マイコンボードを使用した
組込みプログラミングの基礎と実践

第 1.2版: 2020/06/19（内容は2015年当時）

contact@flogics.com

FIRMLOGICS

Embedded Design and Partnerships

License Agreement

- This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
  http://creativecommons.org/licenses/by-nc-sa/4.0/

班分け

- 以下のように班分けさせて頂きました
  （深い意図は、たぶんありません!）

1班:

2班:

3班:

4班:

5班:

今日の予定

- 13:00 頃～
  - オリエンテーション
  - 自己紹介
- 13:30 頃～
  - Arduino について
  - 部品の入手方法
  - Arduino 開発環境の導入
  - Hello World
- 14:30 頃～
  - Arduino プログラミング
  - デジタル回路の基礎
- 15:30 頃～
  - ハンダ付け工作
  - サンプルプログラムの導入
  - Q&A
- 17:30 頃
  - 予備時間
  - 解散

自己紹介タイム

- 皆さんに簡単に自己紹介して頂きます!
- 時間はおおむね 1分です
- 含めて頂きたい内容は以下の通りです
  - お名前
  - 「私はこんな人です」
  - 趣味、あるいはライフワーク
  - 最近興味を持っていること、取り組んでいること

事前準備編（必要な道具と部材）

- 今回の実践編（工作）では、以下の道具および部材が必要、あるいはあると便利です。次ページからは、必要度、市場価格など詳しい情報を含めて説明します。
  - ハンダごて, ハンダごて台
  - ニッパ
  - ラジオペンチ
  - ピンセット
  - ワイヤストリッパ
  - 老眼鏡（ハンズフリー虫眼鏡）
  - デジタルマルチメータ（DMM）あるいはテスター
  - ノートパソコン, 標準 USB ケーブル
  - ハンダ（鉛フリー）
  - リード線
  - ハンダ吸取線
  - 養生テープ, マスキングテープ

道具1: ハンダごて

- まず必要なのが「ハンダごて」です。
  - ブレッドボードという道具を使うと、必ずしも「無くてはならない工具」ではありませんが、一台持っていて損はないと思います。
  - 昔の 15W 以下くらいのコテをお持ちの場合、鉛フリー対応に買い換えたほうが良いかも知れません。
    参考:
    http://www.hozan.co.jp/catalog/Soldering_Tools/Solder_Flux.htm
  - 必要度: ★★★★★
  - 共用できる度: ★★☆☆☆
  - 個人的なおすすめ
    HAKKO PRESTO 984-01（ボタンを押すと130W になり、コネクタやベタ面の作業が楽。キャップも便利。2,000円くらい）

道具2: ハンダごて台

- ハンダごては高熱になるので、台があると便利です。（というか、プロには必須）
  - 使い勝手からは、スタンド型が便利ですが、たまにしか使わない人には邪魔なので、携帯型というのもあります
  - 必要度: ★★★★☆
  - 共用できる度: ☆☆☆☆☆
  - 個人的なおすすめ
    ホーザン H-11（写真下。蓋ができるので安心。1,000円くらい）

おまけ: ハンダごて台のスポンジ

- ハンダごて台には、耐熱性のあるスポンジ（海綿）が付属しています。台所スポンジは耐熱性がないのでダメです。（溶けます）
- スポンジはそのままだと使いづらいので、プロは切ったり、切れ目を入れたりして使います。（以下、ホーザン様のサイトより引用 → 削除）

道具3: ニッパ

- これも、お持ちの方は、それで良いです。もし新たに購入する場合は、電子工作には小型のものが使いやすいです。
  - 必要度: ★★★★★
  - 共用できる度: ★★★★☆
    プロは刃先にこだわるので注意しましょう。(^^)
  - 個人的に使ってるもの
    ホーザン N-31（刃渡り 10mm くらい。1,500円くらい）

道具4: ラジオペンチ

- すでにお持ちの方は、それでも良いです。もし新たに購入する場合は、電子工作には小型のものが使いやすいです。
  - 必要度: ★★★★☆
  - 共用できる度: ★★★★★
    貸し借りしても良いと思います。
  - 個人的に使ってるもの
    ENGINEER PS-04（くわえ長30mmくらい。2,000円弱）

おまけ: お金持ち(?)向けセット

- いちいち道具を買いそろえるのはメンドーだな、という方には、こんなセットもあります(これはホーザン社製)。
- ケースに入っているので見た目が綺麗というメリットがあります。ただし、余計な工具が多すぎという話もあります。

道具5: ピンセット

- これも、お持ちの方はそれでも良いです。もし新たに購入する場合は、先の尖ったものが使いやすいです。
  - プロはピンセットを大切にするので、貸し借りしないほうが無難です。(^^)
  - 必要度: ★★★★☆
  - 共用できる度: ★★★☆☆
  - 私がもし次に買うなら…
    ホーザン P-670（完全非磁性。チップ部品を扱いやすいと思う。4,500円くらい）
  - 安いのであれば、1,000円くらいであるのではないでしょうか。

道具6: ワイヤストリッパ

- 電子工作以外には用途が思いつかないのですが、この工具は必要だと思います。ニッパで被覆を剥くのは、ちと無理と思います。
  - ただし、貸し借り可能と思います。今回は無理に買わなくても良いかも知れません。
  - 必要度: ★★★★★
  - 共用できる度: ★★★★★
  - 個人的に使っているもの
    ホーザン P-963（3,000円弱くらい）と P-960。一つだけ買うなら、P-967のほうが利用範囲が広いかも。

道具7: 老眼鏡（ハンズフリー虫眼鏡）

- 講師も数年前には予想だにしなかったことですが、寄る年波には勝てず、近くの小さなものが見えにくくなりました。
  - 手をふさぐことのない、右写真のようなものがお勧めです。両眼だと立体的に見えるので作業がはかどります。この製品は米国 Texas Instruments 社の技術者が使っているのを見て、講師も購入しました。
  - 必要度: 目と年齢に応じて
  - 共用できる度: ★★★★☆
  - 個人的なおすすめ
    OptiVISOR DA-10（4,500円くらい。DA-7 でも良いかもです。安価なプラスチックレンズ版もあります）

道具8: デジタルマルチメータ

- デジタルマルチメータ（DMM）かテスタは、あったほうが良いと思います。
  - 今回は講師と学校で用意できると思いますので、無理に買わなくて良いです。
  - 必要度: ★★★★☆
  - 共用できる度: ★★★★★
  - 個人的なおすすめ
    安価な DMM なら、サンワ CD771（7,000円くらい）とか。導通が見られればいいだけなら、アナログテスターがホームセンターで 1,000円くらいからあると思います。

道具9: ノートパソコンと USB ケーブル

- 現代の電子工作にパソコンは必須と言えます。
  - 部品の手配、回路図の作成、回路シミュレーション、プログラミングなどに欠かせません。
  - あと、マイコンボードを繋ぐのに USB 2.0 標準ケーブル（mini, micro は不可）が必要です。
  - Windows でも Mac でも良いと思います。講師は Mac をメインに、ときどき Windows も使います。
  - 必要度: ★★★★★
  - 共用できる度: ★★★★☆

材料1: ハンダ（鉛フリー）

- ここからは、「ねじ釘」に相当する工作の材料（部材）です。
- ハンダは、かつては鉛とスズの共晶ハンダが用いられましたが、環境負荷が重いので「鉛フリー」に取って代わられました。
  - 鉛フリーはハンダ付けが難しいと言われますが、最近のハンダごてを使えば問題ないはずです。特に家庭にお子さんや小動物がいる方には、毒性の低い鉛フリーを強くお勧めします。
  - 必要度: ★★★★★
  - 共用できる度: ★★★★★
  - 個人的なおすすめ
    ホーザン HS-361 （11グラムで 1,000円くらい。銀が入っているので高いのです!）
    太さは 0.6ミリくらいで、Sn-3Ag-0.5Cu が良いと思います。（詳しくないが）

材料2: リード線

- これも「ねじ釘」に相当します。部品の間をつなぐ電線ですね。
  - より線はヒゲが出たりして使いづらいので、「単線のラッピングワイヤ」と呼ばれるものがお勧めです。私はAWG26という太さのものを使っています。やや太めですが、最近はマイコン工作でバス配線とかしないので、これくらいが使いやすいと思います。
  - 必要度: ★★★★★
  - 共用できる度: ★★★★☆
    たくさん買ってもしようがないので、今回は共用します。
  - 個人的なおすすめ
    というか業界標準は、潤工社のジュンフロンETFE電線だと思います（独断。50メートルで1,800円くらい）。

材料3: ハンダ吸取線

- ハンダ付けを間違えた際、ハンダを取り除くのは意外と大変です。
  - はんだ吸取線は持っていたほうが良いでしょう。ほかに、バキュームで吸い取るものも売られています。
  - 必要度: ★★★★☆
  - 共用できる度: ★★★★★
    講師が持参します
  - 個人的なおすすめ
    ホーザン HS-380-1.5（300円くらい）

材料4: 養生テープ, マスキングテープ

- ハンダ付けをしていて必ず思うのは、「ああ、もう一本手があればいいのに」ということです。専用の固定台とかあります。
  - 養生テープ, マスキングテープが意外と便利です。リード線を持ちながらハンダ付けすると手が熱いですが、テープで固定すれば OK です。
  - 必要度: ★★★★☆
  - 共用できる度: ★★★★★
  - 個人的なおすすめ
    ホームセンターで買いましょう。今回は講師が持参します。

その他

- 今回の工作会で、あると良いもの
  - あらかじめ Arduino 開発ツールをインストールしておくと良い（講師は USB メモリで持参します）
  - 赤外線の見えるデジカメ（最近の iPhone などは見えなくなっているのでダメです。赤外線リモコンのボタンを押しながらデジカメのファインダーを覗き、赤外線が出ているのが分かるものがあれば持参すると便利です）
  - 何か赤外線の送信できるリモコン。できるだけ単純そうなものがお勧めです

まずはじめに

ARDUINO で何を作ろう

Arduino ってなに? (1)

- 最近は、高性能なパソコンやスマートフォンが身近になり、初めてプログラミングをする学生さんも、それらを使ってプログラミングを覚えることが多いと思います
- しかし、次のようなことを思ったことはないでしょうか?
  - なんでコンピュータってこんなに複雑なんだろう? 学校で理論を教わっても、現実のパソコンの仕組は理解できないしょ?
  - プログラムで main() 関数を書く方法は分かった。でも、どういう仕組でこのプログラムがコンピュータ上で動作するのだろう?
  - パソコンを使えば、インターネットやクラウドと連携するプログラムはすぐに書ける。でも、実際に身近な機器を制御するようなプログラムはどうやって書いたらいいのだろう?

Arduino ってなに?（2）

- パソコンやスマートホンが、現在のようにブラックボックス化したコモディティとなってしまったことで、コンピュータの基本的な動きを勉強する機会は、かえって減ってしまったように思います
- そのような中で近年、非常に安価かつ小型な Linux コンピュータ Raspberry Pi や、ワンボードマイコン Arduino が注目されています
- ちょうど、UNIX OS が複雑になりすぎたことを嘆き、1990年ころに教育用 OS MINIX をタネンバウム教授が作り出したことと似ているかも知れません
  - MINIX と違うのは、Raspberry Pi も Arduino も十分パワフルで、そのまま実用製品に応用することができる点です
  - 半導体技術の進歩が、そのような高性能を実現しているとも言えます

Arduino ってなに?（3）

Arduino

Raspberry Pi

Arduino ってなに? (4)

- Raspberry Pi も Arduino も安価なので、子供のお小遣いでも買うことができます。昔(1970～80年代くらい)、ロジック IC 等を秋葉原で買ってきて電子工作をすることが流行りましたが、そのブームに少し近いと思います
- 一つ違うのは、当時にはなかったインターネットがあることで、情報が世界中に素早く広まり、たいていの応用例が、ネット上ですぐに見つかることです
  - Make: Magazine
  - Maker Faire Tokyo 2015
- 最近の子供たちにとって、パソコンやスマホ上でゲームが動くのが当たり前になってしまいましたが、まずは大人が最近の潮流を知り、子供たちには電子/機械/情報技術の楽しさを知って欲しいと思います
- 学生さんには、大学で教わる理論を、パソコンだけでなく「このようなおもちゃ」でいろいろ実験し、技術を身につけて貰いたいと思います

補足: でも、おもちゃでしょ?

- 過去にわれわれが親しんだコンピュータと、どれくらいの性能差があるのでしょうか? “Lies, damned lies, and benchmarks”（嘘があり、大嘘があり、そしてベンチマークがある）というジョークがありますが、とりあえず数字だけ持ってきますと…
  - VAX 11/780: 1 DMIPS
  - Sun SPARCstation 1+: だいたい 15 DMIPS と言われている
  - Pentium P5-133: 180 DMIPS くらい
  - Arduino: 5 DMIPS くらい（誰かが調べたらしい!）
  - Raspberry Pi 2: 1500 DMIPS くらい
  - Intel Core i7: 10000 DMIPS くらい
- というわけで、おうちサーバーくらいには十分に使えます（何しろ、低消費電力!）

ネット対応の学習リモコンを作ろう(1)

- さて、今回は Arduino を取り上げたいと思います。Raspberry Pi は Linux で動くので、Linux が分かる人にとっては小さなパソコンに過ぎません。(Raspberry でも機器制御はできます)
- 一方 Arduino はマイコンなので、その上で動作するプログラムを、一からすべて理解することも困難ではありません
  - Arduino に搭載されている Atmel 社のマイコン IC のマニュアルは、400～500ページに過ぎません(!?)。Windows や iOS の開発ドキュメントを読むことに比べたら、実に簡単です。(実際にはすべて読まなくても、PDF ファイルを検索して必要なところだけ読めば良い)
  - Arduino で工作をするには、あとは簡単なデジタル回路理論、C言語の使い方、そしてハンダ付けの方法が分かれば OK です

ネット対応の学習リモコンを作ろう(2)

- 先に述べたように、Arduino 工作の醍醐味は、プログラミングにより身近な機器を制御できることにあります。たとえば、
  - LED (豆電球)の点滅
  - モーターの動作, ロボットの制御
  - センサー(温度, 明るさ, 振動, 加速度, GPS など)のデータを取得
- そのために、既存の製品を買ってきて USB などで繋ぐだけでも実現できるかも知れませんが、ハンダ付けができると、もっと自在な工作ができるようになります
  - ネット上に溢れる回路例を参考にすれば良いので、電子回路を一から設計できなくとも大丈夫です (みんな真似をして覚えるのです!)

ネット対応の学習リモコンを作ろう(3)

- 前置きが長くなりました
- 今回は、Arduino で家電製品の制御ができる学習リモコンを作りましょう
  - あとは、Arduino をうまくパソコンや LAN に繋げられれば、家の外から部屋の照明をオン/オフすることだって可能です
  - 重要な注意: くれぐれも、暖房器具や電熱器、そのほか人体や建造物、ペットなどに影響を与えるような遠隔制御はしないよう、お願いします（講師は責任を負いかねます）

赤外線リモコンの仕組(1)

- 家電に多く使われる赤外線リモコンは、その名の通り、発光部にある赤外線 LED(発光ダイオード)を点滅させて、指示を相手に送ります
- デジカメのファインダーで覗くと、点滅が見えます。(光学ファインダーは無理です)
  - 最近の iPhone などでは、フィルタがかかって、見られなくなったようです。(残念)

赤外線リモコンの仕組(2)

- フォーマットは次のようになっています(一例)
  - Electronic Lives Manufacturing 様のサイト から引用

T は 400～500us (マイクロ秒)くらい

赤外線リモコンの仕組(3)

- 送信側については、このパターンで赤外線 LED を点滅させればOK です
  - 正確に点滅させるには、38kHz の倍である 76kHz の「割込」を発生させ、そのタイミング毎に LED を on/off すれば良いです
  - もちろん、これはサブキャリアですので、そこにさらにビットパターンの変調が必要です
  - 余談ですが、前ページの資料を見ると、点滅のデューティ比は 1 : 2 が正しいようです。私は知らなかったので 1 : 1 で作ってしまいましたが一応動いています。1 : 2 への変更と動作比較は、皆様への宿題とします

赤外線リモコンの仕組（4）

- 受信については、専用の受信デバイスがあるので、それを利用します
  - と言っても、このデバイスは 38kHz サブキャリアの復調（38kHz 点滅の取り除き）と、信号の整形しかしてくれません
  - つまり、次のような信号が得られるだけなので、それを 1/0 に戻すには、自作のプログラムで実現することになります

  - 私のサンプル（?）プログラムでは、この信号のオン部分とオフ部分のタイミングを計算していき、最終的に 1/0 信号に戻します

プログラムについて

- 言い訳: 今回の講義の主題は、あくまでも Arduino の使い方とハンダ付けですので、リモコンコマンドの解析アルゴリズムまでは詳説しません。（皆様のソフトウェア技術を投入し、私のサンプルプログラムよりも優れたコードを実装いただければ幸いです）

まずは部品がないと始まらない

さっそく部品を入手しよう

必要な部品

- 道具とハンダ付け部材については最初に説明しましたので、ここでは今回の工作に必要な部品について説明します
- 必要な部品を大きく分類すると、次のようになります
  - Arduino マイコンボードと、その周辺パーツ
  - 今回作るテーマのための電子部品
- 最近は、いろいろなネットショップで部品を購入できます。まずは、上記それぞれについて、部品を購入できる店を紹介しましょう

秋葉原に足を運んで購入する(1)

- 昔は、関東近郊ではこのような方法が主流でした。一方、大阪では日本橋(にっぽんばし)に電子部品を扱うお店が多くあります(ありました)
- 地方在住の方は、「トランジスタ技術」や「ラジオの製作」といった雑誌に掲載された通信販売を利用されたと思います
- もし秋葉原が近い場合は、次のようなお店を回ってみるのはいかがでしょうか?
  - 最近はネット通販が便利なので、送料をケチるくらいしか役立ちませんが、急ぎの場合には便利です。また、一度このようなお店を覗くことは勉強になると思います

秋葉原に足を運んで購入する(2)

- マルツエレック
  - 最近、大きくなってきたと思われる電子部品屋さん。部品の展示方法が面白い。流行の部品はなんでも扱っていると思われる
- 千石電商
  - 昔からの老舗。電子部品だけでなく、配線材とかケースとかが豊富
- 秋月電子通商
  - 同。秋月の電子キットは昔から有名。(役に立つ部品が多く売られていて、電子工作ファンで知らない人はいない!?)
- 秋葉原ラジオセンター, 秋葉原ラジオデパート
  - まるで市場のように小さなお店が列をなしていて、初めて見る人はカルチャーショック間違いなし。近年は元気がない

ネットショッピングで購入する

- 店によって得意不得意があるので、個人的な経験から列挙します
  - Arduino, Raspberry Pi 関連
    - スイッチサイエンス社 が便利だと思います
  - 一般的な電子部品
    - 共立エレショップ が便利だと思います。配送オプションが豊富です
  - ちょっと特殊な部品
    - 型番は分かるんだけど、共立では売ってないとき。マルツオンライン が良いかもです。
  - マニア好みの部品、風変わりなキット
    - やはり 秋月電子通商 でしょうか
  - ケースとか
    - タカチ社製ケースなどは、せんごくネット通販 をよく使います。配線材なども豊富です
  - 本気で入手難な部品、特に外国製 IC など
    - RS オンラインや Digi-Key が有名です。個人相手でも売ってくれると思います。Digi-Key は送料が高めですが、一定額以下では消費税がかかりません

それでは実作業に入りましょう

マイコンボード ARDUINO の導入

Arduino をパソコンに繋ぐ（1）

- Arduino マイコンボードの箱を開けると、中には電子基板が一枚納められています。それだけです
- Arduino を動かすには電源が必要ですが、3つの方法があります
  - パソコンや USB 電源（あるいはモバイルバッテリー）に USB ケーブルで接続する。なお、通常の Arduino には USB 標準ケーブルが必要です。Mini や micro ケーブルではありません。なお、互換ボードなどではまた異なります
  - AC アダプタを別途購入して接続する（最近は USB 電源が便利なので、あまり利用されないかも知れません）
  - 独自に電源回路を設計して、それを接続する

Arduino をパソコンに繋ぐ（2）

- Arduino をパソコンに繋ぐことには、3つの意味があります
  - その1: 前述の通り、Arduino の電源になります
  - その2: プログラムを Arduino 内の ROM（フラッシュメモリ）に書き込むために必要です
  - その3: Arduino にはモニタ画面がありませんので、print デバッグをするには、パソコンの画面を借ります（USB 経由の仮想シリアルポートで通信できます）
- という訳で、まずは USB ケーブルでパソコンに繋いでみましょう。Windows でも Mac でも結構です
  - 購入したばかりの Arduino には、LED 点滅プログラムがあらかじめ書き込まれているので、繋ぐだけで、LED の点滅が見えると思います

Arduino の開発環境をインストールする

- Arduino のプログラミングには、通常、専用の開発環境ソフトをインストールします。いわゆる IDE (Integrated Development Environment) というものですね
- 以下からダウンロードできます
  - https://www.arduino.cc/en/main/software
  - 今回は、フラッシュメモリか Wi-FI で、適宜コピーしてください
    - SSID: 略
    - http:略
  - そして、インストーラを使ってインストールしてください
  - Arduino を、Mac OS X Mavericks 以降で使うとうまく通信できない問題が発生することがあります。今後のバージョンアップに期待してます

Arduino で “Hello World”（1）

- それでは、最初のプログラムをコンパイルして書き込んでみましょう。Linux (Unix) の世界では Hello World という文字を表示するのが一般的ですが、マイコンの世界では LED 点滅（略して「L チカ」）がよく使われます

Arduino で “Hello World”（2）

- 手順
  1. Arduino を繋ぎ、IDE（開発環境）を起動します
  2. メニュー: ツール → ボードで、Arduino Uno を選びます
  3. メニュー: ツール → ポートで、Arduino Uno が繋がったポートを選びます
  4. メニュー: ファイル → スケッチの例 → 01.Basics → Blink を選びます
  5. メニュー: スケッチ → コンパイル・検証を選びます
  6. メニュー: スケッチ → マイコンボードに書き込むを選びます
  7. 数秒間 Arduino 上の LED が高速に点滅（書き込み中）した後、基板中央部上側にある LED が 2秒周期（0.5ヘルツ）で点滅しはじめます
  8. 画面に表示されているプログラムで delay(2000); という部分（2カ所）の値を変えて、点滅速度が変わることを確認してください。なお、「スケッチの例」を変更するには、プログラムファイルを別の場所に保存し直す必要があります
  9. 完成です。あとは、パソコンから外しても（USB 電源にささっていれば）LED 点滅が続きます。御家族・友人に見せて自慢しましょう

Arduino IDE トラブルシュート

- プログラムがうまく書き込めない（Mac の場合にありがち）
  - Arduino IDE を立ち上げ直してみてください
  - Arduino IDE のシリアルポート選択をし直してみてください（一度他を選んで、選び直す）
  - Arduino の USB ケーブルを抜き差ししてみてください
  - Arduino 上のリセットスイッチを押してみてください
  - PC を再起動してみてください
  - USB ケーブルを替えてみてください

ショートブレイク

Arduino プログラミングの基本を覚えよう

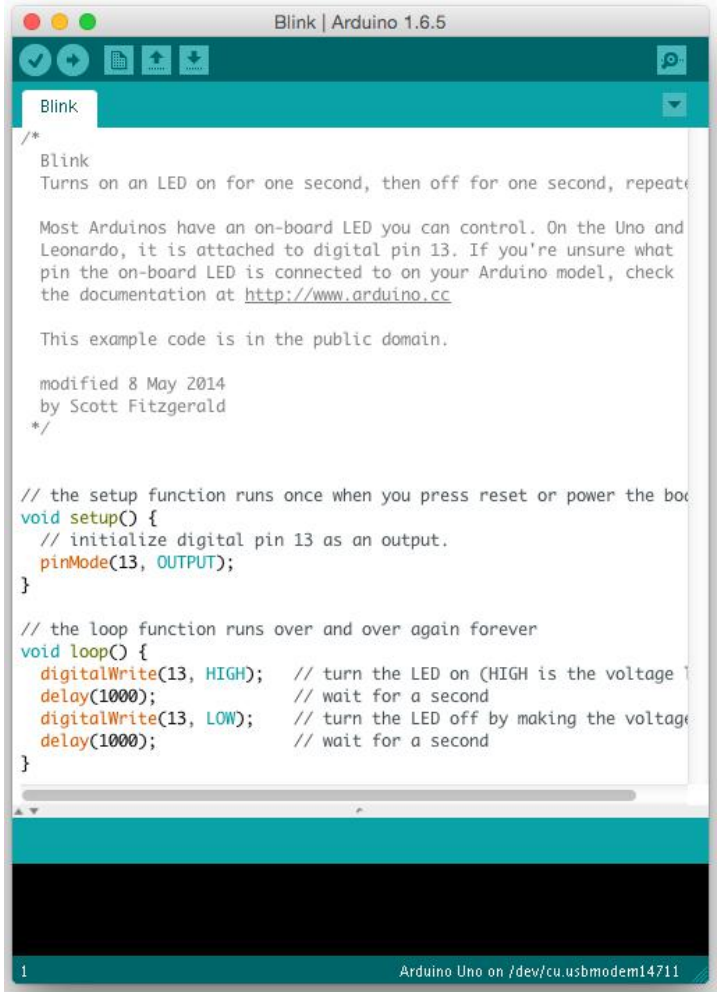
ARDUINO プログラムの基本構造

Arduino のプログラミング言語

- 組込系プログラミングの経験が少ない方にも始めやすいよう、Arduino では C, C++ ベースの特殊なプログラミング言語が用意されています
  - 最近はこの名前で呼ばれることが減った気がしますが、“Sketch” という名称で知られています
  - 組込系プログラミングで一番厄介な「プラットフォーム依存のスタートアップルーチン」、「I/O の取り扱い」が、うまく（やり過ぎない程度に）抽象化されていて、分かりやすいです
  - C++ が分かる方であれば、クラスライブラリの作成までできると思いますが、C 言語の知識があれば、ほぼ十分です

