW（写真）
DipSW（写真）

（基板）




Victor
SD
8MB

機能のわりに、配線はそれほど大変ではありません。これも、マイコンが持つ機能と今日のデバイス開発の結果が有機的に結合した結果です。
また、mikroBasic のようなソフト面での開発環境も整って、どのようなものでも開発が容易になって来ています。

（プログラム）

```
program SD_Card_Logger_V1a
' Declarations section
' Lcd module connections
dim
   LCD_RS as sbit at RB2_bit
   LCD_EN as sbit at RB3_bit
   LCD_D7 as sbit at RB7_bit
   LCD_D6 as sbit at RB6_bit
   LCD_D5 as sbit at RB5_bit
   LCD_D4 as sbit at RB4_bit

dim
   LCD_RS_Direction as sbit at TRISB2_bit
   LCD_EN_Direction as sbit at TRISB3_bit
   LCD_D7_Direction as sbit at TRISB7_bit
   LCD_D6_Direction as sbit at TRISB6_bit
   LCD_D5_Direction as sbit at TRISB5_bit
   LCD_D4_Direction as sbit at TRISB4_bit
' End Lcd module connections

‘下記は、SD カードとの I/F 接続指定です。
' MMC module connections
dim Mmc_Chip_Select as sbit at RC6_bit
dim Mmc_Chip_Select_Direction as sbit at TRISC6_bit
' MMC module connections

dim sw as byte
dim pa as byte
```

```
dim pc as byte
dim CRLF as char[2]
dim temp as short            ' Current Temperatuer(Deg)
dim ft as float              ' temp for float
```
LCD 表示のためのバッファです。
```
dim text4 as string[4]
dim text10 as string[10]
dim text11 as string[11]
dim text20 as char[20]
dim nb as longint

 ‘下記は、SW の状態を取り込むプログラムです。
 ‘
 ‘グローバルなメモリ sw にスイッチの状態が入ります。
sub function SWinf() as short
   ' SW information
   ' Bit0.1:DipSW
   ' Bit2.3:PushSW
   pa=PORTA
   pa=pa and $30
   pa=pa >> 2
   pc=PORTC
   pc=pc and $03
   sw=pa or pc
   result=sw
end sub



sub function Tmp() as short
   ' Temp.=((AD-300mV)/10mV)-30deg
   'temp=((ad-300)/10-30)
   dim ad as word
   ad = Adc_Read(0)                    ' Current tem. A/D comversion.
    ‘A/D 変換値に 1bit の分解能を掛けて温度換算します。
   ft = float(ad*4.8828125-300)/10.0-30.0
```

```
  temp=short(ft)
  result=temp
end sub

'下記は、動作確認用の LED の ON/OFF 用プログラムです。
sub procedure LED_ONOFF()
  ' LED Indicates
  PORTC.2=1
  Delay_ms(125)
  PORTC.2=0
  Delay_ms(125)

  PORTC.2=1
  Delay_ms(125)
  PORTC.2=0
  Delay_ms(375)

end sub
'下記は、SD カードをクイックフォーマットするプログラムです。
sub function SD_QF() as byte
  'use fat16 quick format instead of init routine if a formatting is needed
  result=Mmc_Fat_QuickFormat("SD CARD LOGER 1a")
  select case result
    case 0
      ' Display start message.
      text20="SDC Format OK..."
    case 1          'フォーマットが旨く行かないと 1 が返されます。
      ' Display start message.
      text20="SDC Format Error"
    case 255        'SD カードが挿入されていないと 255 が返されます。
      ' Display start message.
      text20="SDC Not Detect.."
  end select
  Lcd_OUT(2,1,text20)
end sub
```

```
‘下記は、SD カードの初期化プログラムです。
‘これに先立ち、SD カードはフォーマットされていなければいけません。
sub function SD_INI() as byte
  result=Mmc_Fat_Init()
  select case result
    case 0
      ' Display start message.
      text20="SDC Init. Success"
    case 1
      ' Display start message.
      text20="Sector not found."
    case 255
      ' Display start message.
      text20="SDC Not Detect.."
  end select
  Lcd_OUT(2,1,text20)
end sub

main:
'    Main program
  ' for SD card text CR/LF code
  CRLF[0]=$0d          ' for CR
  CRLF[1]=$0a          ' for LF

  PORTC=$00
  LATC=$00

  PORTA=$00
  LATA=$00
  ADCON1=$0E
  TRISA=$3F
  TRISB=$00              'PORT B is output
  TRISC=$BB
  CMCON=$07

  ' LED Indicates
```

```
  PORTC.2=1
  Delay_ms(500)
  PORTC.2=0
  Delay_ms(500)
  PORTC.2=1
  Delay_ms(500)
  PORTC.2=0
  '-----

  ' LCD Initialized and Initial LOGO display
  Lcd_Init()
  Lcd_Cmd(_LCD_CLEAR)                  ' Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF)
  text20="SD CARD LOGER 1a"            ' Title display
  Lcd_OUT(1,1,text20)
  text20=" by wiseman (^^)"
  Lcd_OUT(2,1,text20)
  Delay_ms(2000)                       ' Wait
  '-----



Main_Loop_:
  SWinf()
  if (sw and $04) = $00 then        ' Start SW ON ???
    Delay_ms(10)                       ' Wait
    SWinf()
    if (sw and $04) = $00 then        ' Start SW ON ???
      goto Measur_
    else
      goto Main_Loop_
    end if

Measur_:
    text20="Now Initialize.."
    Lcd_Out(1,1,text20)
    ' Set SPI to master mode, clock = Fosc/4, data sampled at the middle of interval, clock idle state
```

low and data transmitted at low to high edge:

```
    ' Initialize SPI1 module and set pointer(s) to SPI1 functions
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64, _SPI_DATA_SAMPLE_MIDDLE,
_SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH)

    'use fat16 quick format instead of init routine if a formatting is needed
    if SD_QF() <> 0 then goto Exit_
    end if
    if (SD_INI() <> 0) then goto Exit_
    end if

    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV16, _SPI_DATA_SAMPLE_MIDDLE,
_SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH)

    Mmc_Fat_Assign("log.txt",$A0)
    Mmc_Fat_Rewrite()

    text20="Now Logging...   "
    Lcd_Out(1,1,text20)
    nb=1
    while True
      Tmp()    ' measurement
      text20="CurrentTemp:      "
      Lcd_OUT(2,1,text20)
      ShortToStr(temp,text4)
      Lcd_OUT(2,13,text4)

      LongintToStr(nb,text11)
      Mmc_Fat_Write(text11 ,11)
      Mmc_Fat_Write("," ,1)
      Mmc_Fat_Write(text4 ,4)
      Mmc_Fat_Write(CRLF ,2)


      LED_ONOFF
      SWinf()
```

```
        if (sw and $08) = $00 then          ' Stop SW ON ???
          Delay_ms(10)                      ' Wait
          SWinf()
          if (sw and $08) = $00 then        ' Stop SW ON ???
            goto Stop_
          end if
        end if
        nb=nb+1
      wend
    else
      Tmp()     ' measurement
      text20="CurrentTemp:       "
      Lcd_OUT(2,1,text20)
      ShortToStr(temp,text4)
      Lcd_OUT(2,13,text4)
      goto Main_Loop_
    end if


Stop_:
    ' Display complete message.
    text20="Logging Complete"
    Lcd_OUT(1,1,text20)
    text20=" by wiseman (^^)"
    Lcd_OUT(2,1,text20)

Exit_:

end.
```

４．通信

通信と聞くと、遠く離れた場所との電気信号による情報のやりとりをイメージしがちですが、もっと近く僅か数 cm あるいは数 mm しか離れていないデバイス間の情報のやりとりも含みます。

特に、ここで説明したいのは、その方法が技術の進歩によっていろいろな規格が生まれているが、その通信プロトコルをソフトウエアでも専用のポートを持った CPU がハード的にサポートする事もできる事です。

現在、マイコンや PC 絡みでの通信は、ほぼシリアル通信で行われています。機器の制御などには、パラレルルインターフェースを用いることもありますが、デバイスと・PC 間は、シリアル通信が利用されています。

４．１　センサー・マイコン間通信

これまで、見てきたようにセンサーはマイコンのアナログ入力ポートに直接接続できます。使用するマイコンによって A/D 変換の分解能が違いますが、A/D 変換器を持つどのマイコンもおおむね 5V を 10bit または 12bit で A/D 変換できます。もっと精度を上げたいときは、OP アンプでセンサーの温度勾配を大きくしたり、A/D 変換専用 IC などを使用したりしますが、農業においては 1℃未満の精度が必要な事はまれで、**温度変化の傾向を知ることが重要**ですから、センサー直結で十分だと思います。

これまで使用したセンサーは、環境に応じて出力電圧が変化するので、その電圧を測っていましたが、直接デジタルで温度を通知してくれるデジタル温度センサーもあります。

このデジタルセンサーを使う場合は、A/D 変換は必要がない代わりに、センサーに対して初期化や現在温度の通知を要求するためのやりとりが必要です。この章で通信と云っているのは、このような相手のデバイスとの『やりとり』のこと全般を指しています。

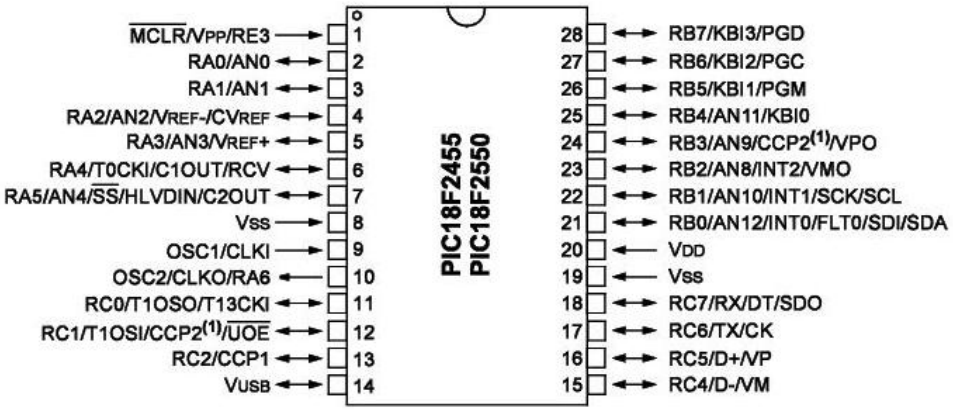
相手のデバイスとやりとりするためには、お互いにタイミングを図る必要がありそのために、クロック信号が流れる線と、デバイスからの出力線と、デバイスへの入力線の３本を使って、やりとりをするのが３線式のインターフェース SPI になります。電気的には、これ以外に GND（0V）の線も必要になります。

４．２　マイコン・メモリ間通信

３章のまとめで作成した、SD カードロガーは、マイコンと SD カードという外部メモリ間で独特のやりとりをしています。このやりとりには、規格が定まっていて、今回は SPI インターフェースを使っています。SPI とは、Serial Peripheral Interface　の頭文字

を取った呼称で、3 線式の同期式シリアル通信インターフェースのことです。

3 本の信号線は、データ出力信号（SDO）、データ入力信号（SDI）、クロック信号（SCK）に使います。回路図で PIC マイコンとの接続されている端子の名称を見ると、どの端子がどの信号に割り当てられて、SD カードのどの端子と接続すれば良いかが分かります。



PIC マイコンのピンダイアグラム

SD カードの信号は次の 6 つです。

(22pin) SCK：システムクロック

(21pin) Dout：データアウト

(18pin) Din：データイン

(任意)　CS：チップセレクト

+3.3V：電源

GND：グラウンド

+3.3V と GND は、マイコンとの接続ではないので、上の 4 つの信号をマイコンの端子に接続すれば良く、CS 以外は、マイコンに該当する名称が付いているピンに 2kΩの抵抗を介して接続します。CS は任意のピンに接続して、SD カードのフォーマット指示に使われています。

