Maker Faire 2019

# Network Stomper 開発 Project 基板設計記

株式会社 IIJ イノベーションインスティテュート

末永洋樹

## はじめに

Network Stomper 開発 Project では、通常足で切り替えるギターエフェクタを Wi-Fi 経由で切り替える装置を作っています。これにより、ギタリストの立ち位置の制限を緩和できたら楽しいのではないかというアイデアです。エフェクタを改造してしまうのも一つ

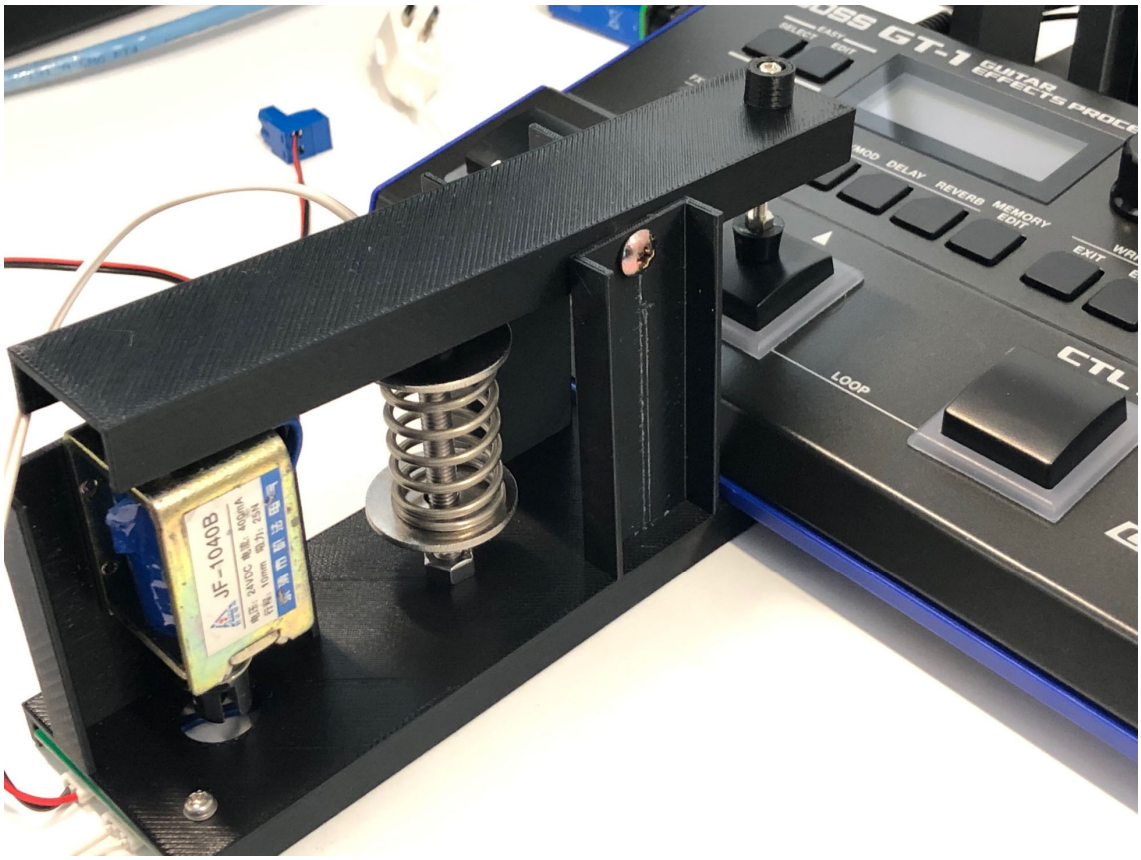
の方法ですが、この Project ではエフェクタを操作する装置を作ります。どんなエフェクタでも無改造で使えれば余計な手間もかからなくて便利でしょう。

