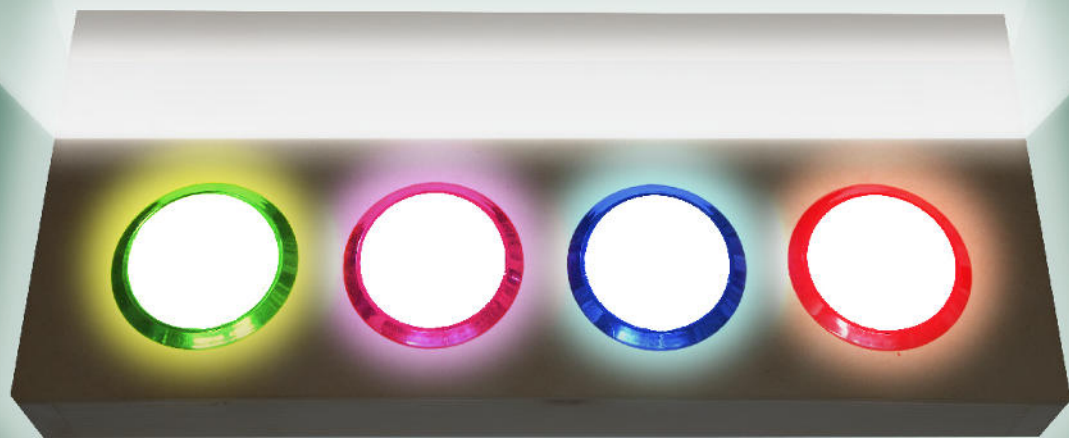


タッチセンサと無接点スイッチのお手軽アケコン設計ド入門

DIVAアーケード コントローラの 創りかた



ぼる 著



DIVA アーケードコントローラの創りかた

◆ 目次

◆	この本は何?	6
◆	お買い物	7
	正しくケチる工具	7
	部品	7
	海外通販	12
◆	環境構築について	16
	Pro Micro	16
	PSoC 4200 Prototyping Kit (CY8CKIT-049-42XX)	18
◆	HID ゲームパッド	23
	まずはボタンだけから	23
	チャタリング対策は?	26
◆	大きな 4 つの丸いボタン~無接点スイッチを添えて~	27
	海を越えるボタン	27
	フォトスイッチを再現	28
	静音化	31
	ボタンの重さとバネ定数	32
◆	ボタンを光らせる~順方向電圧降下~	35
	とりあえず光らせよう	35
	順方向電圧降下	36
	出力電流	37
◆	実際に作って理解する静電容量タッチセンサの仕組み	40
	タッチセンサは作れる!	40
	静電容量と過渡応答	40
	人体は抵抗とコンデンサだ	42
	Arduino で実際にタッチセンサを作ってみる	44
◆	PSoC でタッチセンサを動作させる	47
	自習課題	47
	実物のスライダーの中身は?	48
	ハードウェアの実装	50
	手の動く方向検出のアルゴリズム	54

	実際に書き込む.....	54
	タッチ感度の調整について	56
◆	タッチスライダーを光らせる.....	57
◆	GIMX で HID コントローラを PS4 に認識させる.....	59
	必要なハードウェア	59
	USB アダプタをつくる.....	60
	PC へのソフトのインストール.....	60
	ソフトの設定.....	61
	USB アダプタへのファームウェアのインストール	61
	PS4 に繋ぐ!.....	62
◆	製作例.....	63
	ブロック図.....	63
	回路図.....	64
	実装例.....	65
	筐体.....	66
◆	実機を自宅で動作させるには?.....	67
◆	おわりに	68

◆ この本は何？

『初音ミク Project DIVA Arcade Future Tone』は、Project DIVA シリーズのアーケード版として稼働中のタイトルで、PlayStation4 向けの移植版が 2016 年に登場しています。Project DIVA シリーズには最初から家庭版向けとして発売されたものが多くありますが、アーケード版の PS4 移植は、アーケード筐体のインターフェースを無理やり PS4 の標準コントローラである Dualshock4 に落とし込んだもので、譜面はアーケードと同一ながらプレイフィールドは完全に別物となっています。

もちろんアーケード版実機で遊べばいいのですが、実際は家でも同じように楽しみたいもの。それに加えてなんだか難しそうだけど意外とどうにかかなりそうな、実装しがいのあるインターフェース。さらに 10 年近く初音ミク周辺のムーブメントを追いつけていたのに未だにコンテンツの消費ばかりしているという危機感(?)、様々な原因が重なり合ってこのコントローラを作り、本を作りました。

この本では、手にとっていただいた方全員がコントローラを実際に作れるようになるべく簡単な表現をするよう心がけていますが、一方でこれは C の入門書とか半田付け指南書とかではないので、そのあたりは申し訳ないですけど別の本で補完してください。

なお、私の開発環境の都合上 Windows10 上での解説が主となります。他の OS では操作体系が違ふとかあるかもしれませんが、そのあたりはご容赦を。

また、以降の解説でウェブサイトのアドレスを多数記載していますが、紙媒体でこの文章を読んでくださっている方向けにすべてのアドレスを以下のページからリンクしています (ここだけは手打ちしてください! すいません)。正誤情報などもこちらから。

【夏コミ 1 日目西も 38a】"DIVA アーケードコントローラの創りかた"について

<http://pol.dip.jp/diva/>

◆ お買い物

それでは、必要なものを順次買い揃えていきましょう。秋葉原に行ったりホームセンターに行ったり通販したりしてください。

正しくケチる工具

冒頭で「指南書ではない」と書いておきながらこういうこともちょっと書いておきますが……
カッコ内は私が実際に使っているものです。

半田ごて(goot TQ-95)

半田付けに必要! (そりゃそうだ) 今ではダイソーで数百円程度から手に入りますが、できれば**セラミックヒーター**のものを用意しておきたいところ。温調ごてまでは必要ないかと思います。最近は海外メーカーなどから非常に安価な製品も買えますが、そういうの正直どうなのでしょうか……? 私はおすすめしません。ただ半田吸い取り器に関しては安物でもまあまあ問題ないかもしれません。

ラジオペンチ(型番不明)・ニッパ(藤原産業 N-125PF)

どちらも簡単な電子工作に使えるのは 1000 円程度からでしょうか。ニッパは 100 均のものだけでなくホームセンターで安価(500 円前後)なプライベートブランド商品も避けたいところ。

ピンセット(ダイソー 精密道具 C8)

私が持っているのはダイソーのものなのですが、同じく 100 均のダイヤモンドヤスリで先端を研いでいます。もちろん高いものを用意するに越したことはないですが、この程度の電子工作なら 100 均でも使えます。

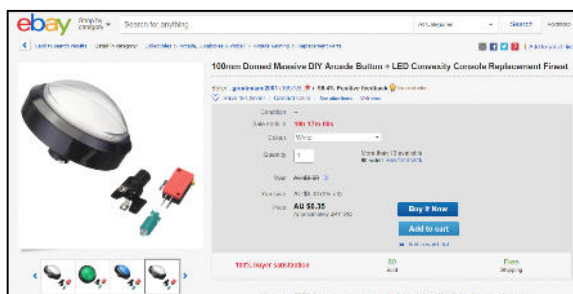
引廻し鋸(藤元 160mm)

適度に大きくて丸い穴を開けるのに必要です。もちろん電動工具を使ってもいいのですが、こういった木工になれてない自分が下手に扱ってけがをするのも怖かったので……。数百円程度から。

部品

ボタン

DIVA に使われている純正のボタンを使うとすると全部で何万円とかするので、見てくれだけそっくりな別のものを用意します。



eBay の互換ボタン(<https://www.ebay.com/itm/332404254762>)

これはその一例。コントローラ上に露出する大きなドーム状のボタンと、内部で押下検出するためのマイ

クロスイッチ、さらに装飾用 LED のセットで購入当時 4 個 2000 円ほど。eBay でノーブランドものを購入したときの価格です。

なお、この手の形だけ似ているボタンは、主にマイクロスイッチのせいで非常に重く・そしてうるさく、そのまま自宅用 DIVA コントローラを構成するにはあまりにも不向きです。なるべく実機に近づけつつ家庭でのプレイに大きな問題を出さないような改造についてはまた後々。

参考までに、おそらく実機に使われていると思われるものを紹介します。

押しボタン

照光式押しボタン ILLUMINATED BUTTON

ワンタッチランプホルダー(フォトスイッチ一体型)
XF lamp holder built in photo sensor switch

OBSA-LHSXF-LN

特徴
スイッチ部を無接点式にすることで、従来の有接点スイッチ(マイクロスイッチ、リードスイッチ)に比べて、電気的寿命を飛躍的に向上しました。

※ フォトセンサ方式を採用しており、電圧電圧 DC 5V が必要になるため、従来スイッチとの互換性はありません。

ATTENTION: This is incompatible for 1F and DF lamp holders, if s needed DCSV.

スイッチの定格及び性能

照光式押しスイッチの形式	薄型、A 型、特大型、厚型、薄型クリップ、薄型取付
スイッチの機能	OBSA-LHSXF
スイッチ電源電圧	DC 5V ±5%
ランプ定格(最大)	DC 50V 3W
断電時	充電時間 AC 50V 1分間
充電時	充電時間 AC 50V 1分間
絶縁抵抗	充電時 100MΩ以上 (DC 250V メガ)
充電時	充電時 100MΩ以上 (DC 250V メガ)
使用温度範囲	ウェッジ部使用時 -20 ~ +80℃
	LED 使用時 -20 ~ +65℃
電気的、機械的寿命	定格 400g、操作頻度 300 回/分、全ストロークにて、500 万回以上。
押圧	70g

コネクター配列

コネクターNo.	仕様
1	ランプ +
2	ランプ -
3	DCSV
4	OUT
5	GND

コネクター: S05B-PASK-2 (JST製)
相 手 例: PAP-05V-S (JST製)

LHSX-H
ワンタッチランプホルダー
XF専用ハーネス

OBSA-LHSXF

写真・図面は OBSA-100UMQ-B-1F

OBSA-100UMQ-□-1F□

薄型100Φドーム
ワンタッチランプホルダー
Thin 100Φ dome 1F lamp holder

取付板厚: 7 (MAX)

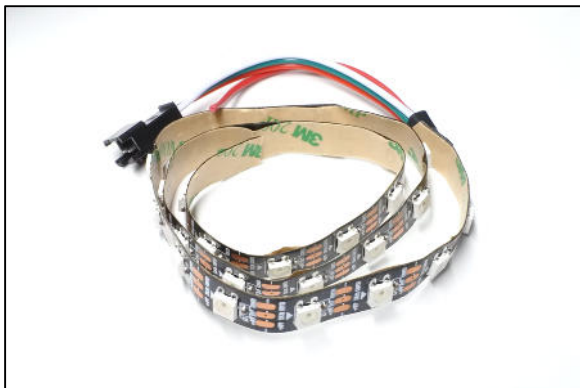
OBSA-100UMQ

(http://www.sanwa-d.co.jp/product/web_c/index.html)

1 セット用意するのに合計 8294 円かかります……4 つで 33k 円……出せなくはないのですがちょっと厳し

いのでパスです。

LED テープ



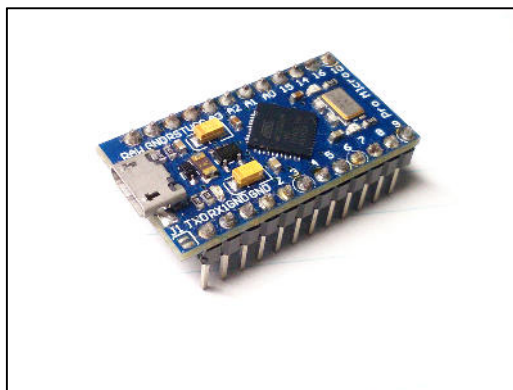
WS2812B 搭載 LED テープ

LED を薄い基板に配線して並べた「LED テープ」なるものは、PC ケースや車の装飾などの改造パーツ用に製品の 1 ジャンルとして確立されていて、海外通販だけでなく国内の通販やパーツ店などでも比較的簡単に入手できます。購入したのは全長 100cm で 60 個「WS2812B」という LED が並べられているもの。スライダーを光らせるために使います。

……ええ、この WS2812B というのが重要です。詳細は後ほど書いていきますが、これによってふつうに LED を並べるよりも大幅に配線を減らすことが可能になります。

ATmega32U4 搭載マイコンボード(Pro Micro)

2 個買います! Pro Micro は、SparkFun 社が製造する Arduino の互換ボードです。形は Arduino のシリーズの 1 つ「Arduino Micro」と似ていますが、公式にライセンスを得ているわけではないので製品名に「Arduino」がそのまま使えないのです。ショップによっては「Arduino 互換機」など書いているところもありますが。なおここで購入したのはさらにその互換ボード(SparkFun 社が製造しているものではない、の意味)です。最近流行りの自作キーボードによく使われているのもこれですね。



Pro Micro(ピンヘッダ実装後)

なお、Pro Micro は動作電圧や動作周波数が異なるものが流通していますが、ここで使うのは **5V/16MHz** のものとなります。

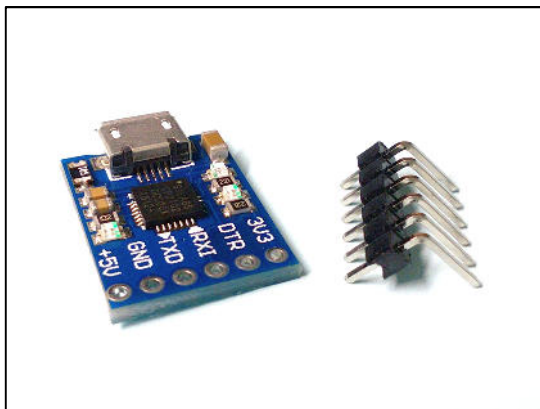
PSoC 4200 Prototyping Kit (CY8CKIT-049-42XX)

PSoC は、一般的なマイコンに加えて様々な「モジュール」と呼ばれるハードウェアの周辺回路をチップ内で構成できる……という、文字で説明してもうまく伝わりませんが、本当によくわからないチップです。そんな PSoC の中でも特に安価に購入できる開発ボードである 4200 Prototyping Kit を使います。秋月電子通商など秋葉原の実店舗をはじめとした様々なパーツ屋で、数百円程度で購入できます。



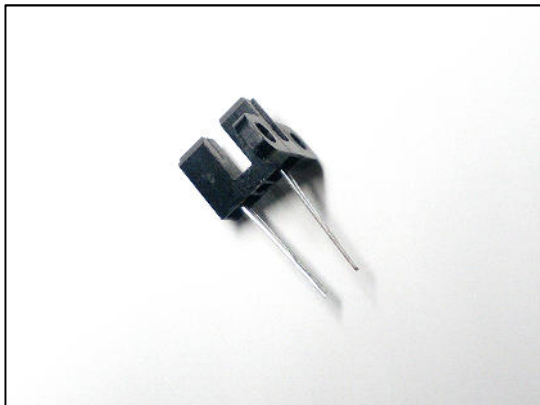
USB-UART 変換アダプタ

ここをあまり詳しく解説しても仕方無いので、そういうボードが必要だと思ってください(なんと投げやりな!)。使えるチップの型番に限られますので注意。基本的には **CP2102** もしくは **FT230X** が搭載されたものを買っておけば問題ないと思われます。



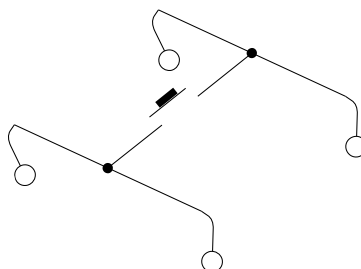
フォトインタラプタ

フォトインタラプタとは、発光素子と受光センサを向かい合わせに並べて、その間を遮る物体を検出することができるようなセンサです。センサと遮る物体との間には電氣的にも物理的にも接点が必要ないので、まったく無音で物体の動作を検出できるわけです。ここで購入したのは **CNZ1023** というもの。センサのサイズの関係上これを推奨します。秋月で非常に安価に購入できます。



タクトスイッチ

ボタンを押しているときだけ導通する、シンプルで比較的小さいスイッチ。どこのパーツ屋でも手に入るものです。一般的な足が 4 本生えているタクトスイッチでは、次の図のように 2 本同士で最初から導通しているので注意。90 度向きを間違えてしまうと、思いもよらず「ボタンを押していないのに常に導通している」状態になってしまいますので、向きには十分注意してください。



トランジスタアレイ

トランジスタアレイは、その名の通り一つの IC の中にトランジスタを並べているものです。ここで使用したのは **TD62783A** というもので、基本的にマイコンなど出力できる電流に限界があるときに、その出力する電流を増強するために使われます。

AND ゲート IC

いよいよもって「CPU の創りかた」っぽくなってきてしまいましたが、ここではちょっとしたグルーロジックとして使います。使用したのは皆さんおなじみの **74HC08**。「2 入力 AND ゲート」ですね。

MDF 材

天板として使用します。ボタンをはめる穴を開けるために、加工のしやすい材料として採用しました。

SPF 材(1x3)

ツープイフォー
2 x 4 とかのアレです。側板として使用します。非常に安価で、強度も十分です。

半透明 PP 板

ダイソーで購入。スライダーのプラスチック部に使用します。

この他には、**ナメクジよけテープ**や **IDE ケーブル**も必要になります。これに関しては「PSoC でタッチセンサを動作させる」の章で詳しく解説します。さらに、

- 抵抗
- コンデンサ(電解/積層セラミック)
- ユニバーサル基板
- コネクタ

などなどが必要なのですが、このあたりは正直何を買っても問題ないと思います。

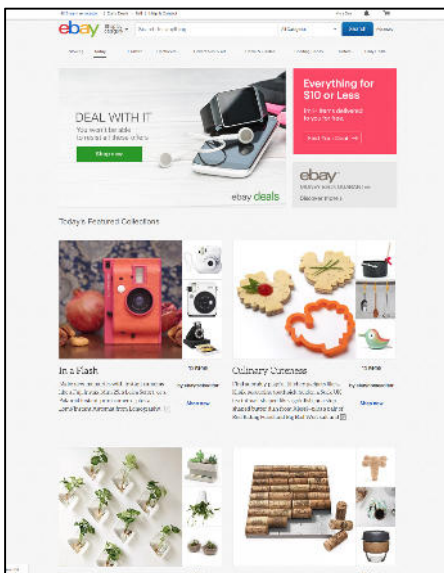


海外通販

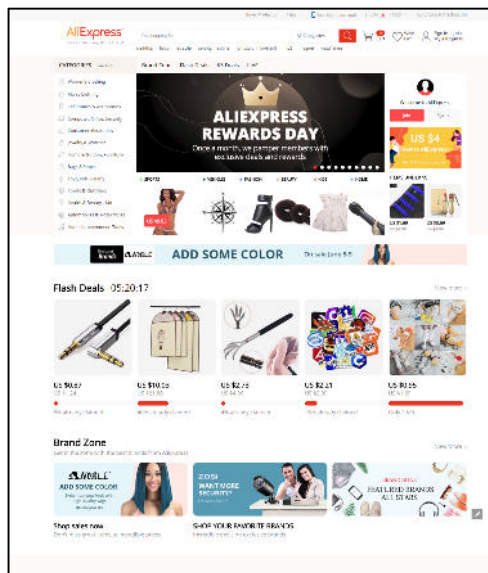
さて、先程さっと触れてしまいましたが、皆さん「海外通販」を利用したことはあるでしょうか？

eBay や AliExpress がそういったサイトの一例です。製品の品質や保証に関しては国内のようにはいきませんが、国内通販ではありえないくらいとんでもなく安く製品を購入できたりします。多少の癖はありますが、国内の普通の通販サイトを使ったことがあって、ちょっとでも英語が読めるならほぼ問題なく使えると思います。ただし注意すべき点として、到着までものすごーく時間がかかるということを忘れないでください。

なお、支払い方法について eBay はクレジットカードもしくは PayPal、AliExpress については実質クレジットカードのみとなりますので、そちらもちょっと注意。



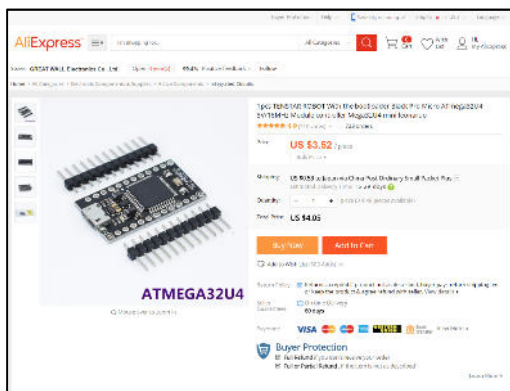
eBay (<https://www.ebay.com>)



AliExpress (<https://www.aliexpress.com>)

登録自体は簡単で、それぞれ"resister" "join"から簡単な個人情報を入力して終了です。

ここではAliExpressを使って実際に買い物をしてみましょう。



AliExpress の商品ページ

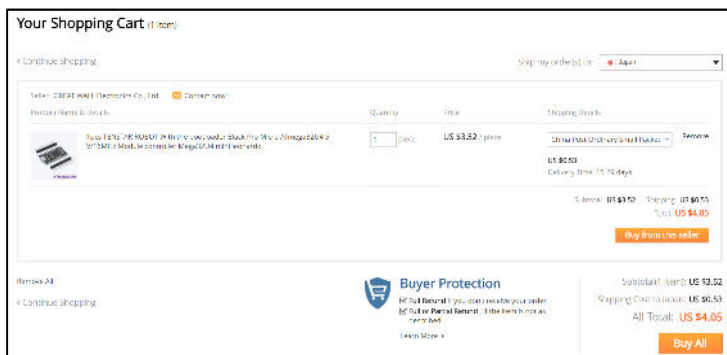
商品の検索欄に半角英数で商品名を入力して検索し、適当に商品を選んで商品ページに飛んだら、まずは商品名と実際の写真が本当に正しいか、写真を拡大したり商品ページ下の説明文を読んだりしてよくよく確かめます。また、送料や発送にかかる時間などの情報を正しく得るために、ページ最上部右側の Ship to が日本になっているか確認しましょう。

特にこういった基板の場合、「〇〇 replacement for ××」といったように、ここでは使われていない(が、一定の使用条件の上では問題なく代替品として使用できる) ××の乗った基板がほしいのに実際には〇〇が乗っている、という検索する上で紛らわしいタイトルの商品が数多くあります。こればかりは自分自身でよく商品タイトルと写真を見て確かめる他ありません。

確認した上で本当にこれが自分の購入したい商品であるとわかったら、ついに Add to Cart をクリック!