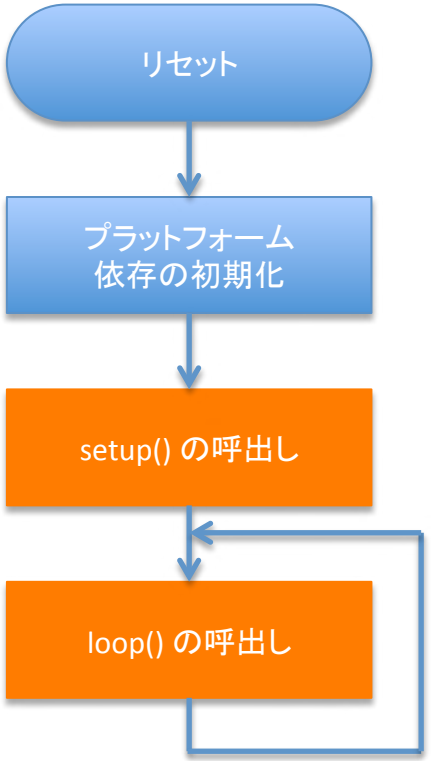
Sketch プログラムの構造（1）

- Arduino 開発環境（IDE）で先ほどのサンプルプログラムを開いたところです
- /* */ で挟まれたところ、// で始まる行はコメントです。C, C++ と同じです
- Sketch では setup() と loop() という 2つの関数が必須となっています
  - void setup(void);
  - void loop(void);



Sketch プログラムの構造（2）

- プログラムをビルドして Arduino 上の Flash ROM に書き込むと、Arduino の電源を投入するたび（あるいはリセットボタンを押すたび）に、右図フローチャートのように動作します

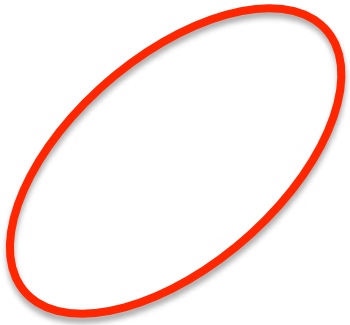


汎用 I/O ポート（GPIO）（1）

- 一般のパソコンやタブレット、スマートフォンでは馴染みのない概念ですが、コンピュータが登場して外部機器の制御や監視に使われるようになってから、基本的なインターフェイスとして利用されてきたのが、「I/O ポート」というものです（I: Input（入力）, O: Output（出力））
- 最近は GPIO（General-Purpose I/O）という、洗練された I/O ポートが一般的です。入力や出力としてプログラムで選択し、その特性まで設定することのできるもので、最近のマイコンではその使いやすさや特性の優れたものが高く評価されます
- 多くのマイコン（マイクロコンピュータ IC）には、GPIO 端子が複数付いています
  - Arduino では一般に、15ピン程度の GPIO が用意されています
  - Arduino では、1ピン毎に入出力の別を指定することができます

汎用 I/O ポート（GPIO）（2）

- Arduino の GPIO ピン



入力（Input）ポート

- あるGPIOピンを入力に設定すると、そのピン（端子）から外部の電気状態を（通常は二値として）知ることができます
- ピンに繋がれた「電気」が低い電圧か（0ボルトとか）、高い電圧か（5ボルトとか）を識別し、プログラムから「二値変数」として読み取ることができます
- 次のようなものを繋ぐことができます
  - スイッチ（押しボタン式, トグル式, リードスイッチなど）
  - 各種センサー（光センサー, 温度センサーなど）
  - 他のマイコンの出力（Output）ポート
- 今回は説明しませんが、別にアナログ入力（ADC）もあります

出力（Output）ポート

- ある GPIO を出力に設定すると、そのピン（端子）を使って外部機器を制御（コントロール）することができます
- 例えばプログラムから、ある「変数」に 0 あるいは 1 を書き込むことで、ピンから出力される「電気」を低い電圧か（0ボルト）や、高い電圧（5ボルト）に設定することができます
- 次のようなものを制御できます
  - LED, 豆電球（LED が一般的です。赤外線 LED を含む）
  - モーター, アクチュエータ（直接繋げるのは電気性能的に難しいので、トランジスタなどを介します）
  - 液晶ディスプレイ（LCD）
  - スピーカー, ブザー
  - トランジスタ（信号増幅, 補強回路）

Arduino で GPIO（出力）を使う

- ここでは、Arduino に標準で搭載されている LED をプログラムで制御してみましょう（サンプルプログラムにあった通りです）
- LED は、GPIO ピン 13番に繋がっているので、まずはそれを出力用に設定します。setup() の中で実行すると良いでしょう

```
pinMode(13, OUTPUT);
```

- このピンを高い電圧（Arduino Uno では 5ボルト）を出力するには、ここに HIGH を、低い電圧（0ボルト）を出力するには LOW を書き込みます

```
digitalWrite(13, HIGH);
digitalWrite(13, LOW);
```

参考: Arduino で GPIO（入力）を使う

- ここでは実例を示しませんが、参考までに、入力は次のようにします

```
void setup() {
    pinMode(7, INPUT);
}
void loop() {
    int a;
    a = digitalRead(7);
    // ピンが低電圧なら 0 が、高電圧なら 1 が a に入る
}
```

ビジーウェイト（時間待ち）

- LED を目で見えるように点滅させたり、ある一定間隔でセンサーの値を読み取ったりするために、時間待ちが必要なことがあります。Arduino ではリアルタイム OS を使わないので、ビジーウェイトで処理するのが一般的です
- 割込（わりこみ）という仕組を使って、高精度にタイミングを取ったり、マイコンの消費電力を下げたりすることもできますが、まずは簡単な方法から紹介します
- Arduino で、例えば 1000ミリ秒のビジーウェイトを入れる（1秒間何もしないで待つ）には、次のようなコードを書きます

```
delay(1000);
```

シリアルポートの利用（1）

- Arduino にはディスプレイ出力がありませんので、そのままでは「print デバッグ」ができません
- その代わりに UART（RS-232）と呼ばれるシリアルインターフェイスが付いていますので、それを利用します
  - UART は「ユーアート」と読みます
  - シリアルインターフェイスとは、データの送受信に 1本ずつの信号線を用いる通信方式の一般的な名称ですが、狭い意味では RS-232 のことを呼びます
  - 昔のパソコンには RS-232 コネクタが一般についていましたが、最近は USB ブリッジを使って仮想的に RS-232 をエミュレートします。中の信号は（ほぼ）RS-232 なのですが、USB の中にトンネルを通して通信します
  - 本来の RS-232 規格は +/- 15V（ボルト）でしたが、単純に UART と呼んだ場合、信号電圧はさまざまです。+5V か +3.3V が一般的です
  - これにより、最近のパソコンでも Arduino と簡単に通信ができます

シリアルポートの利用（2）

- UART (RS-232) を使うには、いくつかの約束事があります
  - UART では、原則として一対一の通信しかできません
  - 通信相手と信号電圧を合わせる必要があります。Arduino で USB 経由の場合、電圧は Arduino 上で終端されてますので気にしなくて大丈夫です
  - 原始的な通信規格なので、通信速度や信号フォーマットを、通信する相手と、あらかじめ以下のように決めておく必要があります。（USB のように、相手が USB 2.0 か 3.0 か、のように自動的に判断してくれたりしません）
    - 通信速度: 9600 bps（ビット毎秒）
    - パリティビット: 無し
    - データビット: 8ビット
    - ストップビット: 1ビット
    - フロー制御: なし

      （いずれも、Arduino 開発ソフトを使う場合はデフォルト設定で大丈夫です）
- Arduino をパソコンに繋ぐと、OS からは次のように接続が見えます
  - Windows の場合: COM ポート（OS が番号を割り当てます）
  - Mac OS の場合: “/dev/tty.usb なんとか”（OS が名前を割り当てます）

Arduino とパソコンでシリアル通信する

- 例えば、次のようなコードを書きます

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    // パソコンに文字列を送信する
    Serial.println("Hello World");