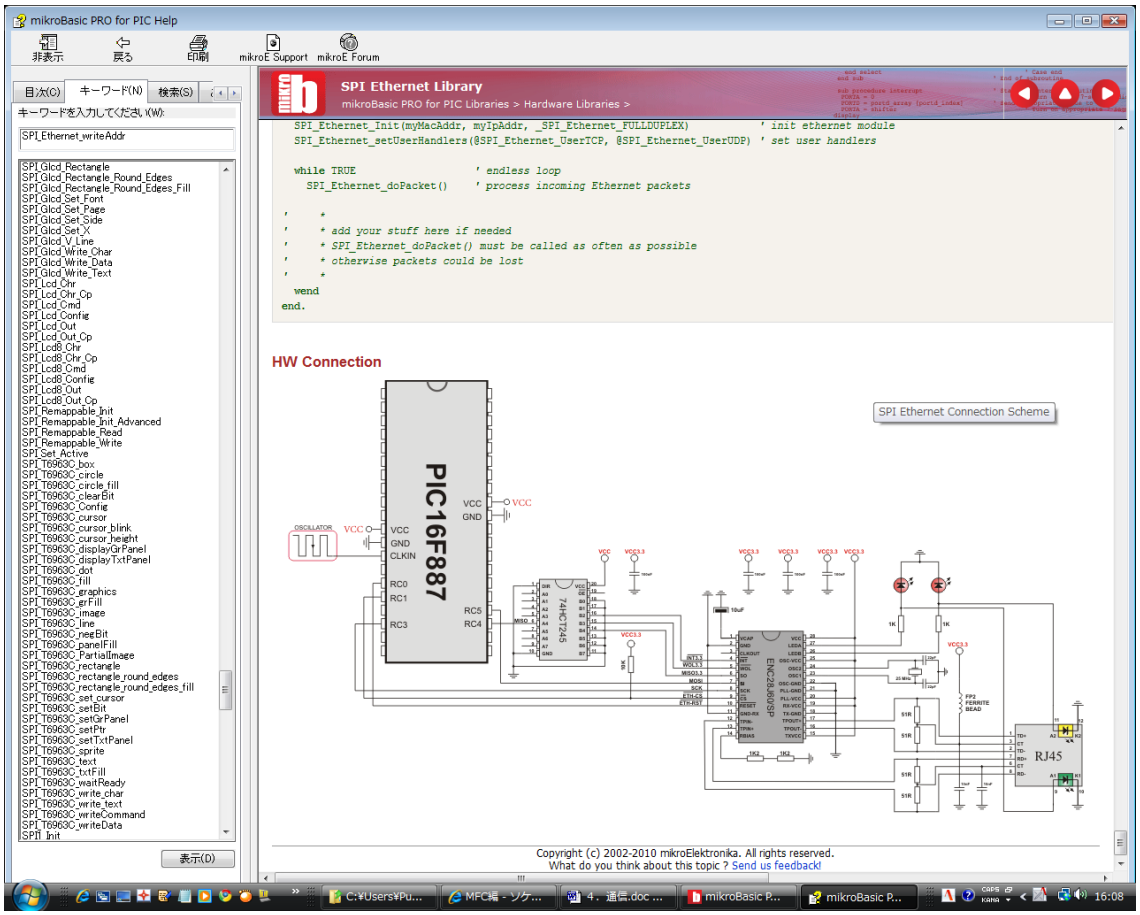
４．３　マイコン・ＰＣ間通信

マイコンとＰＣの間での通信には、分かりやすさの点で、RS-232c がもっとも多く用いられると思います。接続する線の数は 3 本で TxD と RxD と GND です。これを見ると SPI 方式と何ら変わらないように思えますが、RS-232c の信号レベルが 12V なので、レベ

ル変換を行う IC が必要になります。この方式はクロック同期式ではなく、スタートビットで通信を開始するので調歩同期通信と呼ばれ、多用されています。RS-232c が多く使われるのは、マイコンからの送信を PC で受信するだけならば、信号 1 本と GND の 2 本だけの接続線で通信ができるからでもあります。

本研究で使用する mikroBasic では、Ethernet による通信ライブラリも完備していて、PIC マイコンに Ethernet Controler を接続して、ネットワーク上につなぐ事例が HELP に解説されています。参考プログラムも例示されているので、是非一読してみてください。

HELP に含まれる回路図サンプルを掲載しておきます。



４．４　ＰＣ・ＰＣ間通信

RS-232c は、マイコンと PC 間だけでなく、PC・PC 間通信でも使われます。また、PC とモデムなどにも使われていて、老舗の通信方式です。

現在はネットワーク全盛ですから、ftp やソケット(Winsock)などを用いた通信が行われています。これらの通信方法については、本研究では触れませんが、これまでの章の内

容を理解できれば、容易に構築できるでしょう。本研究の発展として、チャレンジしてみてください。ソケット通信は、VB で開発する例が多数 WEB にも公開されています。

５．制御

農業において、一番汎用的な制御手法は、ON/OFF 制御です。これは、農業に限らず、もっとも一般的な制御の手法です。このシステムを自力で構築できれば、農業家はもちろん、他の分野でも役に立ちます。

５．１　ON/OFF 制御

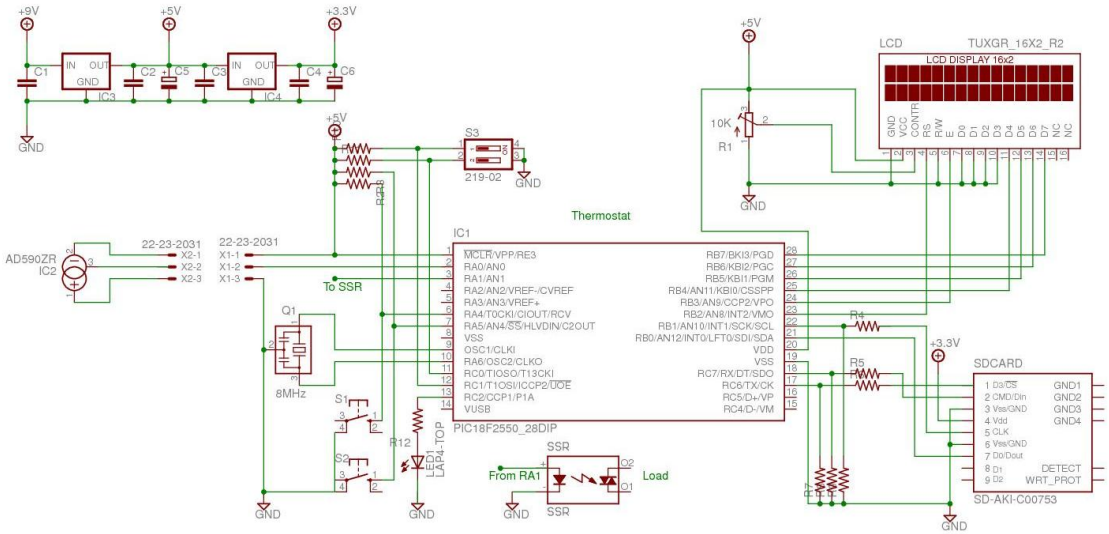
『ON/OFF 制御』と言っても、難しいことは何もありません。３章の『LED による閾値表現』で説明したように、ある基準の値とセンサーから得た値の比較により、大きいか小さいかで、スイッチを ON/OFF することで実現できます。温度センサーを使い、設定温度になるようなシステムを組んでみましょう。

基準値は、LCD 値を見ながら SW で自由に設定できるようにします。温度が低い場合は、出力を ON して、電球（AC100V）を点灯して、その熱で温度センサーが暖まります。ヒステリシス制御を行い、基準値より一定温度高くなったら、出力を OFF して、温度が下がるのを待ちます。

これを繰り返して、センサー検出温度が、設定温度になるように制御します。

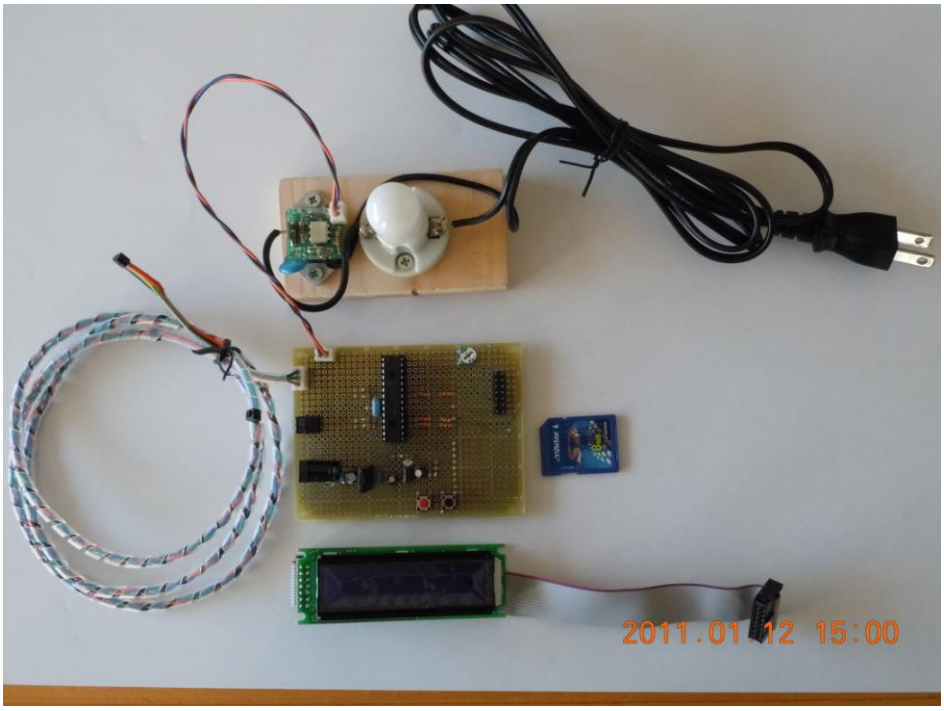
これまでと違うのは、SW で基準温度を設定することと、AC100V の電球を制御することです。電池駆動のマイコンで AC100V をコントロールするには、リレーか SSR（ソリッドステートリレー）を使います。ここでは、AC100V を電気的に絶縁してコントロールする SSR を使います。AC100V の電球・20A 級で 300 円ほどの SSR が市販されていて、マイコンの出力をそのまま、SSR の制御入力につなぐことで、AC100V もコントロールできます。

回路は、LCD や SW、SSR 等が付加されるので、多少込み入ってきますが、実際作って見るとそんなにごちゃごちゃした配線にはなりませんので、安心してください。



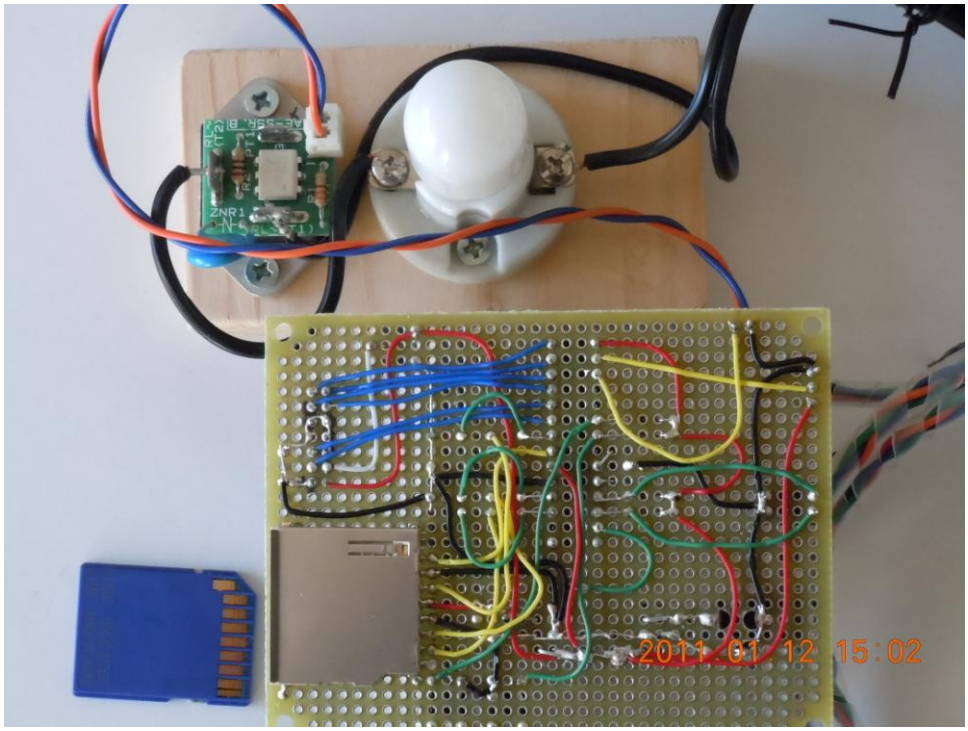
回路図上の Load の部分に、AC100V で点灯するランプを接続します。このランプが光ることで、センサーの検出温度が高まり、目的の温度になったところでランプを消灯します。すると今度は、温度が下がりはじめて、一定の温度にまで下がったところで、再びランプを点灯します。これを繰り返して設定温度を保つように制御します。この制御を行いながら、SD カードに検出温度を記録します。記録されたデータを後で分析すると、先の例のようにヒステリシス制御が行われている様子がわかるはずです。