……する前に、今度は一応ショップと商品の評価を見ておきましょう。AliExpress や eBay はたくさんのショップがそれぞれ商品を販売している楽天市場のような場所で、ショップの評価と商品の評価がそれぞれ表示されます。AliExpress の場合ショップの評価は商品ページ内の検索欄のすぐ下、商品の評価は商品名のすぐ下にあります。

あくまで私個人の感覚によるのですが、ショップの評価は Positive feedback が 98%以上ならほぼ問題なしと言えます。商品の評価については☆の平均というよりも注文数や評価の数あたりを見ておくといいかもしれません。評価平均が☆5.0 だとしても注文数や評価数が一桁だとちょっとだけギャンブルかも。



ショッピングカート

カートに追加したら、もう一度商品そして価格におかしいところはないか確認して Buy All しましょう。

1. Please fill in your shipping address. Don't forget to save!

Contact Name:

Country/Region:

Street Address:

Apartment, suite, unit etc. (optional):

City:

State/Province/Region:

Zip/Postal Code:

☐ My address does not have a ZIP code.

Mobile:

Country Code - Mobile Number:

☐ Set as default

配送先の記入

さて住所の入力です。

Contact Name	姓名
Country/Region	国
Street Address	町丁目名、番地／アパート、マンション名(任意)
City	市区町村
State/Province/Region	都道府県
Zip/Postal Code	郵便番号
Mobile	国コード付き電話番号

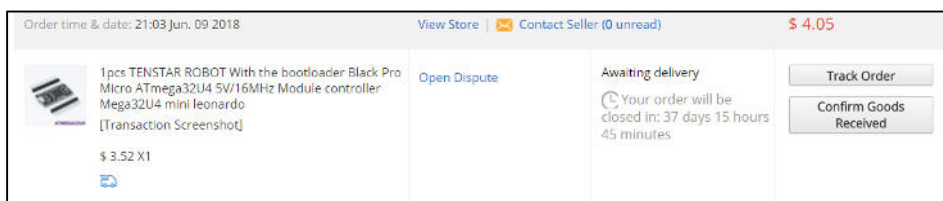
こんな感じで日本での一般的な表記の順番と若干異なるので注意が必要です。また、基本的にローマ字表記で入力したほうが良いと思います。



同ページ下のクレジットカード情報入力欄にも正しく入力したら、Confirm & Pay です! (ここをクリックした時に決済が完了します)



お疲れ様でした。到着まで気長に待ちましょう。



製品が無事到着したら、動作確認をした上で AliExpress マイページ上の "My Orders" から "Confirm Goods Received" をクリックしてセラーに商品が到着したことを伝えます。

◆ 環境構築について

使用するボードの Pro Micro と PSoC 4200 Prototyping Kit を実際に使えるようにパソコンの環境構築をしましょう。

Pro Micro

Pro Micro は、SparkFun 社が製造する Arduino の互換ボードです。Arduino そのものではありませんが、互換性を考えて作られたものなので、Arduino IDE での開発が可能です。使用しているチップは **ATmega32U4** というもの。Arduino は(その名称は自由に使えないものの)回路やソースコードがオープンソースであるため、他のメーカー (や、個人も!) が自由に互換機を作ることができるのです。

それでは、開発環境である『Arduino IDE』をインストールします。



Arduino – Software

<https://www.arduino.cc/en/Main/Software>

とりあえず一番扱いやすいのは"Windows Installer"だと思うので、そちらをクリックしてインストールします。

……と、一応リンクを紹介しましたが、この本を手にとった方なら一応 Arduino を触ったことくらいはありますよね? 多分……

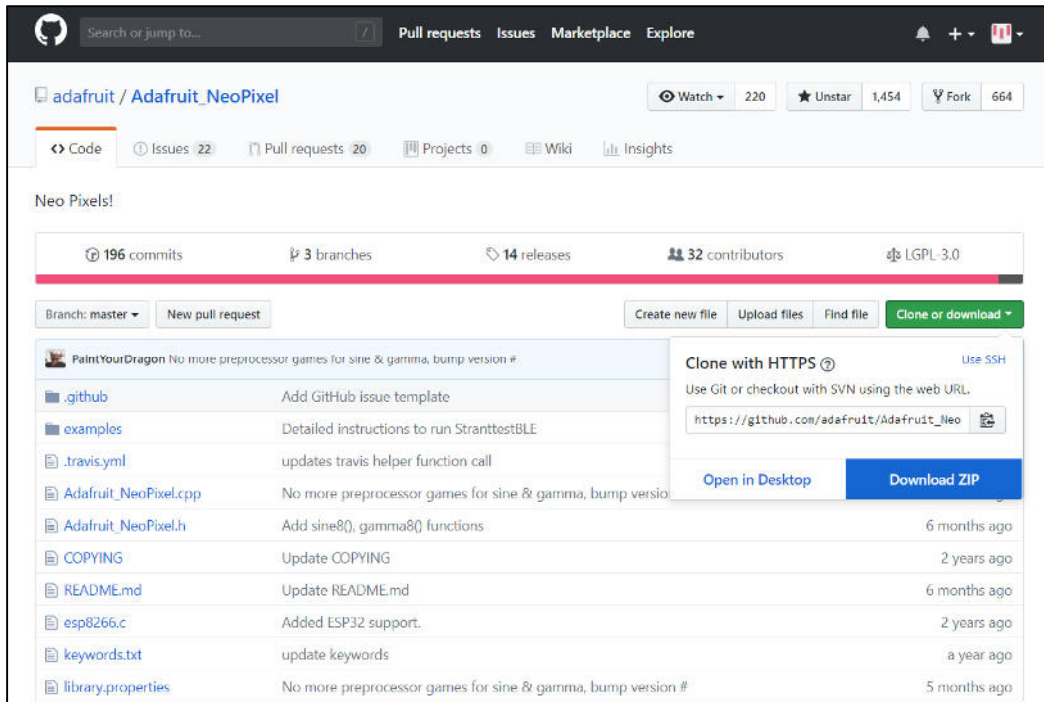
次に、今回使用するライブラリを以下の URL から ZIP 形式でダウンロードします。

NicoHood/HID • GitHub

<https://github.com/NicoHood/HID>

adafruit/Adafruit_NeoPixel · GitHub

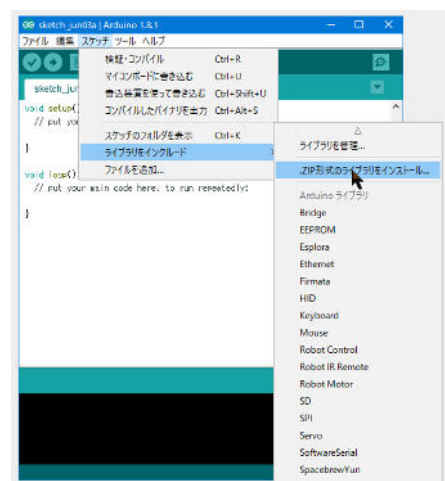
https://github.com/adafruit/Adafruit_NeoPixel



画面右側「Clone or download」→「Download ZIP」でライブラリの ZIP ファイルがダウンロードできます。

次に、Arduino IDE を起動し、「スケッチ」→「ライブラリをインクルード」→「.ZIP 形式のライブラリをインストール」の順に進み、先程ダウンロードしたファイルを選択しインストールします。

「ライブラリが追加されました。……」というメッセージが緑色のバーの部分に表示されたら、今回 Pro Micro を使うための Arduino IDE の環境構築はひとまず終了です。



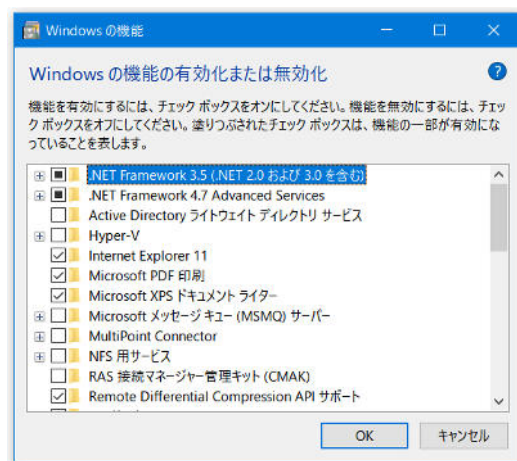
PSoC 4200 Prototyping Kit (CY8CKIT-049-42XX)

PSoC は、一般的なマイコンに加えて様々な「モジュール」と呼ばれるハードウェア的周辺回路をチップ内で構成できる……という、文字で説明してもうまく伝わりませんが、本当によく分からないチップです。

なお、インストールの際、.NET Framework 3.5 と Visual C++ 2008 再頒布可能パッケージがそれぞれ必要になります。これらのインストールを先に済ませておきましょう。

.NET Framework 3.5 は、Windows10 の場合、「Windows の機能の有効化または無効化」から「.NET Framework 3.5 (略)」を選択して OK を押し、手順に従ってしばらく待ちます。

Visual C++ 2008 再頒布可能パッケージは以下からインストールします。



Download Microsoft Visual C++ 2008 Service Pack 1 再頒布可能パッケージ MFC のセキュリティ更新プログラム from Official Microsoft Download Center

<https://www.microsoft.com/ja-jp/download/details.aspx?id=26368>

vcredist_x86.exe のみのダウンロード・インストールで OK です。

なお、ちょっとしたハマりポイントとして、以下の URL などからインストールできるパッケージをインストールしても**意味がありません**!

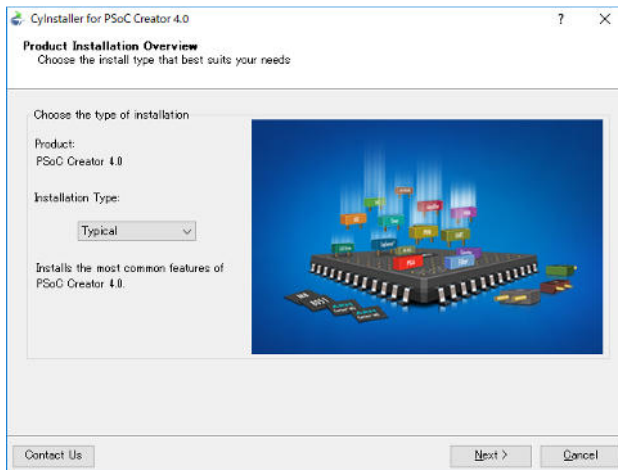
<https://www.microsoft.com/ja-jp/download/details.aspx?id=29>

しかも「Microsoft VC++ 2008 Redistributable」などのいかにもな語でググるとこれがトップに来ます (「VC++ 2008 MFC」でググって見たら正しい方のリンクが 2 番目に来ました)。同じ年代のパッケージでもバージョンが違うためこういった事が起こるのですが、かなり罠ですね……

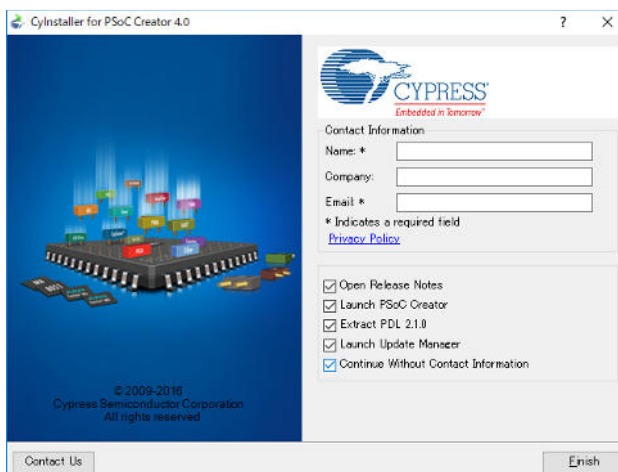
さて、実際にソースコードを書く開発環境である『PSoC Creator』、このボード特有のデバイスドライバである『USB-Serial Software Development Kit』、サンプルファイルを含む『CY8CKIT-049-42xx Kit Only (Kit Design Files, Documentation, Examples)』をそれぞれダウンロード・インストールします。ただし、それらのインストールの前に、まずは Cypress 社のユーザーアカウント登録を済ませておくことをおすすめします。

PSoC® Creator™ Integrated Design Environment (IDE)

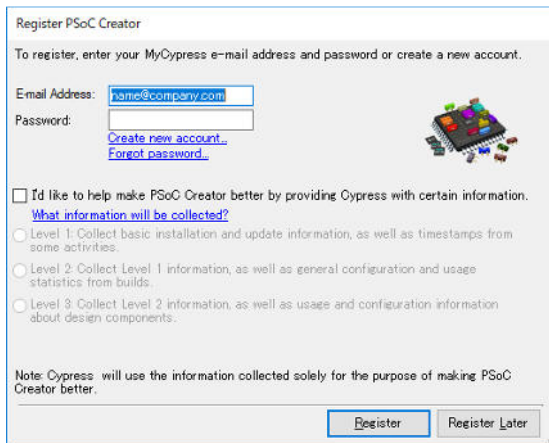
<http://japan.cypress.com/products/psoc-creator-integrated-design-environment-ide>



Installation Type は Typical のままで大丈夫です。このまま手順に従って、ライセンス契約に同意し、インストールしましょう。



インストール終盤、ユーザー情報を求められますが「Continue Without Contact Information」にチェックを入れればそのままインストールを完了することができます。



Register PSoC Creator

To register, enter your MyCypress e-mail address and password or create a new account.

Email Address:

Password:

[Create new account.](#)
[Forgot password.](#)

☐ I'd like to help make PSoC Creator better by providing Cypress with certain information.
[What information will be collected?](#)

☐ Level 1: Collect basic installation and update information, as well as timestamps from some activities.

☐ Level 2: Collect Level 1 information, as well as general configuration and usage statistics from builds.

☐ Level 3: Collect Level 2 information, as well as usage and configuration information about design components.

Note: Cypress will use the information collected solely for the purpose of making PSoC Creator better.

PSoC Creator インストール終了後の初回起動時に Cypress のアカウント情報を求められます。こっちは入力したほうが良さげ？

初回起動がうまくいったら、下記 2 つのインストールが完了するまで終了しておきましょう。

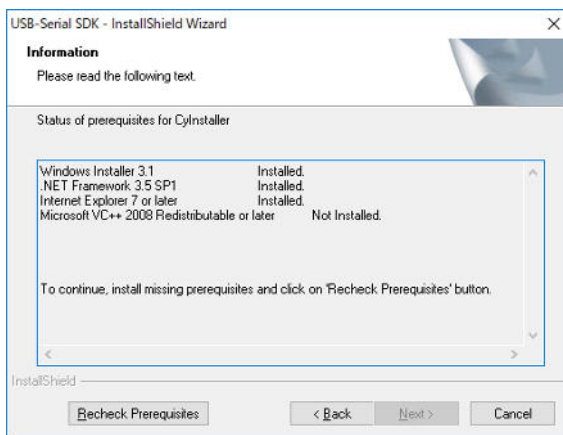
次は USB-Serial Software Development Kit です。

USB-Serial Software Development Kit

<http://www.cypress.com/documentation/software-and-drivers/usb-serial-software-development-kit>

ページ下の関連ファイル群から USBSerialSDKSetup.exe をダウンロードします。

ここで以下のような画面が出て次に進めない場合、Visual C++ 2008 再頒布可能パッケージのインストール失敗、もしくはバージョン選択をミスしている場合があります。



USB-Serial SDK - InstallShield Wizard

Information
Please read the following text.


Status of prerequisites for CyInstaller

Windows Installer 3.1	Installed.
.NET Framework 3.5 SP1	Installed.
Internet Explorer 7 or later	Installed.
Microsoft VC++ 2008 Redistributable or later	Not Installed.

To continue, install missing prerequisites and click on 'Recheck Prerequisites' button.

InstallShield

なおこの時ブラウザで次のようなページが自動で表示されたりしますが、

 **Prerequisite Software Requirements to Install CyInstaller for USB-Serial SDK**

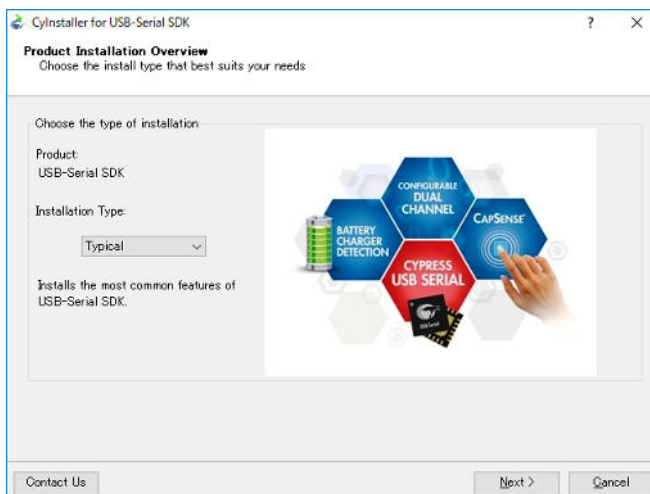
The following software needs to be installed before the CyInstaller for USB-Serial SDK can be installed on your system.