    // パソコンから文字(1バイト)を受信する
    if (Serial.available() > 0)
        Serial.println(Serial.read());
}
```

- 連続して何度も文字列を送出すると、バッファがオーバーフローして内容が壊れる恐れがありますので、注意してください。詳しくはマニュアルを参照してください
  https://www.arduino.cc/en/reference/serial

ブレークタイム

- ここで少し休憩を挟みます
- Arduino でいろいろプログラムを試して頂いて結構です
  - 「メニュー: ファイル → スケッチの例」に、たくさんサンプルコードがあります

GPIO とデジタル IC の基本を勉強します

少しだけハードウェアを学ぼう

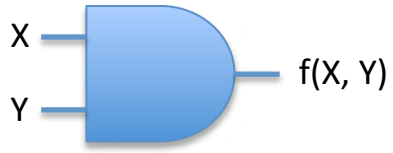
論理式とデジタル回路(1)

- あまり思い出したくない方もあるかと思いますが、少し復習します
- 情報科学や計算機工学を専攻すると、(たぶん) こんなブール代数(二値論理)式を学びます

  $f(X, Y) = X \cdot Y$

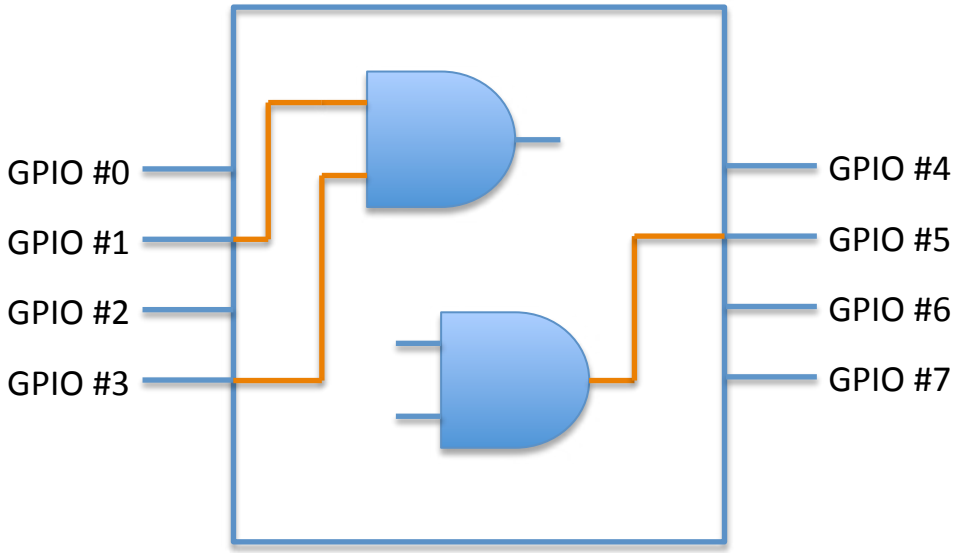
  $g(X, Y) = \overline{X + Y}$

- そして、こんな記号を学んだりします



論理式とデジタル回路(2)

- Arduino マイコンもデジタル回路で構成されており、そしてその入出力(先ほど学んだ GPIO)も、このようなデジタル論理回路の入出力と繋がっています(イメージとして捉えてください)



論理式とデジタル回路(3)

- つまり GPIO というのは
  - 入力の時は論理回路の入力に繋がっていて、
  - 出力の時は論理回路の出力に繋がっている

  というイメージです
  - 既に説明したように、Arduino の GPIO ではピン単位に入力として使うか、出力として使うかをプログラムで(動的に)決められます
- 結局のところ、GPIO というのはデジタル論理回路の入出力だということは分かりましたが、ここにスイッチやセンサー、LED などを繋ぐにはどうしたら良いのでしょうか? スイッチや LED は電気で動きますよね?

論理式とデジタル回路（4）

- Arduino などのマイコンでは、論理式の「真: true」 や「偽: false」 を、電気のあるなし、より正確に言うと電位の高い/低いで表します
- 今回の Arduino Uno は 5V（ボルト）のマイコンなので、電位を 5V と 0V の 2レベル（つまり二値）で表現します
- なお一般に、「真（true）が 5V」であることを正論理、「真が 0V 」であること負論理と呼びます （この辺はなんとなく理解できればオーケーです）

正論理

真	5V
偽	0V

負論理

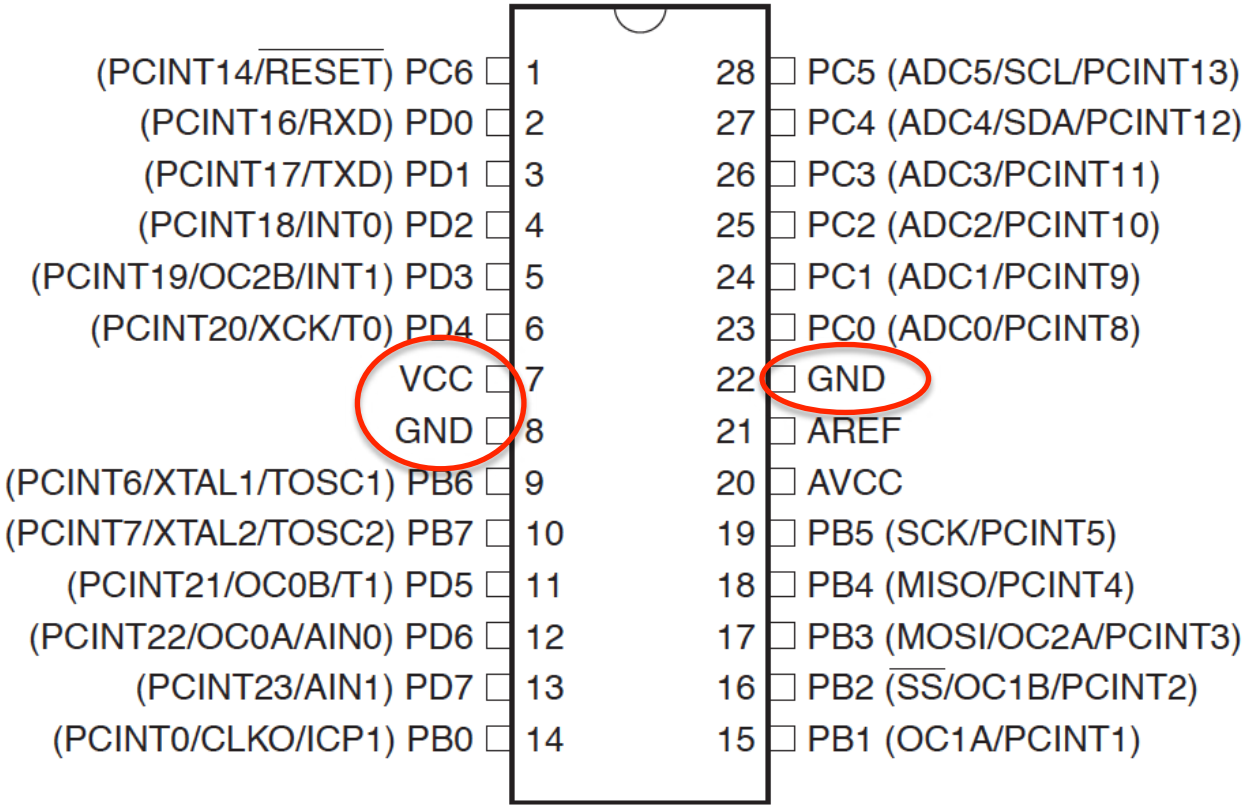
真	0V
偽	5V

マイコン IC と GPIO の使い方（1）

- ここからは、実際に Arduino の GPIO を電気回路に繋ぐ基本を勉強します
  - いちばん眠くなりそうな部分かと思いますが、頑張りましょう
- まず最初に、マイコンには電源を繋がないと動きません。（Arduino では USB コネクタを使って簡単に電源を供給できるようになっていますが、仕組を簡単に理解しましょう）
- マイコンなどのデジタル IC には、多くの場合 2つの電源端子があります
  - V_{DD} あるいは V_{CC} : 5V や 3.3V の電源のプラス側を繋ぎます。Arduino Uno は 5V です
  - GND: 電源のマイナス側を繋ぎます

マイコン IC と GPIO の使い方（2）

- Arduino で使われる Atmel マイコンの実例です

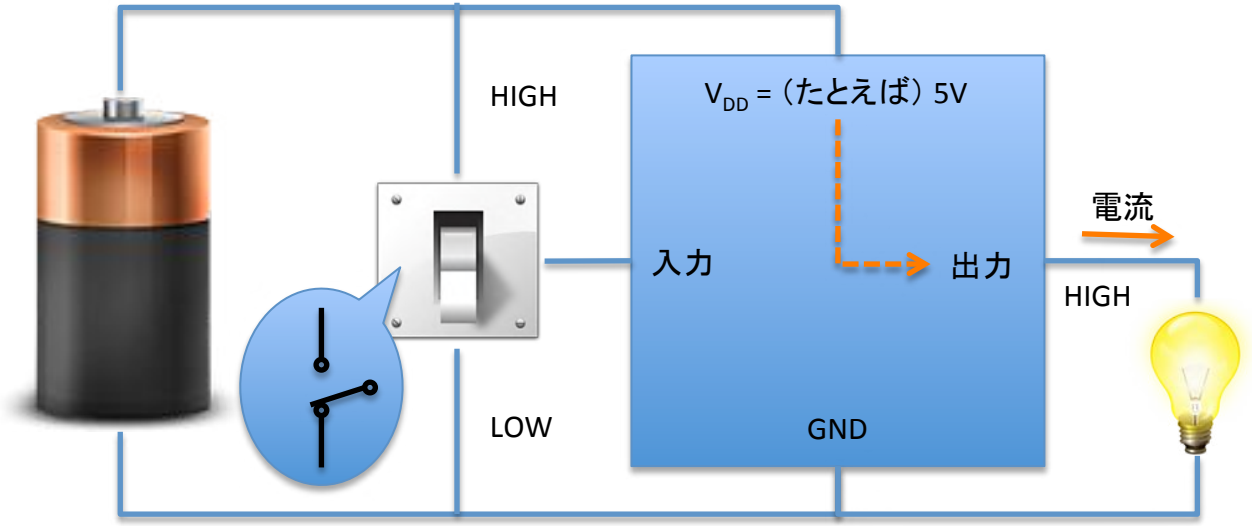


マイコン IC と GPIO の使い方（3）

- GPIO の各端子は、入力あるいは出力に設定すると、それぞれ次のように動作します
- 入力のとき
  - 端子に繋いだ電気の電位を、論理値 true か false として読み取ります。通常、正論理/負論理かに関わらず、電位が高い（Arduino Uno R3 では5V）のときに 1 として読み出され、電位が低い（同 0V）のときに、0 として読み出されます
  - 一般に、電位が高いこと（V_{DD} に等しいこと）を HIGH と呼び、低いこと（GND に等しいこと）を LOW と呼びます
- 出力のとき
  - 一般に、プログラムから端子に書き込んだ値が 1 のとき、端子の電位は HIGH になり、0 を書き込むと LOW になります

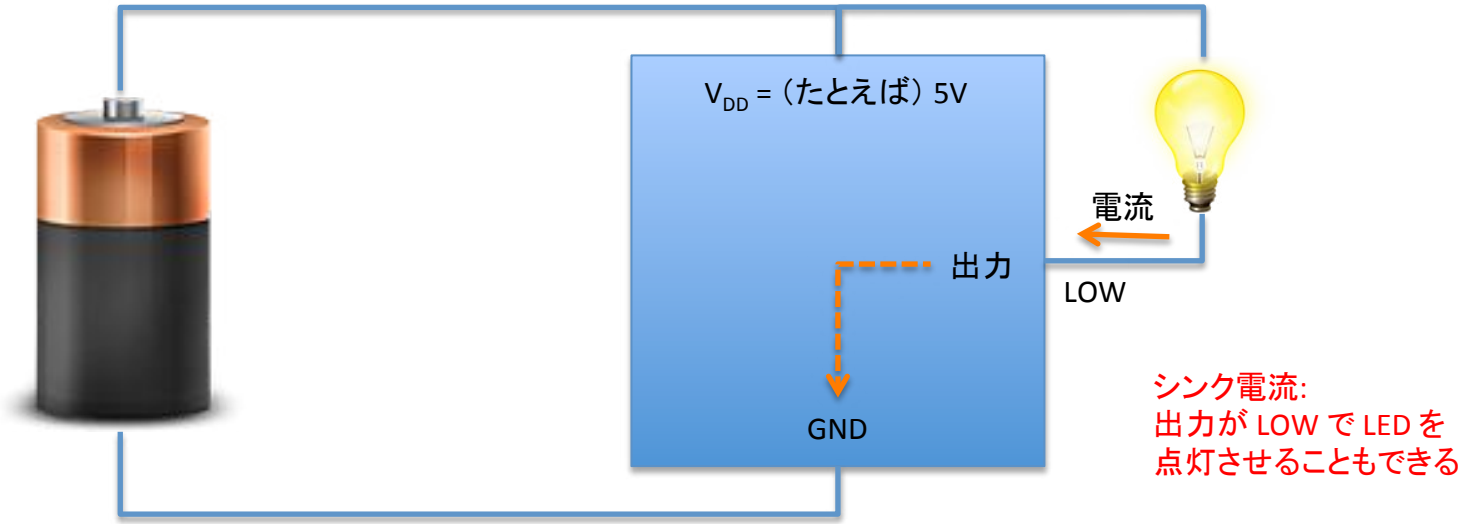
マイコン IC と GPIO の使い方（4）

- GPIO にスイッチと LED を繋ぎたいとしましょう（正確なつなぎ方は後で説明します）
- 入力側: スイッチをオンにすると、GPIO 入力に 5V が繋がり、オフにすると 0V が繋がるようにします
- 出力側: GPIO に 1 を書き込むと HIGH になるので、そこに LED を繋いで電流が流れるようにします



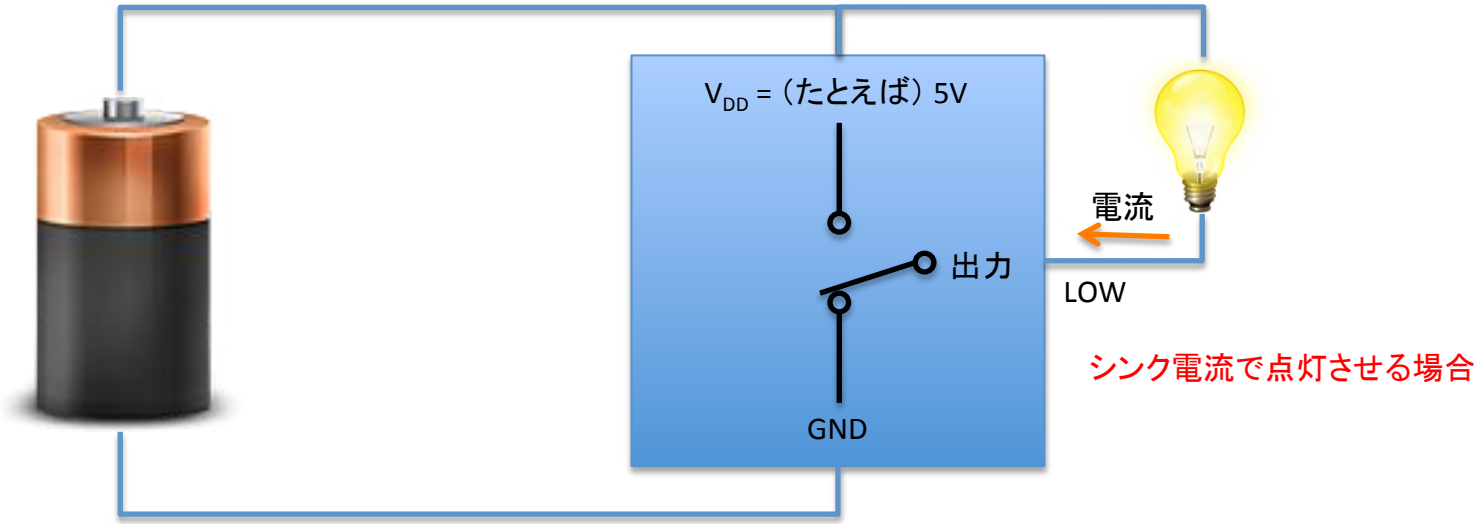
マイコン IC と GPIO の使い方（5）

- 少しむずかしいのですが、出力からは電気（電流）を取り出して負荷を動かすこともできますし、逆に電気（電流）を取り込んで負荷を動かすこともできます
- 前者をソース（source）電流、後者をシンク（sink）電流と言います



マイコン IC と GPIO の使い方（6）

- より分かりやすく言うと、デジタル IC の出力には次のようなスイッチが入っていると考えてください（実際にはトランジスタが入ってます）
- スイッチが V_{DD} に繋がっているときはソース電流を得ることができ、GND に繋がっているときはシンク電流を得ることができます

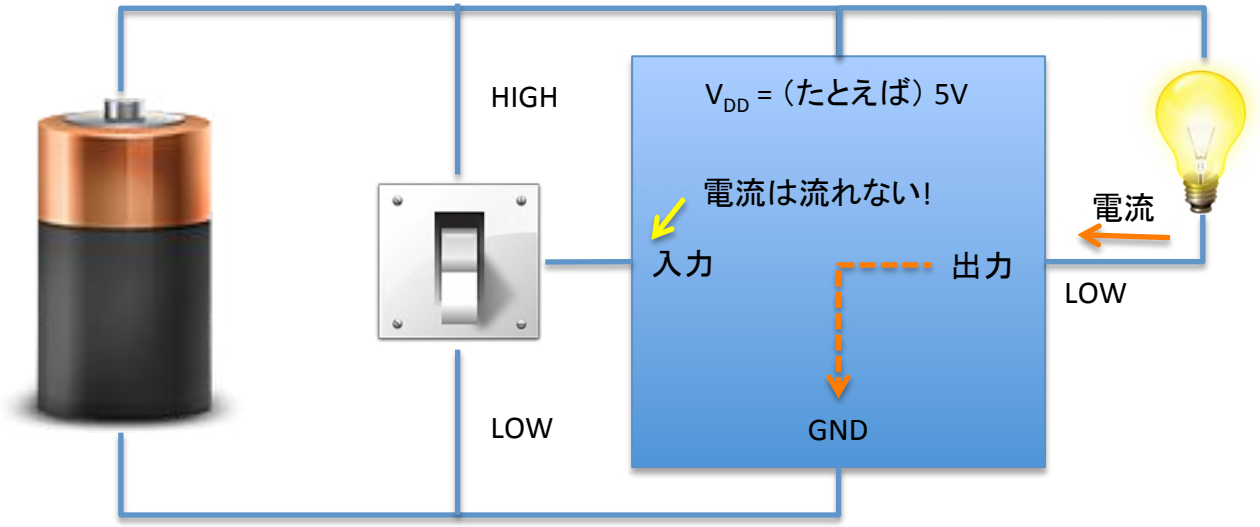


補足1: 少し電気的なこと

- 出力端子から電流を流し出したり、流し入れたりできることは分かりました。ところで、入力端子には電流は流れるのでしょうか?
- 最近の電子回路に一般に使われる MOS IC では、入力に MOS FET (Metal–Oxide–Semiconductor Field-Effect Transistor) という素子が使われています。この素子は入力に電位（電圧）さえ与えれば、論理回路として動作し、入力端子には電流はほとんど流れません。（これを、「入力インピーダンス（抵抗）が高い」という）
- MOS FET の入力は入力インピーダンスが高いので、人が手で触っただけで電位を読み取れるほどです。逆に、入力インピーダンスが高いために、静電気などに弱いという欠点があります。（最近のマイコンには保護回路が付いているので、滅多に壊れない）

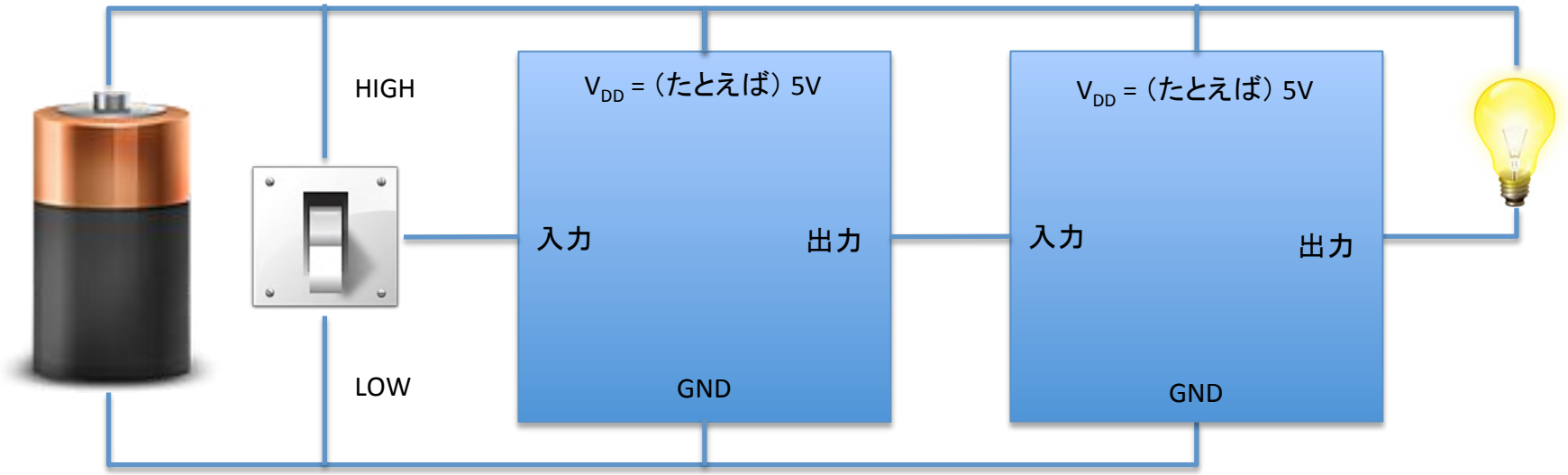
補足2: 少し電気的なこと

- 入力端子には電流が(ほとんど)流れません



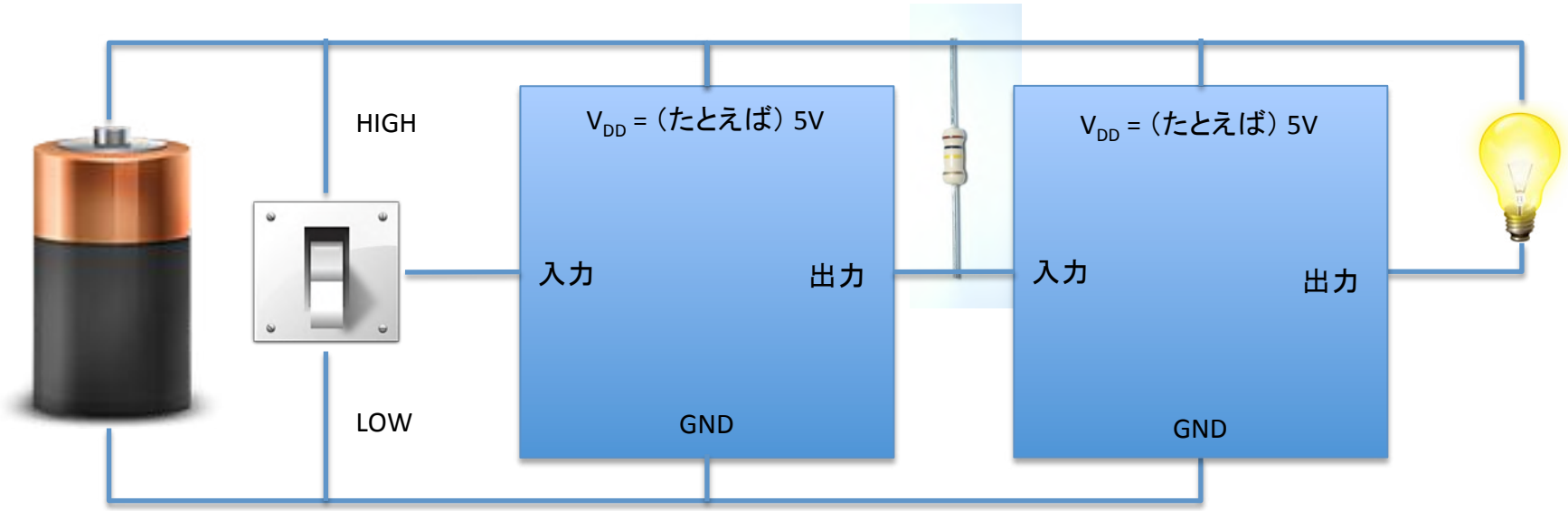
デジタル IC 同士のつなぎ方（1）

- マイコンなどデジタル IC の出力を他の IC の入力に繋ぐには、基本的にはそのまま繋げば大丈夫です（お互いの V_{DD} 電位が一致していないとダメです）
- 前段の IC 出力がハイインピーダンス（V_{DD} 電位にも GND 電位にも繋がらない「仮想的な未接続」状態）のときに、後段の IC が誤動作しないように、プルアップ抵抗、プルダウン抵抗というものを入れることがあります（次頁）



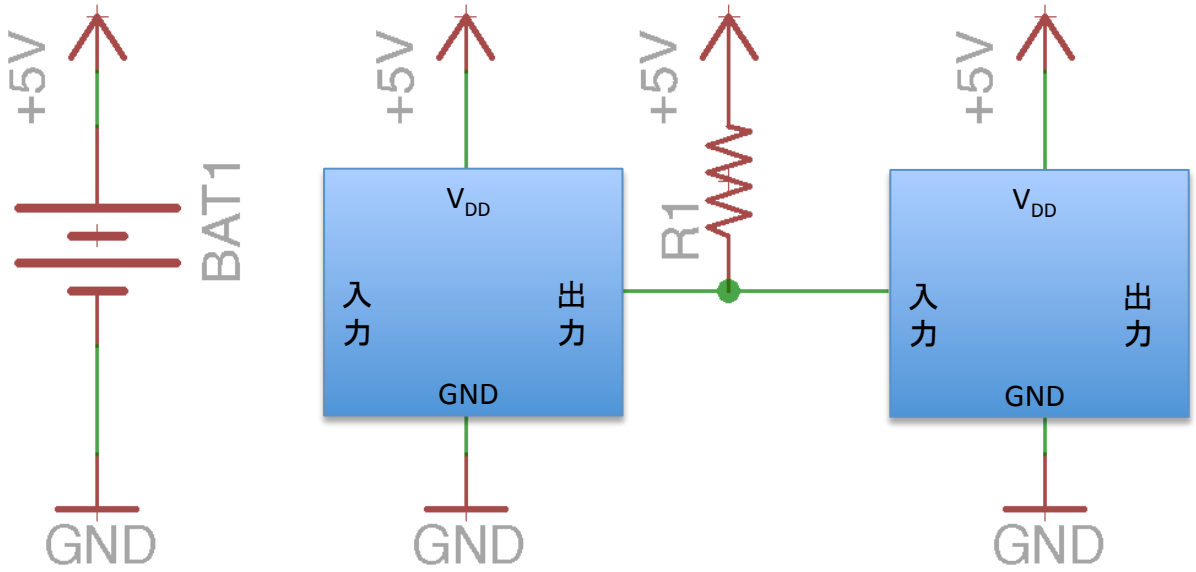
デジタル IC 同士のつなぎ方（2）

- 下図はプルアップの例です。MOS IC では、数十 kΩ（キロオーム）くらい入れておけば大丈夫です。（かなりアバウト）
- 最近のマイコンの入力端子には、内部にプルアップ抵抗が入っていて、結線状態をプログラムできるものもあります



デジタル回路図の描き方