作成したユニットを示します。



写真上部の、ナツメ球が付いているユニットが、SSR ユニットです。ナツメ球の代わりに、電磁弁やファン等を接続することもできます。

基板の裏側は、3 章で作成した SD カードロガーとほぼ同じです。



プログラムは少し複雑ですが、難しい事をしているわけではありません。

大まかな流れは、次のとおりです。

１．PIC レジスタや I/O ポートの初期化

２．LCD 初期表示

３．LED、SSR（AC100 制御）のテスト

４．目標温度の設定

ここでは、基板に実装した DIP SW を ON にして、STAT SW（温度＋）・STOPSW（温度―）で目標温度（15～30℃）を設定出来るようにしています。

５．SD カード初期化

SD カードを、フォーマットして LOG.TXT というファイルを作ります。

６．A/D 変換・温度表示・データロギングと SSR への出力

SSR につながっている、RA1 を High(1)にすると、LOAD に接続されている AC 機器の電源が入ります。今回の研究では、屋内用のナツメ球を ON/OFF しています。

７．停止 SW が押されるまで、６を繰り返す

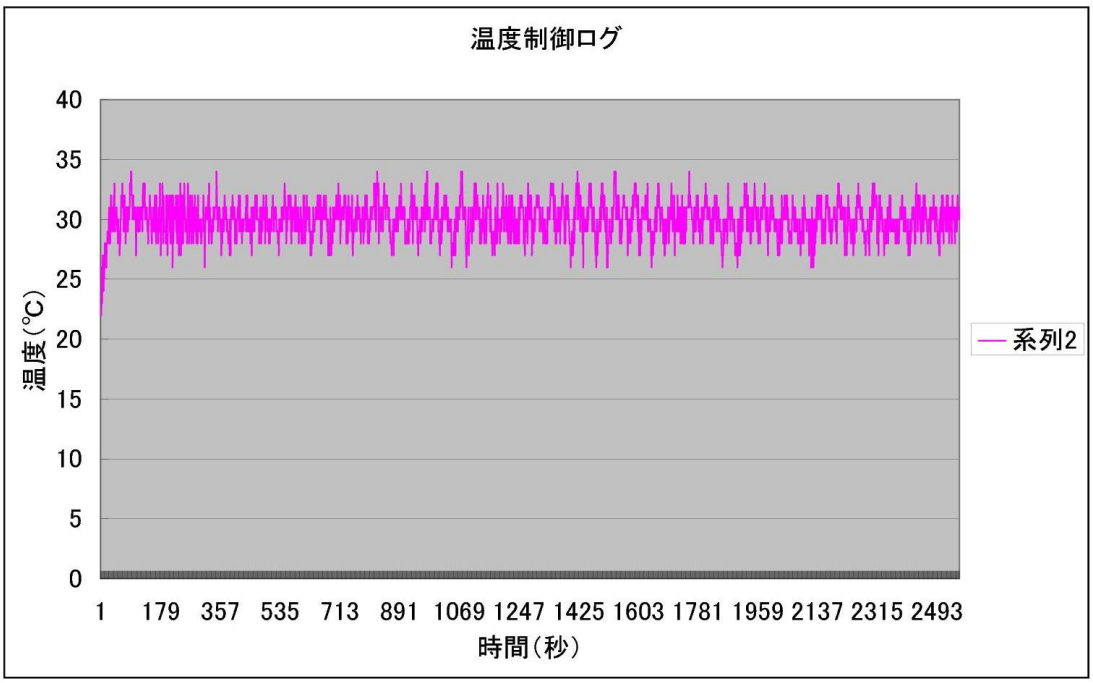
STOP SW を押下して、LOG を停止した後、SD カードを PC にセットして、できあがった LOG.TXT の内容を、Excel 等で分析すると、センサー部を指定温度にするために、どのような温度経過があったのかがわかります。

また、SSR の ON/OFF も、SD カードにロギングすると、さらにわかりやすい記録がと

れます。リアルタイムクロック IC を使えば、正確な時刻も記録できることは、前に少し触れました。

【参考のため】SSR の代わりにリレーを使う場合は、マイコンの出力をトランジスタに接続して電流を増幅して、リレーのコイル電流を供給します。このときは、ダイオードを入れておかないと、リレーコイルの逆起電力でトランジスタが破壊されてしまいます。

実際に、温度を制御した際に取得した LOG を Excel に取込み、そのままグラフにした物を示します。制御開始から温度が上昇し、30℃を中心に振れていますが、目標値に対して温度制御が働いていることが分かります。



最後に、作成したプログラム（Thermostat）の全リストを下記に示します。

program Thermostat

```
' Declarations section
' Lcd module connections
dim
   LCD_RS as sbit at RB2_bit
   LCD_EN as sbit at RB3_bit
   LCD_D7 as sbit at RB7_bit
   LCD_D6 as sbit at RB6_bit
   LCD_D5 as sbit at RB5_bit
   LCD_D4 as sbit at RB4_bit

dim
   LCD_RS_Direction as sbit at TRISB2_bit
   LCD_EN_Direction as sbit at TRISB3_bit
   LCD_D7_Direction as sbit at TRISB7_bit
   LCD_D6_Direction as sbit at TRISB6_bit
   LCD_D5_Direction as sbit at TRISB5_bit
   LCD_D4_Direction as sbit at TRISB4_bit
' End Lcd module connections

' MMC module connections
dim Mmc_Chip_Select as sbit at RC6_bit
dim Mmc_Chip_Select_Direction as sbit at TRISC6_bit
' MMC module connections


 const CON_his = 1


dim sw as byte
dim pa as byte
dim pc as byte
dim CRLF as char[2]
dim temp as short          ' Current Temperatuer(Deg)
dim tt as short            ' Target Temperature
dim ft as float            ' temp for float
dim text4 as string[4]
```

```
dim text10 as string[10]
dim text11 as string[11]
dim text20 as char[20]
dim nb as longint


sub function SWinf() as short
   ' SW information
   ' Bit0.1:DipSW
   ' Bit2.3:PushSW

   ' PushSW information
   pa=PORTA
   pa=pa and $30
   pa=pa >> 2

   ' DipSW information
   pc=PORTC
   pc=pc and $03
   sw=pa or pc

   result=sw
end sub


sub function Tmp() as short
   ' Temp.=((AD-300mV)/10mV)-30deg
   'temp=((ad-300)/10-30)
   dim ad as word

   ad = Adc_Read(0)                    ' Current tem. A/D comversion.

   ft = float(ad*4.8828125-300)/10.0-30.0
   temp=short(ft)

   result=temp
```

```
end sub


sub procedure LED_ONOFF()
   ' LED Indicates
   PORTC.2=1
   Delay_ms(125)
   PORTC.2=0
   Delay_ms(125)

   PORTC.2=1
   Delay_ms(125)
   PORTC.2=0
   Delay_ms(375)


end sub

sub function SD_QF() as byte
   'use fat16 quick format instead of init routine if a formatting is needed
   result=Mmc_Fat_QuickFormat("SD CARD LOGER 1a")
   select case result
      case 0
         ' Display start message.
         text20="SDC Format OK..."

      case 1
         ' Display start message.
         text20="SDC Format Error"

      case 255
         ' Display start message.
         text20="SDC Not Detect.."
   end select

   Lcd_OUT(2,1,text20)
```

```
end sub



sub function SD_INI() as byte
   result=Mmc_Fat_Init()
   select case result
      case 0
         ' Display start message.
         text20="SDC Init. Success"

      case 1
         ' Display start message.
         text20="Sector not found."

      case 255
         ' Display start message.
         text20="SDC Not Detect.."
   end select

   Lcd_OUT(2,1,text20)
end sub



'-------------------------
' Target temperature setting
'-------------------------
sub procedure TTS()

   while true
      SWinf()
      if (sw and $0F)=$0E then
         continue
      end if
```

```
if (sw and $0F)=$0F then
   break
end if

if (sw and $04) = $00 then         ' (+) SW ON ???
   Delay_ms(200)
   if (sw and $04) = $00 then ' (+) SW ON ???
      ' Yes (+) Increase Target TEMP. !!
      tt=tt+1
      if tt>30 then
         tt=30
      end if
      ShortToStr(tt,text4)       ' Target temperature.
      Lcd_OUT(2,7,text4)
   end if
end if

while true
   SWinf()
   if (sw and $04) = $04 then         ' (+) SW OFF ???
      break
   end if
wend

SWinf()
if (sw and $08) = $00 then         ' (-) SW ON ???
   Delay_ms(200)
   if (sw and $08) = $00 then ' (-) SW ON ???
      ' Yes (-) Decrease Target TEMP. !!
      tt=tt-1
      if tt<15 then
         tt=15
      end if
      ShortToStr(tt,text4)       ' Target temperature.
      Lcd_OUT(2,7,text4)
   end if
```

```
      end if


      while true
        SWinf()
        if (sw and $08) = $08 then          ' (-) SW OFF ???
          break
        end if
      wend

      Tmp()     ' measurement
      'text20="CurrentTemp:       "
      text20="Temp:(       )         "
      Lcd_OUT(2,1,text20)
      ShortToStr(temp,text4)
      Lcd_OUT(2,13,text4)
      ShortToStr(tt,text4)       ' Target temperature.
      Lcd_OUT(2,7,text4)



      SWinf()
      if (sw and $01) = $01 then          ' Dip SW_1 OFF ???
        ' Yes Exit Loop
        exit
      end if

    wend

end sub


sub procedure Control()
    if temp > (tt + CON_his) then
      ' Heater OFF ---> Cooling
      PORTa.1=0
```

```
    else
      if temp < (tt - CON_his) then
        ' Heater Power ON
        PORTA.1=1
      end if
    end if
end sub


main:
'    Main program
    ' for SD card text CR/LF code
    CRLF[0]=$0d             ' for CR
    CRLF[1]=$0a             ' for LF

    PORTC=$00
    LATC=$00

    PORTA=$00
    LATA=$00
    ADCON1=$0E
    TRISA=$3D
    TRISB=$00                'PORT B is output
    TRISC=$BB
    CMCON=$07

    ' LED Indicates
    PORTC.2=1
    Delay_ms(500)
    PORTC.2=0
    Delay_ms(500)
    PORTC.2=1
    Delay_ms(500)
    PORTC.2=0
    '-----
    ' SSR
```

```
  PORTA.1=1
  Delay_ms(500)
  PORTA.1=0
  Delay_ms(500)
  PORTA.1=1
  Delay_ms(500)
  PORTA.1=0
  '-----

  ' LCD Initialized and Initial LOGO display
  Lcd_Init()
  Lcd_Cmd(_LCD_CLEAR)                  ' Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF)
  text20="Thermostat v1a.."            ' Title display
  Lcd_OUT(1,1,text20)
  text20=" by wiseman (^^)"
  Lcd_OUT(2,1,text20)
  Delay_ms(2000)                       ' Wait
  '-----

  'Target Tempperature setting
  tt=23
  text20="Target TEMP.= 23"
  Lcd_OUT(2,1,text20)
  Delay_ms(2000)                       ' Wait
  '-----


Main_Loop_:
  SWinf()
  if (sw and $04) = $00 then        ' Start SW ON ???
    ' Yes Stert SW ON(=Low)
    Delay_ms(10)                       ' Wait
    SWinf()
    if (sw and $04) = $00 then        ' Start SW ON ???
      ' Yes Stert SW ON(=Low)
```

```
    goto Measur_
  else
    goto Main_Loop_
  end if

Measur_:
  text20="Now Initialize.."
  Lcd_Out(1,1,text20)
  ' Set SPI to master mode, clock = Fosc/4, data sampled at the middle of interval, clock idle state low and data transmitted at low to high edge:
  ' Initialize SPI1 module and set pointer(s) to SPI1 functions
  SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH)

  'use fat16 quick format instead of init routine if a formatting is needed
  if SD_QF() <> 0 then goto Exit_
  end if
  if (SD_INI() <> 0) then goto Exit_
  end if

  SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV16, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH)

  Mmc_Fat_Assign("log.txt",$A0)
  Mmc_Fat_Rewrite()

  text20="Now Logging...  "
  Lcd_Out(1,1,text20)
  nb=1
  while True
    Tmp()   ' measurement
    'text20="CurrentTemp:     "
    text20="Temp:(     )     "
    Lcd_OUT(2,1,text20)
    ShortToStr(tt,text4)  ' Target Temperature.
    Lcd_OUT(2,7,text4)
```

```
      ShortToStr(temp,text4)    ' Current Temp.
      Lcd_OUT(2,13,text4)

      LongintToStr(nb,text11)
      Mmc_Fat_Write(text11 ,11)
      Mmc_Fat_Write("," ,1)
      Mmc_Fat_Write(text4 ,4)
      Mmc_Fat_Write(CRLF ,2)

      Control()

      LED_ONOFF()

      SWinf()
      if (sw and $08) = $00 then         ' Stop SW ON ???
        Delay_ms(10)                      ' Wait
        SWinf()
        if (sw and $08) = $00 then        ' Stop SW ON ???
          goto Stop_
        end if
      end if
      nb=nb+1
    wend
  else
    ' Wait for Start...
    Tmp()     ' measurement
    'text20="CurrentTemp:       "
    text20="Temp:(       )        "
    Lcd_OUT(2,1,text20)
    ShortToStr(temp,text4)
    Lcd_OUT(2,13,text4)
    ShortToStr(tt,text4)
    Lcd_OUT(2,7,text4)

    Delay_ms(2000)                    ' Wait
```

```
      SWinf()
      if (sw and $01) = $00 then         ' Dip SW_1 ON ???
         Delay_ms(10)                    ' Wait
         SWinf()
         if (sw and $01) = $00 then      ' Dip SW_1 ON ???
            ' Yes Target TEMP. Setting!!
            TTS()
         end if
      end if

      goto Main_Loop_
   end if


Stop_:
   ' Display complete message.
   text20="Logging Complete"
   Lcd_OUT(1,1,text20)
   text20=" by wiseman (^^)"
   Lcd_OUT(2,1,text20)

Exit_:

   PORTA.1=0    ' Heater OFF

end.
```