Product	Download Link	Why it is required
Microsoft Visual C++ 2008 Redistributable	Download	USB-Serial SDK requires the Microsoft Visual C++ 2008 Redistributable package to be installed for execution of the software tools available as part of the kit.

Copyright © 2010 Cypress Semiconductor Corp. All rights reserved.

「Download」を押してもリンク切れです。うーん……

逆に、



この画面まで進んだらそのまま進めてインストール完了です。

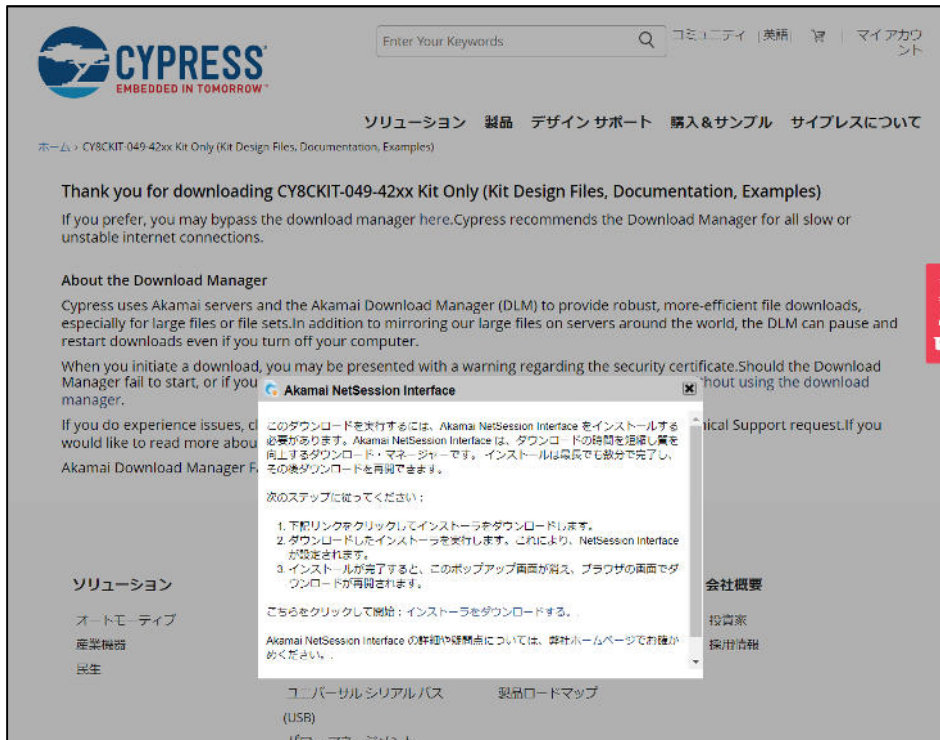
最後に、CY8CKIT-049-42xx Kit Only (Kit Design Files, Documentation, Examples)です。

PSoC® 4 CY8CKIT-049 4xxx Prototyping Kits

<http://japan.cypress.com/documentation/development-kitsboards/psoc-4-cy8ckit-049-4xxx-prototyping-kits>

ページ下の関連ファイル群から CY8CKIT-049-42xx Kit Only (Kit Design Files, Documentation, Examples) をダウンロードします。

なお、先程の USB-Serial Software Development Kit のときもそうなのですが、このファイルのダウンロード時下のように Akamai NetSession Interface というダウンロード支援ツールのインストールを促されます。これはポップアップウィンドウ風の通知右上の「×」を押してから "If you prefer, you may bypass the download manager here." より直接ファイルのダウンロードも可能です。



お疲れ様でした。プログラミングにおける最大の難関って環境構築ですよ。

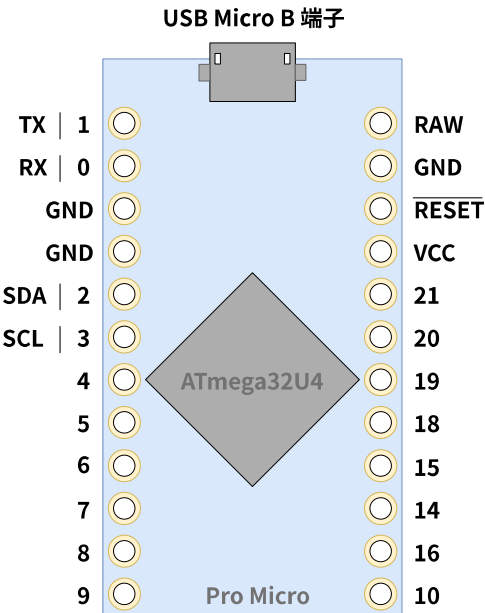
◆ HID ゲームパッド

まずはボタンだけから

それでは、早速 PC で動作する 1 ボタンのゲームパッドを作ってみましょう。PS4 で動作させるのはまた本の後半で。

ここで使うのは、**Pro Micro** です。ピン配置を見ていただくとわかりますが、Arduino のシリーズの中で最も一般的と思われる Arduino Uno とはサイズも全く違いますね。

Pro Micro に乗っているチップは ATmega32U4、一方 Uno に載っているチップは ATmega328P という別のもので、どちらも"AVR"というマイコンのシリーズですが、これらは(形だけでなく)一部使える機能に違いがあります。その中で重要なのが、ATmega32U4 は「**チップ自身を簡単に USB HID デバイスに化けさせることができる**」というもの。



つまり、プログラムを書き込むことで Pro Micro が PC からはキーボードに見えたりマウスに見えたり、ゲームコントローラに見えたりする、というものです。自作キーボード界隈で Pro Micro が使われるのはそれが理由なのです(実際には Arduino Uno も HID にすることは可能なのですが、手順がより簡単である上基板面積が小さく安価なため Pro Micro がよく使われています)。

Arduino はシリーズによって使われているチップの種類が違います。以下がその一例となります。すべての Arduino シリーズを列挙するのは流石に多すぎるため、使われることが多い(と思われる)シリーズを一部抜粋してリストに載せました。上記の通り Pro Micro は Arduino ではありませんが、同じ表にまとめてみます。

Arduino Uno	ATmega328P
Arduino Micro	ATmega32U4
Arduino Nano	ATmega168/ATmega328P
Arduino Leonardo	ATmega32U4
Arduino Mega 2560	ATmega2560
Pro Micro	ATmega32U4

このように、Pro Micro は Arduino Micro や Arduino Leonardo と全く同じチップを使用しています。更に言うと、現在海外通販で入手可能な安価な Pro Micro 互換機は、最初から Arduino Leonardo もしくは Arduino Micro のブートローダ(不正確ですが、AVR を Arduino たらしめるために最初から書き込まれている小さなプログラム、と思ってください)が書き込まれているのです。

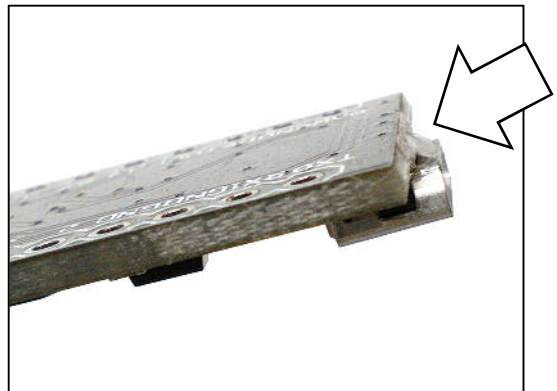
つまり、**Pro Micro が接続されたパソコンからはそれが Arduino Leonardo/Micro に見えるのです!**

互換機の互換機でありながら本家を名乗るとはなんと偉そうな。しかしユーザーとしては便利なのでなんとも言い難いところです。

ここでは、Arduino HID Project ライブラリを使って Pro Micro を HID ゲームコントローラにしてみます。実際に Pro Micro にスケッチを書いていきましょう。

.....の前に、まずは Pro Micro の Micro USB 端子の”もげ”対策をしておきます。

Micro USB 端子は力をかけると非常にもげやすく、特に Pro Micro は端子が一部基板からはみ出していることもあり(?) 力がかかりやすく本当によくもげます。私自身もやらかしたことがあります。もげる前に、瞬間接着剤などでしっかり固定しておきましょう。



```
#include "HID-Project.h"

#define PAD_BUTTON 2
#define PAD_PIN 4

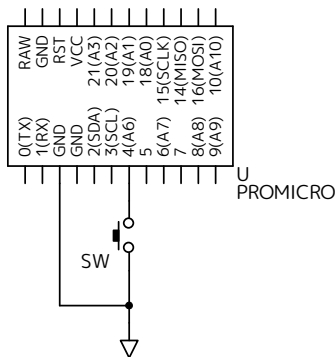
void setup() {
    Gamepad.begin();
    pinMode(PAD_PIN, INPUT_PULLUP);
}

void loop() {
    if(!digitalRead(PAD_PIN)) {
        Gamepad.press(PAD_BUTTON);
    } else {
        Gamepad.release(PAD_BUTTON);
    }
    Gamepad.write();
}
```

スケッチはたったこれだけです。どういうスタイルで書くかにもよりますが、おおむね 10 数行程度でしょう。たったこれだけのコードで Pro Micro はゲームコントローラになるのです。

これが書けたら、Pro Micro を PC に接続し、Arduino IDE の「ツール」→「ボード:...」の順に進み、「Arduino Leonardo」(もしくは「Arduino Micro」)を選択した上で「マイコンボードに書き込む」をしましょう。

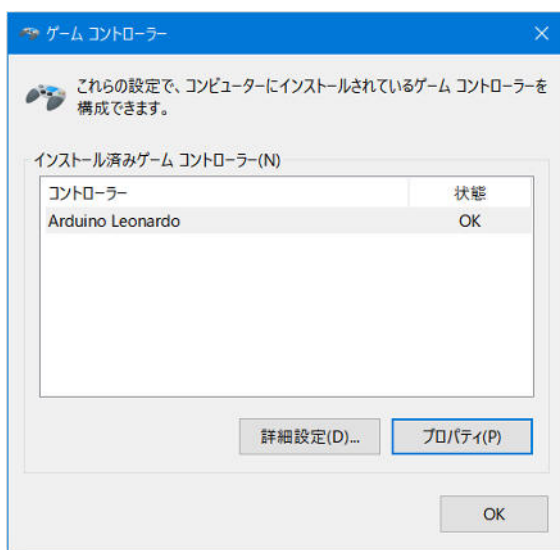
スケッチが書き込めたら、いったん Pro Micro を PC から外して、基板上もしくはブレッドボードで次のようにタクトスイッチを取り付けてください。



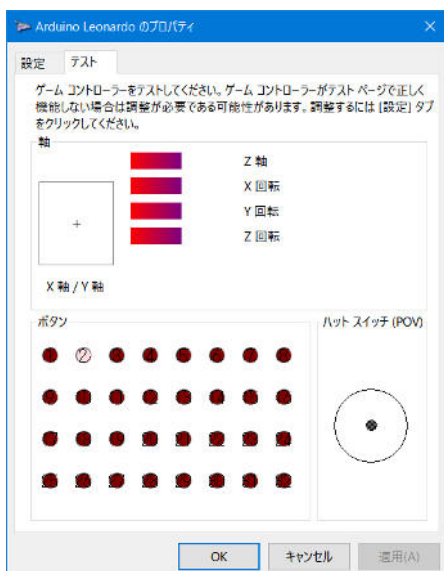
わざわざ回路図を書くかというレベルのものですね。本当に Pro Micro とスイッチしか無いです。必要なことは全部チップとライブラリがやってくれているので、扱う側の我々はこんな簡単な作業だけをしていればいいのです。

ここでもう一度 PC に Pro Micro を接続しましょう。そしてコントロールパネルから検索欄に「game」などを入力して「USB ゲームコントローラのセットアップ」を開きます。

ここで、おそらくコントローラの一覧の中に「Arduino Leonardo(Micro)」が出ていていると思います。このプロパティを開いてみましょう。



ここで基板 or ブレッドボードに配線したタクトスイッチを押してみると……



スイッチを押すのに合わせてウィンドウのボタン番号2が光ります！これがHIDゲームパッドの基本的動作となります。

ここまでできたら、あとはボタンを追加したり、スライダーの情報を受け取ったりして、それを全部PCに送れば大丈夫ですね。なんだか一気に道が開けたしませんか？

チャタリング対策は？

さてここまできて、Arduino やマイコンを触ったことのある方の中には、ボタンの入力処理にチャタリング対策がされてない！と思う方もいるでしょう。

この本の本題ではないのであまり深くは触れませんが、チャタリングとは簡単に言うと「スイッチを押した時に接点がバウンドしてとても短い間(一般的に数ミリ秒程度)に何度もオンオフを繰り返してしまう現象」を言います。一般的にマイコンでスイッチの入力を取り扱う場合、チャタリングの対策をしないとボタンを押す人間は1回しか押してないつもりでもマイコンは一瞬の間に数回連打したと解釈してしまうことがあります。そのため、チャタリング対策として「マイコンがボタンの入力を読み取るタイミングを長く(数十ミリ秒など)設定する」「一度ボタンが押されたら一定期間ボタンの入力状態を無視する」などといった処理を組み込むのが普通です(なお、この接点がバウンドする時間というのはスイッチが劣化すると長くなる傾向にあります。古いマウスが1度クリックしただけなのにダブルクリックしたように見えてしまうのは、チャタリング対策のために指定していた時間を超えて接点がバウンドしていることによります)。

さてこの回路でチャタリング対策をしていない理由ですが、端的に言うと「フォトインタラプタを使った無接点スイッチは、接点のバウンドが構造上発生しないので(無接点なのだからそりゃそうです)、チャタリングがそもそも起こらないから」です。もっと言うと「最終的にPS4に接続する時には1秒間に100回以上の入力は受け付けられないから」というのも理由だったりします。つまりもっと後段のソフトが結果としてチャタリング対策をしているのです。

◆ 大きな4つの丸いボタン~無接点スイッチを添えて~

先程はタクトスイッチをゲームパッドのボタンとして使いましたが、これではアーケードゲームっぽくありませんね。さっそく大きな丸いボタンに繋ぎ変えてみましょう。

海を越えるボタン



ついに海外からはるばる届いたボタンを開封します。ダンボールがボコボコなのは最初からです(これが海外通販です!)。とにかく、ボタンを分解してみましょう。

上部の実際に押されるドーム状のボタン部(以降、単に「ボタン部」と書きます)と、筐体に固定される真ん中の黒い土台部、バネ、下部はマイクロスイッチやボタン部を光らせるためのLED、そしてそれらのホルダーです。



このセットの一番下についているのはマイクロスイッチ。これが実際に配線の電氣的な切り替えをしているのですが、これがあまりにもうるさい! スイッチ単体ではそこそこの音(イメージとしては多少音の

大きいマウスのクリック音程度)ですが、ドーム状のボタンとともに押し込むとこのセット全体で共鳴して大きな音が響きます。これではとても自宅で遊べる環境にはなりません。

また、ドーム状のボタンを上押しするためのバネもそうなのですがこのマイクロスイッチ自体もかなり反発力が強く、DIVA AC 実機のものとは比べると非常にボタンが"重い"のです。



フォトスイッチを再現

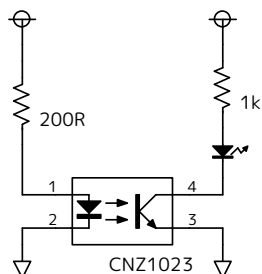
実は、実機ではこういったマイクロスイッチは使われておらず、「フォトスイッチ」という光学的にボタンの動きを検出するスイッチが使われているようです。通販の項でちょっと触れた「フォトインタラプタ」を内蔵したスイッチのようですね。しかしこれも触れましたがとにかく価格が高い。というわけで、ここではこのフォトスイッチを安価なフォトインタラプタを使って再現します。本当に安価なんですよこれ。



フォトインタラプタさん再掲

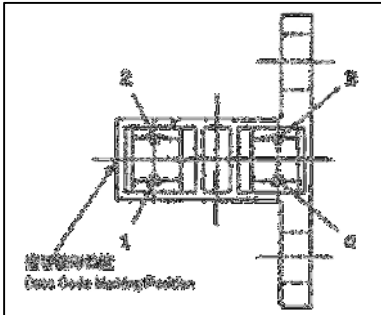
なんていったって単価 20 円ですからね。誇張とかでなく実機のスイッチと 2 桁違います……

さてこのフォトインタラプタ、実際にボタンに組み込む前に使い方を確認しておきましょう。このような回路をブレッドボードなどに組んでみてください。



フォトインタラプタは発光素子と受光素子が向かい合わせに並べられていると紹介しました。CNZ1023 の場合、具体的にはコの字型の樹脂に、向かい合わせに赤外線 LED とフォトトランジスタが入っているのです。回路図でもフォトインタラプタは LED とフォトトランジスタを並べたような図で表現します。回路図

では枠内の左にLED、右にフォトトランジスタが書かれています。そのまんまですね。



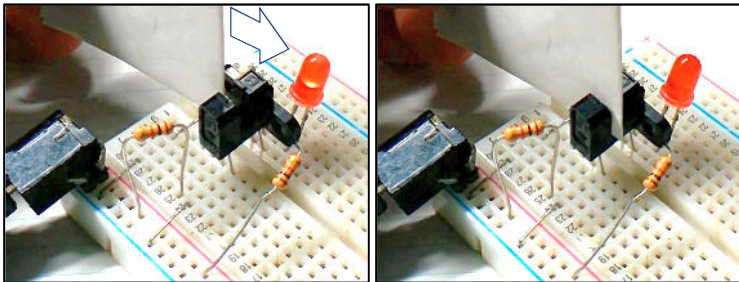
端子に注意！

1	アノード
2	カソード
3	コレクタ
4	エミッタ

[http://akizukidenshi.com/download/ds/panasonic/CNZ1023\(ON1023\).pdf](http://akizukidenshi.com/download/ds/panasonic/CNZ1023(ON1023).pdf)

回路図におけるLEDとフォトトランジスタの向きと実物のそれとは異なっているので、端子を間違えないようにしてください。

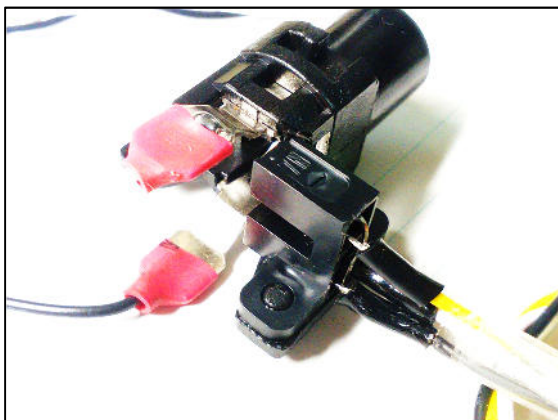
ここで、フォトインタラプタの間になにか薄い板をゆっくり入れてみます。



LEDが消えました。

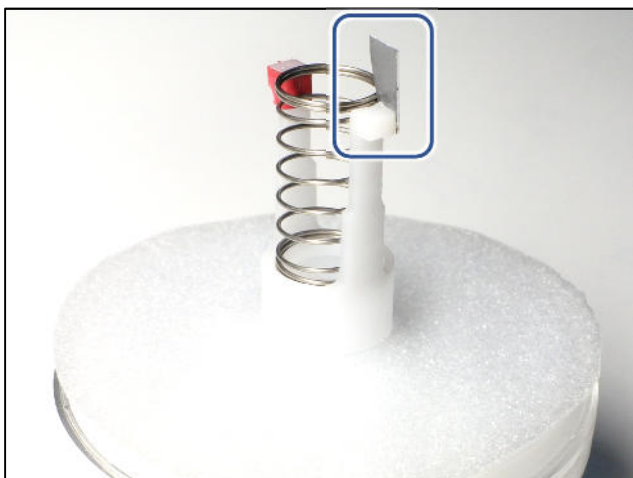
フォトトランジスタという素子は「受け取る光が大きいと電流も多く流れる」という特徴を持ちます。そのため、赤外線LEDとフォトトランジスタの間に何も遮るものがない場合、フォトトランジスタは赤外線LEDの光を受け取り自身に電流が流れるようになります。スイッチが押された状態、とほぼ同じと思ってもらって良いでしょう。しかし光を遮ると、フォトトランジスタは自身に電流を流さなくなります。つまりスイッチが切られた状態です。このような原理で、物理的・電氣的に接点のないスイッチを作ることが可能になるのです。