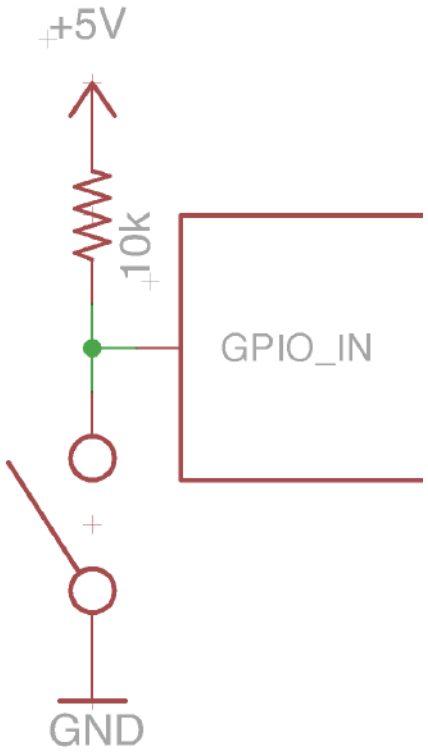
- ここまでは実体配線図を示してきましたが、実際には回路図で表記します。前ページの例を回路図にしてみましょう
- V_{DD} や GND は線で繋いで描いてもいいですが、見やすさのため、このように分けて描くことがあります



GPIO の使いかた実例（1）

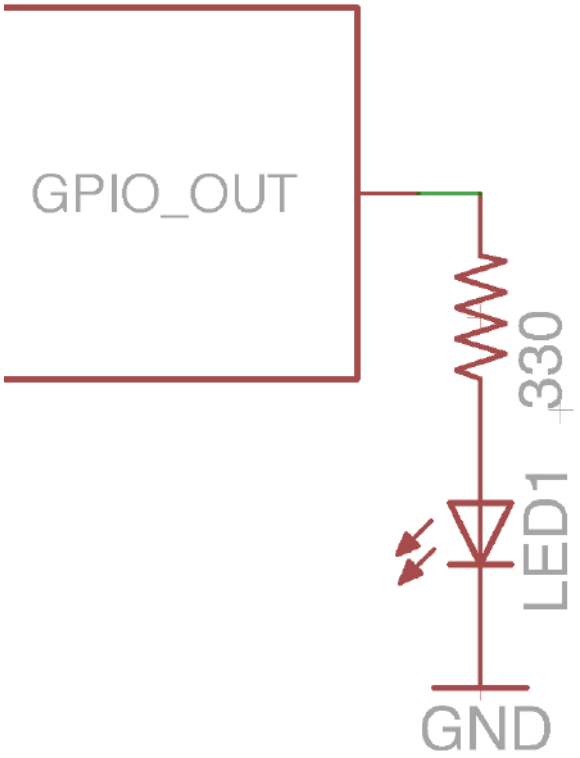
- ここからは、実際に GPIO に入出力を繋ぐ例を示します。理論については省略します

- スイッチ（プッシュボタンなど）の繋ぎかた
  - スイッチは通常、（歴史的な経緯もあって）GND 側に繋ぎ、プルアップ抵抗で電源に繋ぎます（スイッチが入ると GPIO から 0 が読めます。切ると 1 が読めます）
  - プルアップ抵抗は 10kΩ（オーム）がポピュラーですが、少しくらい（10倍くらい）違っても問題ありません



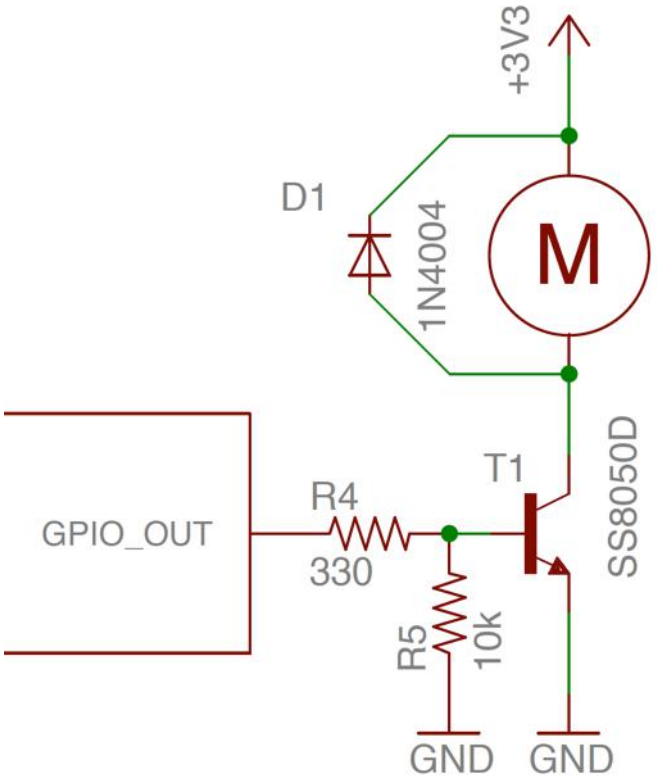
GPIO の使いかた実例（2）

- LED の繋ぎかた
  - ここではソース電流で駆動していますが、シンク電流で駆動することもできます
  - LED には通常、直列に抵抗器を繋いで電流を制限します。そうしないと LED に過大電流が流れて破損します
  - マイコンの電源が 5V の場合、抵抗は 330Ω くらいが標準的です。これでは暗いと思ったら 220Ω 程度、明るいと思ったら 680Ω を試してみましょう



GPIO の使いかた実例（3）

- モーターを回す回路例も、参考までに示しておきます
  - 回路中のトランジスタ SS8050D や、ダイオード 1N4004 は、共立エレショップなどで購入できます
  - ダイオードはモーターの近くに付けてください。極性に注意
  - モーターに加える電圧は、モーターの仕様に合わせてください。（マブチモーターとかだと 3V くらい?）
  - R4 の 330Ω は、5V の Arduino の場合です。3.3V Arduino の場合は 220Ω 程度を使ってください



GPIO の使いかた実例（4）

- 他に、簡単で面白い例として、タッチセンサーがあります。Arduino の GPIO 入力端子を手で触れることを検出してスイッチ入力とすることができます
- Google で「Arduino タッチセンサー」と検索すると簡単な試作例がたくさん見つかります
  - 例: http://inter-arteq.com/arduinoを使ってタッチセンサー/
- 原理としては、建物各階にあるエレベーターのタッチボタンと同等です

ブレークタイム

- ここで少し休憩を挟みます
- 今回製作するものを回覧しますので、見ておいてください

それでは実際に工作を始めましょう

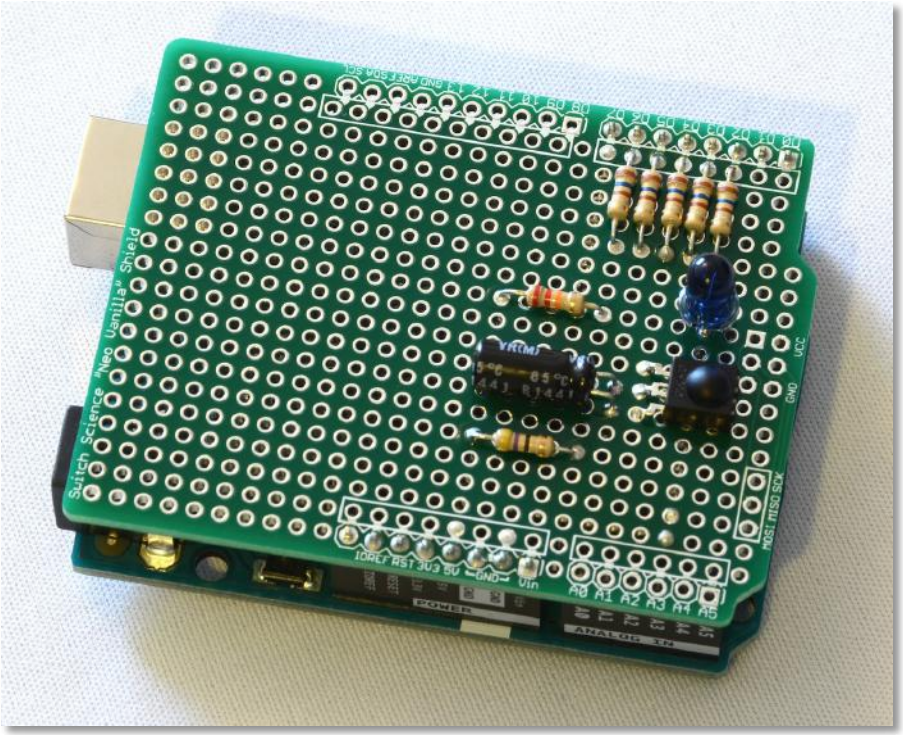
ARDUINO 工作とハンダ付けの実際

大変お待たせしました!

- それでは実際に工作をしてみましょう
- 静電気（ESD）対策
  - 一般に、電子部品（特に MOS IC や MOS FET）は静電気に弱い性質があります
  - 今回のように回路基板に組まれた Arduino などは比較的丈夫ですが、ドアノブに触ってパチっと電気が来るような季節は、MOS IC などに触れる前に、玄関のドアノブなどに手を触れて静電気を逃がしておきましょう
  - 乾燥した季節に、スリッパやゴム底靴などを履いてカーペット上を歩くと静電気が体に溜まりやすいので、特に注意しましょう
- 火傷に注意
  - ハンダごての先端は 400度程度の高熱となります。私も子供のころは誤って先端に触れ、指先が白くなるくらいの火傷を何度か経験しました。見た目が赤くないのでうっかり触ってしまわないよう注意しましょう!

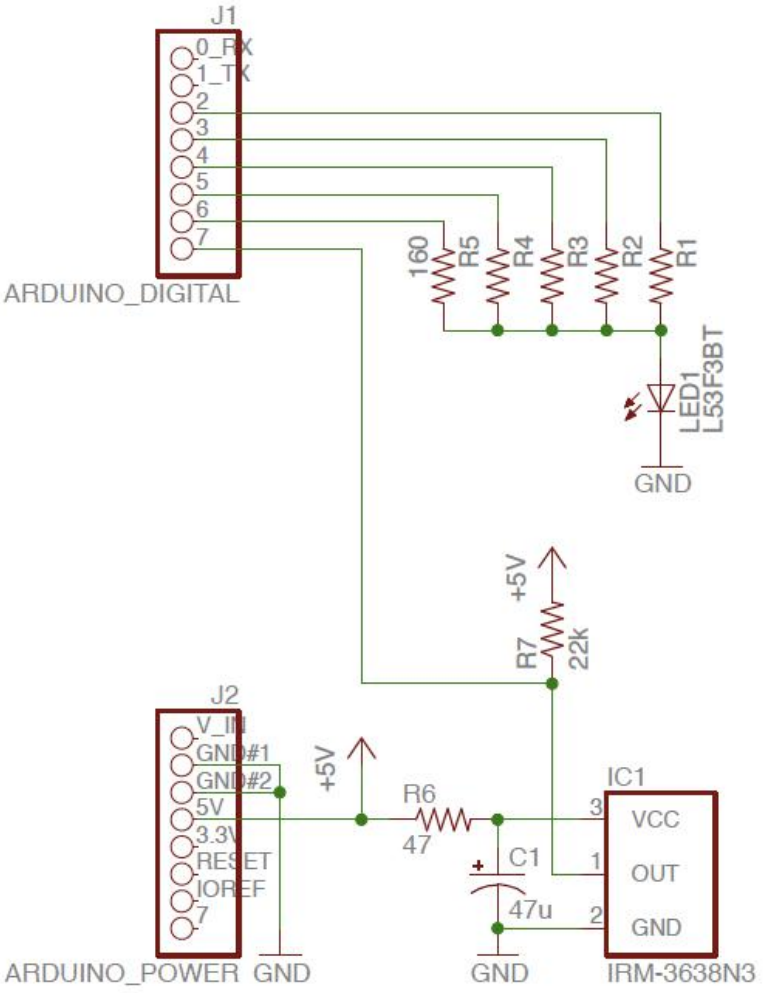
こんなものを作ります

- 休憩時間中に回覧しました
- Arduino Uno の上に子亀のように差し込んで使います



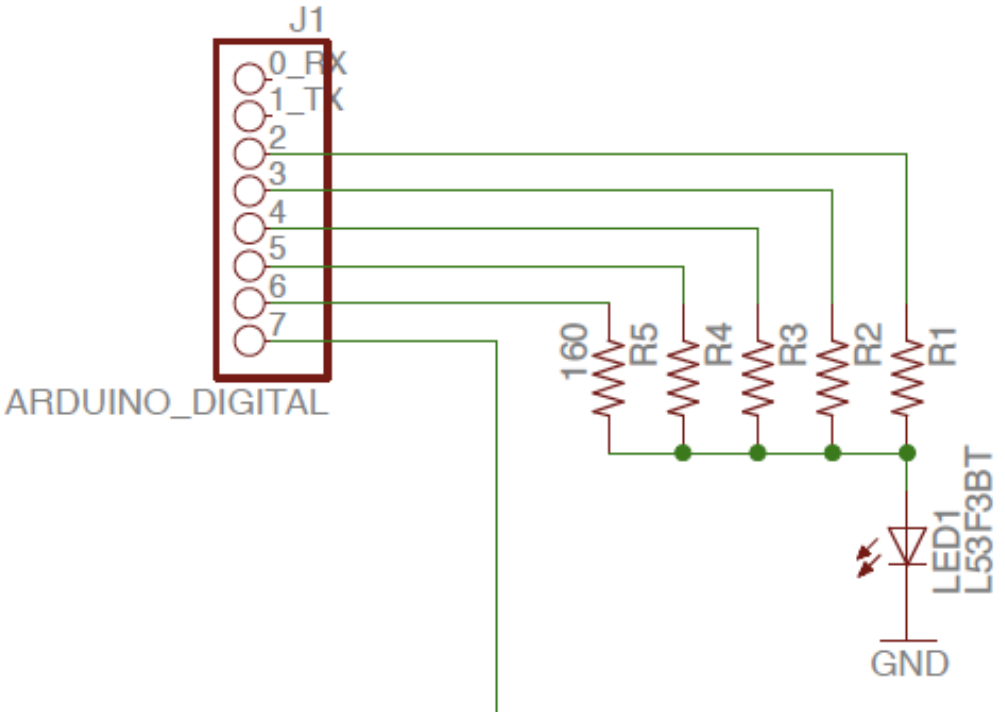
回路図の説明(1)

- 回路図の全体はこのようなものです
- おおむね、上半分が赤外線LEDを点滅してリモコン送信をする回路です
- 下半分は、赤外線を受信するための回路です。受信したコードはArduinoのプログラムで分析します



回路図の説明(2)

- まず、LED 点滅のための回路からです
- これは先ほど説明した「LED の繋ぎかた」の応用です
- Arduino の一つの GPIO 出力では赤外線 LED を十分に明るく点灯させられないので、出力を 5本束ねています（一般にはトランジスタで信号を増幅しますのでこれは裏技です。点滅制御を誤って赤外線 LED を破損しないメリットもあります）



補足: 赤外線 LED の点滅について(1)

- 下の表は、今回使用する赤外線 LED の最大絶対定格(Absolute Maximum Rating) です。最大絶対定格は、電子部品を使う上で非常に大切な特性で、「この値を一瞬でも超えたら、部品が壊れてもメーカーは責任を取りません」という情報です

Absolute Maximum Ratings at T_A=25°C

Item	Symbol	Maximum Rating	Units
Power Dissipation	Pd	100	mW
Forward Current	IF	50	mA
Peak Forward Current	I_P	1.2	A
Reverse Voltage	VR	5	V
Operating Temperature	Topr	-45~ +80	°C
Storage Temperature	Tstg	-45~ +80	°C

Notes:
1. 1/10 Duty Cycle, 0.1ms Pulse Width.
2. 4mm below package base.

補足: 赤外線 LED の点滅について(2)

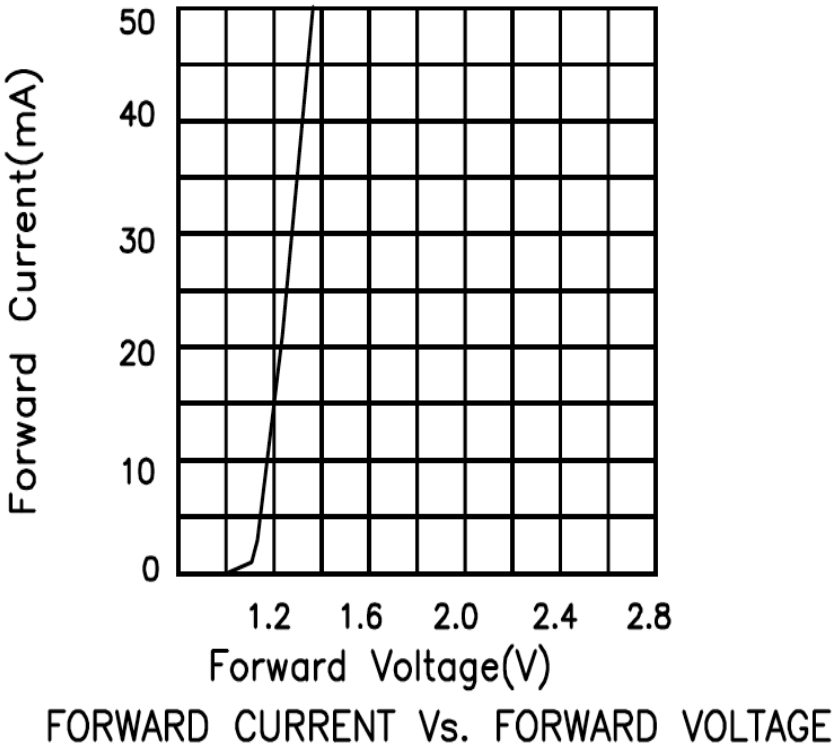
- 最大絶対定格にあるように、この LED は最大で 50mA (ミリアンペア)しか流せないのですが、注釈として、デューティ比(点灯/消灯時間の比率)が短く、絶対的な点灯時間が 0.1ミリ秒以下であれば、最大で 1.2A (アンペア)まで流して良い、ということになっています
  - これは、この LED の典型的な利用例が赤外線リモコンであるということを暗示しています
  - 今回の使い方では LED を 38kHz (キロヘルツ)で点滅させ、デューティ比 1 : 1 で使うことにしており、連続点灯時間は 0.01ミリ秒程度であるので、電流を 100mA まで流して使うことにします
  - もう一つの最大絶対定格は Power Dissipation (電力損失) < 100mW (ミリワット)ですが、今回は 80mW 以下程度で使用することになるので、これも問題ありません
  - なお、先に述べたデューティ比 1 : 2 にする場合、LED の能力を生かすには 150mA 程度流します。(抵抗器 160Ω の変更が必要です)

補足: 赤外線 LED の点滅について(3)

- 結論
  - 今回 GPIO 出力を 5本束ねて LED を点滅しています。実際に点滅プログラムが完成してデューティ比を下げた場合は問題ないのですが、連続点灯で 5本同時に LED を点灯すると最大絶対定格を超えてしまうので、最初は GPIO 出力を 1つだけオンにして実験しましょう
  - 点滅制御までうまくできることが分かったら、5本まとめてオン/オフ制御することにしましょう
  - なお、Arduino マイコン側にも絶対最大定格があって、GPIO ピン 1つあたりの最大電流は 40mA、IC 全体での供給可能電流は 200mA となっています。5本束ねることで、1ピンあたり 20mA (100 / 5) と問題ありませんが、2つの LED を制御してパワーアップするときなどは、最大でも 9本の駆動として、180mA 程度としておいたほうが無難です

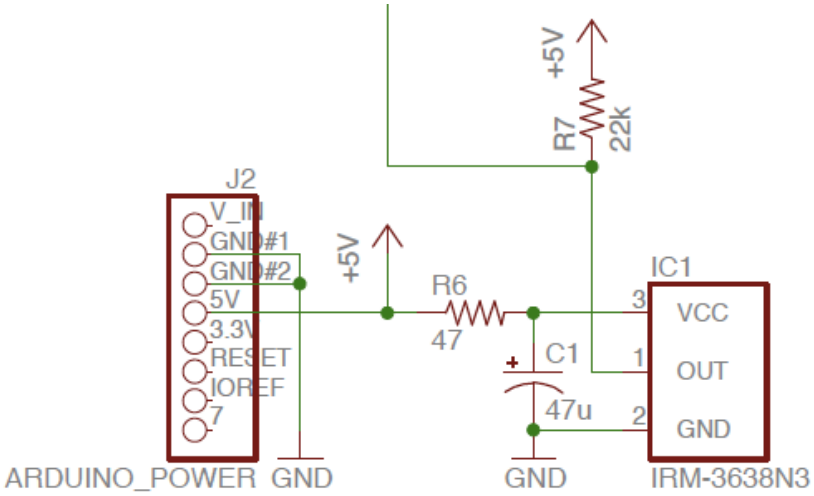
補足: 赤外線 LED の点滅について(4)