要件としては、

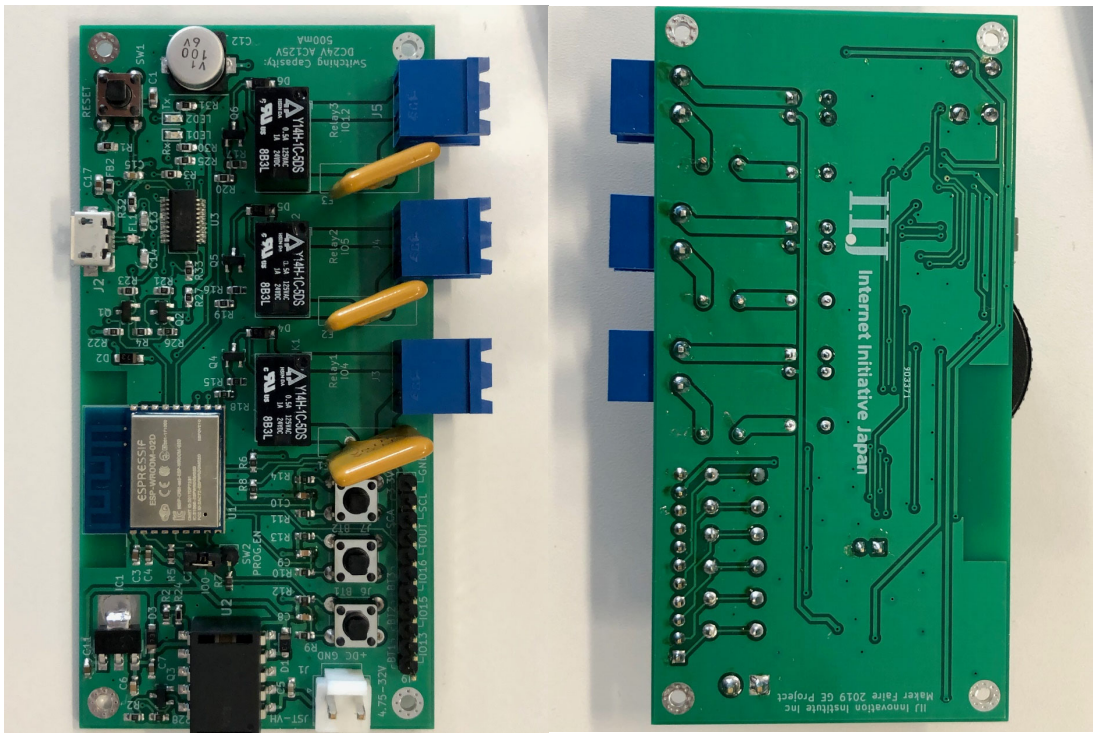
- 足で操作する装置を動かすだけの大きな力を発揮しなければならない
- 操作の遅延は可能な限り短くしなければならない
- Wi-Fi で動作するコントローラが演奏の邪魔になってはならない
- 従来通りの足による操作も受け付けなければならない
- 装置全体の起動時間は可能な限り短くしなければならない
- 秋葉原で売っている普通の部品で作れなければならない

といったところです。

結果として出来上がったのは以下のような駆動装置と、




以下のような 5cm x 10cm の制御ボードです。基板は基板試作サービスの FusionPCB を利用し、部品の実装は家内制手工業により行われています。




本書では制御装置の設計についてゆるく解説します。

## 設計方針

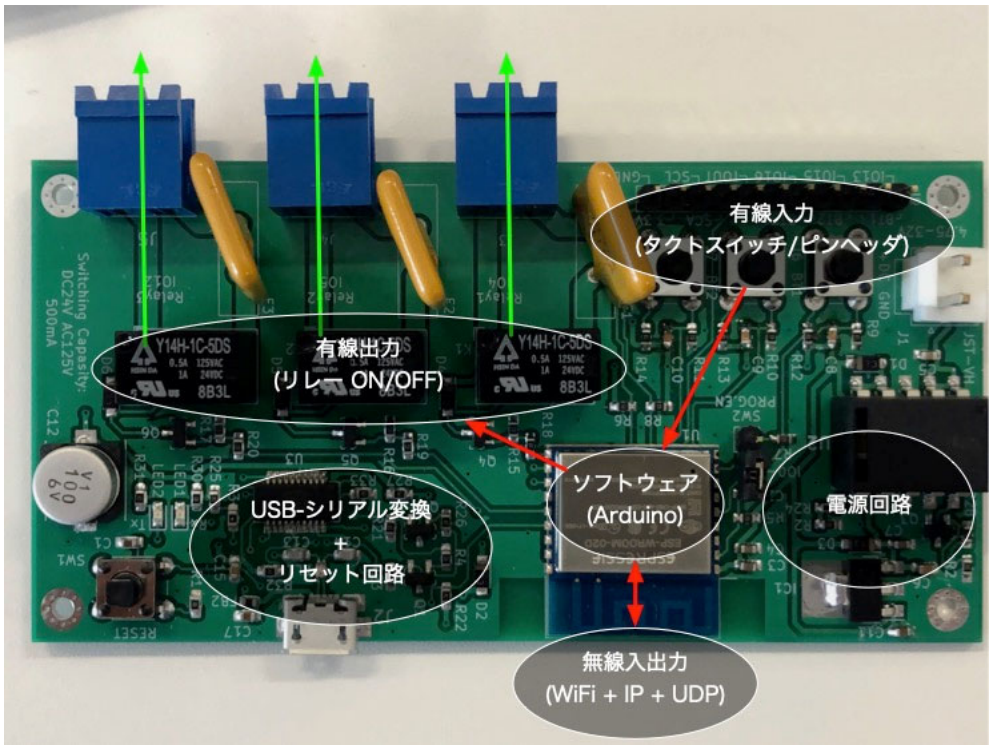
色々と検討した結果、装置の駆動にはソレノイドを利用し、制御機構は Arduino IDE で開発できるマイコンを採用することとしました。