さてここまでは仕組みについて紹介しましたが、このフォトインタラプタを実際にボタンに取り付けてみます。ここから紹介するものはあくまで私が現物合わせで考えたレイアウトなので、実際に作られる際は適宜調整を。

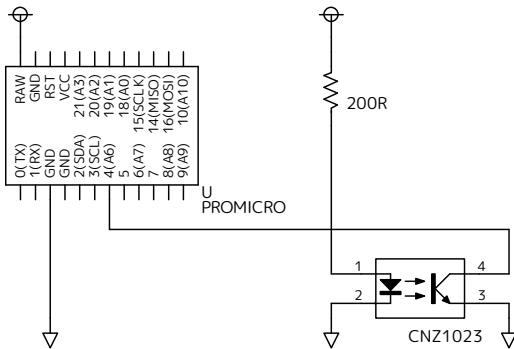


もともとマイクロスイッチのホルダーにあった固定用の突起と、おそらくネジ止め用だったフォトインタラプタの穴を合わせ、瞬間接着剤で固定します。反対側の穴はホルダーと干渉してしまうので適当に削ってしまいます。

ボタン側にも細工をします。ボタンが押し込まれたときにちょうどこのフォトインタラプタの光が遮られるよう、テレホンカードを小さく切った板をボタンの一番下部(写真は撮影のため反転しているので上側ですが)、もともとマイクロスイッチを押し込むための足のところにこれまた瞬間接着剤で貼り付けます。



ボタンが押されることでテレホンカード片がフォトインタラプタの間をうまく遮るように、微調整をしてください。板の長さがそのままだけ押し込んだ時にスイッチが ON になるかに関わるので、好みに応じて長さを調整してもいいかもしれません。私の場合、11mm 程度の長さがちょうど良い感じ。



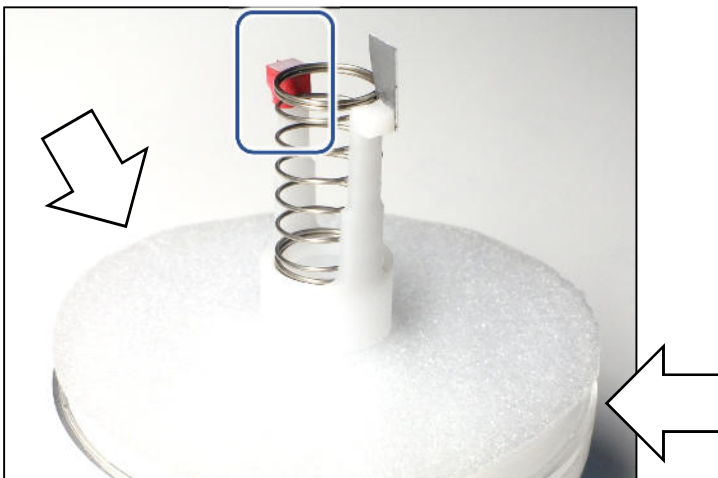
ボタンの改造ができれば、図のような回路でフォトインタラプタを Pro Micro に接続して(ソースコードはタクトスイッチ接続時のものと同じものが使えます)、さらに Pro Micro を PC につないで再度「USB ゲームコントローラのセットアップ」を開き、タクトスイッチでテストしたときと同じようにフォトインタラプタで動作するかどうかを確認しましょう。

静音化

さてフォトインタラプタを使った DIY フォトスイッチの作成により、マイクロスイッチのうるささからは開放されました。

しかし、マイクロスイッチがなくなってもボタンを押し込むとプラスチックどうしがぶつかる音はしてしまいます。こればかりは仕方ない……ですが、できるならばもう少し音を小さくしたいところ。

具体的に私がやってみた改造は以下のとおりです。



- ボタン部の裏側と黒い土台の間にドーナツ状に切った緩衝材を入れます。100 均で購入できる緩衝材をサークルカッター(普通のハサミでも全く問題ないです)でドーナツ状に切り取り土台に敷きます。これが一番効きますがそのぶん感触の変化も大きいというトレードオフ。

- ボタン部の側面に細く切った養生テープを 1~2 周貼り付けます。これはそんなに効果ないかも？
- ボタン部の一番下、本来はマイクロスイッチを押すための足だった部分のうち、フォトインタラプタ認識用のテレホンカード片を貼り付けてない方に熱収縮チューブをかぶせます。緩衝材では押し込むときの音を小さくできますが、こちらはボタン部がバネで押し戻されるとき音を小さくできます。

これによって、ボタンを押したことによる音は、

「カチッッ」(購入時そのままの状態)

「カカッ」(マイクロスイッチをフォトインタラプタに置き換えたところまで)

「フホ」(この項の改造をすべて行う)

程度には静かになりました。しかしながらボタンの感触は実機とはかなり違ってきますので中々難しいところですよ。

ボタンの重さとバネ定数

さて、この手の音ゲーコントローラにありがちな押圧の話です。

音楽ゲームをプレイするにあたって、ボタンがどれくらいの重さで押されると反応するかというのは重要な問題です。一般的に、この「重さ」はバネの硬さとマイクロスイッチの押圧、またボタンそのものの重量によって決まります。音ゲー界隈では、慣習的に(?) バネの硬さを[g]単位で表します。一方マイクロスイッチの押圧の単位は[N]で表されることのほうが多いようです。バネの硬さを表すバネ定数の単位[N/mm]からも分かる通り、バネを押すのに必要な力はバネをどれだけ距離押し込むかによって変わります。[g]単位で表されているバネは、一般的な音楽ゲーム用のボタンを押し込んだ距離で必要な押圧を、標準重力加速度のもとで質量単位に換算したものと考えられます。

しかしながら、私が少し調べた限りでは、単位に[g]を使うバネのバネ定数については解説した文章を見つけることができませんでした。

ですので私が測定しました。

バネ定数を求めることはここでは重要なことではないのですが、せっかく測定したのでその結果だけでもここに記録しておきます。

ここで実際にバネ定数を測定しようとしたバネは、OBSA-SP-200 です。



https://item.rakuten.co.jp/sanwadenshi/ilumb_217/

いわゆる 200g のバネで、これは DIVA の筐体にデフォルトで組み込まれているバネと同じ重さと言われているものです。なお、DIVA の筐体にマイクロスイッチの代わりに組み込まれている純正フォトスイッチは、カタログによると押圧 70g となっています。

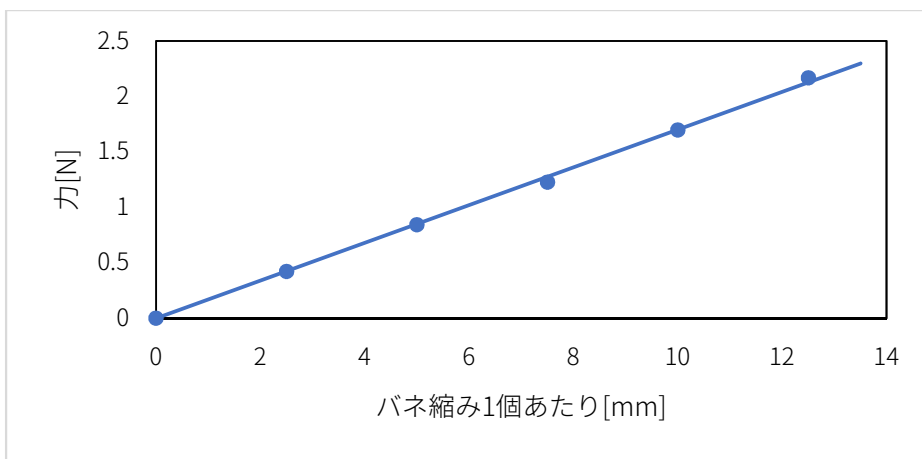
バネ定数 k [N/mm] は、その単位からわかるように、バネの長さを単位長さだけ変えるために必要な力の大きさです。キッチンスケールの読みは見かけ上単位は [(k)g] ですが、実際は質量を (キロ)グラム重で表したのになります。そのため単純な定数倍で力に変換することができます。これとバネの縮を記録できれば、バネ定数が測定できるという理論です。あくまで一般家庭にある機器を使った場合の方法ですが。

まず少しでも分解能を上げようと、バネを 4 つ直列に(セロハンテープで.....)つなげました。これによってバネにかかる力による 1 つあたりのバネの縮みは 4 分の 1 になります。キッチンスケールと定規を使い、指で押さえつつ縮んだ長さとキッチンスケールの読みを記録し、そこからバネ定数を求めました。

実際の測定結果は次のようになりました。

バネ縮み 4 個直列[cm]	キッチンスケール読み[g]	バネ縮み 1 個あたり[mm]	力[N]
0	0	0	0.00
1	43	2.5	0.42
2	86	5	0.84
3	125	7.5	1.23
4	173	10	1.70
5	221	12.5	2.17

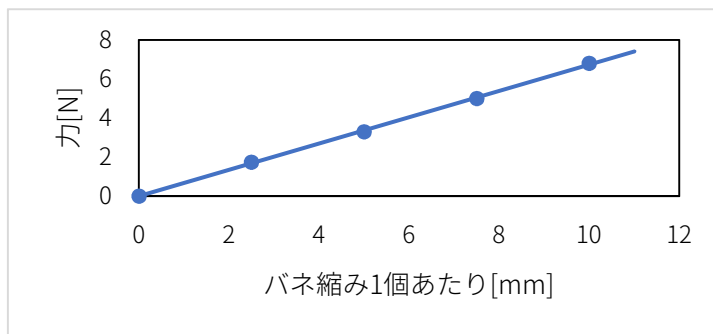
グラフにするとこんな感じ。



ということで、**200g バネのバネ定数は 0.17N/mm**ということになりました。

ここで作った DIY フォトスイッチは、反発力も押圧も何もありません。そのため、ボタンの重さは完全にバネだけで決まります。ここでは 200g バネを使いましたが、これで筐体デフォルト状態の純正コントローラよりは軽い状態となりました。ボタン静音化の関係で若干ボタンの戻りが悪くなっているので、これ以上ボタンを軽くするのは「埋まる」可能性が高くなりそうで難しそうです。

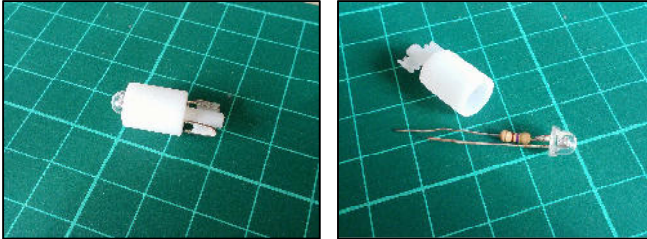
なお、海外ボタンに初めから付属していたバネも同じように測定してみたのですが、下のような結果になり、ばね定数は 0.67N/mm となりました。4 倍ですね。つまり **800g バネ**です (ええ……)。



◆ ボタンを光らせる~順方向電圧降下~

とりあえず光らせよう

さて、ここまでで4つのボタンをPCに認識させるところまでは完了しました。しかし、せっかくですからボタンが光ると面白いですね。抵抗内蔵LEDも最初からボタンのセットに含まれていたことすし、早速このLEDを光るようにしてみましょう。ちなみにこの抵抗内蔵LEDなのですが、



足をほどいてやると中から本当に抵抗が出てきました。(モノクロだとわかりませんが)黄、紫、茶、金なので470Ωですね。

DIVAの場合ボタンが押されていない時のみライトが光る仕様なので、こんな感じにスケッチを書き換えるといい感じです。

```
#include "HID-Project.h"

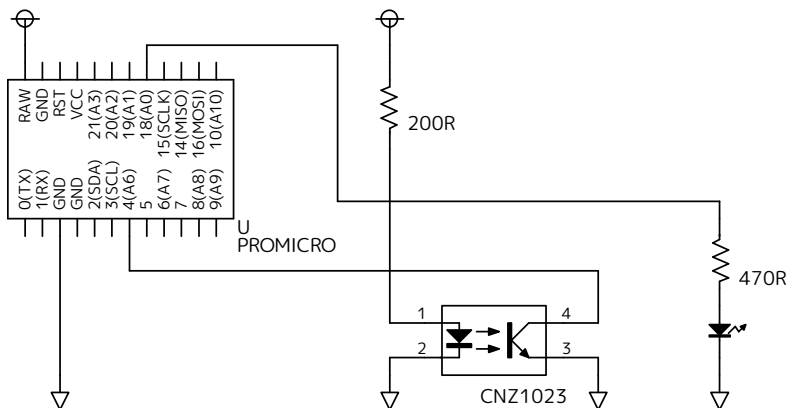
#define PAD_BUTTON 2
#define PAD_PIN 4
#define PAD_LED_PIN 18

void setup() {
    Gamepad.begin();
    pinMode(PAD_PIN, INPUT_PULLUP);
    pinMode(PAD_LED_PIN, OUTPUT);
}

void loop() {
    if(!digitalRead(PAD_PIN)) {
        Gamepad.press(PAD_BUTTON);
        digitalWrite(PAD_LED_PIN, LOW);
    } else {
        Gamepad.release(PAD_BUTTON);
        digitalWrite(PAD_LED_PIN, HIGH);
    }
    Gamepad.write();
}
```

LEDの出力のピンの定義とボタンが押された/押されていない時のピンのそれぞれの出力を書き足してみ

ました。そのうえで回路もこんな感じにしてみましょう。



とりあえずここまでできた時点で光らせてみます。

……が、なんだか思ったのよりも少し暗い。ボタン1つを照らすには少し足りない感じです。

というのもそのはず、この抵抗内蔵 LED は 12V 電源用として作られているものなのです。12V の電圧をかけた時にいい明るさになるように調整されている LED は、5V では少し暗くなってしまうのです。

では、この LED をもともと想定されていた明るさで光らせるにはどうしたらいいでしょう。

順方向電圧降下

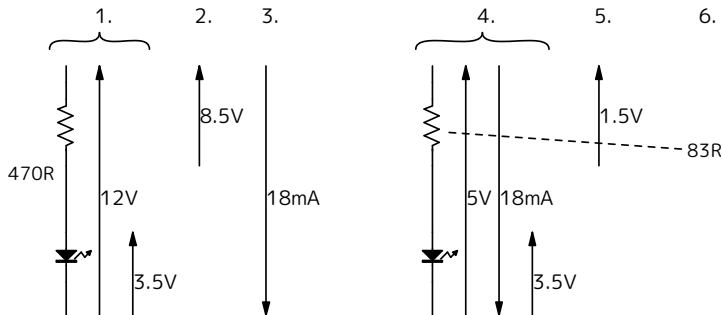
LED は、電流を流した時にそれぞれの端子であるアノード・カソード間に電圧が発生します。その電圧は何 mA 流したという電流の大小に対してそんなに大きくは変化せず、概ね一定の値を取ります。この電圧 (正確にはある電流のときの値を代表して使う) を「**順方向電圧降下**」と呼びます。この値は、LED それぞれによって個別の値になります。しかし完全にバラバラというわけではなく、色によって傾向があり、赤や赤外線が低く、青や紫外線、白が高いです。

もちろんこの順方向電圧降下を確認するにはデータシートを見るのが一番良いのですが、もともとノーブランドの製品ですから LED の型番も不明です。とりあえずここでは 3.5V だとして話を進めます(実は、実際に 12V の電源を用意して光らせたときにアノード・カソード間の電圧をテスターで測定したところ、本当に 3.5V であることを確認しました。まあそれはそれとして、白色 LED の順方向電圧降下はこの程度というのは知識として知っていていいと思います)。

この製品の設計者はこの LED に 470Ω の抵抗を接続し、12V の電圧をかけたときにちょうど良い明るさで光るように設計しているはず。ここで LED(と抵抗)に流れる電流を計算します。

1. まず 12V の電源電圧のうち、まず 3.5V が LED にかかっていると考えます。
2. その結果、残りの 8.5V が 470Ω の抵抗にかかっている電圧になります。
3. オームの法則により $I = E / R$ なので、LED と抵抗に流れる電流はおおよそ 18mA です。

4. では、電源電圧が 5V でも LED に 18mA 流れるように抵抗値を変えましょう。ここでも 3.5V は LED にかかっています。
5. 抵抗両端の電圧は残りの 1.5V です。
6. オームの法則により $R = E / I$ なので、抵抗値としてはおよそ 83Ω がちょうどいい、ということになります。一般的に購入できる抵抗の値として一番近いのは 82Ω でしょう。



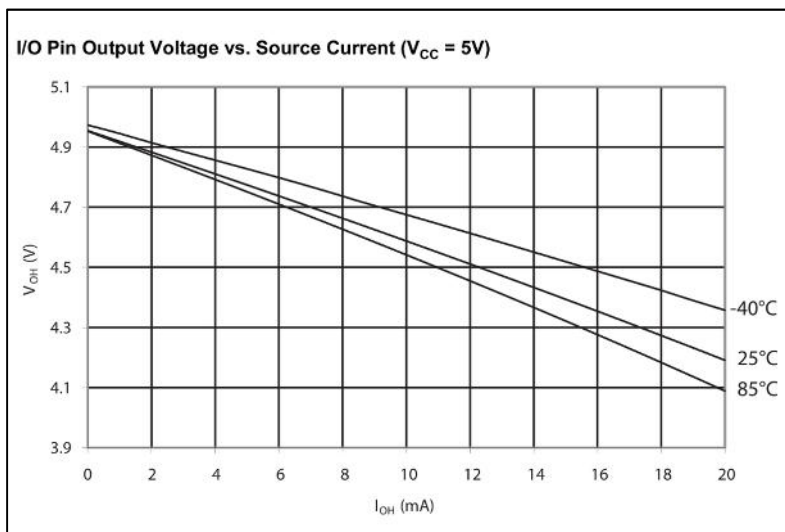
ただし、配線するケーブルやコネクタなどにも若干の電気抵抗があること、回路の設計上電源電圧が 5V を微妙に下回ること、手持ちの部品の都合上、などなどから実際の製作では抵抗値を 75Ω としました。この程度でしたら問題はないと思います。

出力電流

しかし！ ここで単純に抵抗を 82 ないし 75Ω に付け替えて、そのまま Pro Micro で LED を光らせることはおすすめできません！



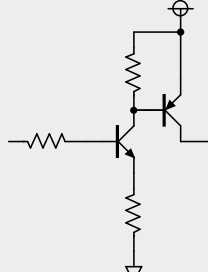
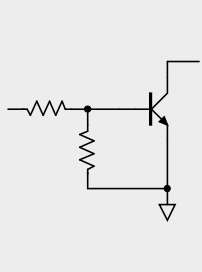
その理由を一言で言うと、「マイコンはそれ自身からそこまでの電流を流すことをそんなに想定していない」からです。

29.1 Absolute Maximum Ratings*	
Operating Temperature	-40°C to +85°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ and VBUS with respect to Ground ^(B)	-0.5V to $V_{CC}+0.5V$
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Voltage on VBUS with respect to Ground	-0.5V to +6.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins	200.0mA

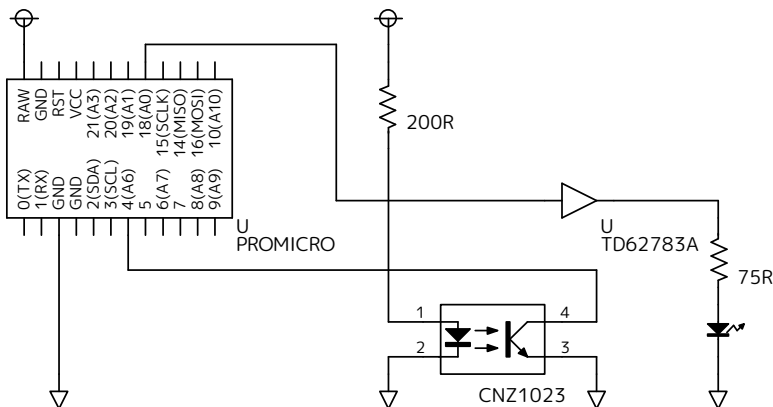


レイ」と名のつく通り、同じ組み合わせのバッファが数個並べられているのが普通です。

なお、トランジスタアレイはその並べられたバッファの個数の他にも、IC の出力が電流を引き出す方向なのか吸い込む方向なのか(ソース駆動、シンク駆動とそれぞれ呼びます)によってそれぞれ分類されますので注意。ここで使用しているのは、ソース駆動です。

	ソース駆動	シンク駆動
電流方向	引き出し	吸い込み
記号		
内部回路(簡略化)		

回路を見れば分かる通り、トランジスタアレイは電流を吸い込む方向か吐き出す方向のどちらかにしか使えません。例えばシンク駆動の場合、データシート上ではまるで普通の NOT ゲートのようにも表記されていますが、74HC04 の代わりには全くなりませんので注意です。



というわけで、光る無接点スイッチを 1 つ構成するための回路はこういった感じにすれば良さそうです。これと同じものを Pro Micro にあと 3 つつなげれば、ボタン部分は完成ですね。

◆ 実際に作って理解する静電容量タッチセンサの仕組み

タッチセンサは作れる!