- なお、この赤外線 LED に 100mA を流すために 160Ω (オーム)の抵抗器を使いましたが、以下が設計根拠です。興味のある方だけ御参考ください
- 右図は、この LED の電圧 – 電流特性です
- 100mA の順電流が 1.6V 程度の電圧降下と想定されるため、(5 – 1.6V) / (100mA / 5) = 160Ω としています



回路図の説明(3)

- 次は赤外線受信の回路ですが、これは今回利用する赤外線受光モジュール IRM-3638N3 のマニュアルにある参考回路を、そのまま利用しています。そのため、特に考慮する点はありません
  - R6 と C1 は、電源入力に対するローパス(低域通過)フィルタとなっています。電源ノイズの削減が目的です
  - R7 はプルアップ抵抗です。IC1 の OUT が HIGH のとき、その電位を十分に 5V に近づけるために設けられています

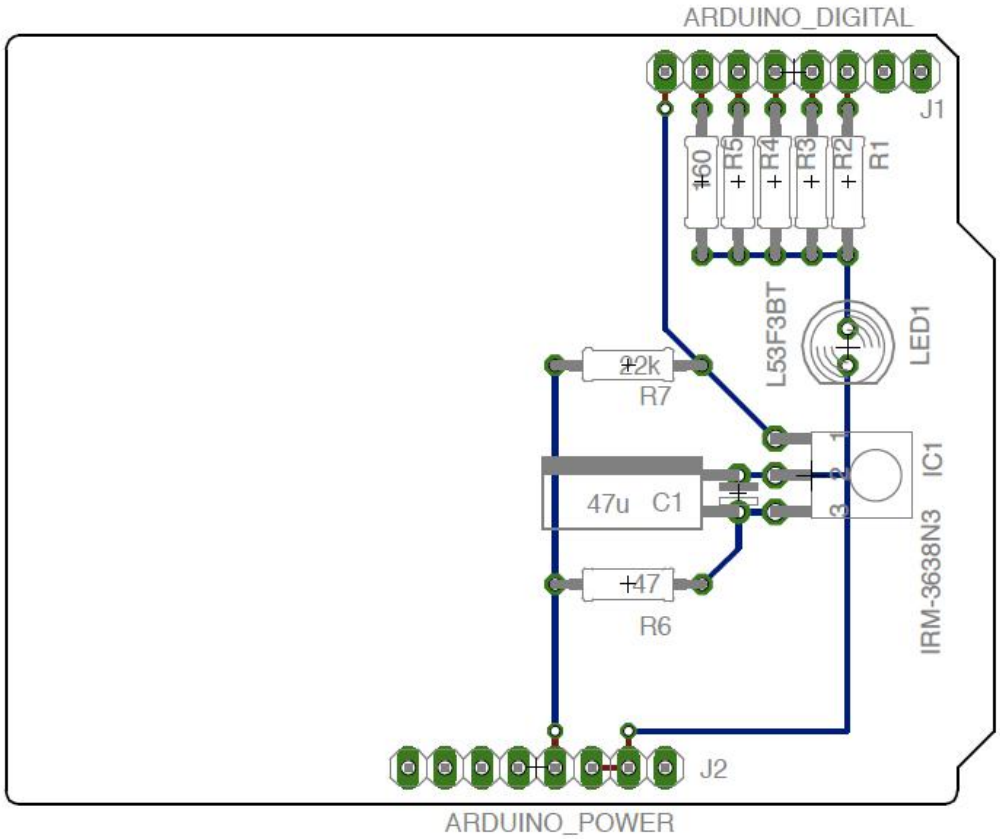


補足: IRM-3638N3 の参考回路について

- IRM-3638N3 のデータシート（マニュアル）にある参考回路を示しておきます（削除）

基板レイアウトについて

- 今回は、Arduino 専用のバニラボードというユニバーサル基板（万能基板）を使います。基板の実体配線図（レイアウト図）は右図のようになります
- 余裕のある方は、先ほどの回路図と見比べてみてください



抵抗器カラーコード

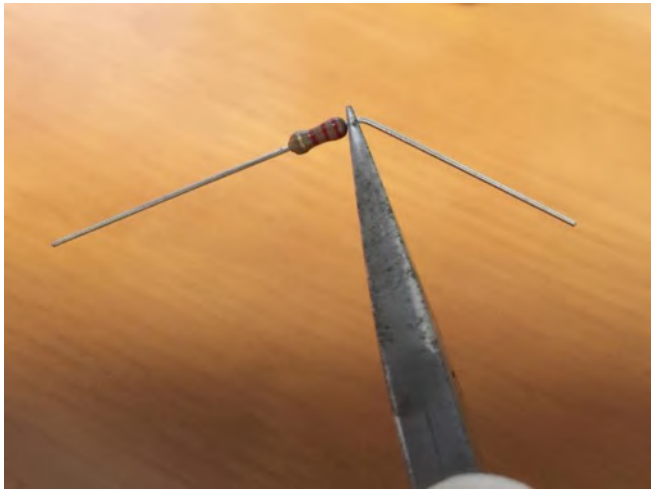
- 今回用いる抵抗器には、色の帯（カラーコード）で抵抗値が示されています
- カラーコードは、左から 2つが抵抗値の絶対値、次が 10の累乗の値、最後が抵抗器の設計精度です
- 色は、黒 = 0, 茶 = 1, 赤 = 2, 橙 = 3, 黄 = 4, 緑 = 5, 青 = 6, 紫 = 7, 灰 = 8, 白 = 9 です
- 右の写真の抵抗値は、
  1 6 × 10 **1** = 160Ωで、精度 +/− 5%（金）です
- カラーコードの覚え方は、ネットなどでいろいろ見つかります

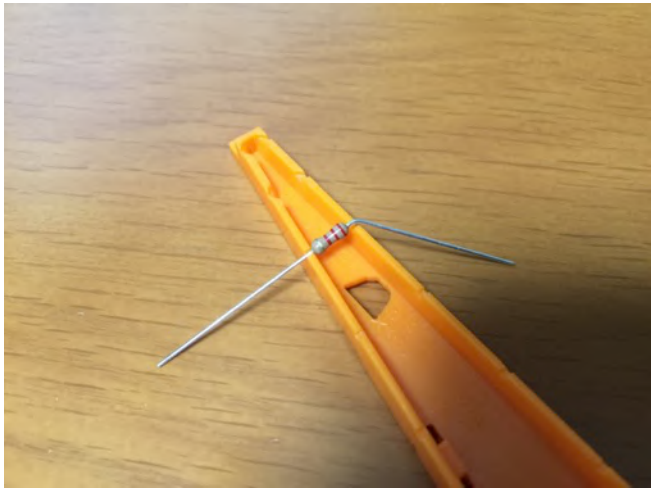
部品の極性

- 多くの電子部品には極性があります。電池の +/– のようなイメージです
  - LED には +/– （anode, cathode） があり、逆向きには電流が流れません。つまり点灯しません。通常、リード線の長いほうが anode（+）で、短いほうが cathode （–）です。（厳密にはデータシート（マニュアル）やでテスターで実測確認したりします。今回の赤外線 LED は目視で点灯が分からないので特に注意が必要です）
  - 電解コンデンサには、+/– があります（他の多くのコンデンサには極性はありません）。一般に、白い帯状のマークが付いているほうが – （マイナス）です
  - 赤外線受光センサーのような IC にも極性があります。データシートで確認するようにしてください
- 抵抗器には極性はありません。ただし、カラーコードが揃うように実装すると見た目が綺麗です

部品のリード線の取り扱い

- 部品のリード線にそのまま力をかけると、部品内部の接合部が傷むことがありますので、写真（上）のようにラジオペンジで押さえて曲げるようにすると良いでしょう
- また、写真（下）のように、2.54ミリピッチの整数倍に曲げられる専用ジグも売られています!
- 細かいことをいうと、リード線に触れる前に手を石鹸で洗っておくとベターです（脂が付くとハンダの乗りが悪くなる）
- なお通常は、リード線はハンダ付けの後に余分な長さを切断します。先に切断してしまうと LED などの極性が分からなくなります





補足1: ユニバーサル基板の注意点

- このスイッチサイエンス社製のバニラ基板（専用ユニバーサル基板）には、いくつか注意点があります
  - Arduino のコネクタには、一部 100mil （0.1インチ = 2.54ミリメートル）ピッチから外れたコネクタ（DIGITAL (PWM～)）があります。逆指し防止のためと思われますが、一般のユニバーサル基板を代用する場合には、注意が必要です
  - この基板は、「両面スルーホール」というタイプの基板です。各穴にはビアという金属製の管のようなものが押し込まれていて、表面と裏面は電気的に導通しています。つまり、表と裏の交差点に、たとえば VDD (5V) と GND を渡すと、ショート（短絡）してしまうので注意しましょう。（そのような場合は、少なくとも一方に被覆線を用いる）
  - また、スルーホールなのでランド（銅箔パターン）が過熱で容易にはがれることはありませんが、逆に、間違ってハンダ付けするとハンダを吸い出すのが難しくなっています。（ハンダを取り除くことができない訳ではない）
  - 基板のサイズは Arduino と同じギリギリサイズになっていますが、USB コネクタ部や AC アダプタコネクタ部は、Arduino 本体と干渉しますので、そこに部品を載せる場合は工夫が必要です

補足2: 赤外線 LED ハンダ付けの工夫

- 今回 LED は基板に直づけしますが、もしかするとあとでケースに納めたいかたもあるかも知れません
  - その場合、あとで LED をケース部に取り付けるため、LED を切り離しても結構です
  - 切り離したあとは、LED の足にリード線を巻き付けてハンダ付けし、基板上の元あった部分にハンダ付けし直します。上級編としては、コネクタを付けてもいいでしょう。LED が点灯しない場合は極性を見直してください
  - そのような理由から、LED の足は長めにハンダ付けしておくことをお勧めします。短く切ってしまうと、ニッパで切ったり、リード線をハンダ付けしたりするのが大変になります
  - 結論としては、10ミリほど浮かせてハンダ付けしておくと良いでしょう

ハンダ付けのしかた

- 私も実際にデモしますが、
  - プロのハンダ付けには適わない
  - 画面に大きく映したほうが分かりやすい
  - よく作成されたビデオがある

  ので、ビデオで説明します
- https://www.youtube.com/watch?v=x0YxEilyQVY
  - （2020年現在、残念ながら視聴できなくなりました）

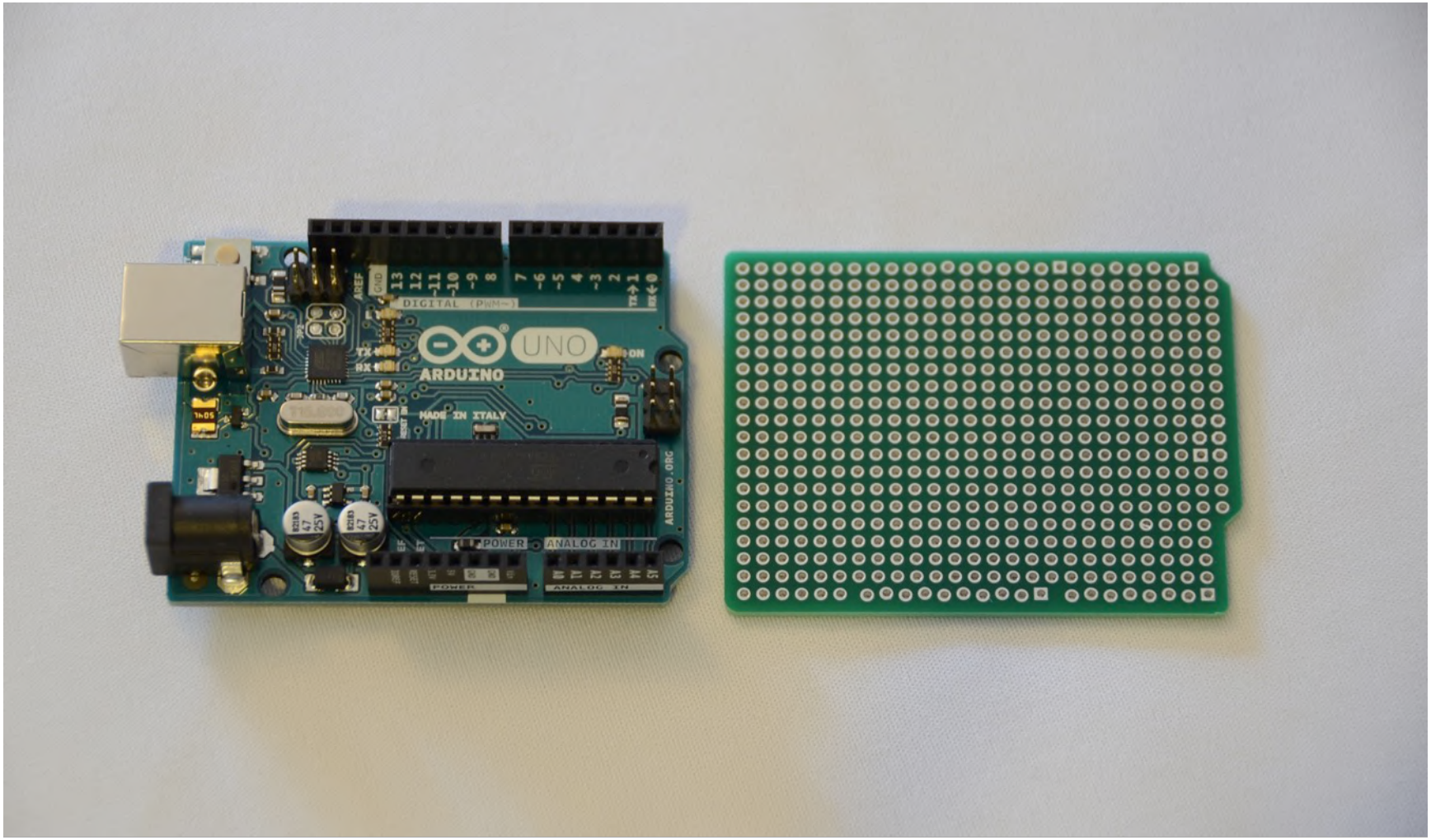
部品表

- 工作を始める前に部品が揃っていることを確認します
  - Arduino Uno R3 × 1（と USB ケーブル）
  - バニラボード（ユニバーサル基板）× 1
  - 8ピンのピンヘッダ × 2（2つ繋がっている場合は、後で切り離します）
  - カーボン抵抗器（許容損失 1/4W（ワット））
    - 160Ω × 9（なお、4つは課題用ですので今日は使いません。以下同様）
    - 22kΩ × 1
    - 47Ω × 1
    - 1MΩ × 1（課題用）
  - 赤外線 LED (L-53F3BT) × 2（1つは課題用）
  - 電解コンデンサ 47uF（マイクロファラッド）, 耐圧 16V, 85℃ 仕様 × 1
  - 赤外線受光モジュール IRM-3638N3 × 1

それでは工作を始めましょう

- ここからは、作業途中の写真を画面に映しておきます
- 実際の作業手順は口頭で説明します

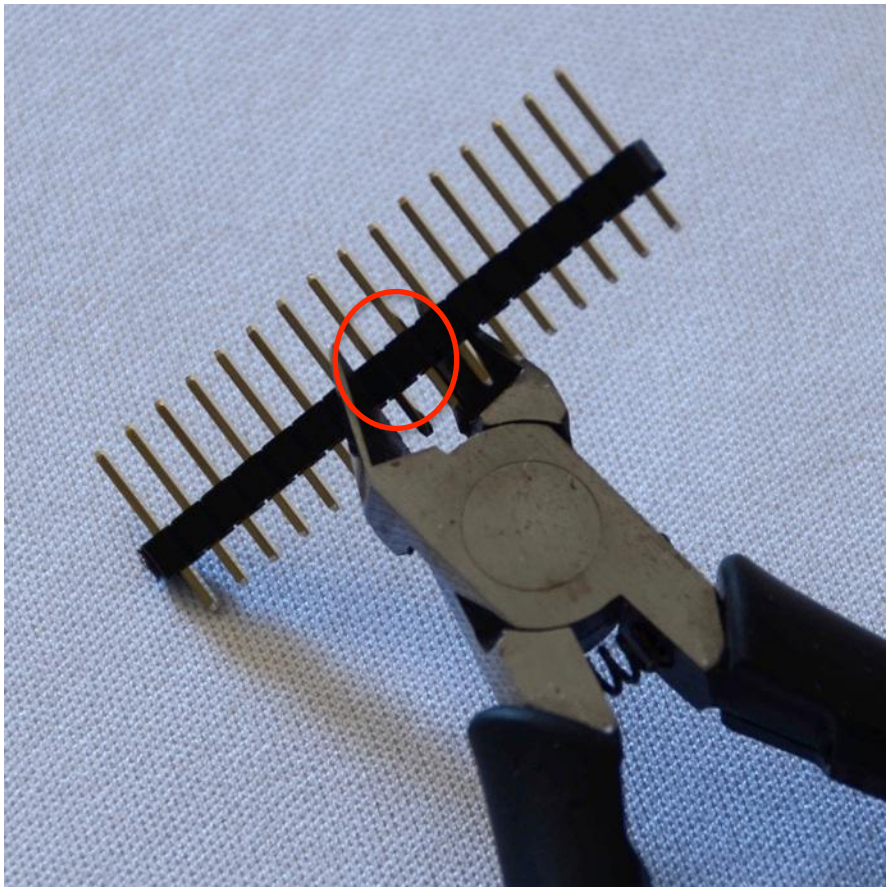
まず基板を用意します



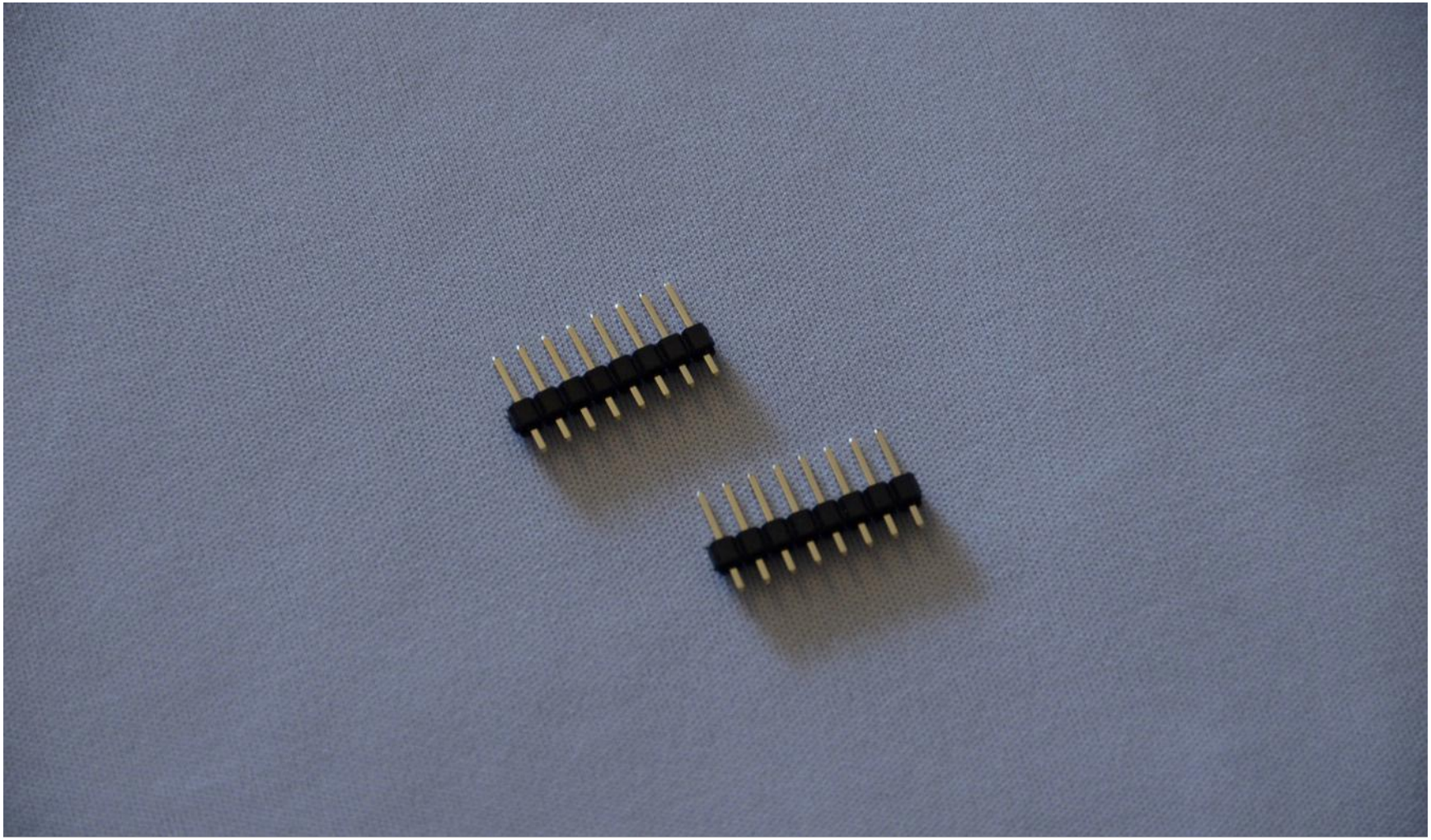
FIRMLOGICS
Embedded Design and Partnerships

最初にコネクタを取り付けましょう

- 部品をハンダ付けする順序の基本は、背の低い部品から高い部品へ、です
- ここでは基板の座りを良くすることも考え、コネクタから取り付けていきます
- 間違って 1ピンだけ切り離すとハンダ付けが大変ですので、8ピンと 8ピンになるように中央で切断するよう、注意しましょう
- ピンを押さえずにニッパで切断すると遠くに飛んでいきますので、注意!

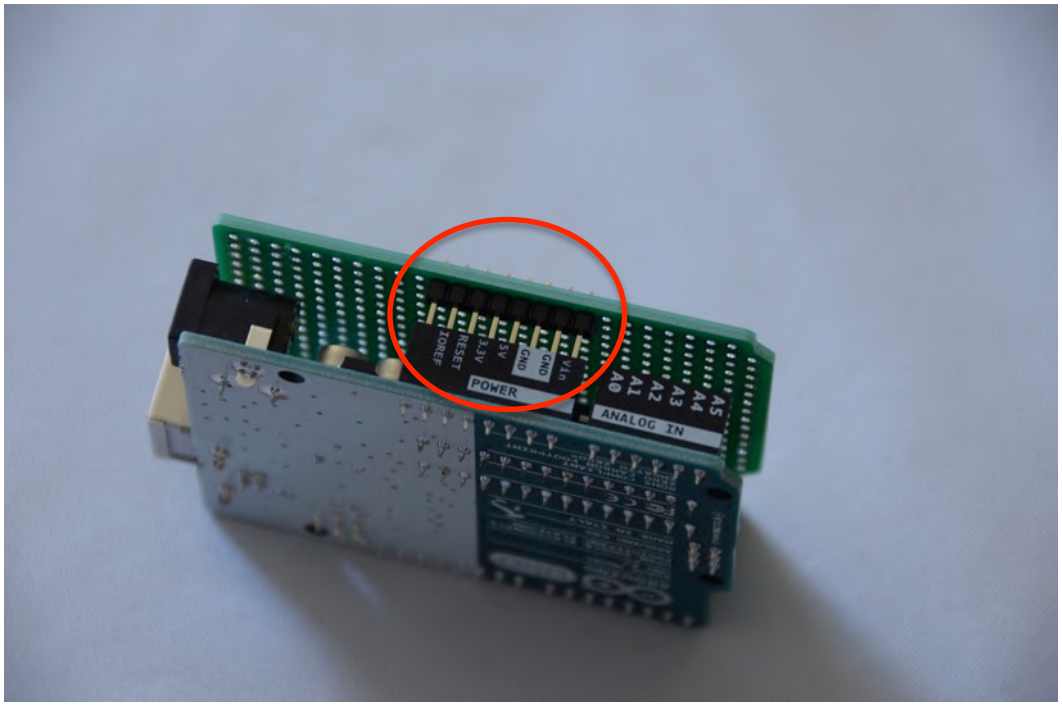


コネクタを 8ピンずつに切り離しました



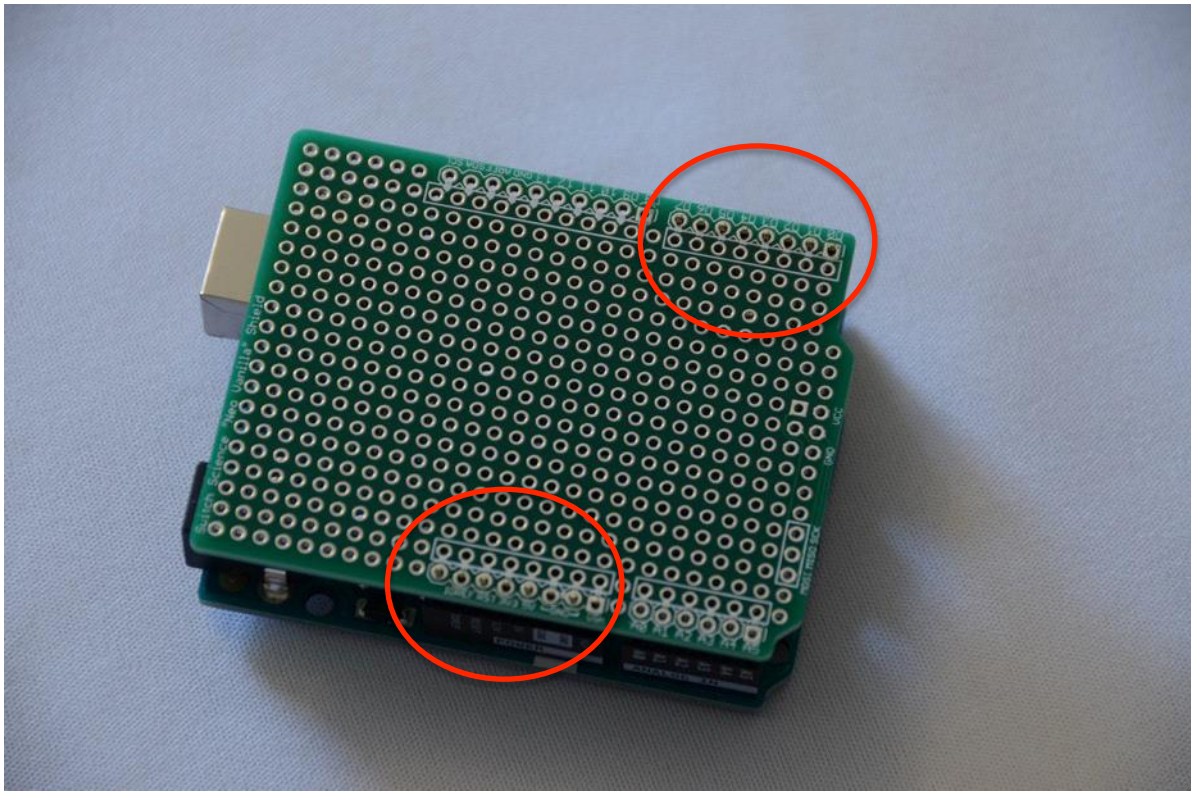
裏技: コネクタを差した状態でハンダ付け