６．フィールドサーバー

６．１　フィールドサーバーとは

「フィールドサーバー」は、カメラとセンサーと通信装置を一体にした、屋外用の簡易計測機器システムです。フィールドサーバーがあれば、他に大掛かりな装置を必要とせずに農園管理や環境モニタリング等の、センサーネットワークを構築できます。

WEB で公開されているフィールドサーバーは、ほぼ同様の形をしています。
カメラとネットに接続する機器が納められているようです。名称を『アグリサーバー』などとしている機器もありますが、基本機能は同じです。

図 6.1 WEB で公開されているフィールドサーバー

風雨にさらされるので、透明のアクリルケースに納められたものが一般的なようです。
ハウス内で使用するようなものでも、塵埃がカメラに影響を及ぼすので、しっかりとしたケースに入れる必要があるでしょう。電源は、AC 電源を取れる場合は簡単ですが、農場の真ん中で使うような場合は、工夫が昼用です。実際、太陽光発電を行うパネルと充電コントローラ＋バッテリを備えたものもありますが、かなり高価なものになるのと、太陽光発電システムが旨く働いてくれるかどうかが問題です。

６．２　フィールドサーバーの開発

本研究では、WEB に公開されている写真を参考にフィールドサーバーも開発してみました。温度センサー部分は、第３章にて作成した SD カードロガーを用いることにして、画像監視用の機器を作りました。

カメラ仕様：

- □ 画像信号：NTSC
- □ イメージセンサー：1/3 インチ CCD カラーイメージセンサー
- □ 画素数：816(H)×495(V)
- □ 電源：12V・DC

カメラ本体に自動絞りのレンズを取付けて使用します。カメラは、一般のビデオ信号(NTSC)を出力するタイプなので、これをネットワークに乗せるためにビデオサーバーが必要です。カメラからビデオサーバーに映像信号を送り、ビデオサーバーが IP アドレスを持ち、ネットワーク経由で動画像を表示します。必要であれば、デジタルビデオレコーダー等の機器を接続すれば、過去の様子を検索することもできます。

パン・チルト・ズーム機能を持つカメラを接続して、ネットワーク越しにリモートコントロールも可能です。

図 6.2 カメラ・レンズ・ビデオサーバー

図 6.3 開発した、フィールドサーバー用カメラ

６．３　フィールドテスト

平成 22 年 12 月 17 日（金)、千葉黎明高校の農場をお借りして、カメラと温度ロガーの動作テストを行いました。

当日は、この冬一番の冷え込みを記録し、農場にも霜が降りた状態で機材の設置を行いました。

農場での電源の取得場所の制約もありカメラは、東南東向きに設置しました。温度ロガーは、農場の土中と地上に設置する予定でしたが、気温が零下まで下がったので、土中温度の変化があまり捉えられないと考え、地上 10cm の位置に設置して温度変化を記録しました。温度データの変化は、記録の間隔が約 1 秒とし、ロギング開始から 2 時間 30 分で、9000 点のデータを取得しました。

図 6.4 フィールドテストの設置場所

図 6.5 ネットワーク経由の画像表示

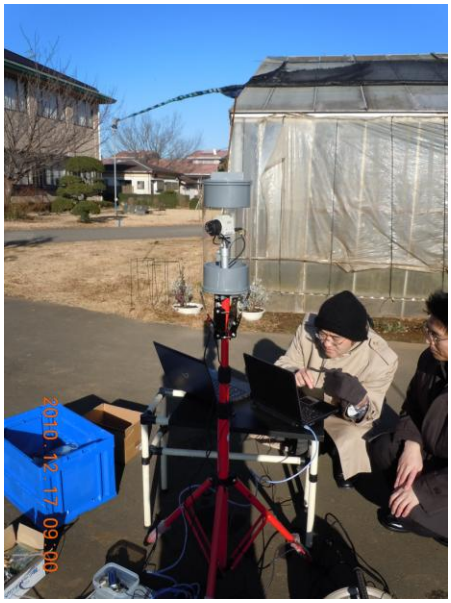


図 6.6 フィールドテストの様子

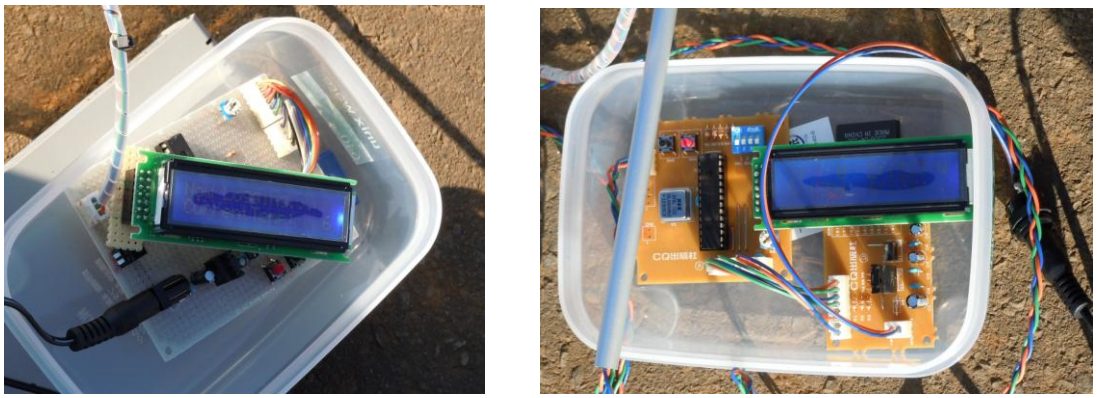


図 6.7 手配線のロガー(左)と学生が作成したロガー(右)

図 6.8 温度センサーの加工(アルミパイプの先端をカシメて封入しました)

実際に取得したデータは、次のような CSV 形式で保存されていて、Excel などで容易に分析が可能です。各行の最初の数字は、シーケンス番号です。

1, 6
2, 6
3, 5
4, 4
5, 4
6, 5
7, 7
8, 4
9, 5
10, 5
11, 4
・

・
・
8989, 14
8990, 13
8991, 15
8992, 17
8993, 12
8994, 14
8995, 11
8996, 14
8997, 16
8998, 13
8999, 16
9000, 14

図 6.9 温度ログファイルの内容

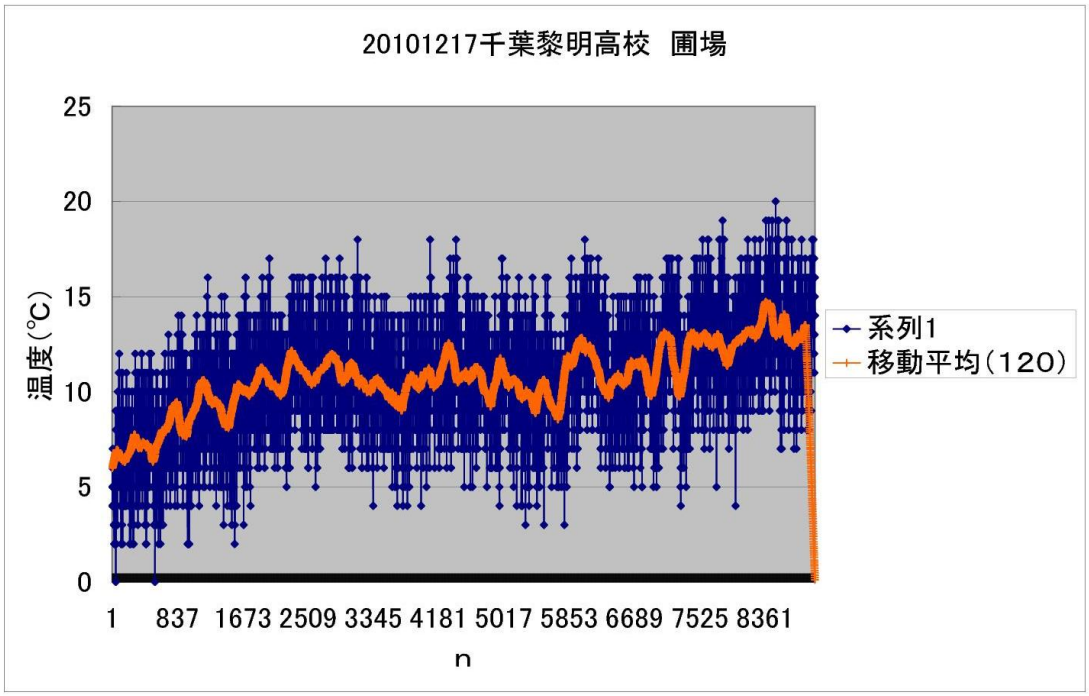


図 6.10 圃場での温度ログ

実際に取得したデータをグラフ化したものが上の図です。

青：系列 1・・・生データ(全 9000 点)