世の中にはタッチセンサやタッチパネルといったインターフェースがありふれてますね。ちょっと思い浮かべるだけでも……

- スマートフォンやタブレット
- 自動券売機
- ATM
- ゲーム機(3DS, Switch, PS vita……)
- iPod などのクリックホイール(もはや懐かしい部類でしょうか)

ざっとこんな感じでしょうか。もはや私達の暮らしの中で、タッチセンサは非常にありふれたものとなっています。

こういったタッチセンサは、物理的に押しているのがよく分かるスイッチなどと比べて何だかいかにも難しそうな仕組みで動いているように感じる、かもしれません。しかし、ものによっては本当に単純な仕組みで動作させることもできるのです。この本を書いた理由はこのことを皆さんに知ってほしかったからでもあります。

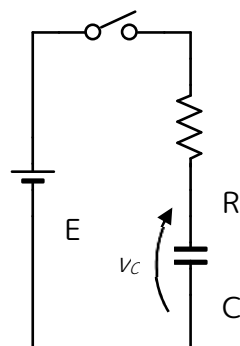
タッチセンサには、それに触る指やペンなどを検出するための方法がいくつかあります。その中で今回は、静電容量式と呼ばれる方式を紹介します。上記の中で言うと、スマートフォン、タブレット、Switch、PS vita、クリックホイールは静電容量式タッチセンサを採用しています。また、DIVA アーケード実機で使われている方式も恐らくこれと考えられています。それでは、早速タッチセンサを作ってみましょう!

……ただ、その前にこのタッチセンサの仕組みを理解してもらうために少々込み入った知識が必要になるので、簡単に解説をしておきます。

静電容量と過渡応答

図のような RC 直列回路を考えてください。急に教科書みたいになってしまいました。嫌ですね。

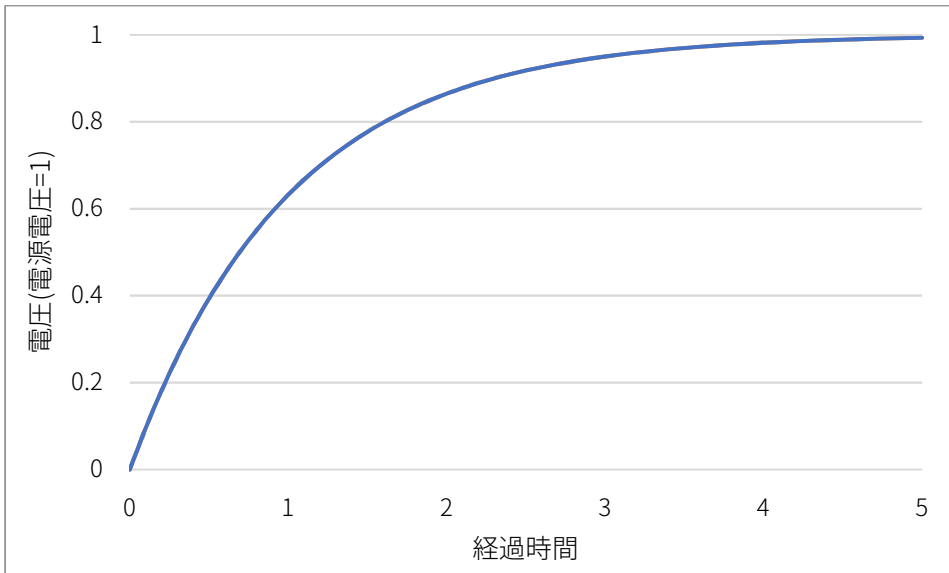
抵抗(R)とコンデンサ(C)が直列に接続されているから RC 直列回路。ついでにここではスイッチも接続されています。なお R に C にスイッチというチャタリング防止のようですが、あくまでここで登場するスイッチというのは概念上の存在というか理想的なものですのであしからず。



最初の時点では、コンデンサには電荷がたまっていないものとします。ここでスイッチを閉じると、抵抗とコンデンサにそれぞれ電流が流れます。するとコンデンサに電荷が蓄えられ、コンデンサ両端の電位差が増加します。スイッチを閉じた瞬間を $t = 0$ として、コンデンサの両端の電位差 v_c は以下の式のようになります。

$$v_c = E \left(1 - e^{-\frac{t}{CR}} \right)$$

グラフにするとこんな感じ。



どうしてこういった曲線になるのか、といったところの解説はこの本ではパスです。詳しい話は回路理論を扱う書籍などを参考にさせていただきとして、ここではとりあえず「こういう式でこういう曲線になるものだ」と思ってください。

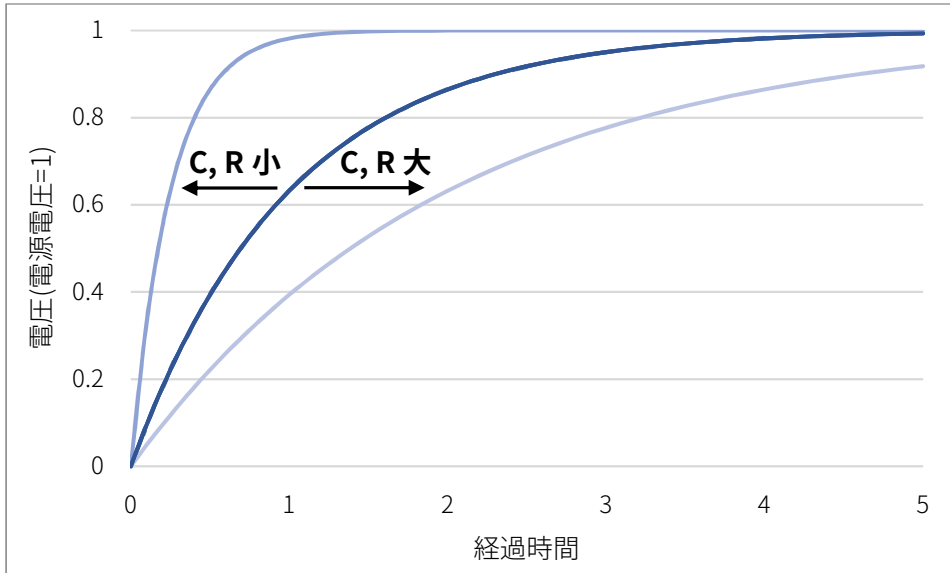
スイッチを閉じた瞬間から電圧は上っていきませんが次第にその傾きは緩やかになり、最後には電源電圧に近づいていって安定する「定常状態」になります。このグラフの傾きは抵抗やコンデンサの値によって変わり、抵抗値が大きかったりコンデンサの容量が大きかったりするとグラフはより緩やかに、抵抗値やコンデンサの容量が小さいとグラフは急峻になります。このグラフで、スイッチを閉じてからコンデンサの電圧が電源電圧の約 63% (より正確には、 $1 - e^{-1}$) になるまでの時間を**時定数**と呼び、コンデンサの充電時間の目安として表されたりします。

RC 直列回路の時定数 τ は、

$$\tau = RC$$

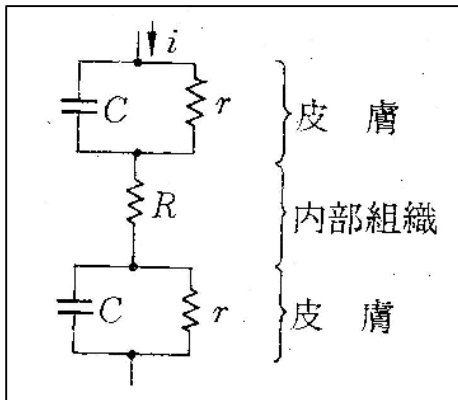
で表されます。上のグラフは、時定数 $\tau = 1$ のときのグラフになるわけです。なお、抵抗の単位は $[\Omega]$ 、コンデンサの単位は $[F]$ であることに注意。

具体的に C や R が大きくなったり小さくなったりすると、グラフは以下のように変化します。時定数が長くなったり短くなったりしているのが分かりますね。



人体は抵抗とコンデンサだ

ところで、皆さんも人体に電流が流れることはご存知と思いますが、実は、皮膚は抵抗とコンデンサが並列に接続されたようなものとして表されます。

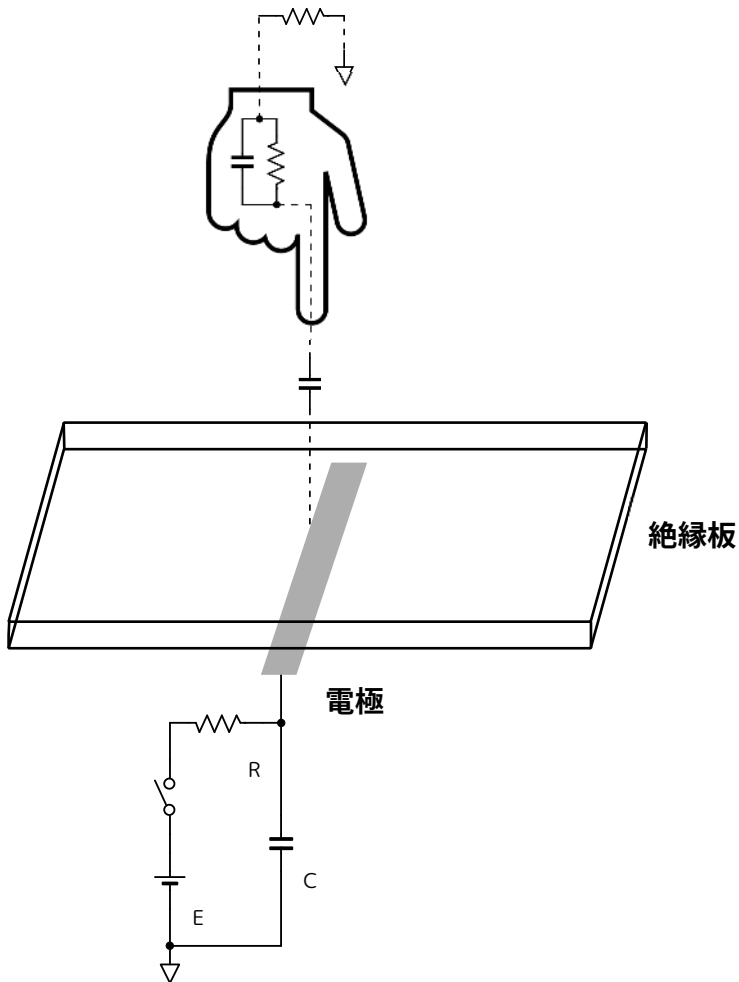


長尾 透, 病院と ME 機器との安全対策 (2), 医用電子と生体工学, 1973,

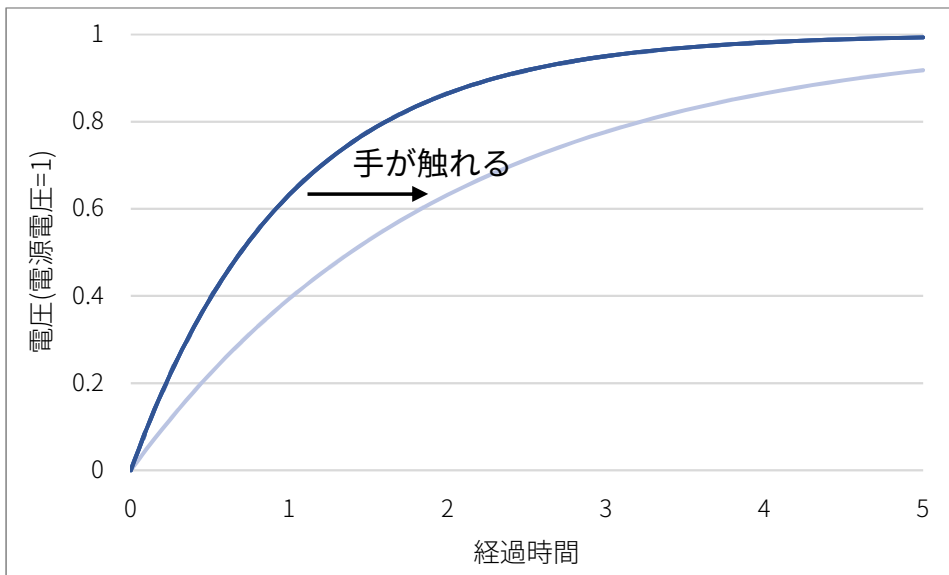
https://www.jstage.jst.go.jp/article/jsmbe1963/11/3/11_3_207/_pdf (2018/07/28 閲覧)

最初の RC 回路の R と C の間に電極を接続し、絶縁板を隔てて手を触れてみます。この時、電極と手の間が擬似的なコンデンサになるので、電極から手がまるで最初のコンデンサに並列に接続されているようになるのです。

なお、人体は内部抵抗がありながらも接地しているかのように見えるのですが、だからといって RC 回路の接地している電極に人体が直接触れているわけではないので注意してください。



このときにスイッチを閉じると、コンデンサの容量成分が増えて時定数が長くなったようなグラフに変化します。



この2つのグラフの形の違いでタッチしているかどうかを検出しているのです。

デジタル回路は、ふつう電圧の高い状態(HIGH)か電圧の低い状態(LOW)だけの状態をとります。電源電圧が5Vなのにデジタル回路の出力が2Vとか、そういったものは普通考えないものとしています。

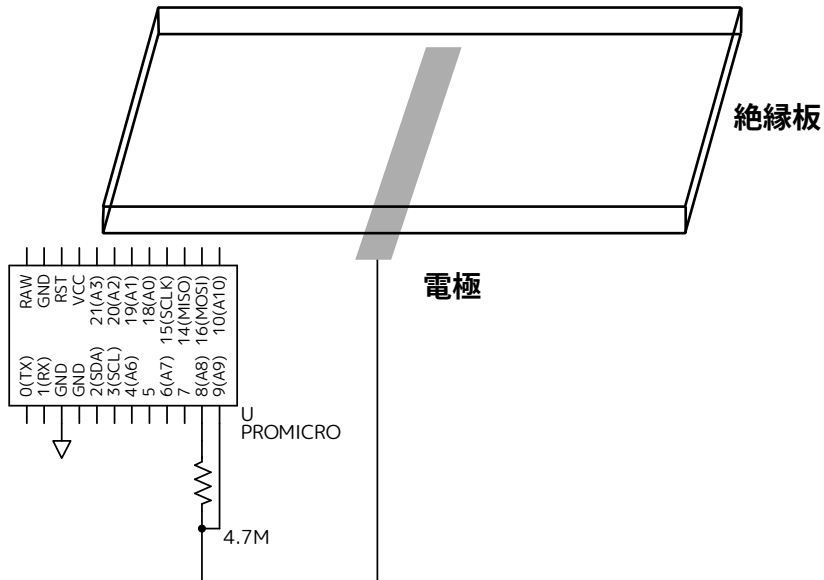
しかしながら、入力にそういった中途半端な電圧が入ってしまった場合、(とりあえずここではわかりやすくするためにあえて不正確な話になりますが)ある電圧に達するまではLOWとして判断していたが、そこを超えた瞬間HIGHとして認識するという挙動になります。

つまり、先程のグラフで言うと、「スイッチを閉じた瞬間からタッチセンサの入力がHIGHになるまでの時間」がタッチしているときとしていない時で違う、ということになります。この時間の違いをマイコンで検出するのです。

Arduino で実際にタッチセンサを作ってみる

さてここまでの仕組みがわかったら、例としてタッチセンサをArduinoに実装してみます。意外と簡単なんですよこれ。

次の図のような回路を作ります。抵抗は4.7MΩでなくても何かしら大きい値のものならOKです。タッチ用電極は何かしら薄くて広い金属を用意してください。絶縁体はプラスチック板でも厚紙でもなんでもどうぞ。



スケッチはこんな感じで。

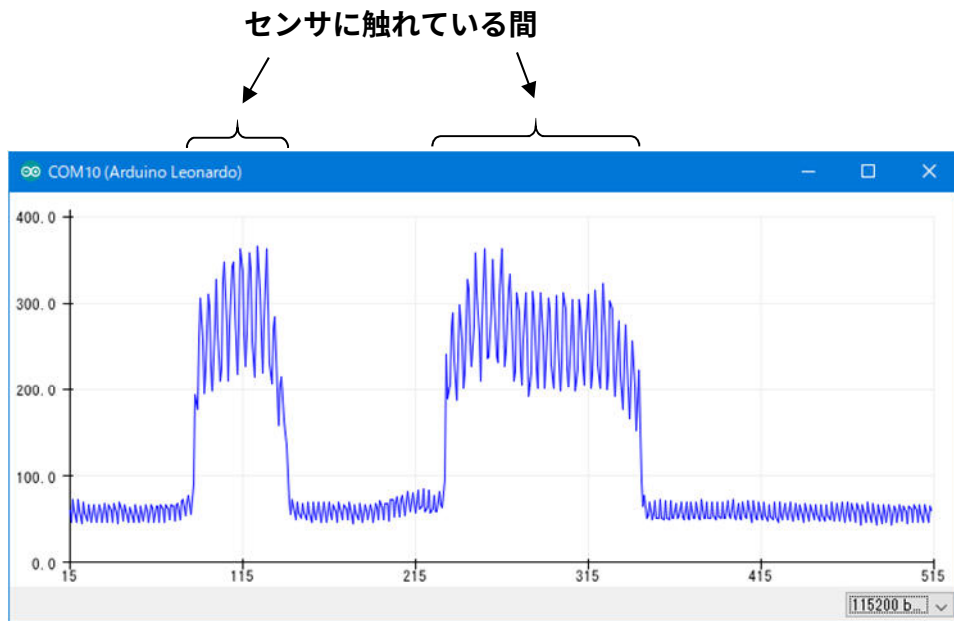
```
#define PIN_SEND 8
#define PIN_RECIEVE 9
#define AVERAGE_FILTER 32

void setup() {
    Serial.begin(115200);
    pinMode(PIN_SEND, OUTPUT);
    pinMode(PIN_RECIEVE, INPUT);
}

void loop() {
    int sensed_total = 0;
    for(int i = 0; i < AVERAGE_FILTER; i++) {
        int sensed_temp = 0;
        digitalWrite(PIN_SEND, 0);
        delay(1);
        digitalWrite(PIN_SEND, 1);
        while(!digitalRead(PIN_RECIEVE)) {
            sensed_temp++;
        }
        sensed_total += sensed_temp;
    }
    Serial.println(sensed_total / AVERAGE_FILTER);
}
```