- このユニバーサル基板には工夫（ジグザグハック）があるらしいですが、一般に「一列にピンのある部品」を垂直にハンダ付けするのは難しいです。そのため、コネクタを嵌合させた状態でハンダ付けする裏技を使います



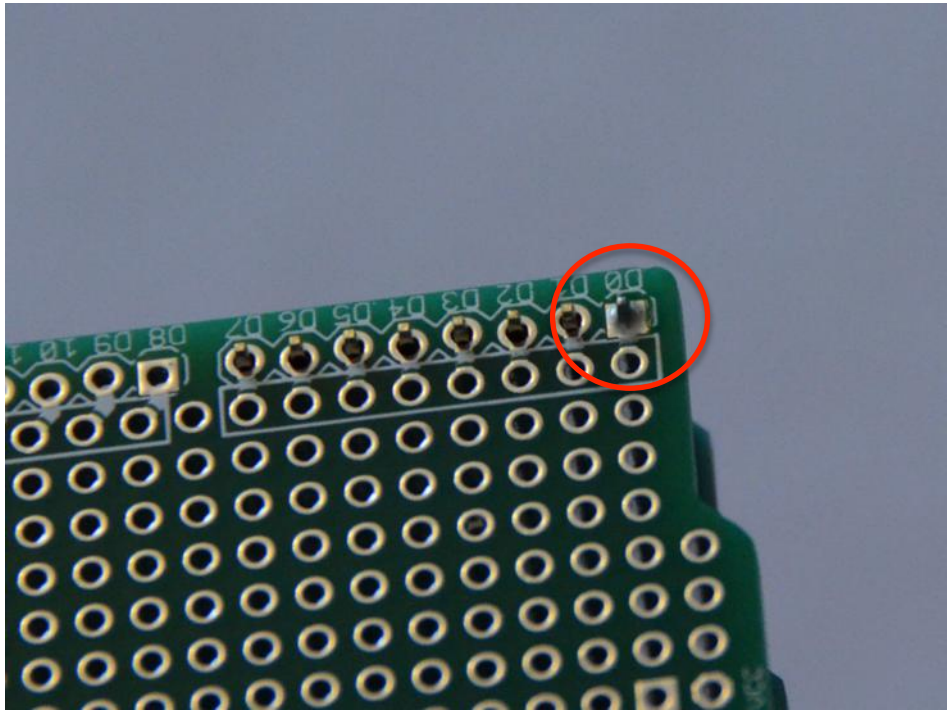
そうすると、コネクタがまっすぐ立ちます

- ハンダ付け位置の間違いも避けられます



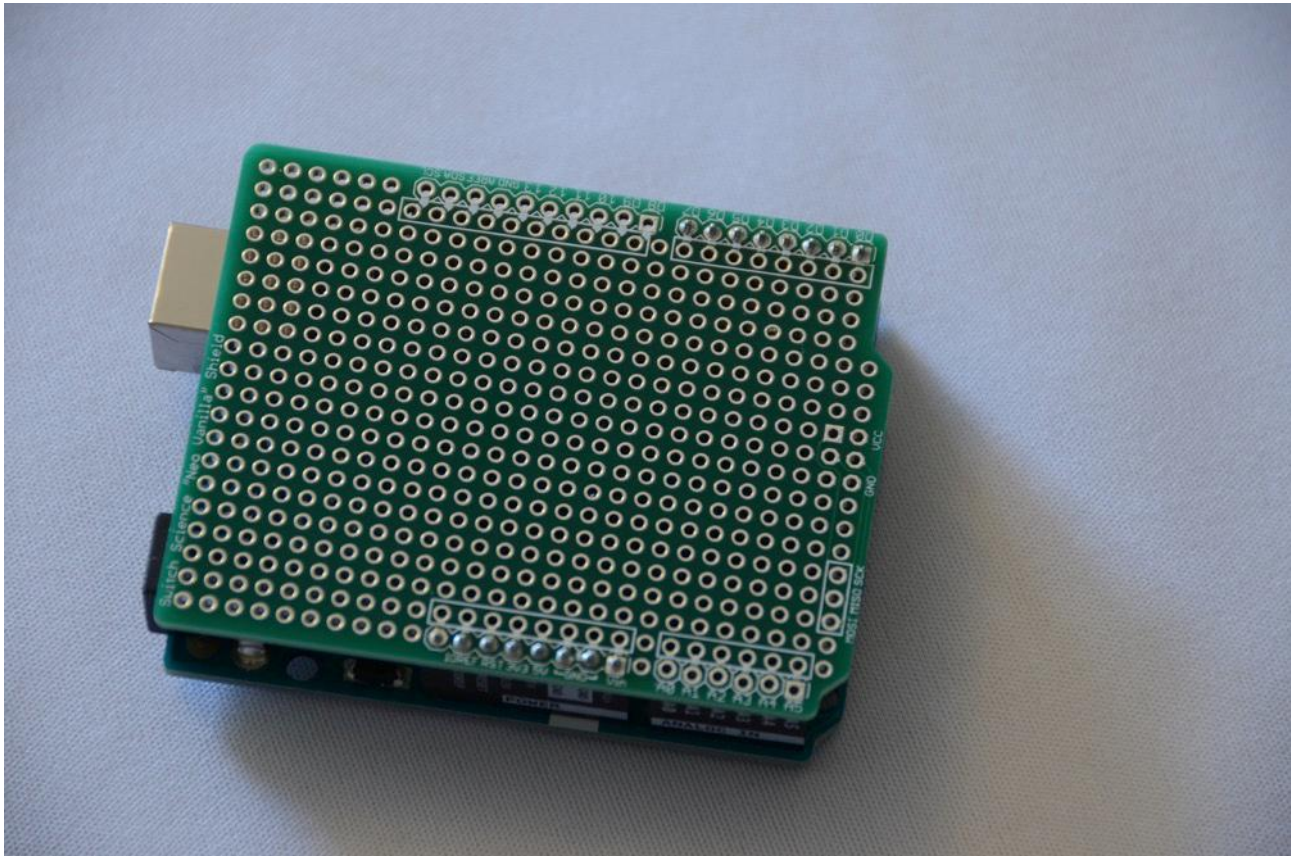
まず最初に、端の 1ピンだけハンダ付け

- 多少の高熱は大丈夫ですので、慌てずに作業しましょう
  - 10秒以上コテを当てた場合は、一度コテを離して冷ましましょう
- ピンが浮いていないか確認しましょう
- ここで、私か周りの経験者に見てもらってください



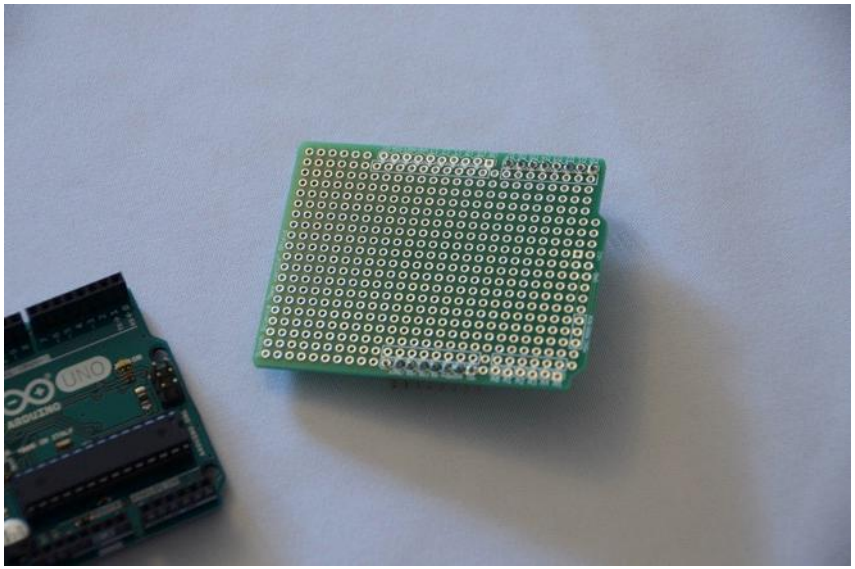
隙間がないことを確認してから

- 残りのピンをすべてハンダ付けします



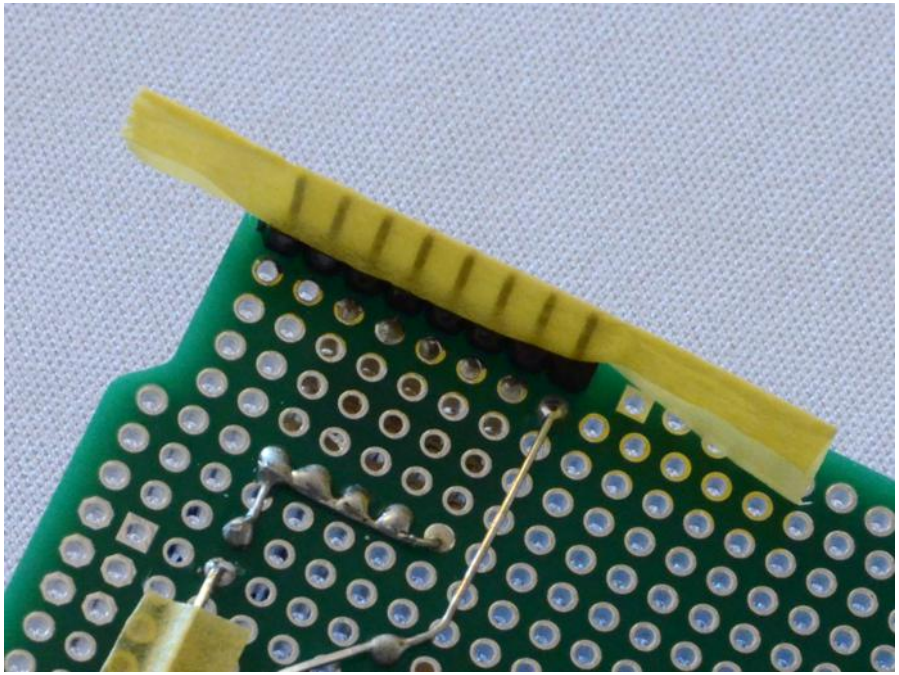
基板を抜き取ります

- かなり固いと思いますので、均等に力をかけて少しずつ抜いてください。コネクタピンが曲がらないように（多少は大丈夫です）
- 全ピンのハンダ付けが終わる前に基板を抜くと、ピンが嵌合相手に残ってしまうことがあるので注意してください



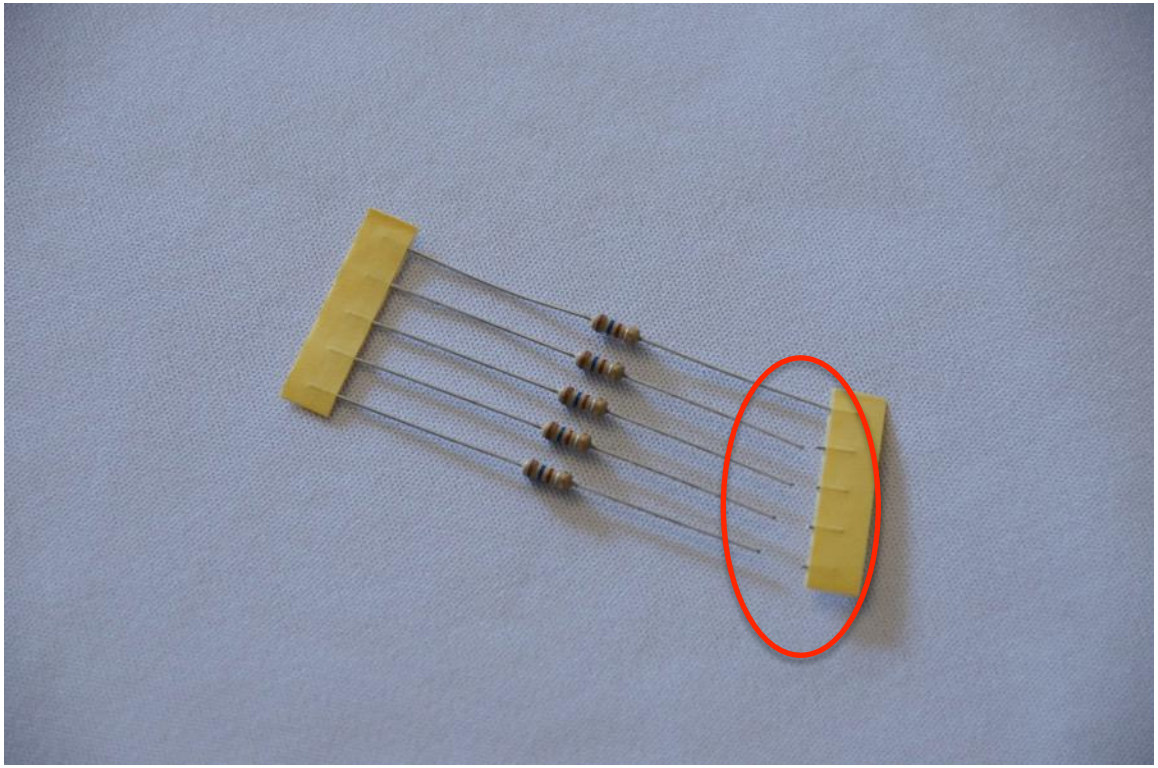
マスキングテープで端子を保護します

- コネクタは、嵌合部の端子（ピン）の接触が「生命線」です。ハンダ付け中にハンダを付着させないように、（糊が残りづらい）マスキングテープで保護しておくことをお勧めします



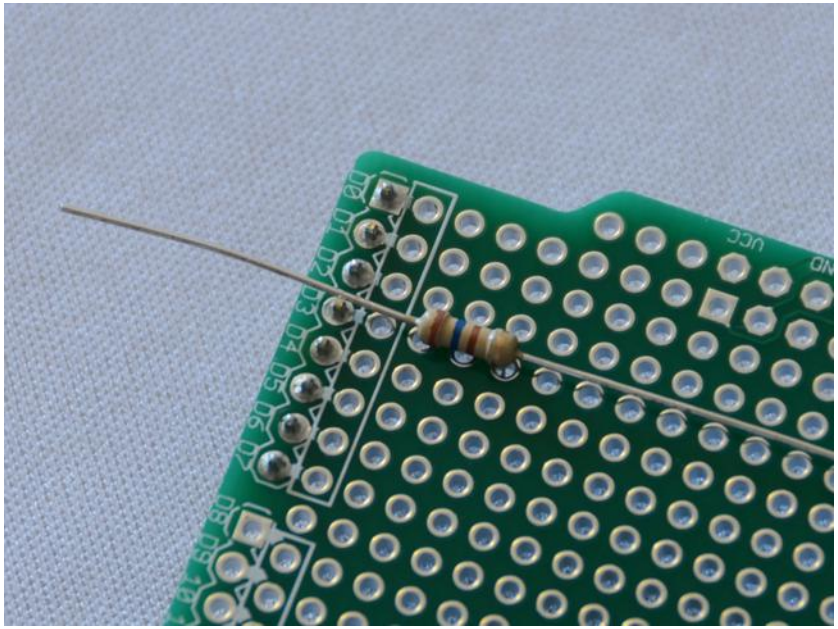
抵抗器を取り付けます

- 抵抗器はリード線がテープに接着されているので、リード線ができるだけ長く残るように切り離してください

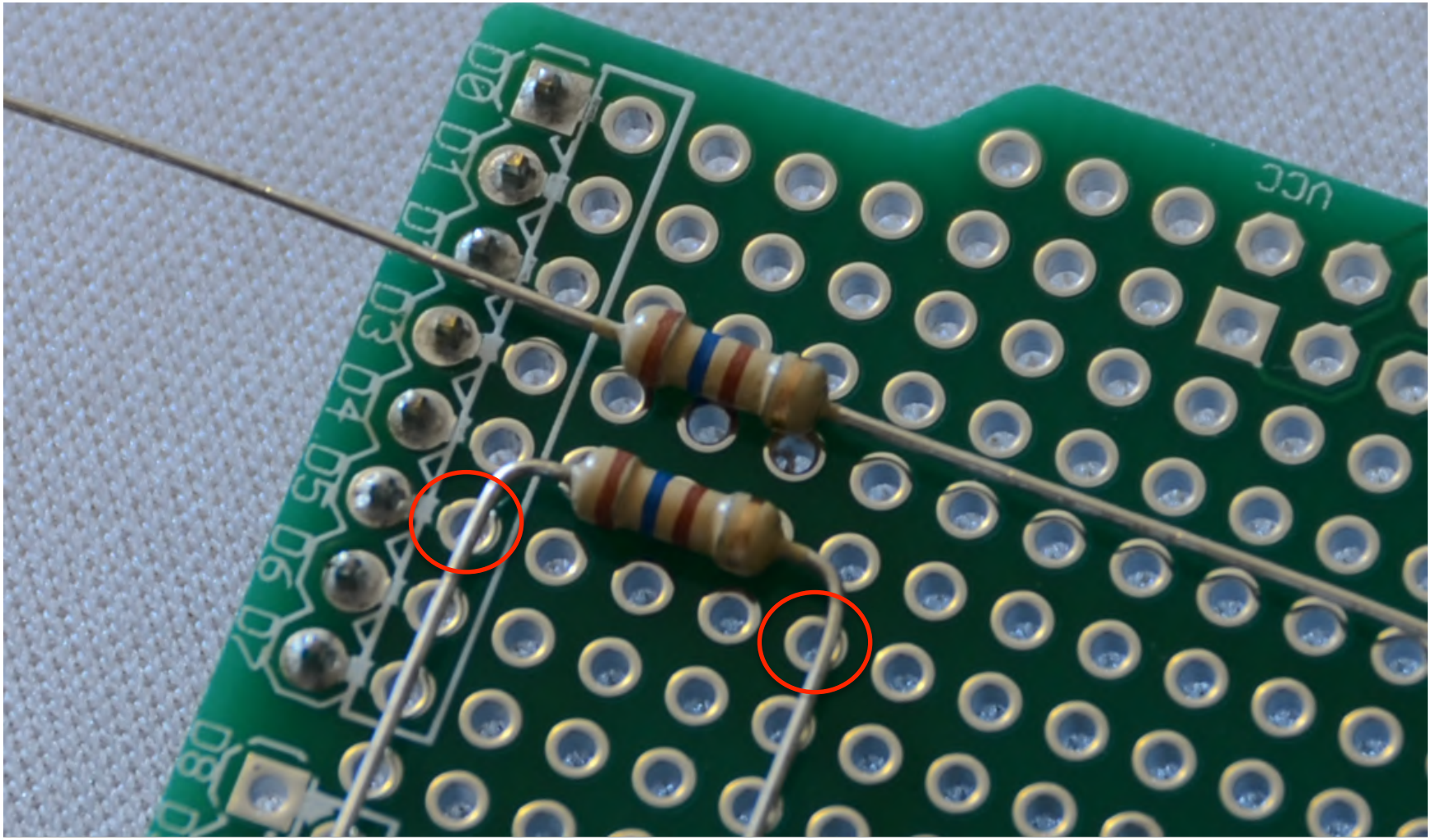


抵抗の値と取り付け場所を確認します

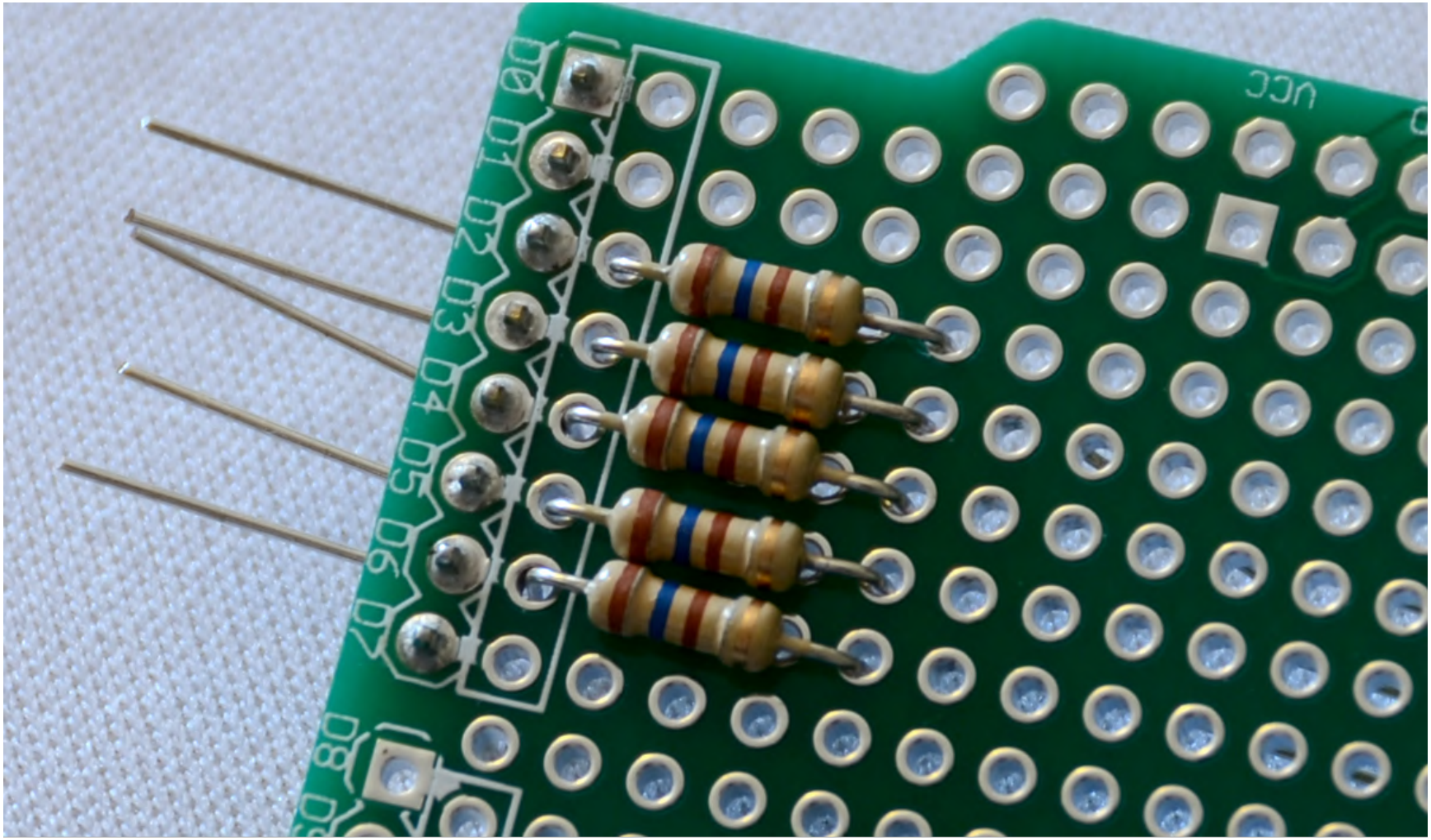
- ここで使う抵抗器は、160Ω（オーム）です。カラーコードは茶・青・茶・金です
- リード線を曲げる長さも、このとき確認します
- 多少長さが違っても反対側のリード線で調整できるので、一度に両方を曲げず、片側を曲げてから一度基板に挿入し、反対側の長さを決めましょう



うまく曲げるとこんな感じです

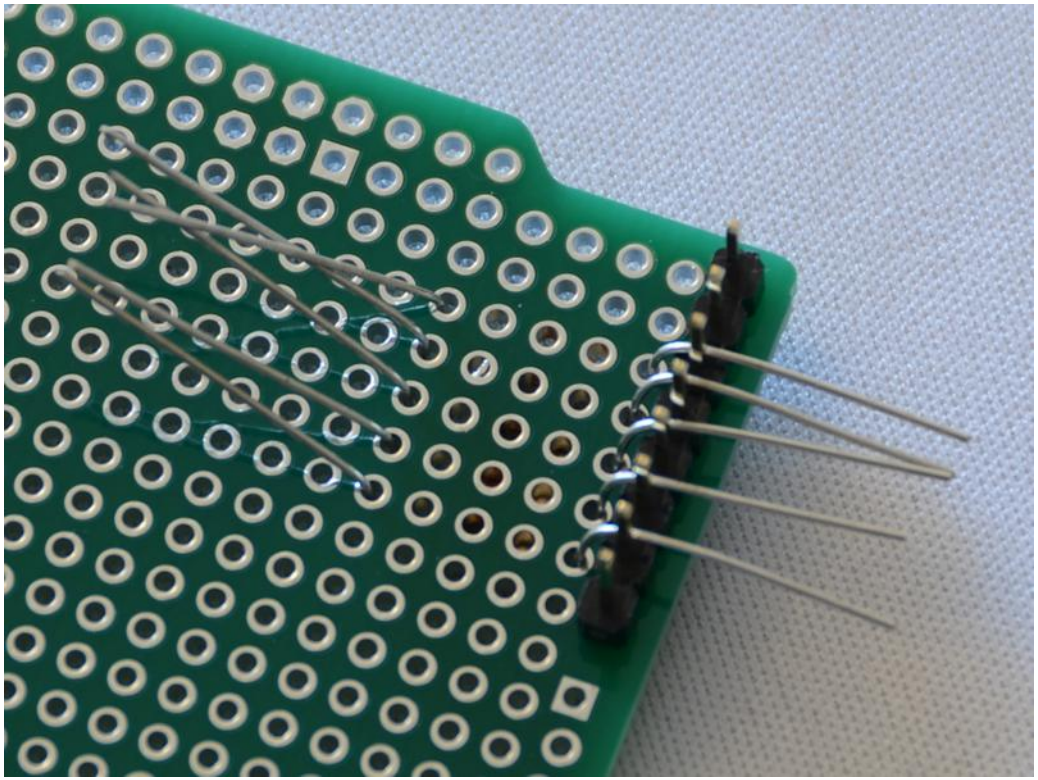


5本できたら挿入します



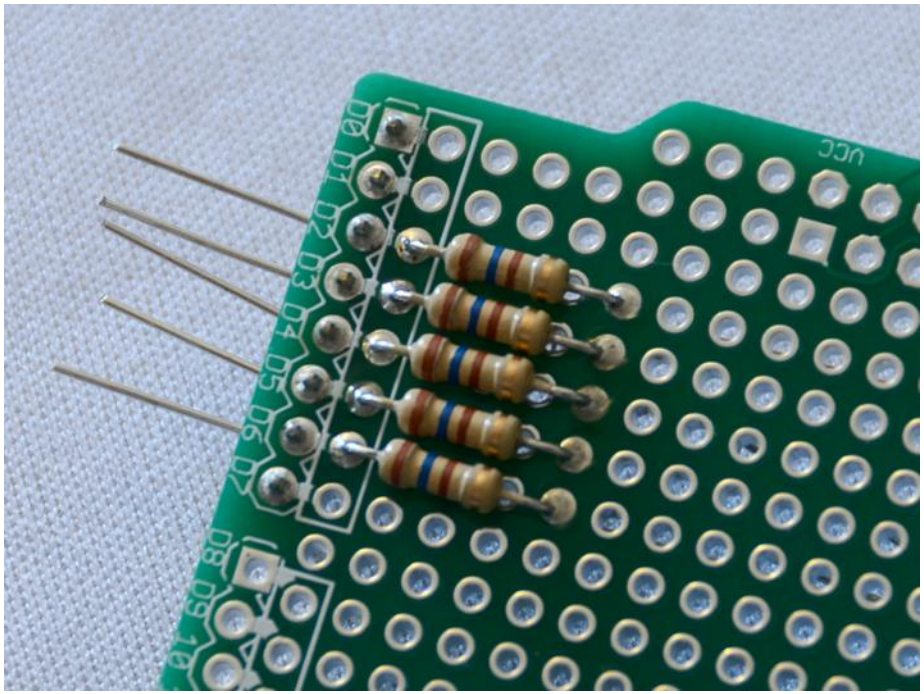
裏側はこんな感じです

- 部品が抜けないよう、リード線を両側に（あとでハンダ付けの邪魔にならない程度に）軽く曲げておきます



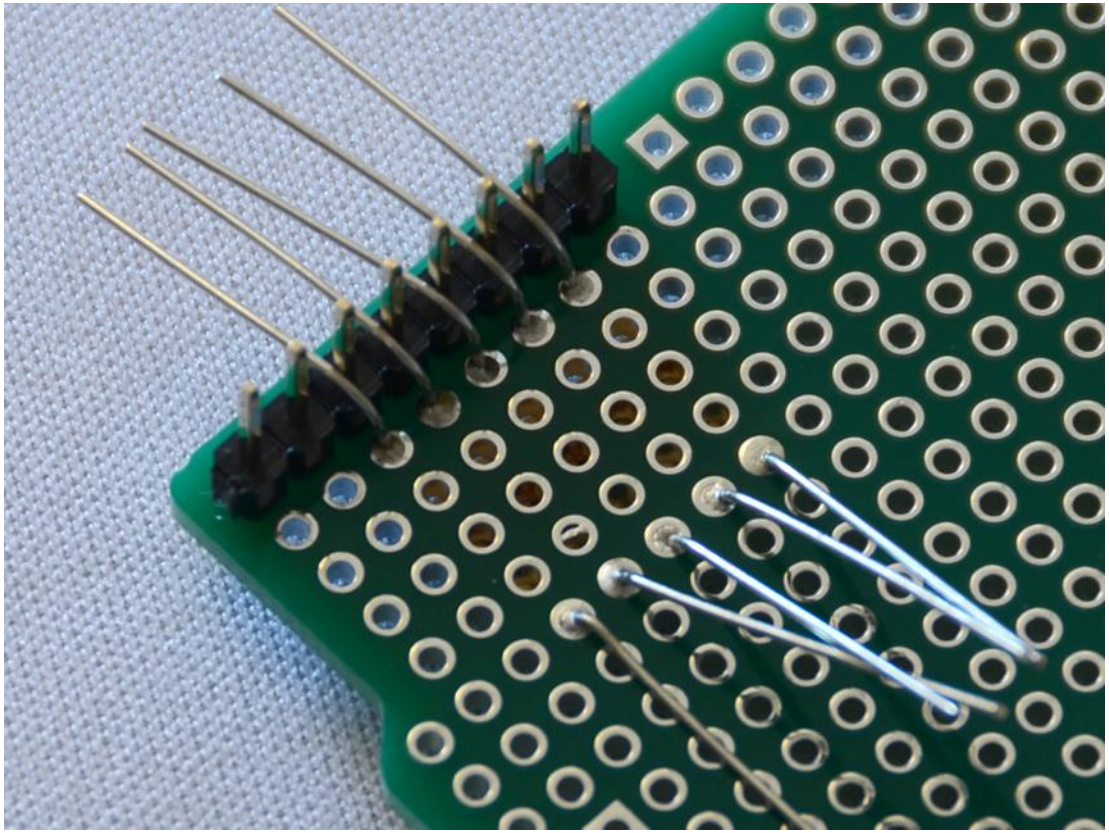
抵抗器もハンダ付けします

- この基板は両面スルホール（through hole）といって、どちらの面からでもハンダ付けできますが、ここでの抵抗器は表面からハンダ付けしたほうが楽だと思います



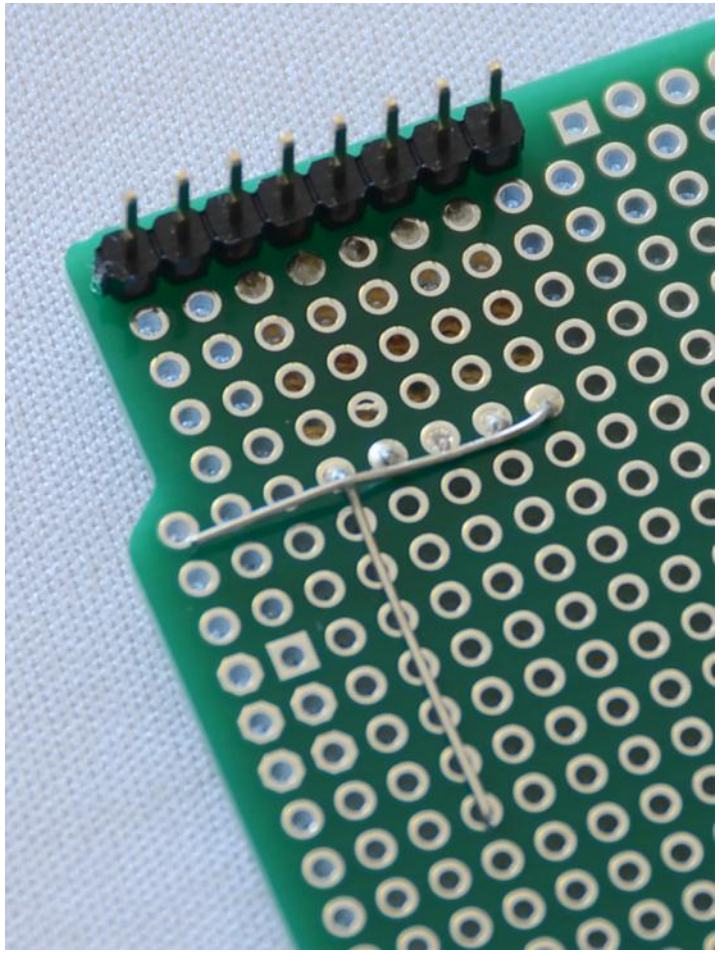
裏側を確認します

- 裏面まである程度ハンダが浸透していることを確認します



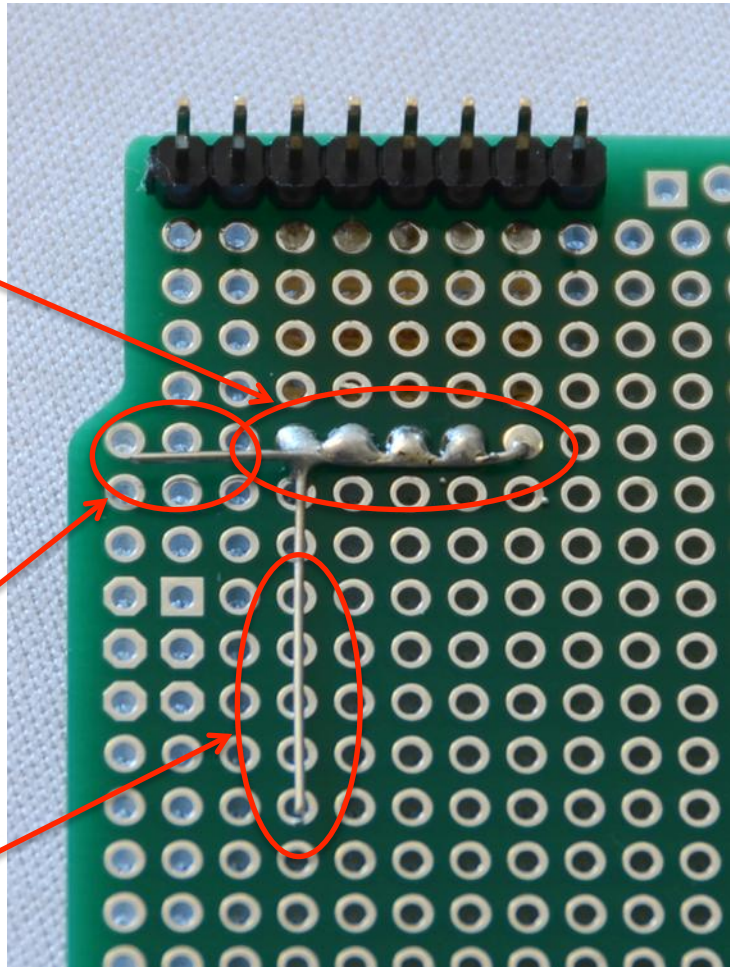
ここからは配線作業を伴います

- 別途リード線を使っても良いのですが、せっかく抵抗器にリード線があるので、それをうまく使うと楽です
- 写真のようにリード線を曲げて残し、一方で不要なリード線は切断してください（間違って切断してしまってもハンダ付けすればいいだけなので、考えすぎなくても大丈夫です）
- 切断したリード線は捨てずに取っておいてください



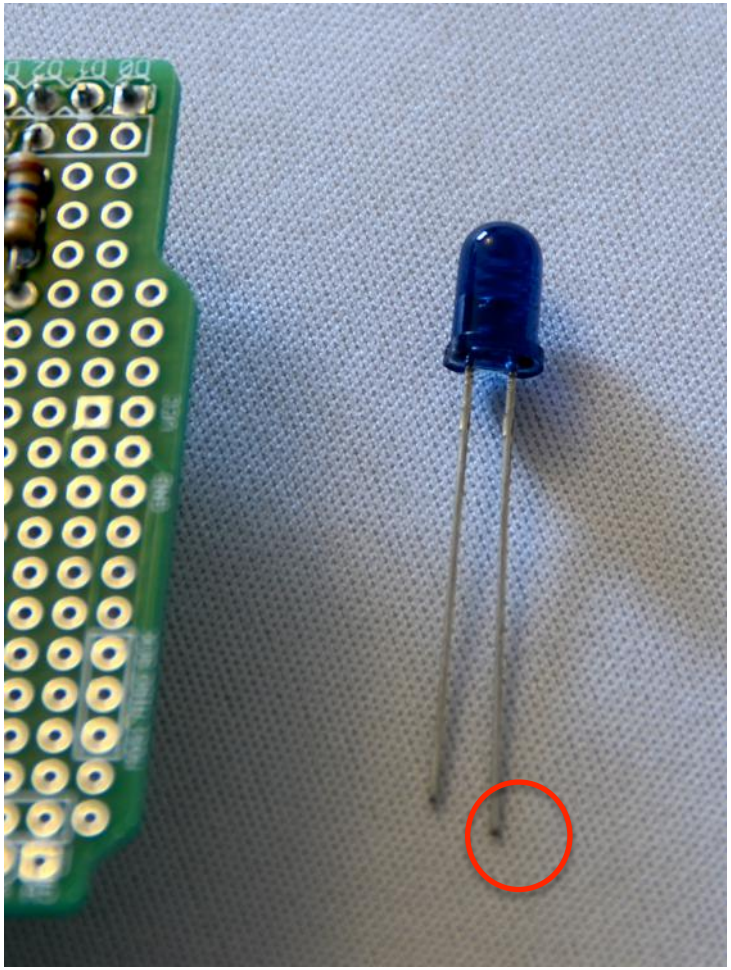
抵抗器の（片側を）繋いでしまいます

- 実体配線図を確認しながら、抵抗器の、コネクタと反対側のリード線をすべてハンダ付けで接続します。コネクタ側は繋がないように!
- 若干ハンダが盛り気味になると思いますが、くっついていれば（たぶん）大丈夫です
- 繋いだら、左にはみ出たリード線は不要なので切断します
- 下側のリード線は、LED に接続するために残しておきます



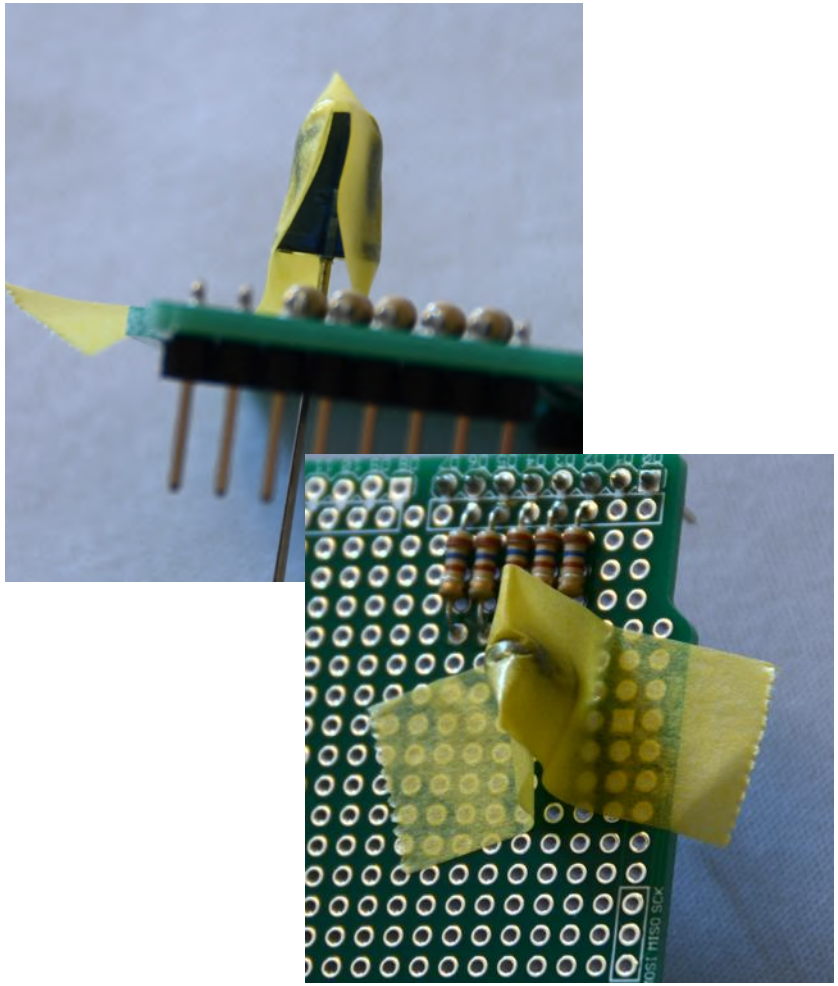
次に LED を取り付けます

- 先ほどの説明の通り、LED には極性があります。写真で右側の少し長いリード線が +（anode）です
- + のリード線が抵抗器側（写真上側）になるように取り付けます



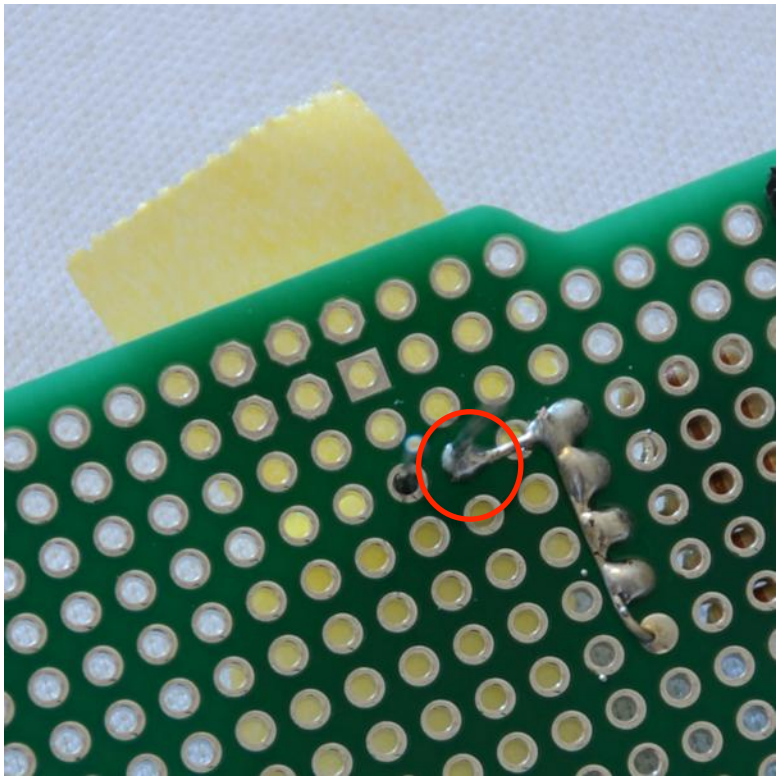
LED を少し浮かせるためテーピングします

- 後で LED をケースの外に出したくなるかも知れませんので、LED は浮かしてハンダ付けしましょう
- このようなとき、紙製のマスキングテープが役立ちます
- 10ミリくらい浮かせておくと良いでしょう。ハンダ付け作業中に埋まってしまいがちなので、気持ち長めのほうが良いと思います
- まずは片側リード線だけハンダ付けします。そうすれば、後で修正がききます（両方ハンダ付けしてしまうと修正しづらい）



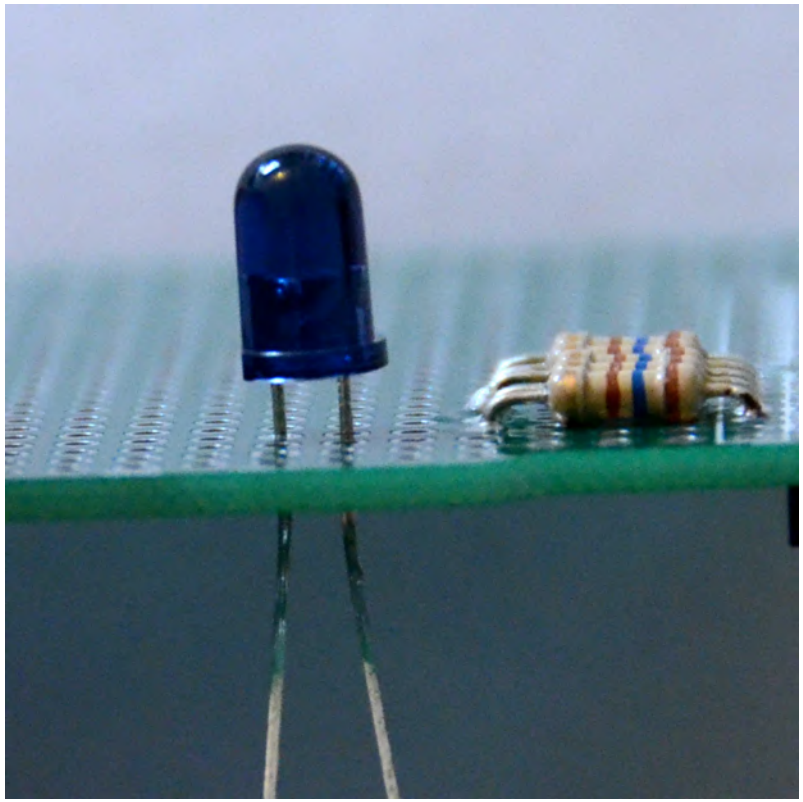
まずは + 側だけハンダ付けします

- 抵抗器からのリード線をちょうど良い長さ（LED の反対側と短絡しない程度）に切り落とし、+ 側（抵抗器に繋がる側）だけハンダ付けします



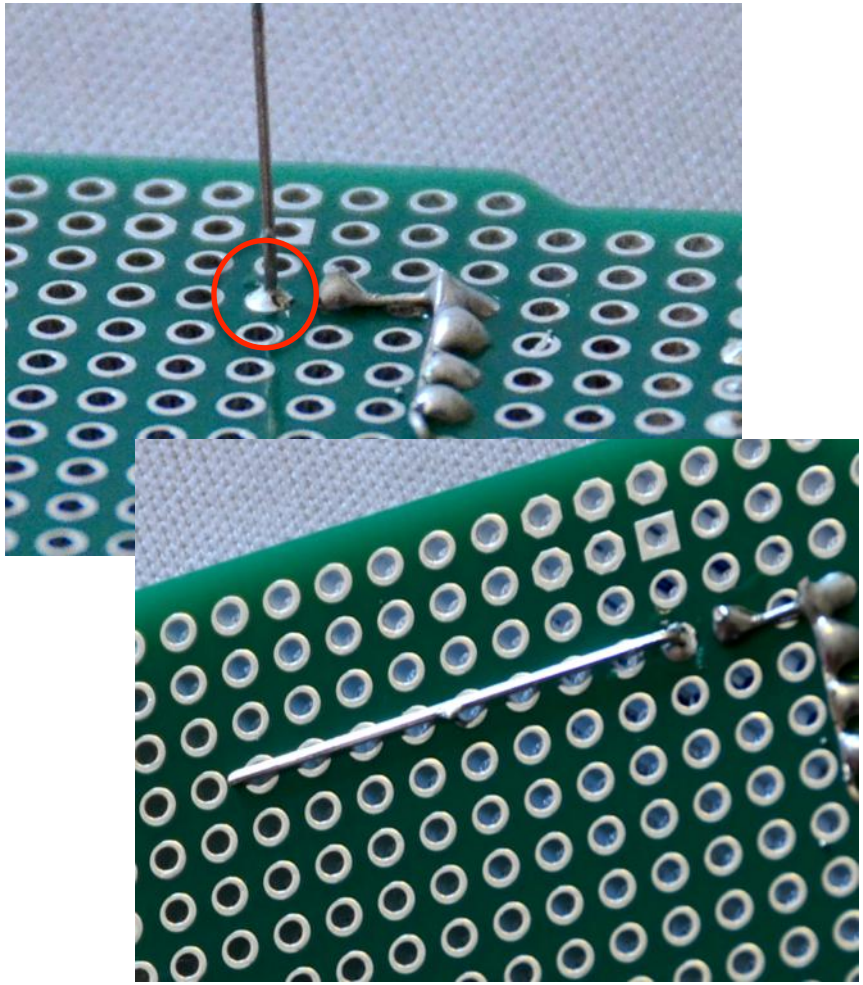
LED が真っ直ぐついているか確認します

- ピンは多少は曲げても大丈夫です。慣れないうちは、無理にハンダ付けし直さないほうが楽でしょう
- たくさん曲がっている場合は、もう一度ハンダを溶かして傾きを直します。難しい場合は、私か周りの人に手伝って貰ってください



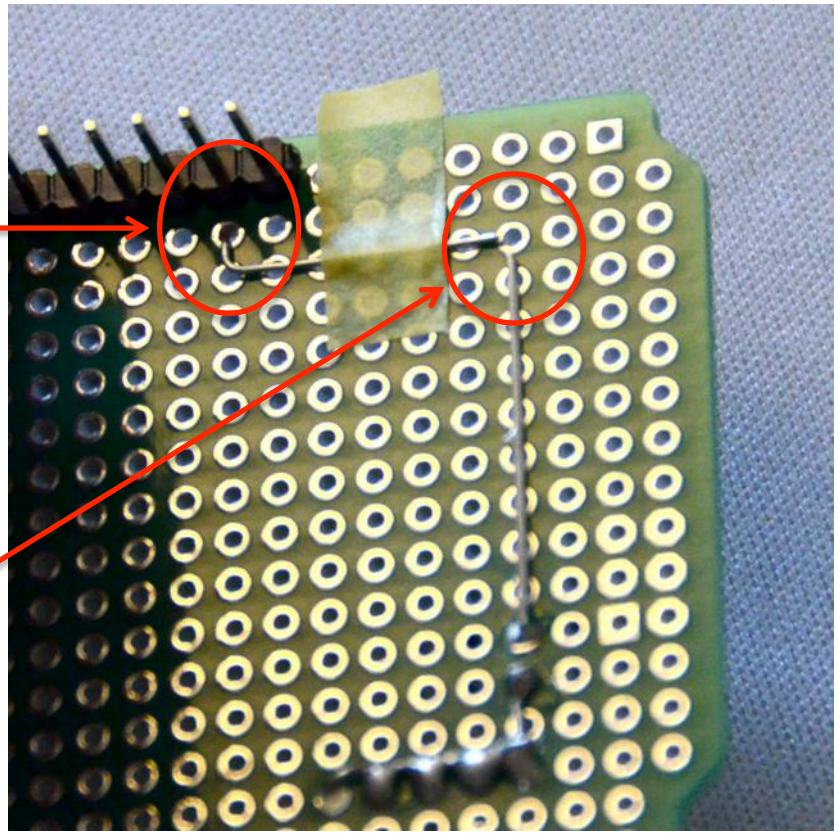
反対側のリード線もハンダ付けします

- 両方のリード線を一度に加熱すると（器用なことだが）LED が傾いてしまうので、傾きをいじらないようにしながら、反対側のリード線を「そっと」ハンダ付けします
- 抵抗器側のリード線は不要になるので、余分なリード線は切断してください。－ 側は配線を続けるので、写真左側に曲げて、残します



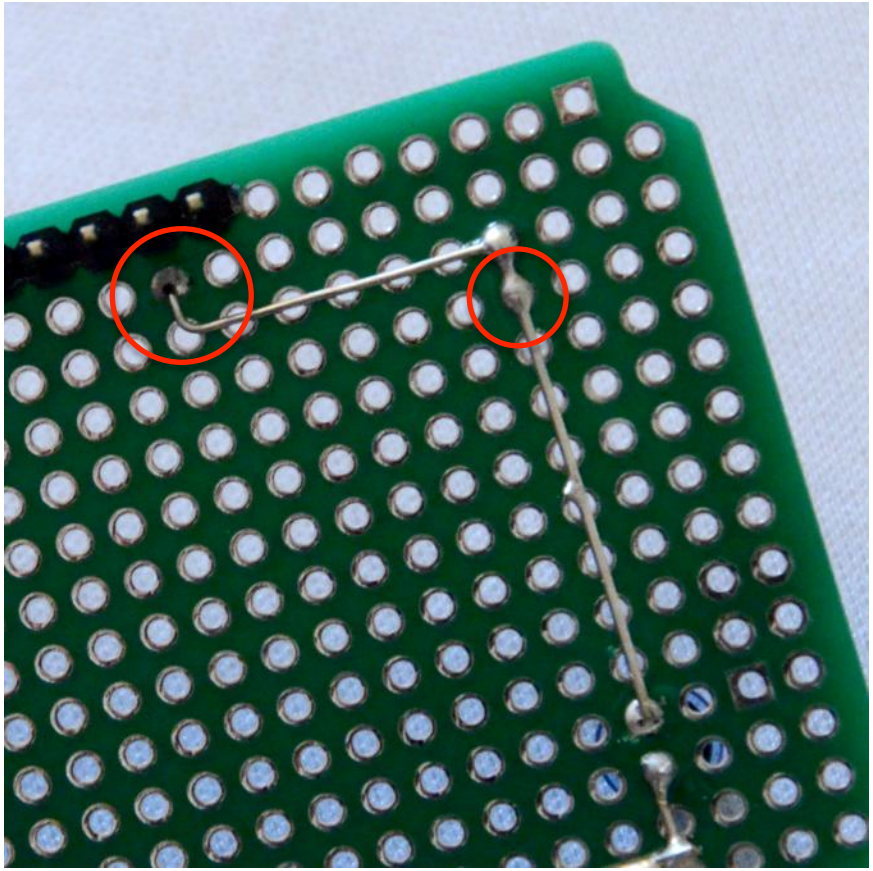
LED の –（マイナス）側を配線します

- LED の –（マイナス）側は、コネクタの GND（写真で右から 2番目か 3番目）と繋ぎます
- 余っているリード線を使って写真のように配線します
- 左上のリード線は、先を少し（1ミリ程度）曲げて穴に差しておきます。指先の器用な人は私よりも綺麗にできるはず!
- ここでもマスキングテープを使って固定すると、作業に非常に楽です
- なお右上の接続点が届かない場合は、双方のリード線を斜めに曲げてハンダ付けしてもオーケーです（リード線を長めに切り落としておく理由はここにあります）



ここまでで、赤外線送信部は完成です

- 写真にあるように、途中途中で基板上の穴（ランド）にハンダ付けしても結構です。機械的強度が高まります。あまりやり過ぎるとハンダの浪費になります
- ここまでできたら、ハンダ付け忘れがないことを確認して、休憩してください



ブレイク: LED の点滅を確認しましょう

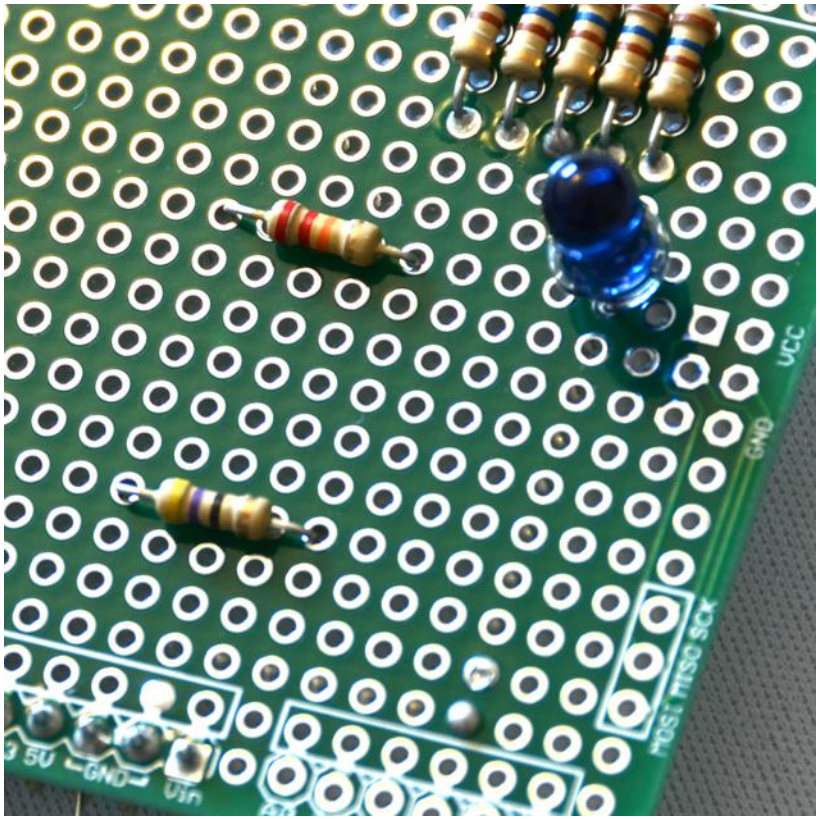
- ここまでの工作で、赤外線の送信ができるようになりました
- Arduino 基板に載せて試してみましょう
- 最初のほうにあった LED 点滅コードに、次のように追加してください。ピン 2〜6 に赤外線 LED が繋がっていますが、ここでは 1つだけドライブ（駆動）します
- 既に説明したように、ここで 5ピン全てをドライブすると LED が壊れる可能性があります（一度の点灯時間が、100 ミリアンペア × 1秒では長すぎるため）
- デジカメ持参の方からお借りして、赤外線が点滅していることを確認してください