橙：移動平均(120 点)

生データの幅が広いのは、地上 10cm に設置したため地、風が止んで日光が当たると周囲の温度が上がるが、風が吹くと冷たい冷気がセンサー部に当たり、すぐに温度が下がるので、このような温度の変化が記録されたと思われます。

また、A/D 変換をほぼ連続で行っている事も、このような結果につながったと考えられます。実際に必要な温度の記録は、短くても 1 分から 5 分程度の間隔で十分です。SD カードには容量的に余裕があるので、時計 IC などを使用して時刻も合わせてロギングすると、より役立つフィールドサーバーが構築できるでしょう。

６．４　システムの構築

圃場の状態監視システムを構築するための機器は、次のように接続してネットワーク上の PC で画像やセンサー情報の取得をおこないます。

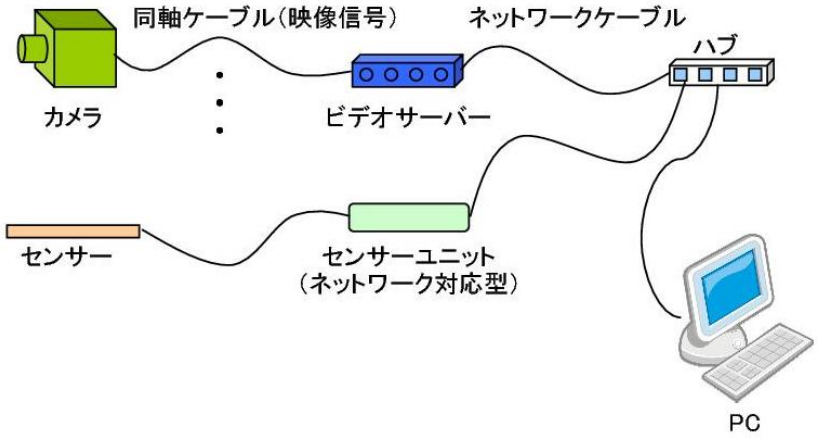


図 6.11 フィールドサーバーの構成

もし、農場が離れている場合や、遠隔地での情報共有を行う場合には、VPN を用いて接続すると良いでしょう。千葉市では、車で 1 時間ほどの距離を NTT のフレッツサービスを利用した接続で、画像監視システムを構築しています。

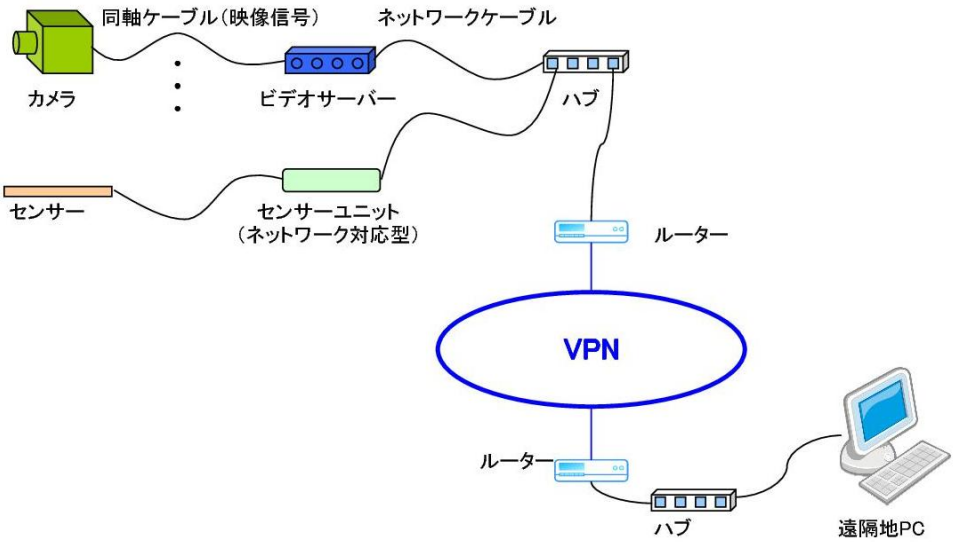


図 6.12 VPN を用いたフィールドサーバー

様々な機器を用いて構築することができるフィールドサーバーですが、カメラ・ビデオサーバー・ハブ・センサーユニットなどは、信号の配線以外に、電源も必要です。
実際にシステムを構築する際には、信号線と電源の引き回しにも工夫をして、すっきりとした形にまとめる事が必要になります。

本研究で開発したカメラドームの脚部はパイプで作成してあるので、センサーユニットとカメラの信号や電源の引き回しを、パイプ内部を通すことで、脚部下に取り出すことができます。

７．記録

農業家の方々は、日々の仕事の記録をどの様にして管理しているでしょうか。
『管理』と云うと大げさに聞こえますが、これは農業の基本です。一昨年、昨年、今年、来年と続けて行く農業家の仕事にどのような変化があったのか。なぜ変化があったのか。旨くいったことは何だったのか、旨くいかなかったのはどのような事であったか。
『どうすれば旨くいき、どうすれば失敗するのか。』・・・農業家の記録は、農業を生業として生活しようとする方々のノウハウ集になるわけで、これほど大切なものはありません。上手に記録を付けて、上手に使えることが、農業成功の秘訣と云っても過言ではないと思います。

では、どのような記録の取り方がよいのでしょうか。記録していないものを後で調べることはできないので、どのような形・方法であれ、とにかく記録する事が大前提です。
農業 IT 技術者に望まれる事が、この辺りに大いに関わってきます。先に開発した SD カードロガーや、フィールドサーバー、カメラなどは各々記録を取るためのツールです。
では、作業の中身はどのように記録すると、便利でしょうか。

ノートに克明な記録をメモする事でも良いと思いますが､やはり IT エンジニアとしては、PC を大いに活用したいところです。記録がデジタルデータになっていれば、何かと便利であることは、云うまでもありません。
農業家向けに市販されている農業日誌ソフトが、いくつかありますが、中でも『ソリマチ農業日誌』は、有名です。カレンダーに記録を付けて、種まきをいつおこなった等の記録を付けると、収穫時期がカレンダーに反映されたり、肥料や資材を購入した記録を付けると、それが経費に自動反映されて会計帳簿も兼ねるなど、パソコンでできそうな事を網羅していますが、ユーザー数が増加しないのは、やはり今の農業家の主たる牽引役が高齢者で PC の苦手意識が強いことと、『これまでそんなもの使わなくてもやれてきたから』という理由であろうと思います。
PC の苦手意識があると、どうしても筆記の記録に頼ります。電気も要らないし、紙と鉛筆さえあれば、記録ができるからです。しかし、その方法では後でその記録をデータとして使うことが困難です。ですから、どうしてもコンピュータへの入力をしてもらいたいのは、誰もが考えることです。では、『紙に書く』＝『コンピュータ入力』となるようにするには、たとえばタブレット PC で手書き入力を使えばよいでしょうか？
もしそれが都合良く機能するようであれば、すでにそのような事は行われているはずですが、それがなかなか普及しないのは、農業の現場にはそぐわないからでしょう。
比較的制約を受けずにどこでもデータ入力ができるような仕組みを作っているシステムがあります。有限会社ワイズマンが開発した携帯電話を利用したシステムで、携帯電話で

入力しておいた記録が、後々Excelのシートに反映され、検索・集計ができる様になるシステム『アグリー』です。携帯電話さえ使うことができれば、記録が取れて電子データに反映されるというのは、とても便利です。

７．１　農業日誌

農業日誌には、どのような事が記録されるでしょうか？　その内容を少し詳しく考えてみましょう。

農業で必要な記録は、次のようなものがあります。

・作業記録（いつ、どの圃場で、どの作物に、何をしたか）
・注文記録（いつ、だれが、何を、いくらで、どれだけ必要か）
・出荷記録（いつ、とこに、何を、いくらで、どれだけ出荷したか）
・生育記録（いつ、どこで、なにが、どのように、育っているか）
・購入記録（いつ、どこで、何を、いくらで、どれだけ購入したか）
・薬剤記録（いつ、どの圃場で、どの作物に、どの薬剤を、どれだけの濃度・量、利用したか）
・施肥記録（いつ、どの圃場で、どの作物に、どの肥料を、どれだけの濃度・量、利用したか）
・天候記録（いつ、どのような天気、気温、風、湿度・・・であったか）
・作業実績（いつ、誰が、どこで、何時間、仕事をしたか）

ほかにも、電気や水・燃料の消費記録や、車、農業機具の使用記録など、考えるといくらでもありそうです。

このような沢山の記録は、記録を取るときに必要ではなく、後で必要になるので、どうしても記録自体が後回し（後でゆっくりと書こう、などと・・・）になりがちです。これを、その場でどんどん記録して行くのは、小型のPCや携帯電話をつかうのが一番良い方法でしょう。小型PCも、ハードディスクがシリコンディスクに変わってきたことで、防塵・防水・耐震対応型のものが出回っています。価格もそれほど高くありません。

では、記録をあとでどう使うかを考えてみましょう。

・作業記録：何にいちばん手間がかかるか、どの作物、どの場所に時間がかかっているかなどが、統計的に導き出せます。天候の記録などとあわせて評価すれば、どのような気候変化が作業の増加につながるかなども判断出来る可能性があります。

・注文記録：出荷予定が組みやすくなるわけで、逆算すれば、どのくらいの作付けが必要か、どれほどの種や苗が必要で、育てる圃場面積もわかりますから、計画が立て

やすくなります。最終的に出荷するわけですから、計画的な農業で計画的な売上につながります。

・出荷記録：これは、まさに売上に特化した記録ですから、購入記録と比較することで、収益性の把握に利用出来ます。購入記録が経費で、出荷記録が売上となりますから、会計処理も行いやすくなります。パート・アルバイトなどの雇用者が居る場合には、その人達の人件費も購入記録に含めて（労務を購入したと考える）統計すれば、毎年申告する税務関係の基礎資料も作れます。

・薬剤記録、施肥記録：安全・安心な農作物の提供は、農家の責任です。それをしっかり管理する事で、品質が保たれ、信用がうまれます。『○○さんの野菜は、安心して消費できる』と評価されることが、現代農家の至上命題です。当然、そんな評判につながれば、売上も向上するでしょう。

記録は、大切です。農業だけではなくすべての生産活動に対して大事な行動です。ですから、農業の記録の方法を考えることは、他のすべての生産活動につながる方法を考えることになります。日頃からいろいろなアイデアを考えて実践してみましょう。

７．２　ＩＴを使った日誌記録の実践

農業日誌を PC で書くとしたらあなたは、どのように書きますか。Word などを使い、文章として記録することはしないでしょう。それは、後で記録を使う事（統計処理などをおこなう事）を考えてのことだと思います。統計処理や、過去の履歴を調べるなどの処理は、情報が DB 化されていれば行いやすいのですが、農業家の方にいきなり DB による記録と情報管理を行ってもらうのは、容易ではありません。しかし、DB に近い形で項目ごとに分けて記録する事は、他の方法でも可能です。あとで使いやすい形で、かつ DB にも近い形というのは、表形式に記録することです。表計算ソフト Excel の機能を利用できれば、統計・検索などは、思ったように行えます。さらに、記録を DB 化する時、Excel は DB との親和性が高いというメリットがあります。Excel を使って表形式に記録をとることに慣れ親しんで、記録の利用・応用ができる様になって来たら、そのときに DB 化を提案すれば、いきなり DB 化を行うときの敷居がずいぶん低くなるのではないでしょうか。
また、Excel を使う事に慣れてもらっていれば、PC 等の IT 機器利用に対する抵抗感も和らぎ、入力にかかる労力よりも、そこから得られる利便性などの方が、より有益生が高いと理解してもらえるようになります。