8 番ピンを HIGH にしてから 9 番ピンが HIGH になるまでの時間を計測している、というプログラムです。

これを Pro Micro に書き込んだら、メニューバーのツールからシリアルプロッタを開きます。Arduino IDE のシリアルプロッタは、シリアル通信で受け取った数値を自動的にグラフにプロットしてくれる超便利機能です。これを使って、Pro Micro から送られてきた数値をグラフにして見てみます。なお、右下のビットレート設定欄を 115200bps にするのを忘れないように注意。



タッチセンサに触れている間だけ値が増加しています。例えばここではしきい値を 150 にしてみたりすると、タッチしている/していないを取得できそうですね。

ここまで説明してきたことが、静電容量タッチセンサの基本的な仕組みになります。

◆ PSoC でタッチセンサを動作させる

自習課題

いよいよタッチスライダーの実装です。

.....と、その前に、PSoC に関する詳しい説明や、CapSense のより詳しい使い方については、以下の公式ドキュメントも参照してください。最初の YouTube プレイリストは英語のみですが、以降の PDF ドキュメントには日本語訳がありますので安心を(?)。でもとりあえず YouTube のを最初に見て PSoC の開発の流れをなんとなく理解しておくのがいいと思います。

PSoC 4 Prototyping Kit 101 - YouTube

https://www.youtube.com/playlist?list=PLIOkqhZiy83Ff72Shqu5uh50Zkq0sC_Lb

AN79953 - PSoC[®] 4 入門

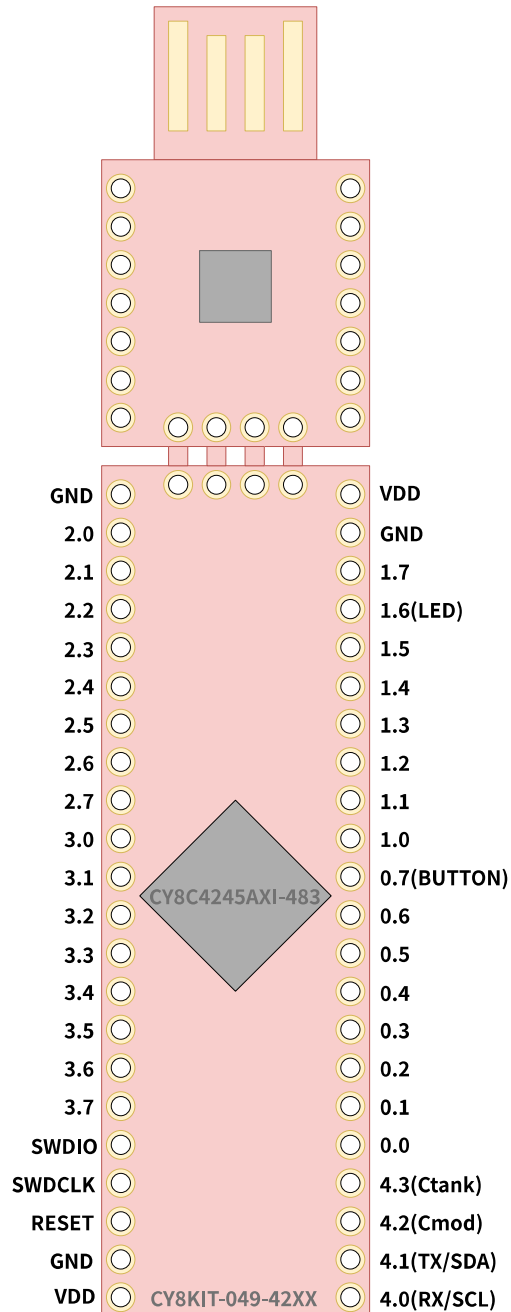
<http://japan.cypress.com/documentation/application-notes/an79953-getting-started-psoc-4>

AN64846 - CapSense[®] 入門

<http://japan.cypress.com/documentation/application-notes/an64846-getting-started-capsense>

AN85951 - PSoC[®] 4 および PSoC アナログ コプロセッサ CapSense[®] デザイン ガイド

<http://japan.cypress.com/documentation/application-notes/an85951-psoc-4-and-psoc-6-mcu-capsense-design-guide>



実物のスライダーの中身は？

前項で散々タッチセンサを Arduino で作るという話をしましたが、実際には PSoC のモジュールの 1 つである **"CapSense"** を使います。これは PSoC の各ピンを静電容量式タッチセンサにしてしまうというもの。ユーザーはスレッシュホールドがどうだとかを意識せずにパラメータの設定をフルオートで決めてくれるようにプログラムを書くこともでき、非常に便利です(もちろんパラメータの詳細な設定を手動で行うこともできます)。先程原理まで説明したのになんだよという感じが……

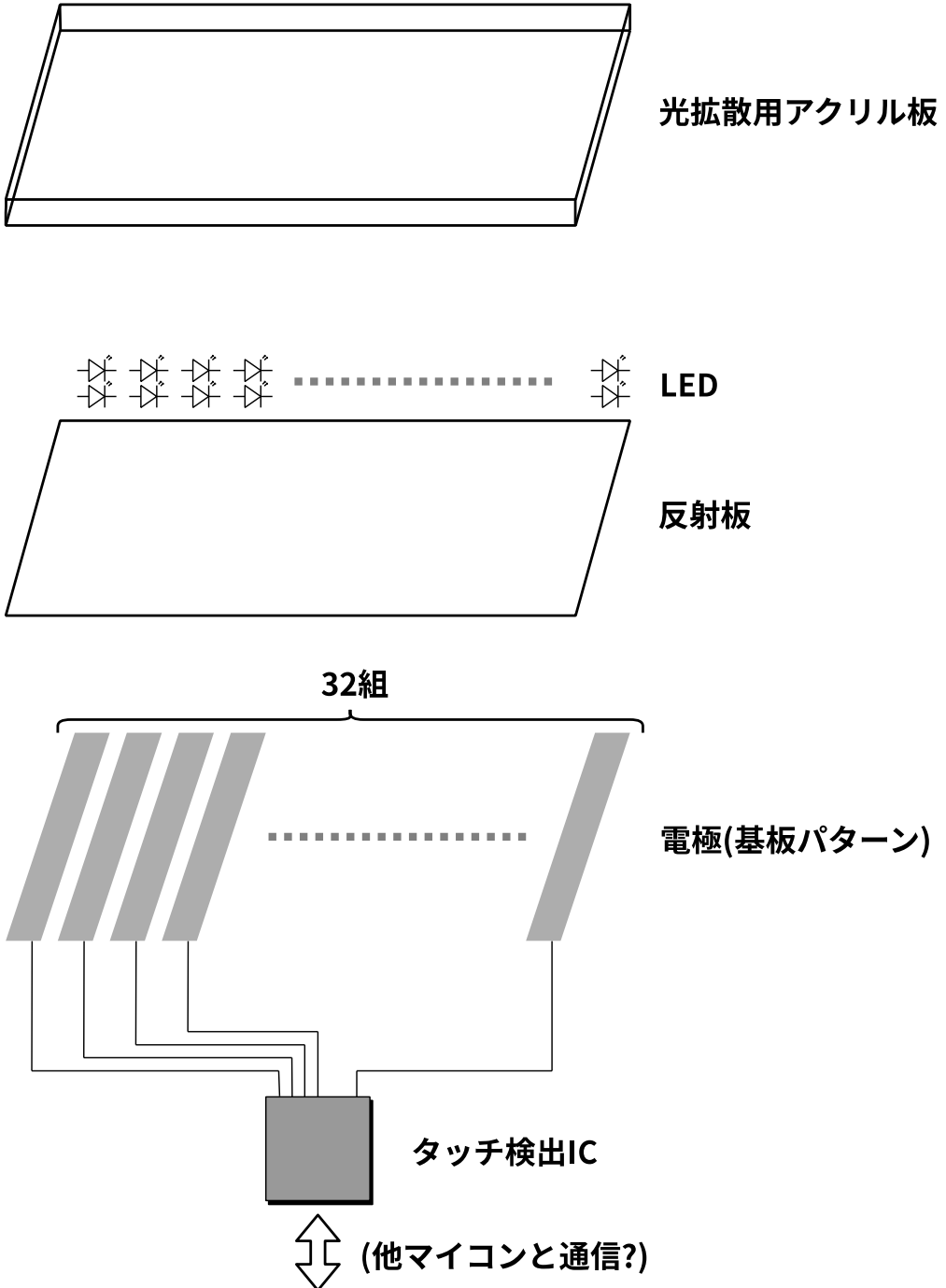
そもそも DIVA のタッチスライダーの内部が（予測も入っていますが）こういった構造なのかをおさらいしておきましょう。

DIVA のスライダーは、光を拡散させる透明なアクリル板、その下の光を反射させる板、そしてその更に下に、基板のパターンとしてつくられた 32 個のタッチセンサが横一列に並べられています。それぞれのタッチセンサは独立しており、1 つ 1 つが自身の真上でタッチされているかどうかを検出できるようになっています。それらのタッチの状況を専用のセンサ IC で検出した上で別のマイコンに送信し、タッチされている手が左右どちらに動いているのかを検出したり、LED の光り方を調節したりしていると考えられます。

今回製作したタッチスライダーも全く同じような構造にしたかったのですが、使用した PSoC のピン数などの関係で、独立したタッチセンサの個数は 30 個となりました。まあ両端 1 つぶんのセンサがなくなった程度と考えれば、実用上はほぼ問題ないと言えるでしょう。割り切りも大事です。

なお、PSoC の CapSense で作成できるものの 1 つとしてその名も「スライダー」(先程のデザインガイド 19 ページを参照してください)があるのですが、これは電極の数よりもより細かい手/指の位置を検出できる代わりに同時に 2 つ以上の場所をタッチする用途には向いていません。そのため、あくまでも PSoC のプログラム上では、実機と同じように独立したセンサを複数用意した、という考え方でタッチスライダーをつくれます。

また、イルミネーション用 LED も実機ではタッチセンサ 1 つにつき 2 個の LED を光らせていますが、ここで製作したものでは調達した部品の関係上センサ 1 つにつき LED 1 個となっています。まあこのあたりはどう作っても大丈夫でしょう。



ハードウェアの実装

まずは、PSoC 単体でタッチセンサとして動かすことを検証します。

実際に書いたプログラムは GitHub を参照していただくとして、ここでは回路方面の解説を主にしていきます。まずは実際にタッチする(といっても透明な板を隔てて、ですが)電極です。

- 適度に広く薄い
- 金属を通す
- もっといってはんだ付けしやすい

などの条件で探すと恐らく最適なのはこういった製品ではないでしょうか。



ナメクジよけなどにつかえる**銅箔テープ**です。

いやナメクジよけにも使えますけどこれ、本当にただの銅箔テープです。銅箔は上に書いたすべての条件を満たしていますからこれを使わない手はありません。100 均でも購入できますが、Amazon などで購入できる幅 5cm のテープを使うのが手っ取り早いかもしれません。

電極から基板そして PSoC に配線するのに使うのは、**80 芯 IDE ケーブル**です。連続で冗談言っているのとかではなく、長さや芯数、コネクタのピンのピッチが 2.54mm でユニバーサル基板に直接刺さる、信号線と GND 線が交互に配線されてノイズに強い、などなど今回の用途には IDE ケーブルが本当にぴったりだったのです。

80 芯 IDE ケーブルはもともとデスクトップパソコンにおいてマザーボードとハードディスクなどを接続するためのケーブルでした。しかし現在は、ストレージの接続にはご存知の通り SATA や

M.2 などより高速な転送方式(を扱うコネクタ)が主流になっており、IDE ケーブルを使用する機会は大幅に減少しました。一方でひと昔前の PC には 1 本や 2 本は必ず入っていたものであり、中古もしくはジャンクではほとんど投げ売りでどこでも手に入るケーブルなのです。

私が製作したレイアウトでは、必要になった長さは 40cm でした。この長さであれば十分かんたんに手に入るはずです。

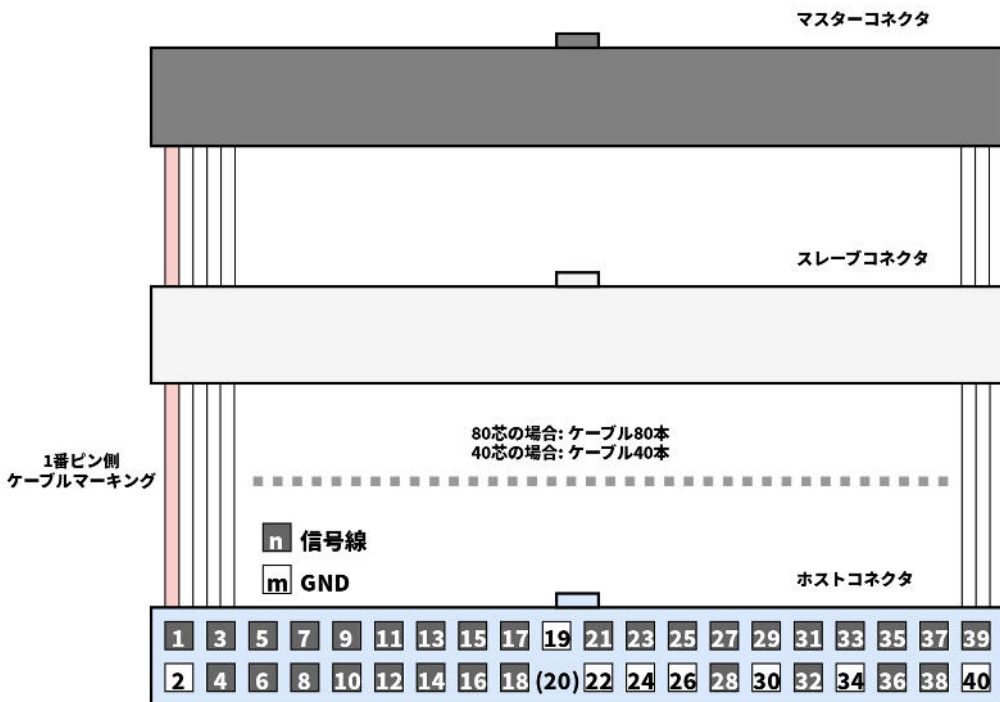
ただ、IDE ケーブルにはケーブル 1 本あたり 2 台の機器を接続することができる「マスタースレーブ接続」

という規格があり、1本のケーブルにマザーボードと機器2台用、合わせて3個のコネクタが接続されていることが多くあります。このうちユニバーサル基板の接続のために残していくのは端の1つ、ホストコネクタのみで、もう片方の端のマスターコネクタは切り落としてよいのですが、問題は中間に挟まっているスレーブコネクタです。IDE ケーブルのコネクタは、コネクタの端にある爪にカッターやピンセットなどを使って引っ掛けて外すことで、きれいに(とはいえともケーブルに跡は残っていますが)取り外すことができます。

IDE ケーブルのコネクタのピン配置は以下になっています。言うまでもないかもしれませんが、GND 線はすべてコネクタ内部で接続されているので、電極の配線には使いません。信号線として使われている31本のケーブルが電極への配線に使えることになります。

コネクタの色は、一般的にホストが青、スレーブが灰、マスターが黒となっています。

20番ピンの「キー」は(実物のコネクタを見るとよくわかりますが)そもそもピンが塞がれており配線には使えない場所です。



なお、ケーブルそのものには80芯IDEケーブルの場合ピンの結線と交互にGND線が通っていることに注意が必要です。GNDの配線が奇数番目と偶数番目のどちらの芯に割り当てられているかはケーブルによります。コネクタの部分をよく見ると、小さく「EVEN GND」とか「ODD GND」とか書かれています。私が使ったケーブルはEVEN GNDでした。一般的にはODD GNDのケーブルのほうが多いそうです。

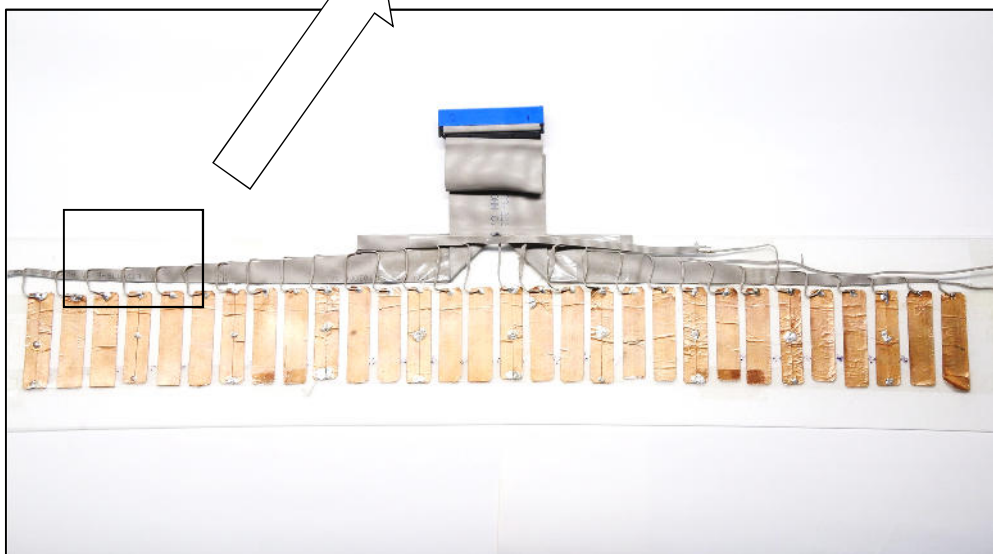
つまりケーブルそのものの配線は、1番ピン側からの端から

	EVEN GND	ODD GND
1 本目	信号線(1 番ピン)	GND
2 本目	GND	信号線(1 番ピン)
3 本目	GND(2 番ピン)	GND
4 本目	GND	GND(2 番ピン)
5 本目	信号線(3 番ピン)	GND
6 本目	GND	信号線(3 番ピン)
⋮	⋮	⋮

と並んでいることになります。

実際の配線では、なるべく融着された部分を残すように配線し、特に交互に配線されている GND 線は電極に配線する信号線のギリギリまで残すようにしています。

なお、ここまでもととの IDE ケーブルの使い方から GND 線という呼び方をしてきましたが、ここではそれらの線はすべて PSoc の 4.2 ピンに接続していることに注意してください。



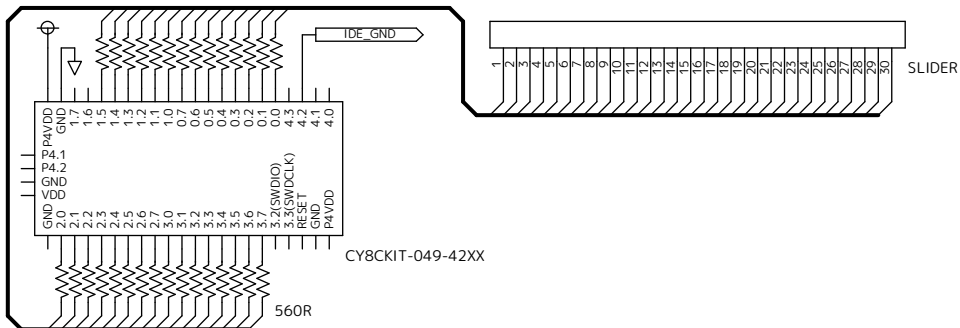
このタッチスライダはそれ自体のサイズが大きいため、センサそのものから PSoC まで多少の長さがあります。こういった配線長があると、その配線から他の配線、またはあらゆる周りの金属との間でどうしても発生してしまう「寄生容量」が問題になります。