```
// the setup function runs once whe
void setup() {
  // initialize digital pin 13 as a
  pinMode(13, OUTPUT);
  pinMode(2, OUTPUT);
}

// the loop function runs over and
void loop() {
  digitalWrite(13, HIGH);   // turn
  digitalWrite(2, HIGH);
  delay(1000);              // wait
  digitalWrite(13, LOW);    // turn
  digitalWrite(2, LOW);
  delay(1000);              // wait
}
```

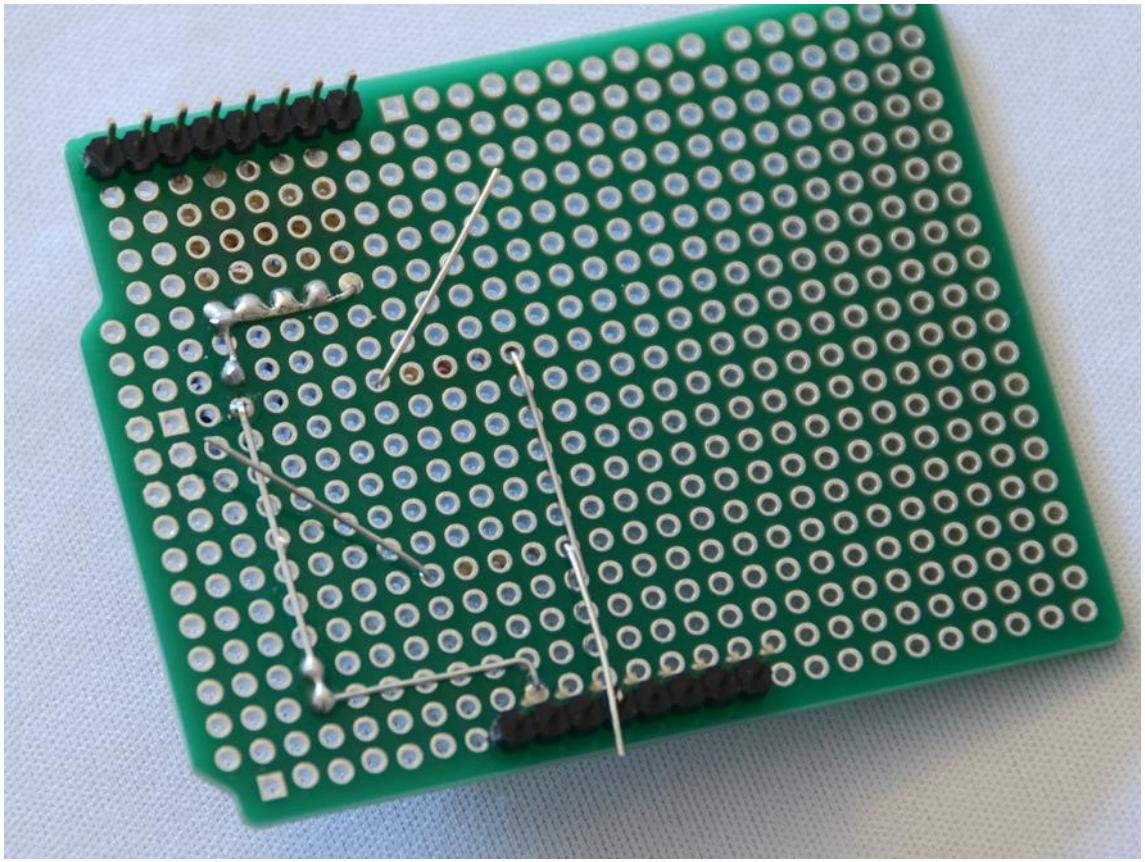
次に赤外線受信部を組み立てます

- まずは 2つの抵抗器を載せます。リード線を適切な方向に曲げてハンダ付けします
- 抵抗器の抵抗値はそれぞれ異なるので注意してください
  - 上は 22kΩ（キロオーム）: 赤・赤・橙・金です
  - 下は 47Ω（オーム）: 黄・紫・黒・金です



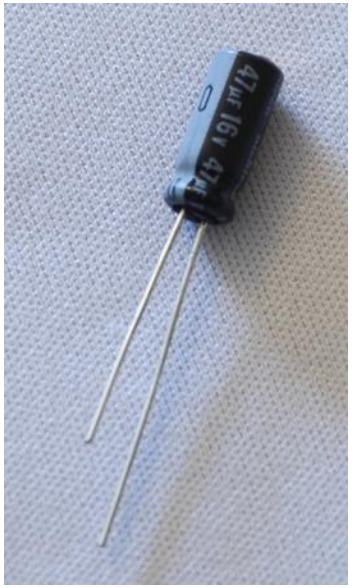
裏側はこんな感じです

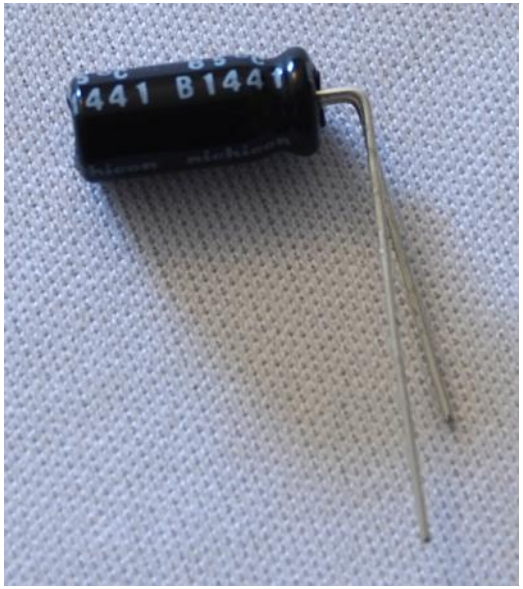
- 実体配線図を参考に、足を曲げています

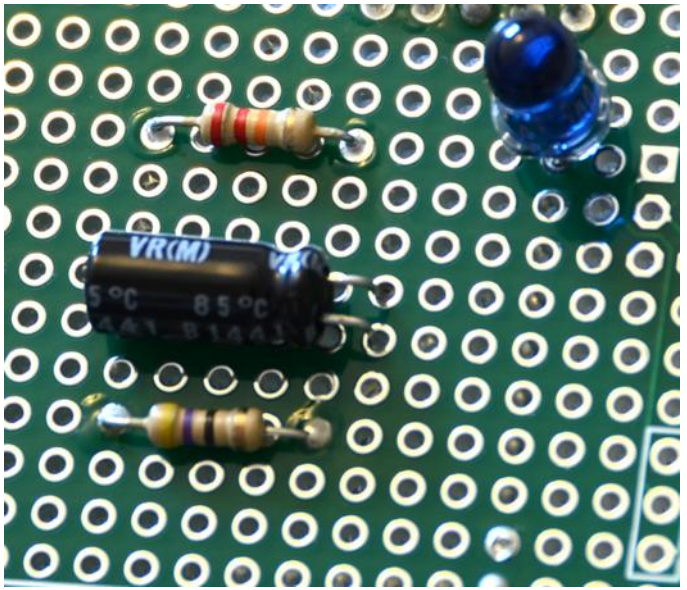


電解コンデンサも載せます

- 先に書いたように、－（マイナス）側には白い帯があります。また、通常は＋（プラス）側のが長くなっています
- リード線を写真（右）のように曲げておきます。ラジオペンチで支えながら曲げると、コンデンサに負担がかかりません

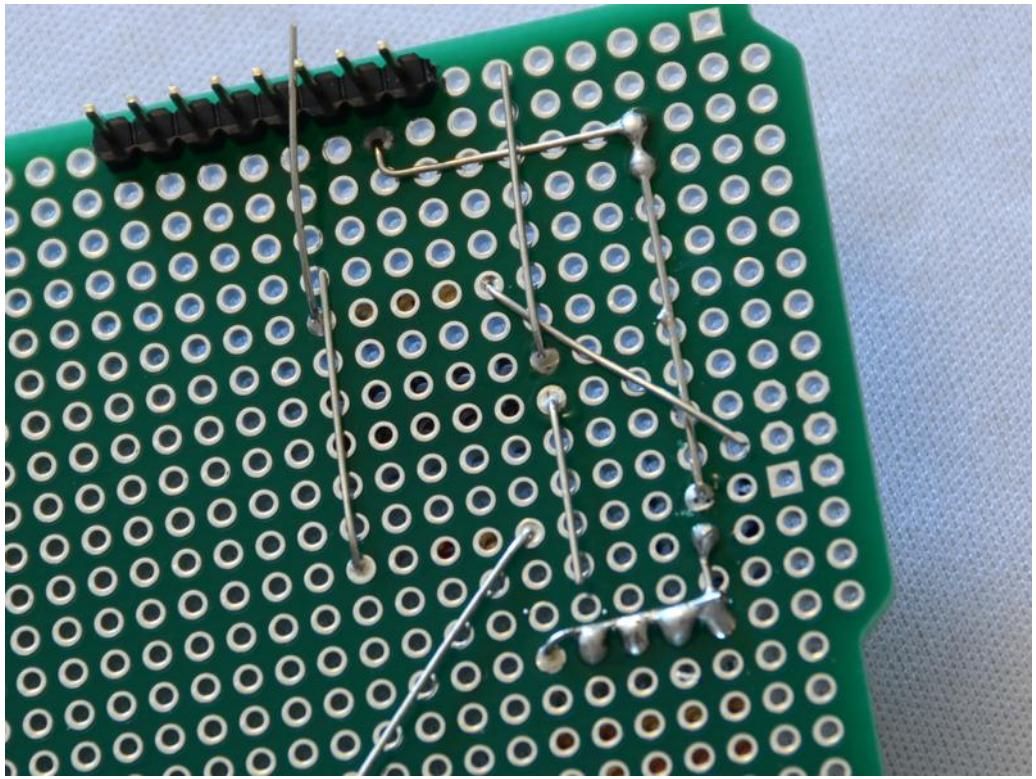






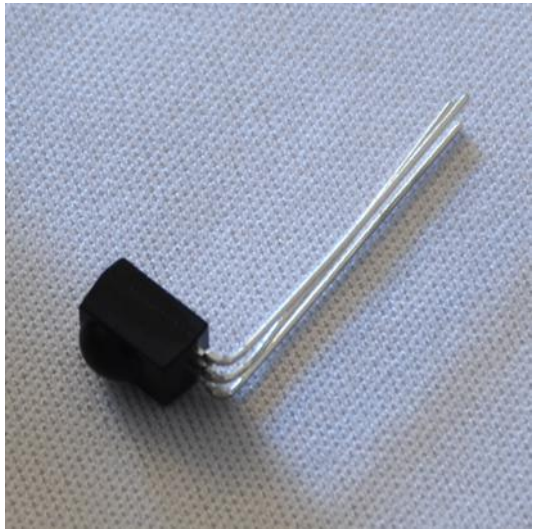
電解コンデンサをハンダ付けしました

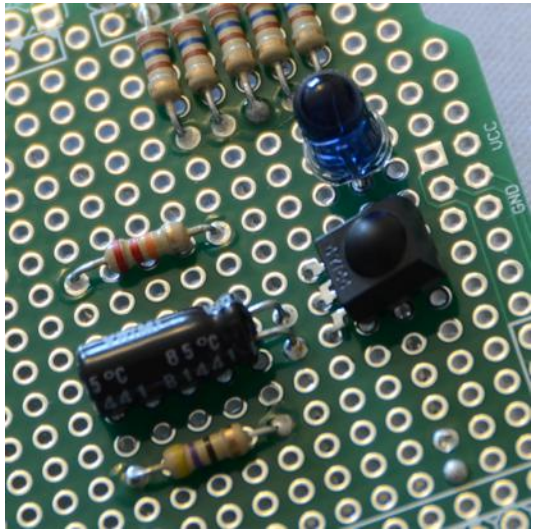
- やはり、リード線は切らずに残していますが、慣れてきたら適切なタイミングで切り落としたほうがハンダ付け作業が楽かも知れません

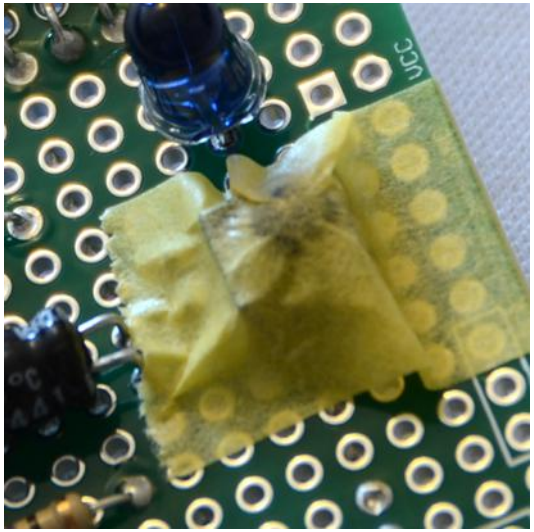


続いて赤外線受光素子を載せます

- これも同様にリード線を曲げてハンダ付けします
- 浮いてしまいハンダ付けしづらいので、マスキングテープで固定しましょう

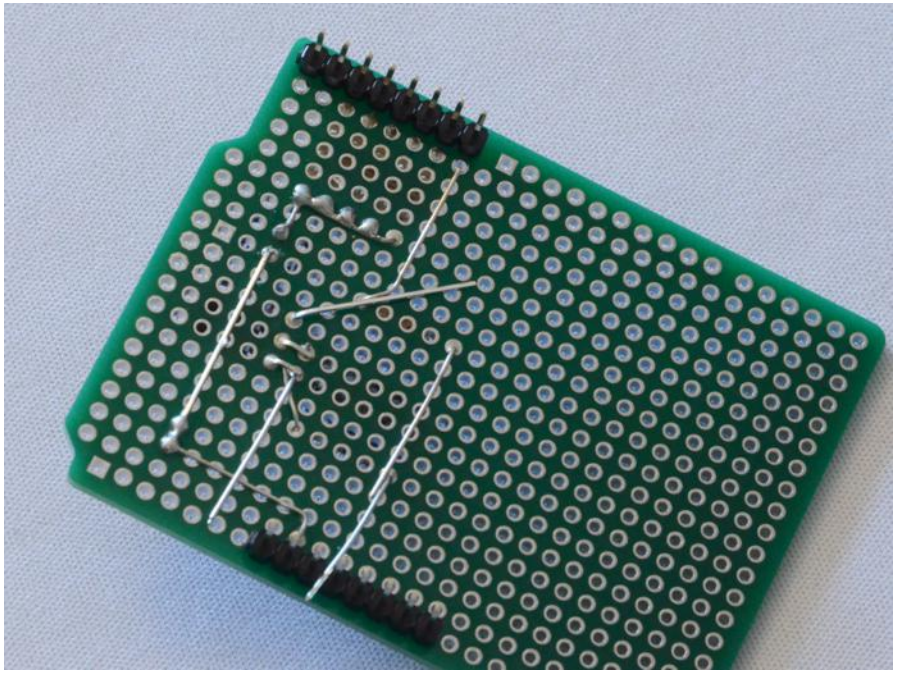




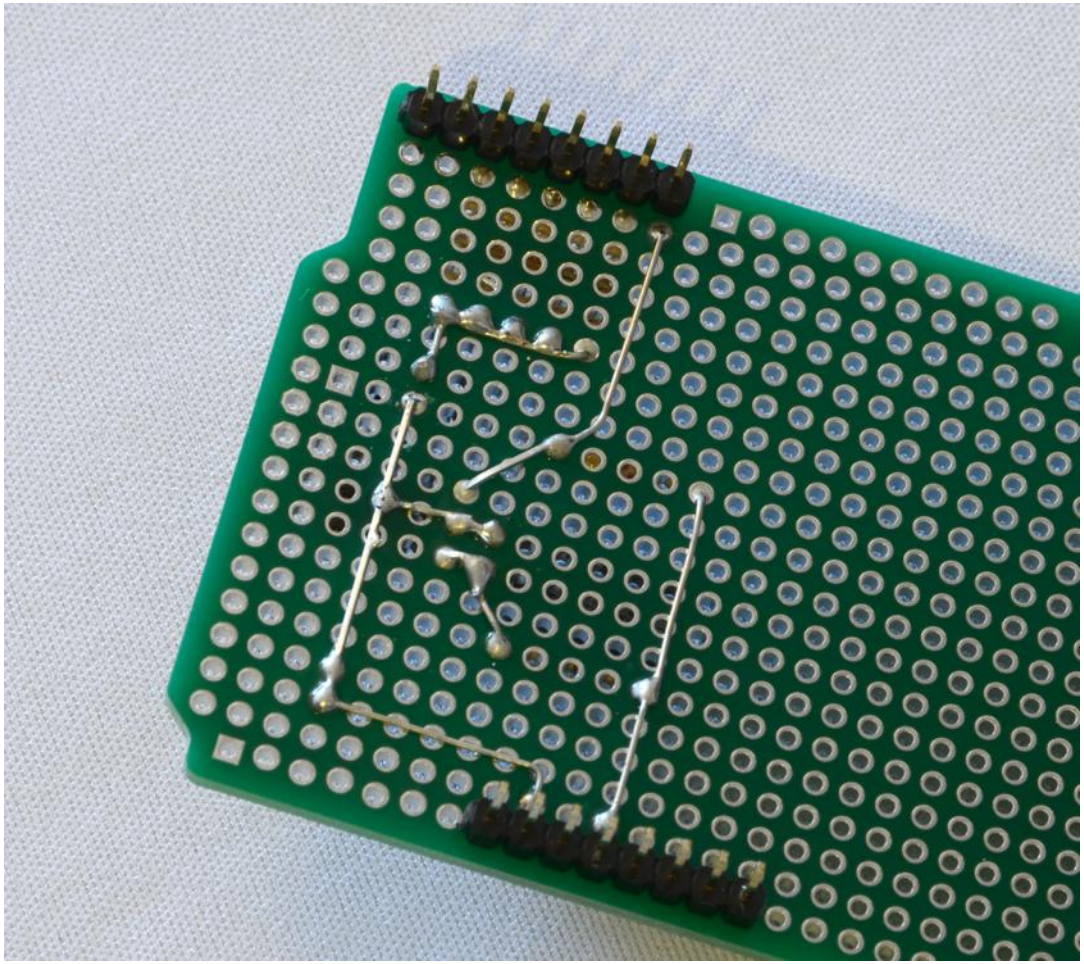


裏側を見てみましょう

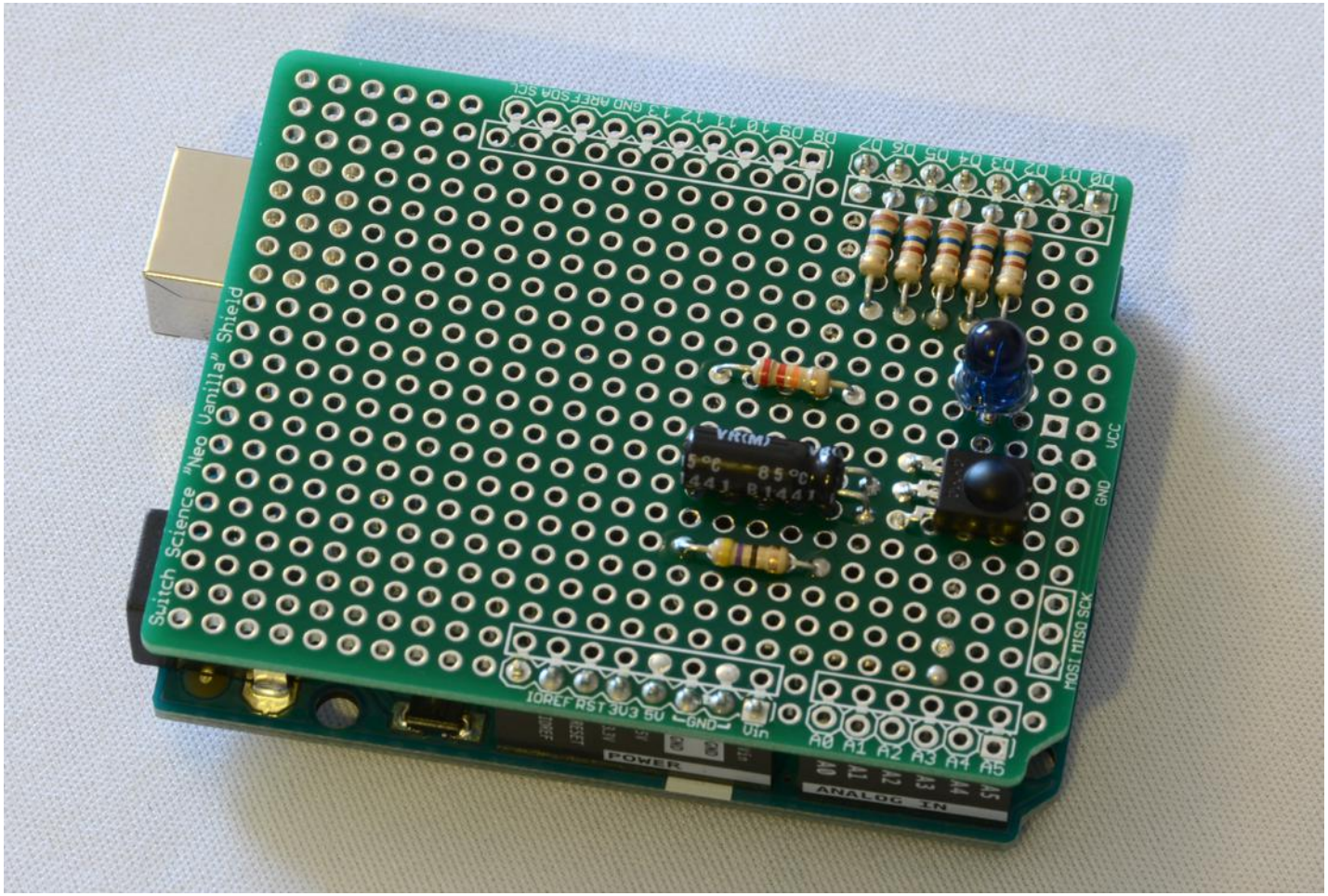
- 同様に実体配線図を見ながらリード線を曲げ、ハンダ付けしていきます
- 余分なリード線は適時、切断していきます



最終的にこのような形になります



表側はこんなです



まずは、おつかれさまでした!

サンプルプログラムを動かそう

- まずは、私が用意したサンプルプログラム ir_recv を動かしてみましょう
  - プログラムコードを配布しますので、サンプルコードと同様にビルドして、Arduino に書き込みます
  - どこかに ir_recv というフォルダ（ディレクトリ）を作り、その中に ir_recv.ino を置きます
  - Arduino IDE を起動し、上記ファイルを開きます
  - プログラムをビルドします（スケッチ → コンパイル・検証）
  - プログラムを Arduino に書き込みます（スケッチ → マイコンボードに書き込む）
  - ツール → シリアルモニタで、動作を確認します

次に学習リモコンのコードを動かそう

- 続いて、学習リモコンプログラム arduino_irrec を動かしてみましょう
  - プログラムコードを配布しますので、サンプルコードと同様にビルドして、Arduino に書き込みます
  - どこかに arduino_irrec というフォルダ（ディレクトリ）を作り、その中に arduino_irrec.ino を置きます
  - Arduino IDE を起動し、プログラムをビルドします
  - プログラムを Arduino に書き込みます
  - ツール → シリアルモニタを起動します
  - 改行コードは「CR のみ」に設定します
  - h と押し、リターンキーを押してみてください
  - 最終的には、Tera Term などのターミナルソフトのほうが使いやすいかも知れません

学習リモコン arduino_irrec 説明（1）

- このプログラムは、Arduino に今回製作した赤外線送受信ボードを学習リモコンとして動作させるものです
- シリアルポートを使い、対話的なユーザーインターフェイスがあります
- 9600bps でターミナル（Tera Term）などから操作することもできますし、同様に Raspberry Pi などから制御することもできます

学習リモコン arduino_irrec 説明(2)

- プロンプト “>” が出たらコマンド操作できます。コマンド “h” + 改行でヘルプが出ます。全ての入力は大文字/小文字の区別をしません
- 以下、コマンドです
  - h: ヘルプ
  - p: 赤外線点滅の変調パターンを設定する（以下の r でリモコンを学習できる）
  - t: 赤外線を発して外部機器にコマンド(シーケンス)を送る
  - r: 赤外線点滅の変調パターンと、受信した赤外線信号のシーケンスを復号(解読する)。結果は 16進数で表示され、変調パターンのほうは内部に記憶する（電源を切るかリセットするまで有効）
  - d: デバッグ情報の表示

学習リモコン arduino_irrec 説明(3)

- 使用例
  - リセットして以下のメッセージを確認する(Arduino Uno では、ターミナルソフトを起動する度にリセットされるようです)