ソレノイドは大きな力を発揮するには向かない素子ではあるのですが、モータをギアで減速して使うような方法と比較すると反応速度を上げやすく、サーボ制御がなくても位置が正確に決まるという利点があります。また、ソレノイドによる駆動機構は明和電機っぽくて音楽とは相性が良いような気がします。原理的には電磁石で鉄心を引っ張るという単純明解な素子となっています。

マイコンについては、起動時間の問題からラズパイ系よりも Arduino 系が良いという判断になりました。数年前までは Arduino と Wi-Fi は面倒な組み合わせの一つでしたが、現在は Espressif 社製の ESP-WROOM-02 が使えるのでとてもコンパクトでお安く実現できるようになっています。ESP-WROOM-02 はアンテナ内蔵で技適認証を取っているという点が大きな利点です。電波暗箱/暗室を利用しなくても堂々と開発できます。

制御ボードの詳細

制御ボードの役割は、Wi-Fi で受け取った指示にしたがってソレノイドの ON/OFF を実行することです。色々と選択肢はありますが、まずはシンプルに ESP-WROOM-02 からリレーを駆動する回路を作りました。リレー制御でインタフェースしておけば、今回の目的以外でも汎用の Wi-Fi スイッチとして活用できるであろうという目論見です。




上図のリレーの先にある青い端子台にソレノイドを接続します。スイッチは基板上にテスト用スイッチを置いているほか、ピンヘッダ経由で外付けスイッチを接続することも可能です。スイッチが押されたことを検出すると、Wi-Fi で他のボードに知らせるか、リレーを駆動するかします。この基板の回路図を以下に示します。