タッチセンサの原理の説明で、人体の静電容量によりタッチ検出をしているという話をしました。センサに人体が触っていないときはセンサが読み取る容量は少なく、人体が触れているとき容量が多くなるので、その比を見ているわけです。しかし、長い配線長による寄生容量が生じてしまうと、人体が触っていないときもセンサはある程度の容量が接続されていると認識してしまうのです。そこから更に人体がセンサに触って容量が増えても、その触れているときと触っていないときの容量の比が、寄生容量が生じない場合に比べて小さくなってしまいます。そのため、正確なタッチ検出ができなくなってしまうのです。

その対策の一つとして、ここのプログラムでは、本来の IDE ケーブルの用途では GND だったピンを「被駆動シールド」として動作させています。被駆動シールドには、センサのピンに出力されるのと同じ波形の信号を、十分に出力インピーダンスを低くして出力されています。こういった信号が出力されたピンをシールドとして配線することで、寄生容量として働いてしまう他の金属の影響を受けにくくなっているのです。ここでは、その被駆動シールドピンを 4.2 ピンに設定しているのです。

また、ノイズ除去と測定精度維持の兼ね合いから、電極と PSoC の端子の間、なるべく PSoC 側に 560Ω 程度の抵抗を直列に追加することが推奨されています。

このタッチスライダの配線はひたすら単純作業なので、頑張って全て配線しましょう。



手の動く方向検出のアルゴリズム

さて、ここまで電気的な話をしてきましたが、ソフトウェア的な話も少しだけしておきます。まず前提として、プログラム上ではタッチセンサにおける「どのセンサがタッチされていてどれがタッチされていないか」という情報を1秒間に何百回も取得しています。この情報から、手の動きを検出していきます。また、最終的にはゲームに対して左右のジョイスティックをどう動かすかという情報を送信します。これに関してはDIVA FTのキーコンフィグに準拠した設定となっているため、実機でのスライダの動作と微妙に違うのですが、まあ動けば良いということで。

座標の取得

まずはある瞬間での手の位置、というか次元の座標を求めます。タッチセンサは1つの幅があまり広くないので、指先で軽く触れたりなどしない限り、複数のタッチセンサに連続して手が触れていることになります。そういった場合、連続して触れている範囲のちょうど中間地点を手が触れている座標として判定します。このタッチしている手の位置は左端から順に確認していて、タッチセンサ1マス以上離れた位置に2つタッチしている手があることを判定したら、それより右側、3箇所目以上のタッチしている部分は無視します。

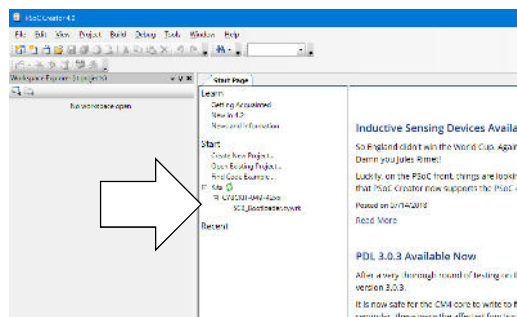
手の動く方向

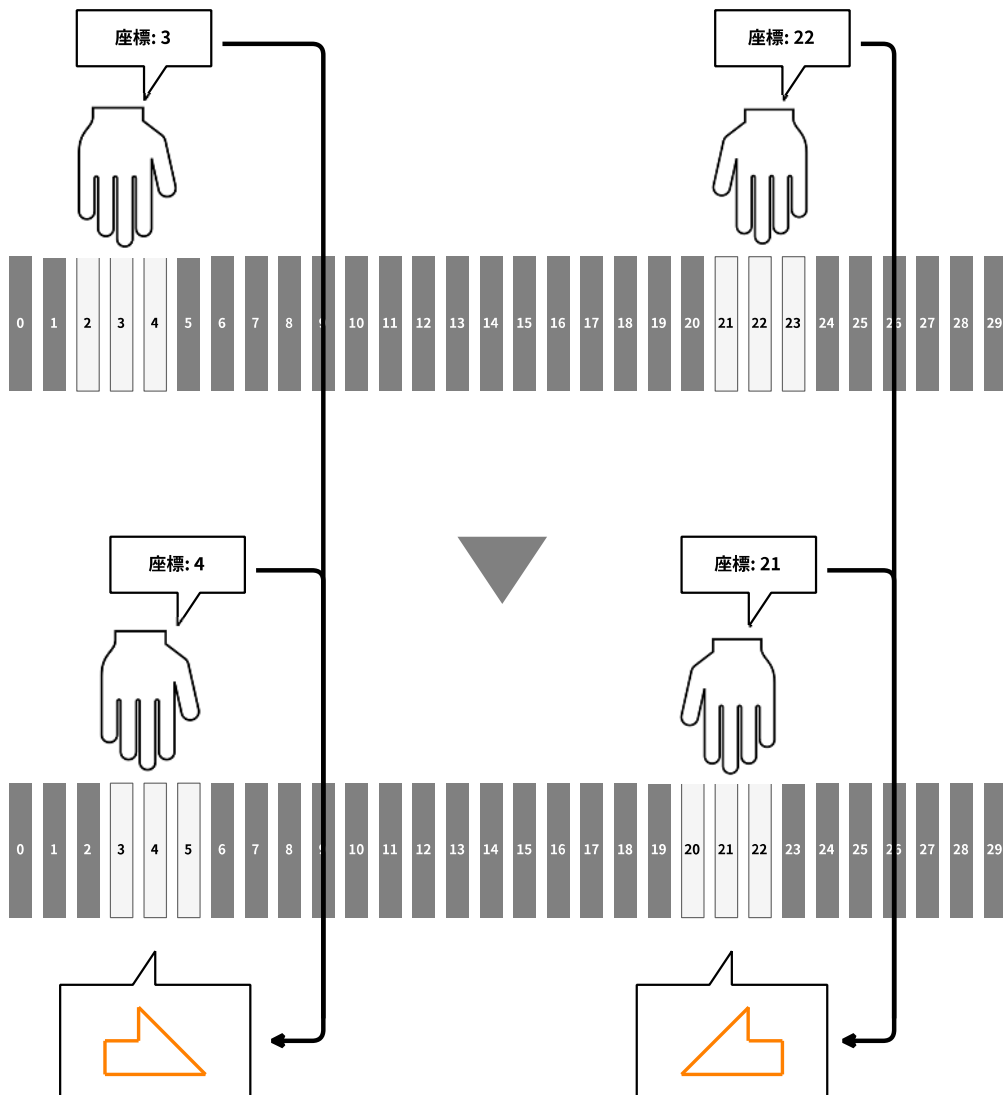
そういった手の座標の取得を、1秒間に数百回のペース、つまり数ミリ秒ごとにタッチセンサの情報が更新されるたびに毎回行っています。手の座標が左右どちらかに1だけ移動したときに「手が移動した」と判定しています。また、タッチスライダを片手で触れているとき、その手が半分より右側にあるときは右側のジョイスティックを動かしているものとして認識させます。片手でスライドさせているときは左右どちらのジョイスティックを動かしてもゲーム上の判定としては同じですが、両手スライドの判定のためのちょっとしたチューニングです。もちろん、例えば両手とも半分より左側にあった場合などでも、それぞれの手の相対的な位置から左右の手を判定しています。

実際に書き込む

ここで、PSoCにこのCapSenseのプログラムを書き込んでみます。ここで使用しているPSoC 4200 Prototyping Kit(CY8CKIT-049-42XX)は、別途通常のライターを使用して書き込むことも可能なのですが、基板内蔵のUSBコネクタから書き込む場合は一般的なPSoCと違い若干特殊な書き込み方法が必要となります。

まず、PSoC Creatorを起動してStart Page内Examples and KitsからSCB_Bootloaderプロジェクトを選択し、適当な場所に保存します。





次に、以下のプロジェクトをダウンロードし、SCB_Bootloader フォルダと同じ場所に置きます。

pol8139/CY8CKIT-049_touch_to_serial

https://github.com/pol8139/CY8CKIT-049_touch_to_serial

CY8CKIT-049_touch_to_serial\CY8CKIT-049_touch_to_serial.cydsn\CY8CKIT-049_touch_to_serial.cypri
を PSoC Creator で開き、メニューバーの「Build」→「Build CY8CKIT-049_touch_to_serial」をクリック
してビルドします。

ビルドが終了したら、PSoC 4200 Prototyping Kit の端にある小さいタクトスイッチを押したまま USB 端
子を PC に接続してください。青い LED が 2Hz で点滅したら OK です。

PSoC はそのままにして、「Tools」→「Bootloader Host」を選択して Bootloader Host ウィンドウを開き

ます。

File は、CY8CKIT-049_touch_to_serial\
CY8CKIT-049_touch_to_serial.cydsn\
CortexM0\ARM_GCC_541\Debug\CY8CKIT-
049_touch_to_serial.cyacd を選択してください。

ポートが PSoC のものであること、Baud が 115200
であることを確認したら、Program ボタンを押しま
す。PSoC へのプログラムの書き込みは数秒で終了し
ます。ウィンドウ下部に成功のメッセージが表示さ
れ青 LED が消えたら書き込み終了です。

タッチ感度の調整について

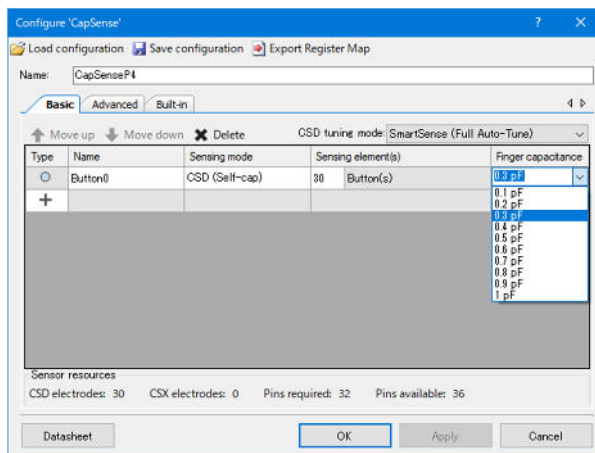
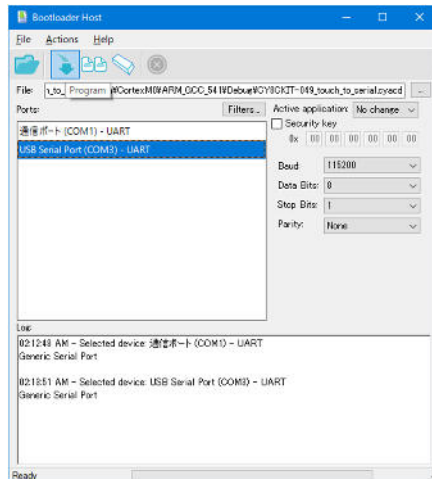
これは実際にコントローラが全て完成してからでない調整できないものなのですが、PSoC Creator で操
作するパラメータなのでここで紹介しておきます。

ここでは、PSoC でタッチセンサを構成するに当たり、調整するパラメータをほぼ全自動で設定してくれ
る"SmartSense (Full Auto-Tune)"というチューニング方法を使いました。これを使うことで、設計者は先
程 Arduino で考えたような、わざわざ値を読み取って最適なきい値を自分で決めて……といったことを
手動でする必要がなく、すべて PSoC 自身が自動でパラメータの調整を行ってくれます。

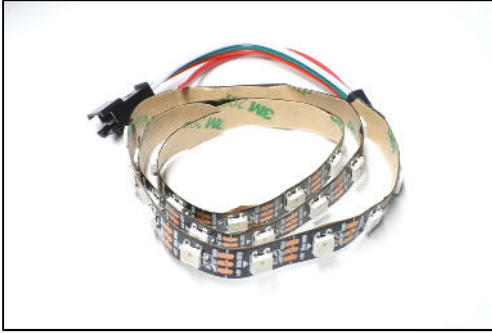
唯一手動での調整が必要なのは、Finger Capacitance というパラメータ。その名の通りタッチする指の静
電容量を設定するところです。

Finger Capacitance の値は、タッチセンサの感度と反比例します。最初はデフォルトの設定でも問題な
いと思いますが、指一本スライダーに軽く触れただけで連続した数個のセンサがタッチ判定してしまうな
ど感度が高すぎる場合は Finger Capacitance 値を上げ、逆に反応しづらい場合は値を下げます。この値は、
SmartSense (Full Auto-Tune)では 0.1pF
から 1pF まで 0.02pF 単位で調整できま
す。ざっくりとした調整なら 0.1pF 単位
で上げ下げする程度がちょうどいいと思
います。

このパラメータを調整するには、PSoC
Creator の TopDesign.cysch タブで、
CapSense モジュールをダブルクリック
して設定ウィンドウを開きます。ここで、
感度の調整を始めとする様々なパラメー
タの調整をします。



◆ タッチスライダーを光らせる



WS2812B 搭載 LED テープさん再掲

便利な LED

WS2812B というフルカラーLED をご存知でしょうか？ 見た目は表面実装の LED チップそのもの。しかし、ピンはそれぞれの色の LED に直結されているのではなく、入力に"こういう色を出してくれ"と、電圧や電流などアナログな信号ではなく、シリアル通信によるデジタルな制御信号を送ると、その色の通りに光ってくれるという優れもの。さらに IC を数珠つなぎに接続することができ、どんなに数が増えようともたった 3 本の線(そのうち 2 本は電源用の線なので、信号を送っている線はたったの 1 本!) で LED の色を個別に指定することが可能なのです。

実機のスライダーを見てみると、2 個 1 組で 32 組のフルカラーLED がそれぞれ光っているのがわかります。ここで普通に考えてしまうと、それこそタッチセンサ本体でやったようにマイコンのピン 1 本に対してそれぞれの LED の 1 つの色を接続して……といったように大変な本数の配線をしなければいけないように思います。しかしこの WS2812B を使うことで、大幅に配線の本数を減らして LED を制御することが可能になるのです。

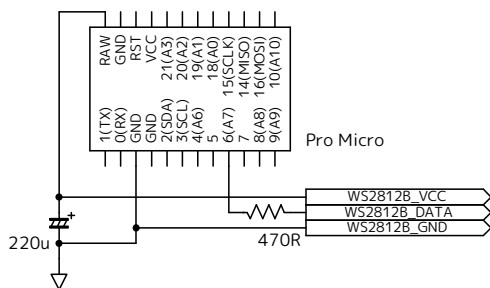
1 つのチップを光らせるために必要な信号は、赤・緑・青のそれぞれの光の強さを 8bit(256 段階)で表した 24bit の信号となります。連続した信号を受け取った最初の WS2812B は、その信号のうち最初の 24bit 分だけを取り出し、自身をその色に光らせます。そして、そのチップの出力からは 24bit 分を切り取った信号を出力するのです。これにより、次に接続された WS2812B は結果的にはマイコンから 2 番目に送信されていた色の情報を受け取って光り、さらに次に接続された WS2812B は……と繰り返すことで、それぞれの LED がそれぞれの色で光らせることができるのです。

もちろん、そういった信号もわざわざ 1 から書かなければならないわけではなく、便利なライブラリがあるのでそれを活用します。Adafruit_NeoPixel ライブラリは、その名の通り Adafruit 社の NeoPixel シリーズを動作させるための Arduino 用のライブラリなのですが、その NeoPixel の実態はただの数珠つなぎにされた WS2812B そのものだったりします。また、上で説明をした配線も WS2812B 使用の LED テープを購入すればすでに完了しています。そのため、かなり簡単な作業だけでスライダーを光らせることができます。

Pro Micro との接続

ここでは動作確認として、サンプルスケッチを動作させてみましょう。

ドキュメント\Arduino\libraries\Adafruit_NeoPixel\examples\simple フォルダ内にある simple.ino を Pro Micro に書き込んでください。その上で、Pro Micro と LED テープを以下のように配線します。



ここでいくつか注意点を。

- LED テープの電源端子になるべく容量が大きめの電解コンデンサを追加してください。ドキュメントによると 1000uF とのことですが、サンプルスケッチを動かす程度ならそこまで必要ないのではと思います。自分がテストしたときはとりあえず手持ちの 220uF を接続しましたがこれで大丈夫でした。もちろん耐圧は 6.3V とかその程度で十分です。
- Pro Micro の信号出力ピンと LED テープの最初の入力との間に抵抗を入れてください。300Ω から 500Ω 程度が推奨されますが、大きいに越したことはありませんのでとりあえずなにか入れておきます。できれば LED テープ側になるべく近いところに追加することがいいのですが、まあ 10cm とかその程度離れているくらいなら問題ないかと思います。
- WS2812B は電源電圧 5V 専用です。この本の中で説明した Pro Micro を購入しているなら電源電圧 5V のはずですが、もし 3.3V 品などを使っている場合、何らかの変換回路が必要です。
- Pro Micro に電源を入れる前にすべての接続を完了させてください。
- WS2812B の信号の流れる向きは、LED テープの基板上に(おそらく三角形や矢印のマークなどで)書かれています。その向きを確認し、間違えて Pro Micro の信号出力に LED テープの出力を繋がないよう気をつけてください。NeoPixel 純正品だけでなく基本的には多くの互換品の LED テープは、メスコネクタ側が入力になっているはずですが。
- このサンプルスケッチ程度であればこのままで問題ないと思いますが、WS2812B は最大輝度で 1 つのチップあたり 60mA を消費します。大量に高輝度で光らせる場合、電源を Pro Micro ではなく他のところから持ってくることも考えてください。

電源を入れて、LED テープの根本から順番に 16 個 LED が緑色に光ったら動作確認完了です。あとはもうタッチセンサの状態に合わせてうまい具合に光らせてあげれば大丈夫ですね。

◆ GIMX で HID コントローラを PS4 に認識させる

これまで DIVA コントローラを HID ゲームパッドとして Pro Micro で作ることを解説してきたわけですが、実は、PS4 は純正コントローラもしくは認証を受けたサードパーティ製コントローラしか動作を受け付けてくれません。その判定はなかなか厳しく、例えば PS3 純正の Dualshock3 を接続しようとする、そのコントローラが Dualshock3 であることを認識した上で「動作しない」と表示するのです。もちろん他の PC 用コントローラも全く動作しません。

そのため、PS4……に限らず従来からのゲームコンソールでもそうですが、そういったコンソールに自作コントローラを認識させる方法として、純正 or 認証済みコントローラを分解して内部の端子に直接はんだ付けをするという方法が行われてきました。

なお、国内でワイヤレスコントローラを分解して改造するのは、電波法、というかいわゆる技適に引っかかる恐れがあるので注意。また、そうでなくても既存のコントローラの改造はある程度リスクを伴うものです。

そのため、今回は全く別のアプローチとして、**GIMX** というシステムを使ったものを紹介します。

GIMX は、PC に繋がれたマウス／キーボード／コントローラを認証済みコントローラとしてゲームコンソールに認識させるコンバータです。PC 上で動作させるソフトウェアと、PC からゲームコンソールへのコントローラの情報の橋渡し役となる USB アダプターケーブル(条件付きで Bluetooth による無線接続も可能)から成り立っています。これを用いることで、コントローラの分解なしに好きなコントローラを PS4 で動作させることができるのです。

GIMX

https://gimx.fr/wiki/index.php?title=Main_Page

GIMX は、動作させたい非純正のコントローラと(接続対象が PS4 の場合)認証のための純正コントローラを PC につなぎ、PC とコンソールを専用の USB アダプタで接続して、PC 上のソフトウェアで設定した(非純正コントローラのどのボタンが純正コントローラのどのボタンに当たるかという)変換テーブルをもとに、コンソールに認証済みコントローラ……より正確には、「ホリパッド FPS プラス」のふりをして情報を送信します。そのために必要なソフトウェア／ハードウェアのセットアップをしていきましょう。