```
Learning Infrared Remote Controller version 20150823
Copyright (c) 2010-2015, Atsushi Yokoyama, Firmlogics

h<CR> to show help

>
```

  - r コマンドを入力および改行して、リモコンを向けてボタンを押す(周りのリモコンを拾わないように手で覆うとベター)。次のようなメッセージを確認する。前者が変調パターン。後者が受信シーケンス(16進数)

```
P 3750 1914 467 1414
T 55 5A F3 08 10 8E 4C 10 10 01 00 0F 82
```

  - 受信機器があれば、この学習リモコンを向けて `T 55 5A F3 08 10 8E 4C 10 10 01 00 0F 82` というコマンドを入力してみましょう(もう一台の学習リモコンで試しても可)

学習リモコン arduino_irrec 説明(4)

- Raspberry Pi などから制御する場合
  - まず最初に、前記のように r コマンドで学習し、p コマンドと t コマンドの列を控えておきます
  - Raspberry Pi (など)を繋ぎます
  - ターミナルソフト(ここでは screen を使います。予め sudo apt-get install screen でインストールしておく)を起動します

    ```
    $ sudo screen /dev/ttyUSB0 9600
    ```
  - まず p コマンドで変調パターンを設定します
    - `P 3750 1914 467 1414`
  - 次に t コマンドで赤外線シーケンスを送信します
    - `T 55 5A F3 08 10 8E 4C 10 10 01 00 0F 82`
  - なお、p コマンドの設定はリセットまで保持されますが、t コマンドのシーケンスは毎回指定する必要があります
  - 実際にプログラムなどから制御したい場合、たとえば Python 言語で pySerial ライブラリなどを使います

例: PySerial で制御するコード例

- PySerial と Pexpect (fdexpect) を使った例です

```
#!/opt/local/bin/python2.7

dev = '/dev/tty.usbmodem144421'
# For Windows, it is said that dev should be '\.\COM1' or similar

def main():
        import fdpexpect
        import serial

        ser = serial.Serial(dev, 9600)
        child = fdpexpect.fdspawn(ser)
        child.expect('\n> ')

        print "Sending modulation parameters..."
        child.sendline("P 3455 1769 428 1314")
        child.expect('\n> ')

        print "Sending command..."
        child.sendline("T 34 4A 9C 01 9D")
        child.expect('\n> ')

        child.close()
        print "Done"

if __name__ == "__main__":
        main()
```

学習リモコン arduino_irrec 説明(5)

- 関数の簡単な説明
  - diag(): 診断メッセージ(文字列)をシリアルポートに出力します
  - sense_ir(): 赤外線受光モジュールの受光状態を読み取ります。1が点灯(サブキャリア 38kHz あり)、0 が消灯です
  - get_cmd(): シリアルポートからコマンドを読み取ります
  - hold_led_len(): LED を指定時間オン/オフ状態に保持します
  - send_led_bit(): 予め設定された変調パターンで LED からビット 0/1 を送出します
  - transmit_ir_msg(): 予め設定された変調パターンで、LED からコマンドシーケンスを送ります
  - detect_raise(): 赤外線受光モジュールが点灯(サブキャリアあり)するまで待ちます。シリアルポートから何か文字を送ると中断します
  - record_ir(): 赤外線受光モジュールから点滅パターン(サブキャリアを復調したもの)を受信し、記録します。detect_raise() から戻ったら速やかに呼び出す必要があります
  - analyze_pat(): record_ir() で取得した点滅パターンから、変調パターンを分析取得します
  - decode_pat(): record_ir() で取得した点滅パターンと analyze_pat() で取得した変調パターンを用い、受信した点滅パターンを復号します
  - parse_cmd(): シリアルポートと対話し、ユーザーからのコマンドを解釈しアクションを起こします
  - ISR(TIMER2_COMPA_vect): サブキャリア周波数の 2倍のレートで呼び出される割込ルーチンです。setup() でタイマと割込を設定しています
  - setup(), loop(): それぞれ、Arduino プログラムの構成関数です

補足: 割込について

- 今回の学習リモコンプログラムでは、「割込」という仕組を使っています
  - 割込は、プログラムの実行順序と非同期にサブルーチン（関数）を呼び出す仕組で、Linux (Unix) ユーザープログラムにおける signal に似ています。（実際には、もっとハードウェアに依存した仕組です）
  - 今回のプログラムでは、赤外線 LED をサブキャリア周波数 38kHz で正確に点滅させるために利用しています
  - 今回は割込の詳細まで踏み込めませんが、今回の実装では 38kHz の倍の 76kHz で ISR(TIMER2_COMPA_vect) という（疑似）関数が呼び出されると理解してください
  - この関数（ISR: Interrupt Service Routine）は、プログラムのどの実行タイミングで呼び出されるか分かりません。今回は割込禁止を使っていませんが、非常にプリミティブな処理（8ビット変数の参照）しかしていませんので、競合（racing）は発生しないはずです

皆さんへの課題

- LED ダブル化
  - 今回は GPIO 端子 2～6（5本並列）を使いましたが、赤外線 LED 一つでは 100mA（絶対最大定格）が限界です。
  - もう一つ赤外線 LED を加えて、より明るくコマンドを送信できるようにしましょう
  - GPIO 端子 8～12（5本並列）を使いたいところですが、Arduino マイコンの絶対最大定格を超える恐れがありますので、8～11（4本並列）にしておいたほうが良いと思います
  - 先に説明したトランジスタ増幅回路を使うと、LED 5つくらい駆動できるかも知れません。ただしボード全体で、USB 電源の場合は 450mA、外部電源を使っても 650mA くらいが限界のようです
  - なお、ESP8266 による Wi-Fi 化（次回テーマ?）をしたい方は、基板に余裕ができるようにしておいてください
- タッチセンサの実験
  - インターネット上でタッチセンサの製作記事を見つけて、実験してみましょう。ハンダ付けは不要だと思います

    例: http://kousaku-kousaku.blogspot.jp/2008/10/arduino.html

次回の予告

- Wi-Fi から直接制御できるようにしてみよう!
  - ESP-WROOM-02（中身は ESP8266 らしい）を使う予定

講師持参品

- 各種データシート
- USB 電源
- DMM
- 名札
- Arduino Sketch インストーラ
- 赤外線の見えるデジカメ
- テーブルタップ
- オシロ
- 高周波 L メータと、サンプルのコンデンサとか
- 1MΩ の抵抗器
- WebEx と Wi-Fi AP の準備
- 小さな部品を入れるケース
- プレゼンター（レーザーポインタ）
- スモック
- ゴミ袋