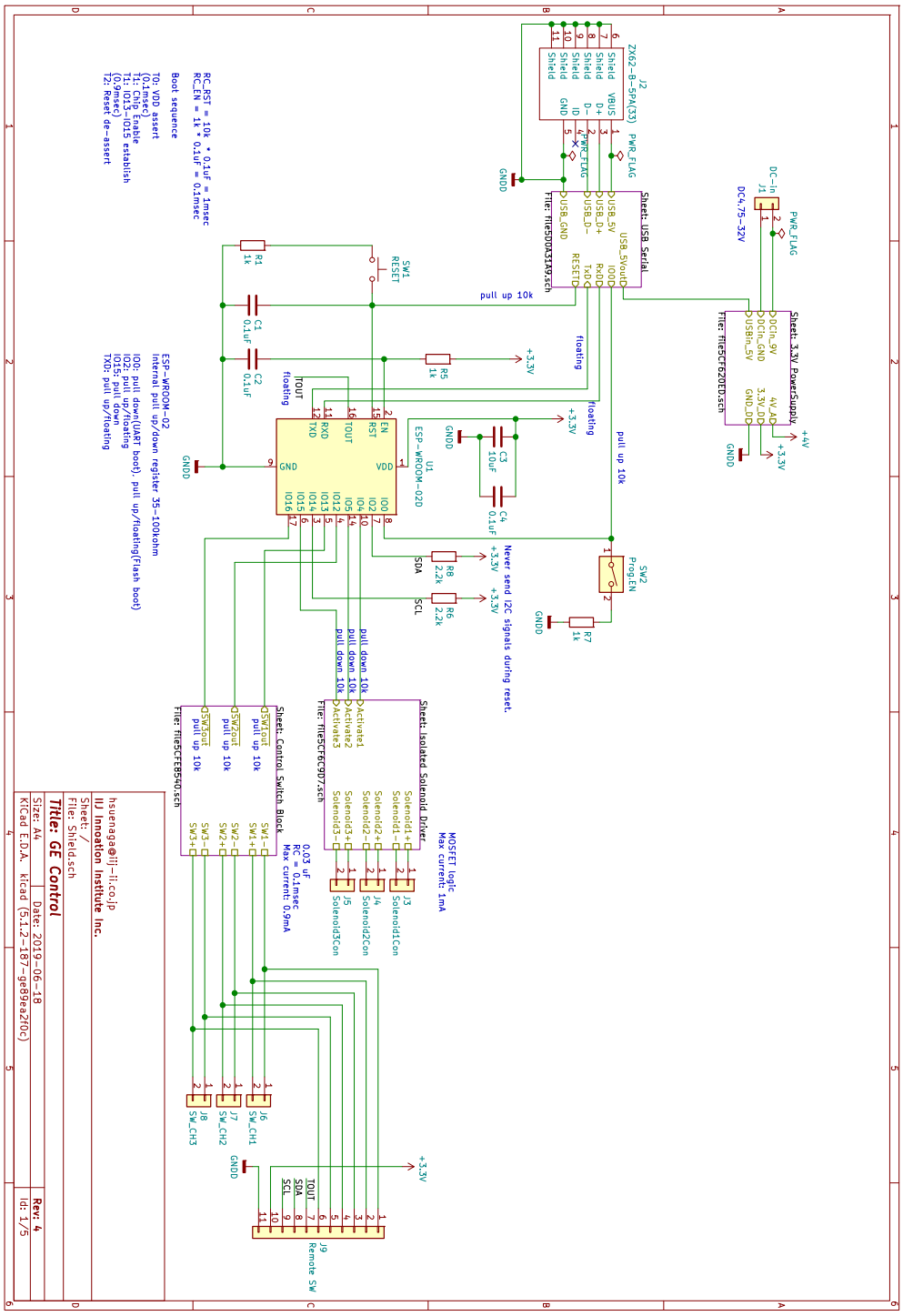



図 1 ESP-WROOM-2 本体周辺回路

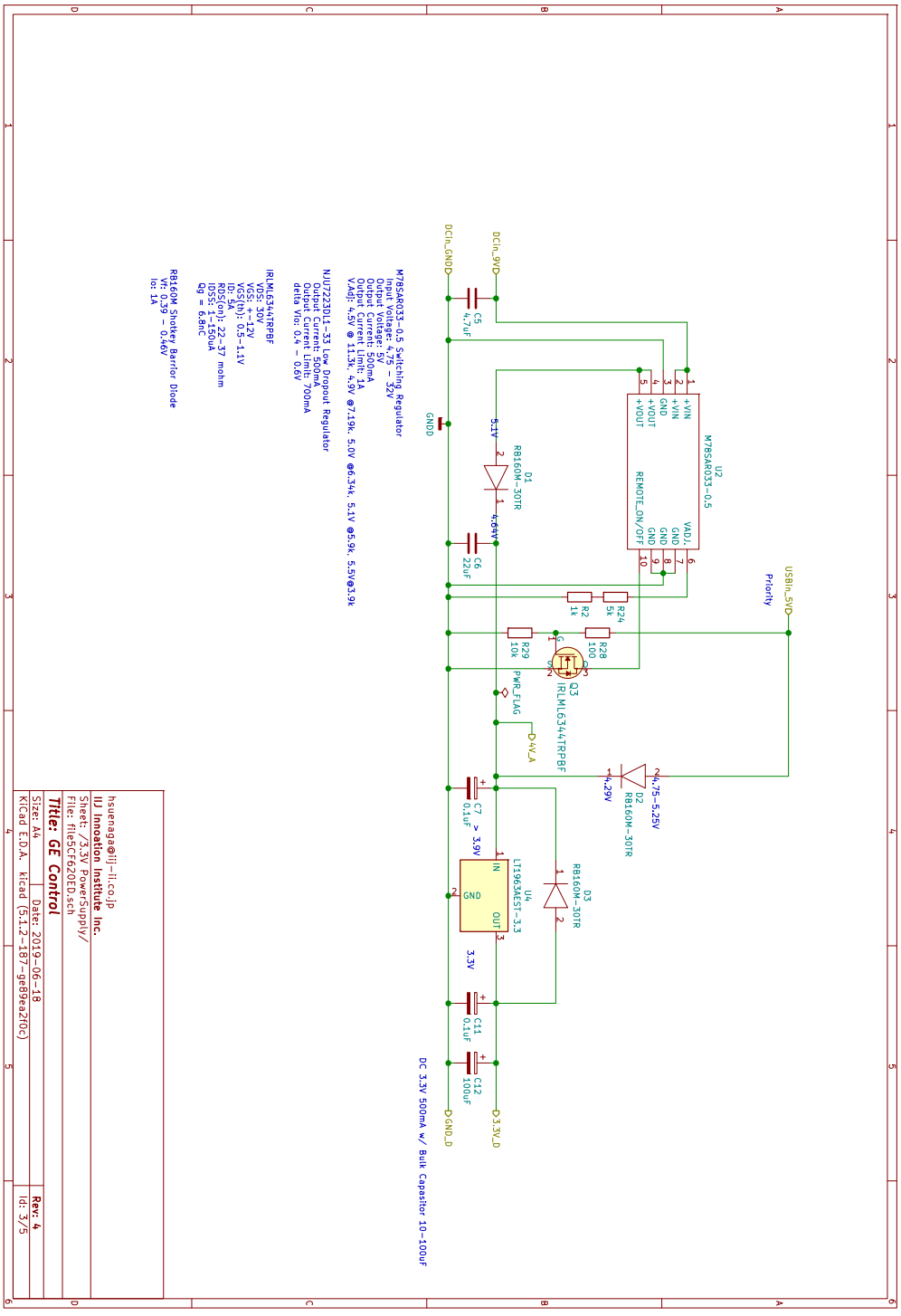

M78SAR033-0.5 Switching Regulator
Input Voltage: 4.75 – 32V
Output Voltage: 3.3V
Output Current: 500mA
Output Current Limit: 1A
V.Adj: 4.5V @ 11.3k, 4.9V @7.19k, 5.0V @6.34k, 5.1V @5.9k, 5.5V@3.9k
NJU7223D1-33 Low Dropout Regulator
Output Current: 500mA
Output Current Limit: 700mA
delta Vio: 0.4 – 0.6V
IRLML6344TRPBF
VDS: 30V
VGS: +-12V
VGS(th): 0.5–1.1V
ID: 5A
RDS(on): 22–37 mohm
IDSS: 1–150uA
Qg = 6.8nC
RB160M Shottkey Barrier Diode
Vf: 0.39 – 0.46V
Io: 1A
DC 3.3V 500mA w/ Buik Capasitor 10–100uF
hsuenaga@iij-ii.co.jp
IIJ Innoation Institute Inc.
Sheet: /3.3V PowerSupply/
File: fileSCF620ED.sch
Title: GE Control
Size: A4
Date: 2019-06-18
KiCad E.D.A. kicad (5.1.2-187-ge89ea2f0c)
Rev: 4
Id: 3/5