必要なハードウェア

DIVA コントローラ

そりゃそうだ。

これまで明記をしていませんでしたが、最終的にコントローラとして動作させるためのスケッチは“製作例”の項を参照してください。

PC

Windows7 以降、もしくは Ubuntu16.04 以降が必要です。もしくは、Raspbian がインストールされた Raspberry Pi でも実験段階ながら動作するようです。ここで解説するのは Windows を使った場合の手順ですが、Ubuntu や Raspbian でもおおむね手順は一緒です。

USB アダプタ

ここで序盤に紹介した **USB-UART 変換アダプタ**と(2 つ目の)**Pro Micro** が必要になります。他のいかなる USB ケーブル、例えば PC から PC にデータ転送が出来るケーブルなどは使えません。

純正 Dualshock4 コントローラ

GIMX が PS4 への認証を突破するために必要です。

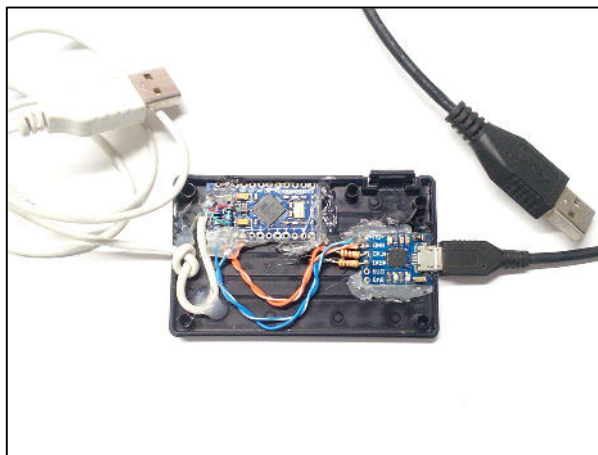
USB アダプタをつくる

先程 Bluetooth での接続もできると書きましたが、有線での接続が出来るようなケーブルの用意をおすすめします。

以下の通り、USB-UART 変換アダプタと Pro Micro をはんだ付けしてください。

Pro Micro	USB-UART
GND	GND
RX	TX
TX	RX

私は写真のようにミンティアのケースにそれぞれの基板をグルーガンで固定し、さらに RX-TX 間に 1kΩ の抵抗を入れています(この抵抗は必ずしも必要はないかと思います)。



PC へのソフトのインストール

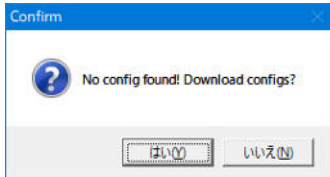
Guide - GIMX

<https://gimx.fr/wiki/index.php?title=Guide&platform=ps4&connectiontype=USB&ostype=windows&device=Pad>

このページの少しスクロールしたところにある"GIMX Installation"のリンクから、自分の PC の bit 数に合わせて適したものをクリックしてダウンロード・インストールします。まあどっちかわからなければ 32-bit の方を押しておけば大丈夫です。

ソフトの設定

無事 PC に GIMX がインストールできたら、まずは gimx-launcher を起動させます。



初めて起動したとき、設定ファイルが無いのでこのようなダイアログが表示されたりしますが、ここから今必要な設定用 xml ファイルをダウンロードすることはできないので「いいえ」を押してください。

ここで必要な xml ファイルは、以下からダウンロードしてください。

gimx で自作 DIVA アケコンを使用するときのコンフィグファイル(Windows 用)

<https://gist.github.com/pol8139/b45cab3076d4c8f9225bbc5ec26424a6>

これをダウンロードしたものを、gimx-launcher のメニューから「File」→「Open config directory」をクリックして開いたフォルダに保存してください。

なお、このファイルは Pro Micro に Arduino Leonardo のブートローダが書き込まれていることを前提としています。Arduino Micro など別のブートローダの場合、「USB ゲームコントローラのセットアップ」から名前を確認して、xml ファイルをテキストエディタで開き「Arduino Leonardo」と書かれている部分をすべてその名前に置換してください。

USB アダプタへのファームウェアのインストール

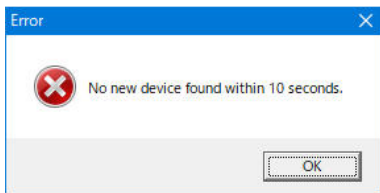
先程製作した USB アダプタの USB コネクタの両方を、PC に接続します。

Gimx-launcher のメニューから「Help」→「Update firmware」をクリックした後、ファームウェアの選択で「EMUPS4.hex」を load します。

手順に従っていくと、途中で USB ケーブルを抜き差しするよう促されます。この時抜き差しするのは Pro Micro のほうです。USB-UART ボードではありません。

Unplug/replug the USB cable from/to computer USB port.

もし何度試しても以下のように「No new device found within 10 seconds」といったエラーメッセージが出てしまう場合、上記のケーブル抜き差しするタイミングで、ケーブルを差したままで GIMX アダプタ内 Pro Micro の RST 端子と GND を一瞬ショートさせてみてください。



ターミナル(コマンドプロンプト)の画面が一瞬表示された後、「Firmware loaded successfully!」のメッセージが表示されたらインストール完了です。

PS4 に繋ぐ!

以下の手順でそれぞれのケーブルを接続していきます。

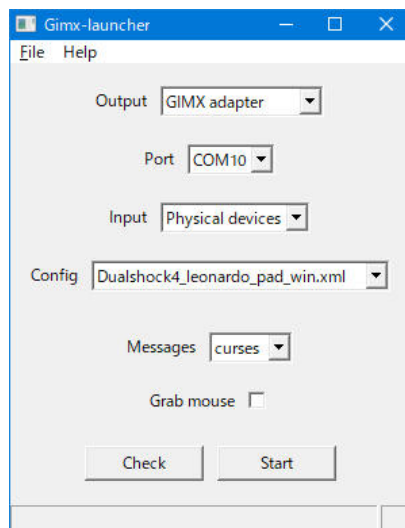
- PS4 と PC を起動しておきます。
- GIMX と DIVA コントローラに関するすべての USB ケーブルを抜きます。
- Pro Micro(DIVA コントローラ)を PC に接続します。
- USB-UART アダプタを PC に接続します。
- Pro Micro(GIMX アダプタ)を PS4 に接続します。
- Dualshock4 の電源を切ってから、PC に接続します。

gimx-launcher の設定をしていきます。

- Port は、USB-UART アダプタのポート番号を選択します。デバイスマネージャーからポート番号を確認してください。
- Config は、先程ダウンロードしたファイルが選ばれているかどうか確認します。
- Messages は none でも良いですが、curses や text にしておくとボタンの状況をリアルタイムに PC で確認できるので便利です。

Start を押して、ターミナル(コマンドプロンプト)の画面が出てきたら、手順に従って Dualshock4 の PS ボタンか DIVA コントローラの Start ボタンを押します。

DIVA コントローラが PS4 に認識されたら成功です!



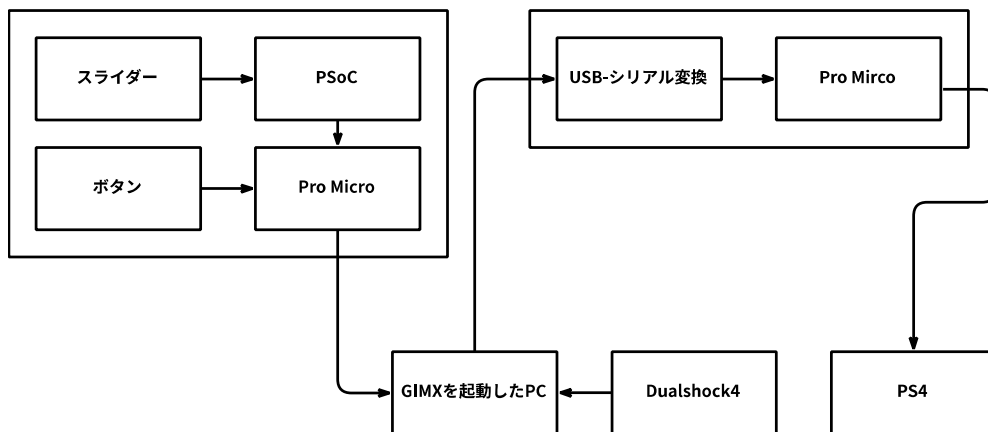
◆ 製作例

私が実際に製作したものを掲載します。

ブロック図

まずは全体のシステムをここで一度おさらいしておきます。

- コントローラを PS4 で使用するとき、前述の「PS4 に繋ぐ!」の手順を毎回最初に行う必要があることに注意してください。
- 今回は既存のコントローラの乗っ取りという一般的な手順を使わずに自作コントローラを使用しているためこのような複雑な構成になっています。乗っ取りを行うなら、コントローラと PS4 を直結できます。



ソースコード

pol8139/CY8CKIT-049_touch_to_serial(GitHub)

https://github.com/pol8139/CY8CKIT-049_touch_to_serial

再掲になりますが、PSoC に書き込むプログラムです。

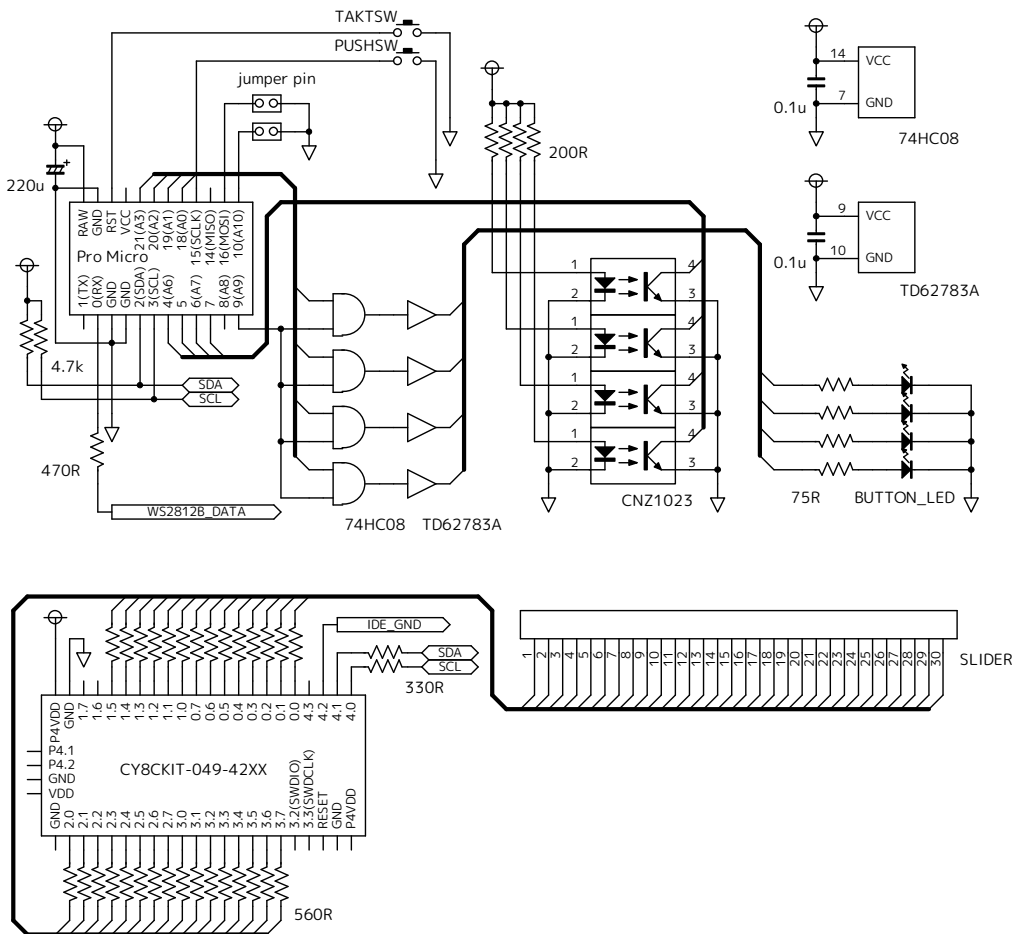
pol8139/leonardo_ser_to_key(GitHub)

https://github.com/pol8139/leonardo_ser_to_key

さんざんテスト用のスケッチを書いてきた Pro Micro に、最終的に書き込むスケッチです。

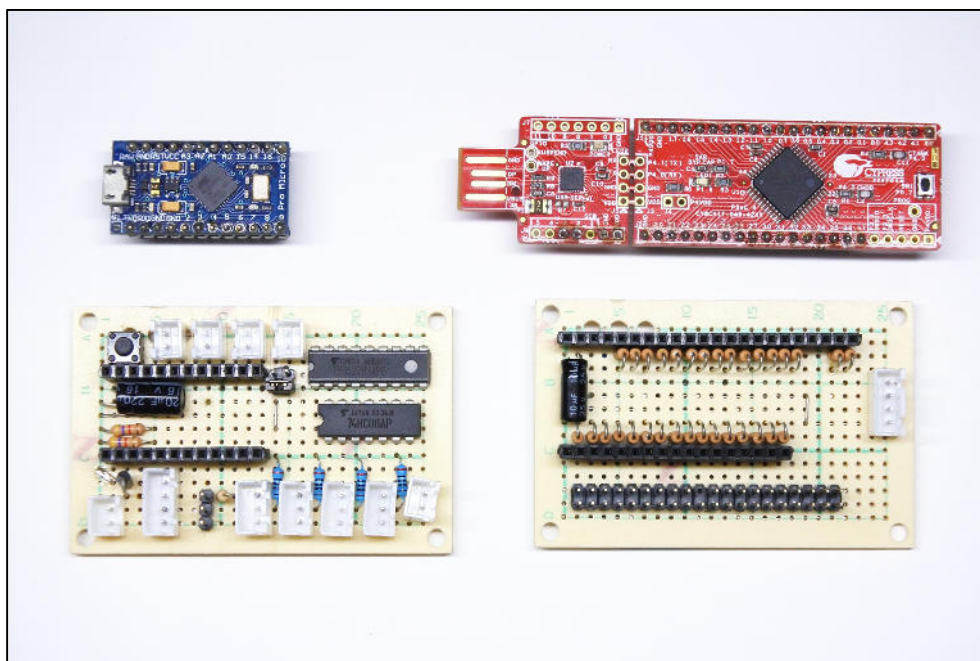
回路図

- 電源はすべて Pro Micro の USB 端子から供給しています。
- デバッグのためにジャンパーピンを用意していたり、リセット用タクトスイッチを実装したりしています。
- 74HC08 によるボタンの明るさ調整の機能を実装しようとしたのですが、電気的な配線は行ったものの、ソフト側が未対応だったりします.....



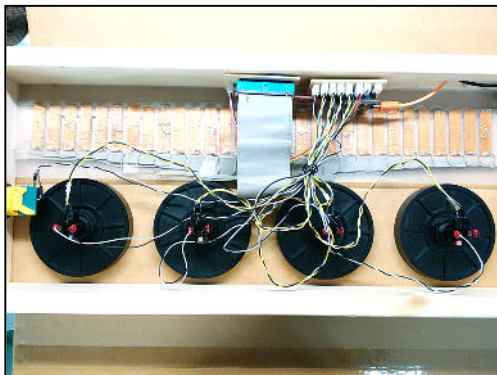
実装例

- ここでは、基板を2枚に分けて「Pro Micro とボタン用コネクタ類」「PSoC とタッチスライダ接続用 IDE ソケット」のそれぞれに役割分担をさせています。両者をつないでいるのは、I²C 信号(SDA、SCL)と電源(VCC、GND)の計4本のワイヤのみです(基板上に乗っている4ピン用コネクタがそれです)。
- 秋月電子のCタイプユニバーサル基板2枚に収めるには多少きつかったので、Cタイプ3枚とか、もしくはもう少し大きい基板を使うとかしたほうが良いかもしれません。
- タッチスライダの製作時に切った IDE ケーブルの余りが基板裏の配線にぴったりでした。
- 電源はすべて Pro Micro に接続する USB ケーブルから供給しています。回路図上では示しませんが、回路の上流(ProMicro の RAW 端子直後など)にポリスイッチなどを保護のために入れておいたほうが良いかもしれません。そうでなくても USB 機器としてはそこその電流を消費するので注意です。



筐体

- 天板は MDF、側板は 1x3 です。タッチスライダー部は PP 板を何枚か重ねています。
- 実機では○ボタン右上あたりにある Start ボタンですが、それに似た小型のボタンを筐体右側面に取り付けています。GIMX のコンフィグファイルでこのボタンは PS ボタンに割り当てていますが、何か他の設定にしてもいいかもしれません。
- テープ LED は、スライダー下から斜め上に光るように角度を調整しています。



◆ 実機を自宅で動作させるには？

ここでは筐体を木材で 1 から作りましたが、最近は個人で DIVA AC 筐体の実機を購入する猛者もいるようです。実機を自宅で動作させるには、以下の方法がありそうです。

- 自宅にゲームセンターと同じネットワーク環境を構築して、アーケードそのものの ROM を起動する。
- DIVA AC 筐体に PS4 を内蔵させ、DIVA FT(DX)を起動する。

有名な話ですが、DIVA のアーケード実機はネットワークに接続した状態でないと基本的に稼働ができないようになっています。そのため、個人が購入してそのまま家で DIVA AC を起動することは普通できません。アーケードそのものの ROM を起動することは普通に考えるとパスです。

ですので、筐体に PS4 を内蔵させるのが良いでしょう。問題はどうやって実機のボタンやスライダーを PS4 のコントローラに変換するか、という話になります。ボタン類はただ単に電氣的な配線だけで済みそうですが、問題はスライダー(と LED)です。

- 実際のコントローラに使われているスライダーのプロトコルをどうにかして解析し、情報を外部から送信/取得する。
- 実機のスライダーの電極に使われている基板上の銅箔パターンを引き出し、CapSense などでもタッチ情報を取得する。
- 実機のスライダーのガワの部分だけを拝借し、スライダーを電極から作る。

現実的なのは 2 番目か 3 番目でしょう。

私自身は実機を所有していないのでまったくの想像になりますが、PS4 版を動作させるだけならそこまで難しいものではなさそうです。ほかにも、実機に使われているであろうスライダー電極と LED が実装された基板と同じ寸法の基板を 1 から設計し、CapSense などでも取得するのも良さそうです。そのサイズの基板を発注するのはなかなかリスクがありそうですが.....実機を所有している方はいろいろ試してください(?)。

◆ おわりに

なんだか、そんなに薄くない(他意は無いです)本になってしまいました。ここまで読んでくださった皆さん、お疲れ様でした。

動画を公開したり Twitter で製作時の進捗を公開したりしていた時に一番多かったフィードバックは、なんとと言っても「タッチスライダーが個人で作れるのか」という驚きの声でした。しかしタッチスライダーで使ったものは PSoC4 という何年も前からあるデバイスです。静電容量式タッチセンサの仕組みも、CR 回路の過渡応答というシンプルな仕組みを利用したものです。何か最先端技術とかを使っているわけではなく、「枯れた」技術を理解し、それらを組み合わせて目的の動作を達成しているわけです。

同時に「作って欲しい!」というフィードバックも複数いただきました。正直なところちょっと嬉しかったりするのですが、すべてお断りいたしました。申し訳ありません。何より人に譲渡できる品質を保つことができないというのが最大の理由です。実際に自分の作ったものも、数十曲ほど遊ぶたびに内部の点検をしています。

この本をお手にとっていただいた皆さまは、ぜひコントローラを自分で作ってみてください。この本が皆さんの創作意欲を刺激するようなものになったら、私はとても嬉しいです。

◆ DIVA アーケードコントローラの創りかた

発行日 2019 年 3 月 18 日 改訂第 4 版
 2018 年 8 月 10 日 初版

発行者 ぼる

連絡先 メールアドレス | pol8139+nr@gmail.com
 ウェブページ | <http://pol.dip.jp>

印刷所 株式会社ポプルス