私も過去 15 年以上、多くの営農されている方々と PC に関連したおつきあいをして来ましたが、先に PC を使い始めていて、『ワープロ』という意識で Word 使用になれていた方々より、後から PC を使い始めたにもかかわらず、Excel を最初に使い始めた方々のほうが、統計・検索やさらに、会計処理を説明するときの理解が、早いという実感を得ています。また、Word 利用に慣れた方々は、Word と Excel とを上手に使い分けようとされますが、Excel を最初に使い始めた方々は、必要なことのほとんどが Excel で実現できるので、別途 Word 利用について多くの時間を使わず、Excel 一つで対応できてしまうというメリットがあります。

Excel は、表の形に情報が記録されるので、表の一番上のセルに項目名を記入してやると、DB のテーブルと同じ形になり、オートフィルタなどでの絞り込みによる検索ができる事も理解し易いようです。セルの中に計算式を記入すれば、様々な計算にも対応でき、さらに進めばマクロの利用で機能の拡張や、専用アプリケーションとしての ExcelBook を開発できます。

MS-Office に含まれる DB である Access は、Execl にデータを出力できる様に、Office Links という機能があり、これを使えば、AccessDB 側から Excel のワークシートを作ることもできます。また、その反対に、WorkSheet を CSV 形式で保存すれば、容易に DB に取り込む（Inport）することもできます。

Excel 側から、クエリを作成・操作することで、AccessDB を代表として、ODBC 接続されている DB ならそのテーブルの内容を WorkSheet に取り込むことができます。このデータは、DB の更新に合わせ、最新の DB 状態が反映されるようになっています。

以上のことから、特別なシステムやソフトウエアを使わずに農業日誌を記録するには、Excel を使い、表の形に記録してゆく事が有益です。表の形になれば、後のデータ活用も行いやすくなるので、もし、仮に何らかのシステムを開発するとすれば、先に解説した『アグリー』のように、最終的に Excel に情報が記録されるようなシステムが最適です。

７．３　記録する項目

では、記録の実際を考えましょう。先にも説明したとおり、記録しないことには後の情報活用はできないので、何でも記録する事です。その項目は、ざっと

□日付　□時刻　□担当者　□作物　□圃場（場所）　□作業
□所要時間　□天候　□気温・・・

こんなかんじでしょうか。
これを、表の一行目の項目として記入します。作物をトマト、ナス、ピーマン　などと記録して行くことで、後にオートフィルタの機能が使えるようになり、トマトだけの記録を検索したり、集計したりできます。

また作業には、『除草』や『播種』などといった言葉でも、『草取り』や『種まき』といった普通につかう言葉でも構わないので、該当する仕事の内容を記録します。オートフィルタでは、『除草 AND 草取り』で検索・絞り込みができますから、最初はあまり堅苦しく考えずに記録する事が大切です。

少し工夫すると、作業に『出荷』、作物に『キュウリ』、数量にケース数、単価に出荷価格など、併せて場所に JA などと記録すると、後の売上・収益管理に便利です。

同様に、作業に『購買』または『仕入れ』、作物に『キュウリ』、品名に『マルチシート』、数量に購入数、単価に購入単価、場所に『JA○○経済センター』などと記録すると、経費の管理ができます。

ここまで説明すれば、後はどのようにもできる事が分かると思います。使うひとの工夫次第です。

応用は、種まきや定植の数と、収穫の数量を記録すれば、作物の効率が把握できますし、圃場の情報を記録しておけば、効率の善し悪しを圃場ごとに把握できて、次の作付け時に圃場利用計画に役立つでしょう。

【参考】実際に使用されているメールの利用図を添付しておきます。

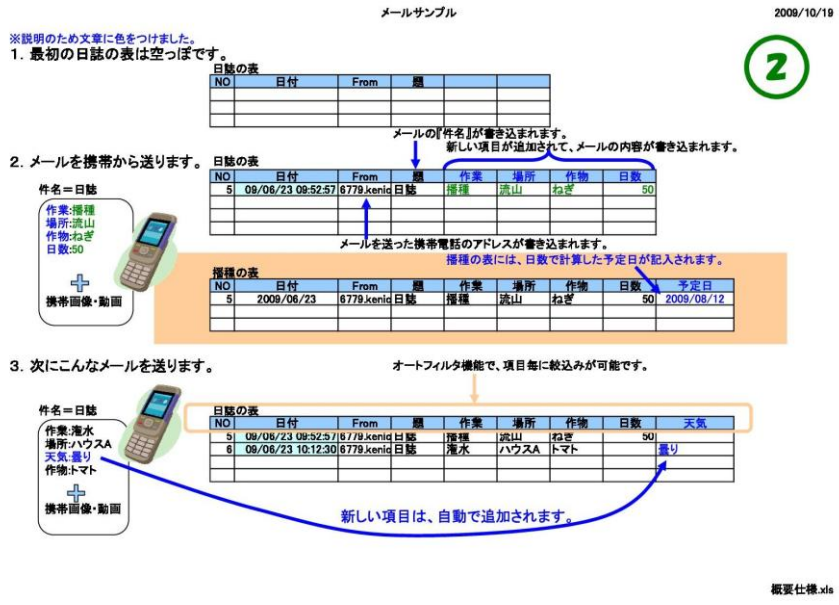


図 7.1 農業日誌の仕組み

８．農業知識のデータベース化

前章で解説した農業の記録は、農業ノウハウの塊『農 HOW』です。ノウハウは、再利用できて意味があります。再利用しやすい形にするには DB 化が必須ですが、さらに一般の方が事典として利用できれば、誰でも農業に参入しやすくなります。それには、数多くの情報が含まれている必要があります。本研究とは別に、農業 IT の研究会が開催されており、その中で前章で説明した、携帯電話を利用した農業日誌の技術を応用して、農魚に関わる DB を構築する流れがあります。具体的な内容はここでは解説できませんが、概観して見ましょう。

８．１　コンテンツ

DB は、後に利用されることを考えて、情報を記録していく必要があります。DB は最初に作った時は空ですから、必要な情報を貯め込んで行く仕組みが必要です。私たちは、これを農業日誌の記録から作ろうと考えて、日誌を書くことが農業 DB の充実につながるという仕組みを考えました。

後に検索することを考えると、１件の情報に複数の検索キーを持たせて DB に記録しておくことで、後からの利用がし易くなるようにしました。

現時点では、基本的な DB の形はできあがっていて、テスト RUN も行っています。今後、コンテンツの種類や拡充（イメージ・動画・その他様々な Object）をおこなう予定です。

８．２　検索インターフェイス

後で検索してヒットした件数がどれだけあるかが DB の有用性の尺度の一つになりますが、検索の手続きが面倒くさいと、誰にも使ってもらえません。特別な道具が不要で、容易に検索ができる方法として、日誌を記録するツールと同様に、携帯電話の利用を重点に考えて DB を構築しています。現在、単純な検索はすでに完成しており、今後コンテンツの充実と並行して検索インターフェイスの改良・改善と、ヒット結果の表現方法と内容などを充実させる計画です。

詳細を説明できませんが、ある研究会で試作されている農業 DB の検索画面の一部を参考に掲載します。

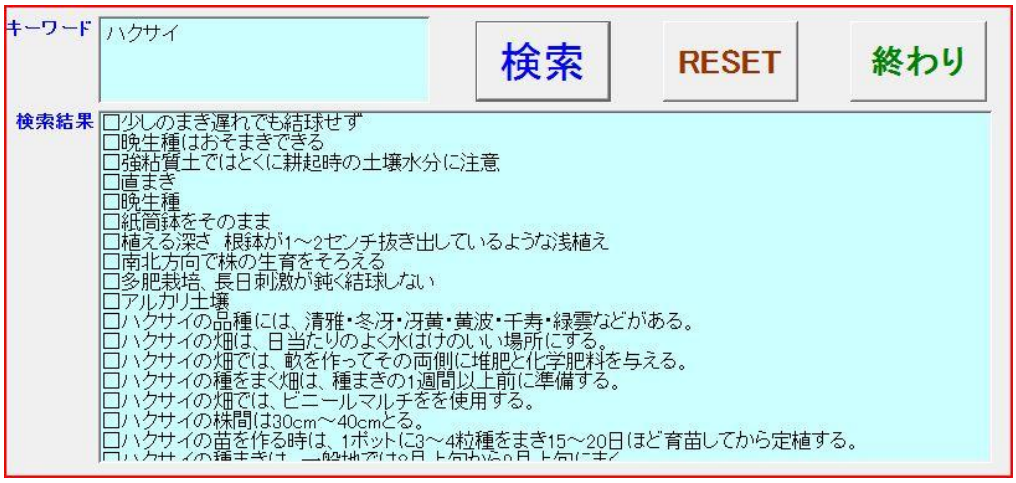


図 8.1 検索画面

このように、すでに DB は構築され、実用域に達しています。実際に使いながら、フォローを行い、コンテンツの充実を図っています。

８．３　実例

次の図は、ある地域の農業関連事業の目論見図です。
その中に、『エキスパートシステム DB』が組み込まれていて、プロも農業家も一般の方も広く使える仕組み作りがすすんでいます。

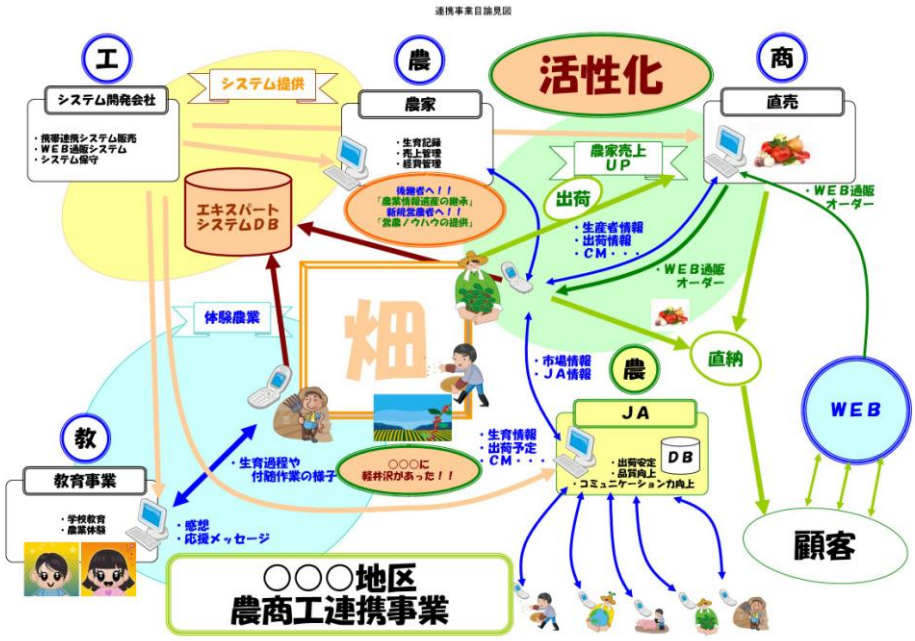


図 8.2 農業関連事業の目論見図

９．農業従事者とのおつきあい

農業を生業にしている方々は、好奇心が旺盛で常に積極的です。いろんな事に対して良く質問されますし、後日時間のあるときに再び質問されたりもします。それは、向上心の表れだと思いますので、根気強くいろいろな方向から物事を解説してあげることが大切です。

私は立派な IT 技術者は IT 技術を理屈で理解していることだけでは不足で、実践はもちろん、加えて相手の聞きたいこと・問題点・ニーズなどを引き出す手法として『インタビュー技術』が必要だと考えています。相手の考えを相手任せで伝えてもらうだけでは不十分で、こちらから引き出す（聞き出す）技術・テクニックが必要だと思います。農業家の皆さんは IT 技術や用語に不慣れな反面、そのことが役立つと思うと徹底して理解しようと努めてくれます。そのような場面で頼ってもらえるのは大きな喜びですが、自分自身が伝えようとしたことが正しく理解されたかどうかを判断したり、どのような点が理解されていないか等を知ることができなければ、無駄に同じ説明を繰り返すことになり、不毛な時間が過ぎてしまいます。『効率よく、相手の知りたいことを引き出し、乾いた土に水がスッと吸い込まれるような』説明ができること。このことが、IT 技術を進める下支えになっていると強く思います。これを常に実践してトレーニングし続ければ、農業だけではなく、どんな対象にでも善戦できる IT 技術者になれると考えます。