**図 2 電源回路**

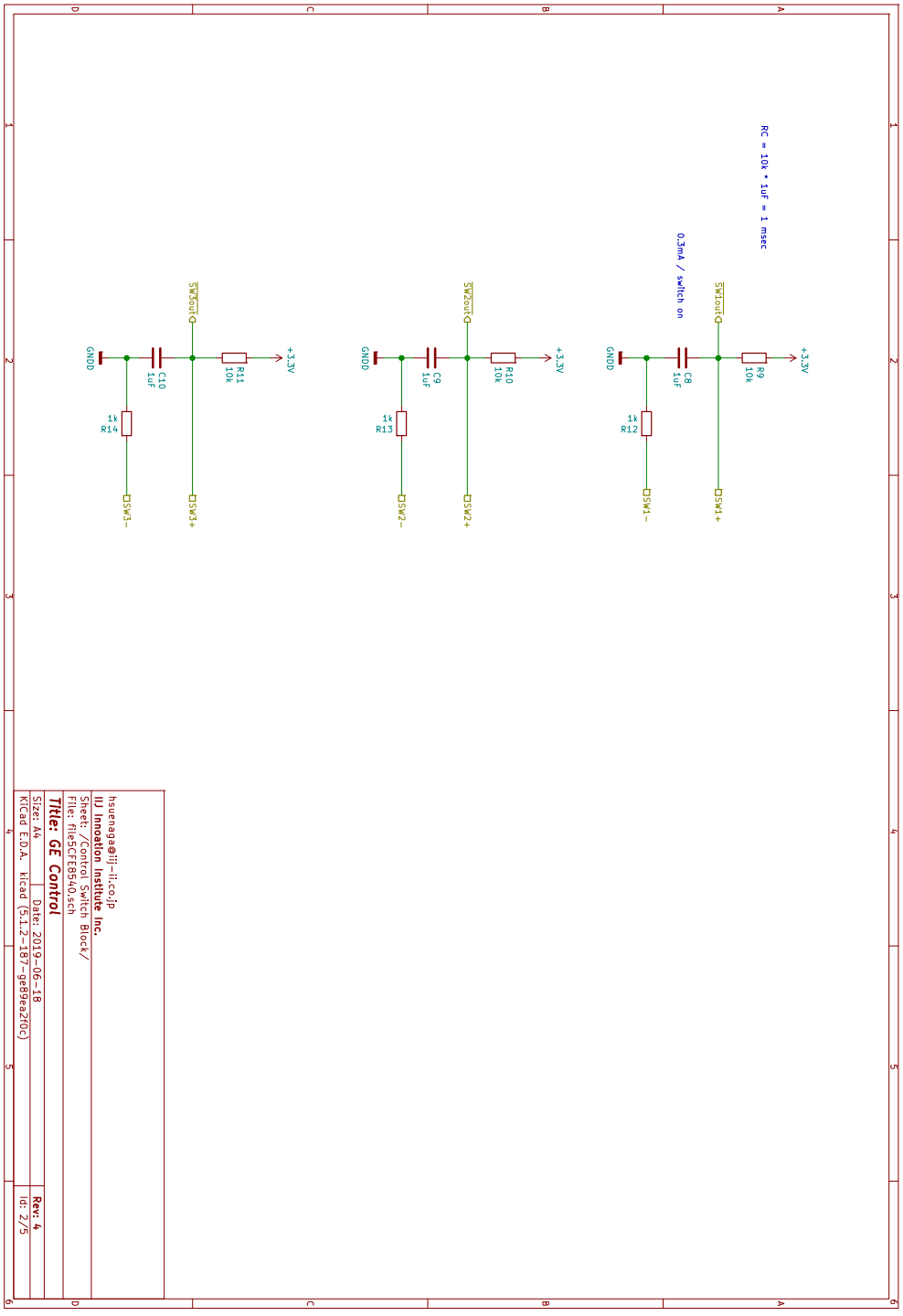

RC = 10k * 1uF = 1 msec
0.3mA / switch on
SW1out
SW2out
SW3out
+3.3V
GNDD
R9
10k
R10
10k
R11
10k
C8
1uF
C9
1uF
C10
1uF
R12
1k
R13
1k
R14
1k
SW1+
SW1-
SW2+
SW2-
SW3+
SW3-
hsuenaga@iij-ii.co.jp
IIJ Innoation Institute Inc.
Sheet: /Control Switch Block/
File: file5CFE8540.sch
Title: GE Control
Size: A4
Date: 2019-06-18
KiCad E.D.A. kicad (5.1.2-187-ge89ea2f0c)
Rev: 4
Id: 2/5