私たちに多くの事を教えてくださる農業家の方々は皆さん、次のような方達です。

力強い。
好奇心が強い。
向上心が強い。
意外に面倒くさがり。
いろんな意味でパワフル。
自然に対する自然な姿勢。
誤りを引きずらない考え方。
地域的なつながりを大切にする。
しきたり、慣行、慣例を大切にする。
多少の力業でも、良い方向に進もうとする行動力。
早朝・深夜・カレンダーなどにとらわれない行動力。

・・・この程度の言葉で、すべての農業家の方を表現することはできませんが、逸れてはいないと思います。

このテキストを執筆している２０１１年元日の夕刻、やはり農業家の方から電話があり

ました。新年の挨拶をすると、電話の相手がいきなり知らない人に変わりました。いろいろ話を聞くと、どうやら自分の PC が不具合なので、知人宅にやってきて PC を使おうとしたら、思った様に使えないので、私の知り合いの農業家の方に電話を掛けてもらい、質問をしたかったようです。

年末までに原稿が仕上がらなかったので、元日から仕事をしていた私には、問題無かったのですが、これからやってきそうな勢いもありました。辺りにも他の人が居る気配がしていて、流石に元日から押しかけるのはどうかと云われたらしく、電話で済みましたが、翌日の早朝、自宅を訪ねてきました。

『自分の知り合いが知りたいことがあるので、チョット電話を替わります。』などと、最初に言葉があると、びっくりもしませんが、このような事がけっこうな頻度で起こります。日時にこだわり無くおつきあいができる様な大きな気持ちと、熱い気持ちを常に持って、多くの農業家の方々と交流が深まると、自分自身も大きくなれるように思います。

これからも、パワフルな農業家の方々のお役に立てるような IT 技術者でありたいと思います。

１０．資料編

１０．１　センサー

現在、たやすく入手できるセンサー

□ アナログ温度センサー

LM35DZ

LM60BIZ

LM61CIZ

LM19CIZ

□ デジタル温度センサー

LM74CIM3NOPB　(SPI：3.3V 動作)

LM74CIM5NOPB　(SPI：5V 動作)

□ 湿度センサー

HS-15P

□　照度センサー

AMS302T (パナソニック電工)

NJL7502L (新日本無線)

□　土壌水分センサー（完成ユニット）

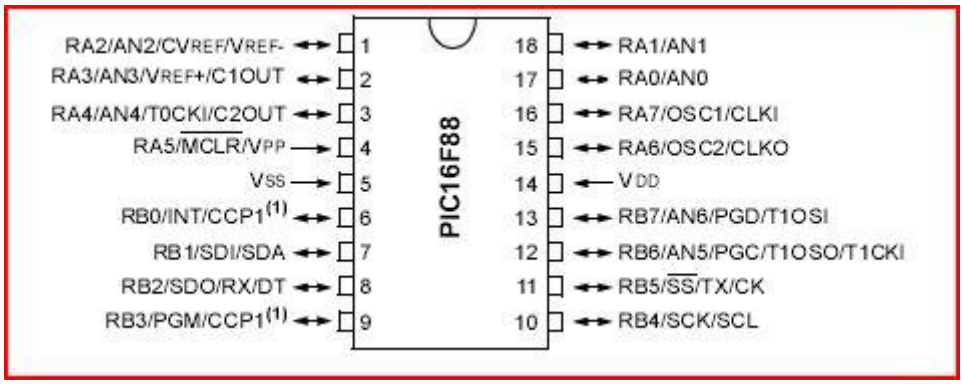
取り扱い会社：アオネクス株式会社

取り扱い会社：タカギ

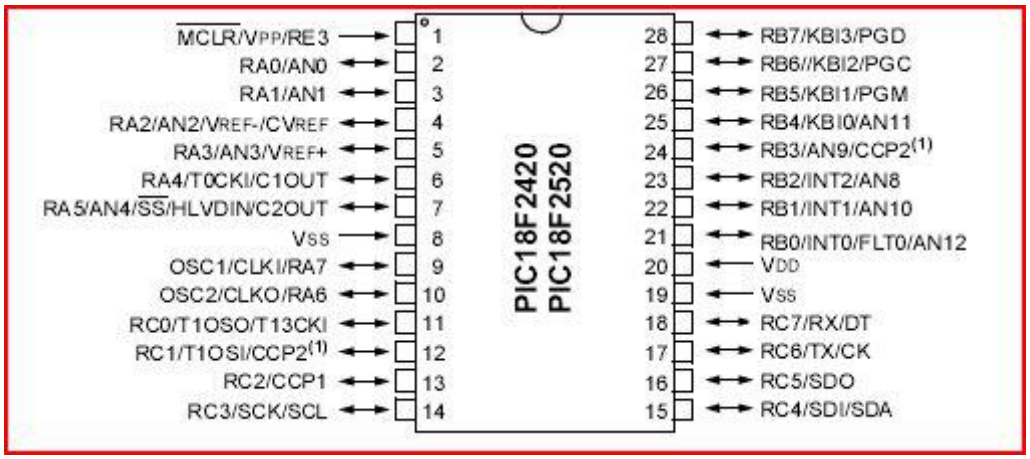
１０．２　マイコン

PIC データシート抜粋：ピンダイアグラム

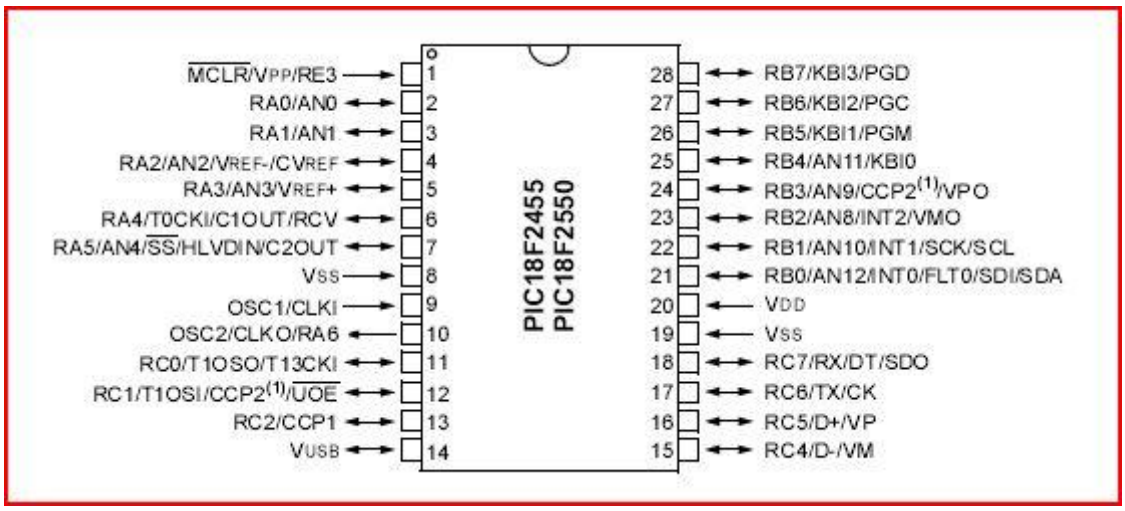
PIC16F88



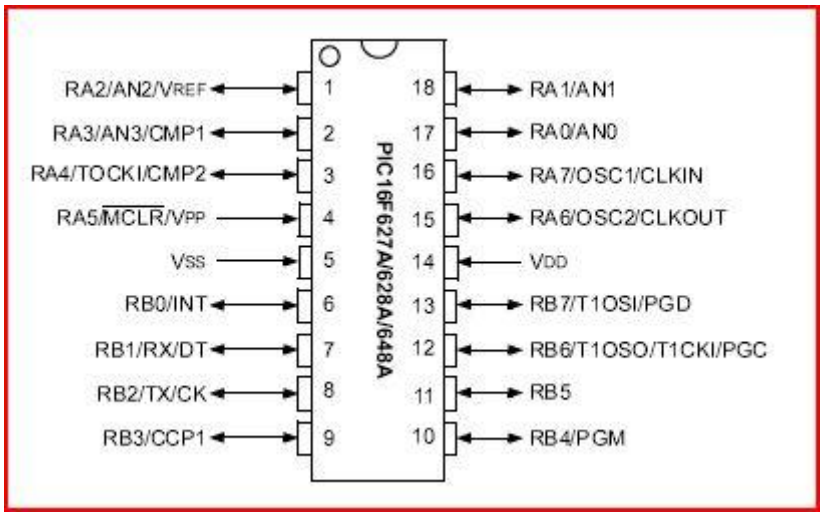
PIC18F2520



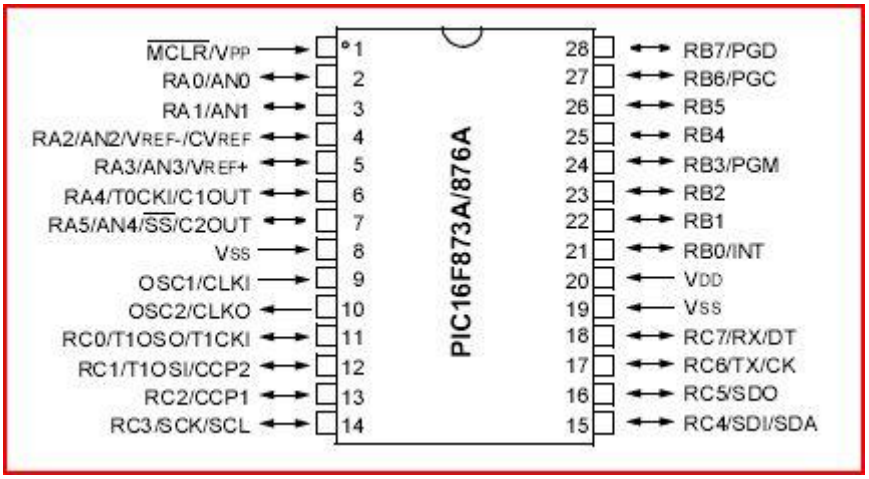
PIC18F2550



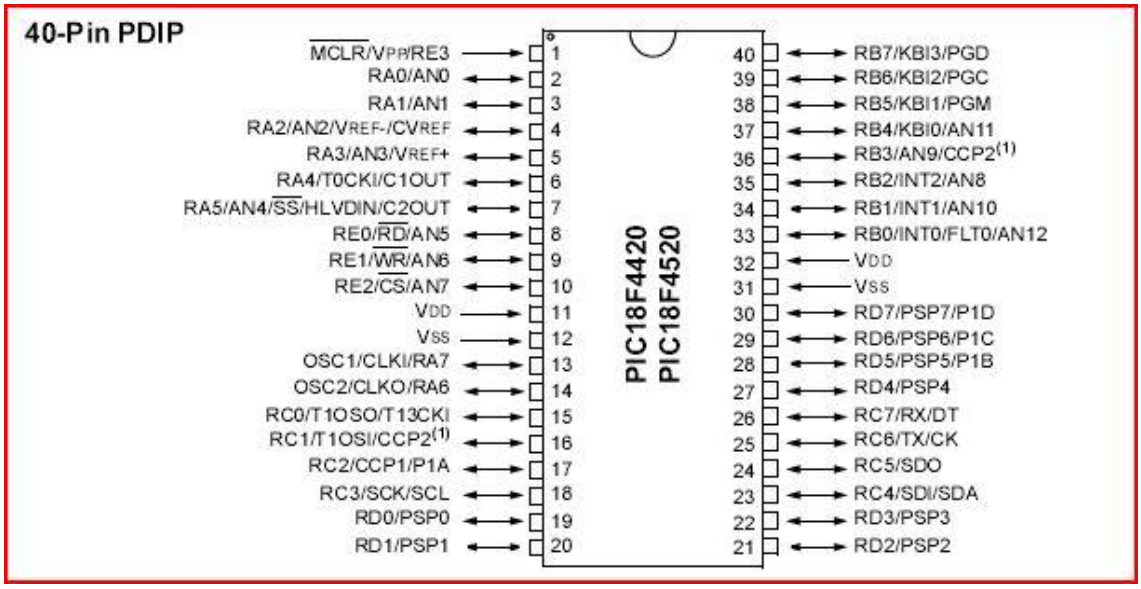
PIC16F648



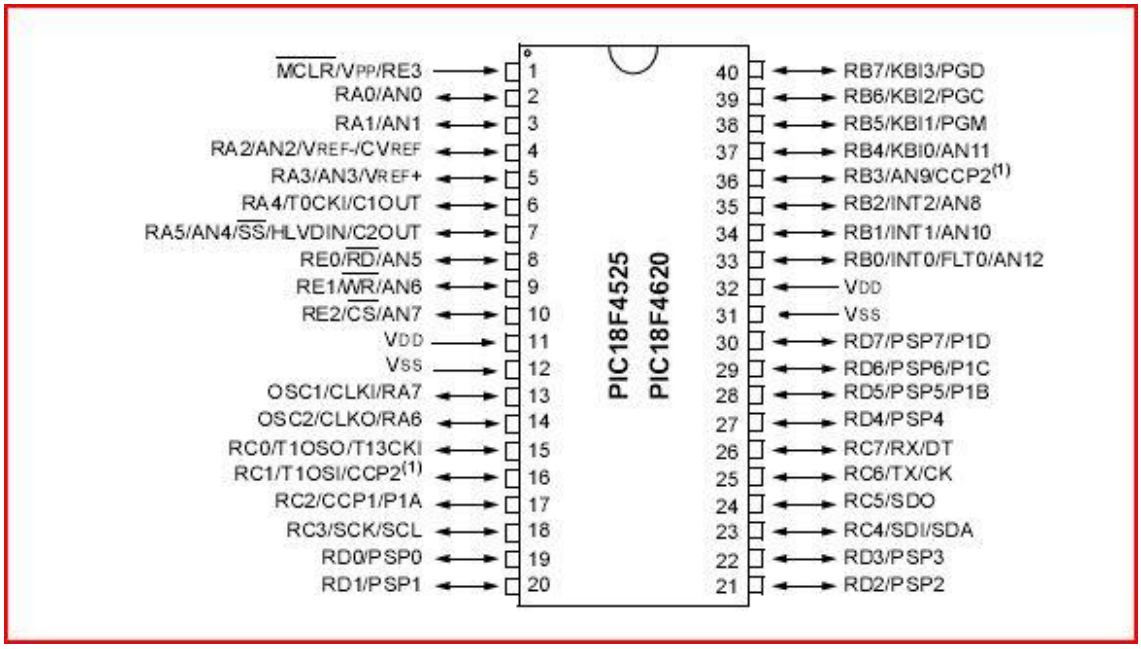
PIC16F873



PIC18F4520



PIC18F4620



１０．３　衛星リモートセンシング

衛星リモートセンシングは、地球を周回する人工衛星から、いろいろな波長の光で地上を撮影し、地球の状態を調べる技術です。農業では、地域的な土地の状態、作柄やそのときの気温・気象の状態などを調べることができます。地上にあるものは、地面も含めて反射光の波長が分かっているので、調べたい物に特有の波長を通すフィルターを掛けて写真を撮ることで、作物の地域的な広がりや生育状態、土地の状態などがわかる仕組みです。

衛星による写真撮影が基礎データですので、ピンポイントでの調査も可能ですが、費用が高価ですから、広い範囲での状態把握の必要な時に利用します。

衛星は雲の遙か上に位置していますので、気象の状態によっては、調べたいときに地上が見えないこともありますので、大きな情報が得られる反面、タイムリーにならない場合もあります。一度撮影タイミング逃してしまうと、もう一度同じ位置に衛星が戻るまでに時間がかかることもあり得ます。

現在では、リクエストに応じた撮影だけでなく、フリーの公開されたデータを利用する方法もあるようです。画像データを処理するソフトもフリーでいくつか公開されています。

実は、私は天文の研究を行っていて、以前から公開天文台の望遠鏡で遠くの星の写真を撮影しています。この写真は、写真として眺めるのではなく、明るさや色やスペクトルを取得する目的の写真です。この時の処理の方法・手法が、リモートセンシングと全く同じ

なのです。地球から宇宙を眺めるのと、向きが違うだけで、行う事は大変似ています。後は、取得したデータから何が読み取れるか、ということですね。

ここに、参考文献を示しておきます。

① リモートセンシング読本（CD 付き）：日本測量協会：岩男 弘毅 著

② 独習リモートセンシング：森北出版：新井 康平 著

いずれも、難しい本ではありませんが、①が入門としては適当だと思います。

JAXA も WEB にて『リモートセンシング基礎講座』を開講しています。

http://www.eorc.jaxa.jp/hatoyama/experience/rm_kiso/top.html

１０．４　GIS

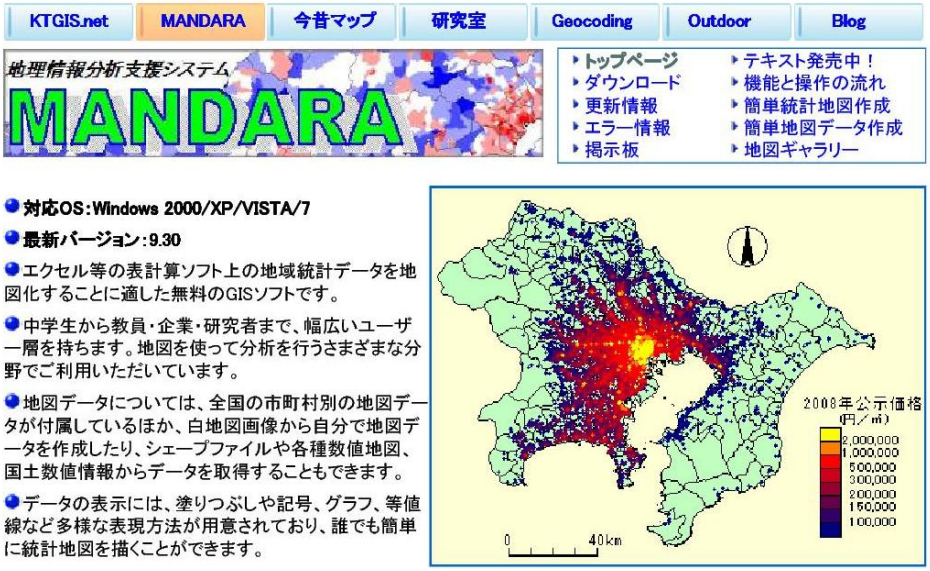
GIS というのは、Geographical Information System の頭文字を取ったもので、日本では『地理情報システム』と呼ばれています。

地理情報システムとは、様々な地図を作成するシステムのことです。カーナビが一般に普及して、GPS が一般に理解される様になってきた現在、広い土地を利用する農業にも、

同じように様々な土地利用図や計画図などを作る必要が出てきています。このような時に、利用出来るシステムとして一般公開されていて誰でも使える GIS として『MANDARA』というソフトがあります。この『MANDARA』は Excel と連動させてさまざまな統計分析を行い、その結果を MANDARA で白地図上に色を変えてグラフィカル表示したりすることができるものです。

地域的な作物の収穫量に対応した地図や、人口密度と消費量との関係などを容易に作成し、各種分析をおこなえます。

リモートセンシングと併用することで、土地の状態、利活用、農業に関するさまざまな情報を平面の広がり地図として表示できる技術です。WEB の抜粋と、URL を下記しておきます。



ＵＲＬは

http://ktgis.net/mandara/

また、参考書籍は次の通りです。

① GIS 講座：古今書店：後藤 真太郎・谷 謙二・酒井 聡一・加藤 一郎 著

１０．５　電子部品の入手先

下記の入手先は、有名なところです。可能であれば、一度店舗に出かけてみると大変勉強になりますし、意欲も湧いてきます。

秋月電子通商：http://akizukidenshi.com
マルツパーツ館：http://www.marutsu.co.jp/
千石電商：http://www.sengoku.co.jp

１１．キーワード集

□ 農業のフェーズ

□ 生産品目

□ ＩＴ技術

□ 農地活用計画

□ 灌水タイマー

□ マルチング

□ マルチ

□ 種まきトレイ

□ ポット

□ 誘引

□ 定植

□ 播種

□ わき芽かき

□ トンネル

□ 寒冷紗

網目状の薄い布。遮光・防寒・防虫用などに使われる。目の粗い布で黒と白のほか銀色なども出回り、用途に応じて使い分ける。主に強光線を遮るためや、防風、防寒に利用。

□ SPI：Serial Peripheral Interface

□ IIC：Inter-Integrated Circuit(I2C)

□ RS-232c：

□ ソケット通信

□ TCP/IP

□ オームの法則　E=IR　（電圧＝電流×抵抗）

□ オペアンプ　オペレーショナル・アンプ　作動増幅器

□ PIC：Peripheral Inteface Controler　アメリカマイクロチップ社が製造販売しているマイコン。安価で種類が多く、入手しやすい。

□ CO2 濃度　計測の用途

フィールドサーバー導入による効果を実感している。以前から課題としていたハウスの換気タイミングが、二酸化炭素濃度のグラフをチェックすることによって、正確に分かる

ようになった。「冬場はハウスを閉め切っているので、二酸化炭素濃度は深夜から明け方にかけて上昇し、光合成が始まる日中にかけて下降します。下降しきる前に炭酸ガス発生装置を稼働させればならないのですが、そのタイミングは日によってまちまち。グラフによってそのポイントがはっきり分かるようになったので、作物にも良い影響が出ていると思います」（WEB より）

□ 平藤 雅之 氏：農業研究センター 研究情報部 ： フィールドサーバー研究者

□ フィードサーバ：

Web サーバ、複数のセンサ、ネットワークカメラ、無線 LAN 通信モジュール、超高輝度 LED 照明など様々な電子機器を搭載し、 フィールド（圃場）に長期間設置して、環境の計測、動植物のモニタリング、農園の監視等を行う超分散モニタリングデバイスです。

１２．参考文献

□ 農を楽しくする人たち：ダイヤモンド社：週刊ダイヤモンド&嶺 竜一

□ マイコンの１線２線３線インターフェース活用入門：CQ 出版社：中尾 司 著

□ 月刊現代農業：農文協

□ 研究室ですぐに役立つ電子回路：工学図書：阿部 寛 著

□ PIC を使ったデータ・ロガーの制作：CQ 出版社：稲崎 弘次 著

１３．参考 URL

□ フィールドサーバー関連

http://model.job.affrc.go.jp/FieldServer/

http://www.blwisdom.com/btrend2/06/2.html

http://www.elab-experience.com/products

http://www.iwatou.net/fs/index.html

http://kankyou.cs.shinshu-u.ac.jp/agri/index.php?as09

http://www.nca.or.jp/shinbun/20080411/keiei080411_01.html

□ WEB 上の記事

（NTT 東日本での記事）

http://www.ntt-east.co.jp/business/column/agri/005/index.html

（NTT コムウエアでの記事）

http://www.nttcom.co.jp/comzine/no086/newdragnet/index.html

平成２２年度文部科学省

産学連携による実践型人材育成事業

―専門人材の基盤的教育推進プログラム―

「農業を対象分野とする IT コンサルタントを目指す人材の育成プログラムの開発と実施」

指導書

平成２３年３月

学校法人三橋学園

船橋情報ビジネス専門学校