**図 3 スイッチ入力回路**

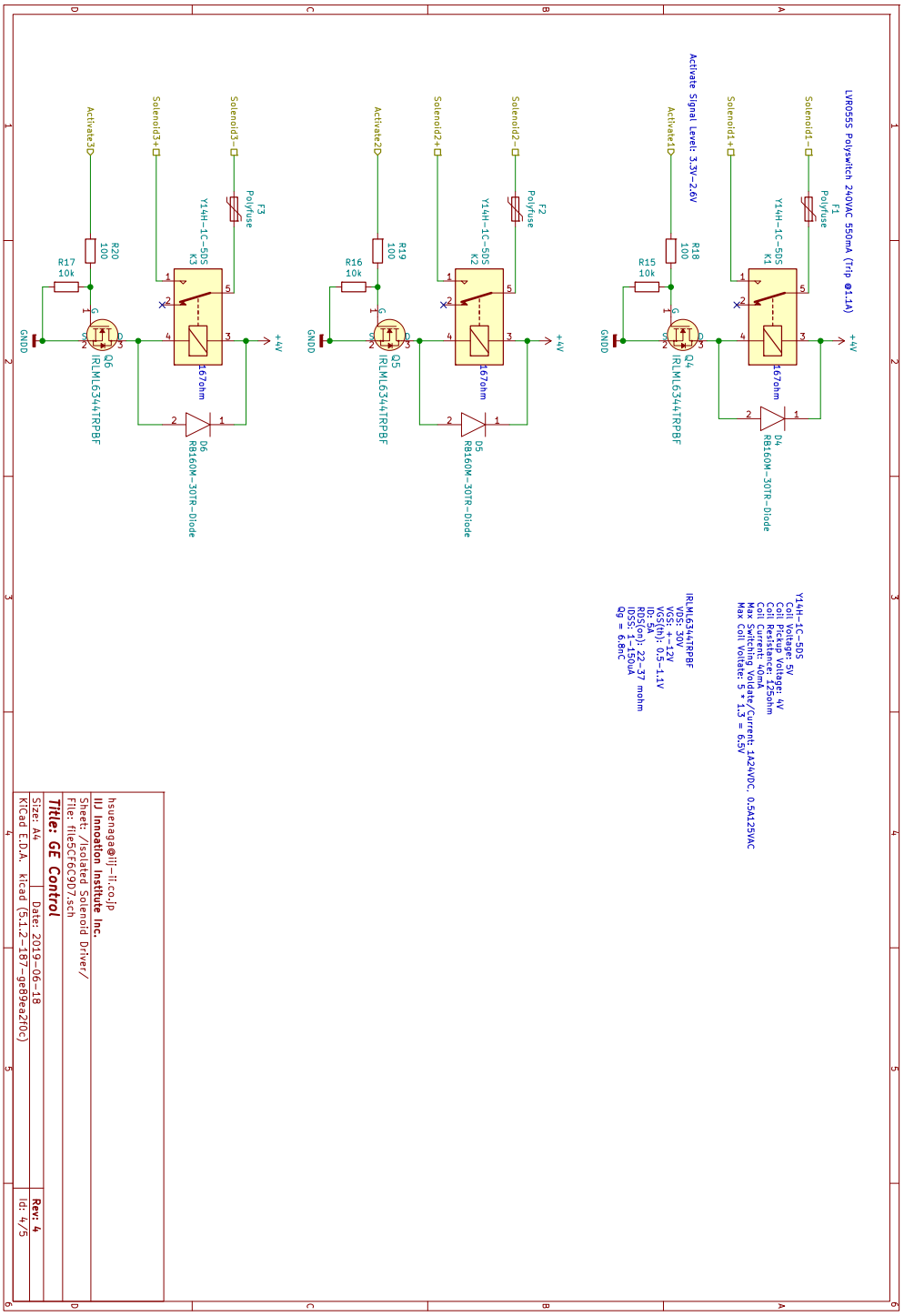

LVR055S Polyswitch 240VAC 550mA (Trip @1.1A)
Activate Signal Level: 3.3V~2.6V
Y14H-1C-5DS
Coil Voltage: 5V
Coil Pickup Voltage: 4V
Coil Resistance: 125ohm
Coil Current: 40mA
Max Switching Voltage/Current: 142VDC, 0.5A125VAC
Max Coil Voltage: 5*1.5 = 6.5V
IRLML6344TRPBF
VDS: 30V
VGS: +-12V
VGS(th): 0.5~1.1V
RDS(on): 22~37 mohm
IDSS: 1~150uA
Qg = 6.8nC
hsuenaga@iij-ii.co.jp
IIJ Innoation Institute Inc.
Sheet: /Isolated Solenoid Driver/
File: file5CF6C9D7.sch
Title: GE Control
Size: A4
Date: 2019-06-18
KiCad E.D.A. kicad (5.1.2-187-ge89ea2f0c)
Rev: 4
Id: 4/5

**図 4 リレー出力回路**

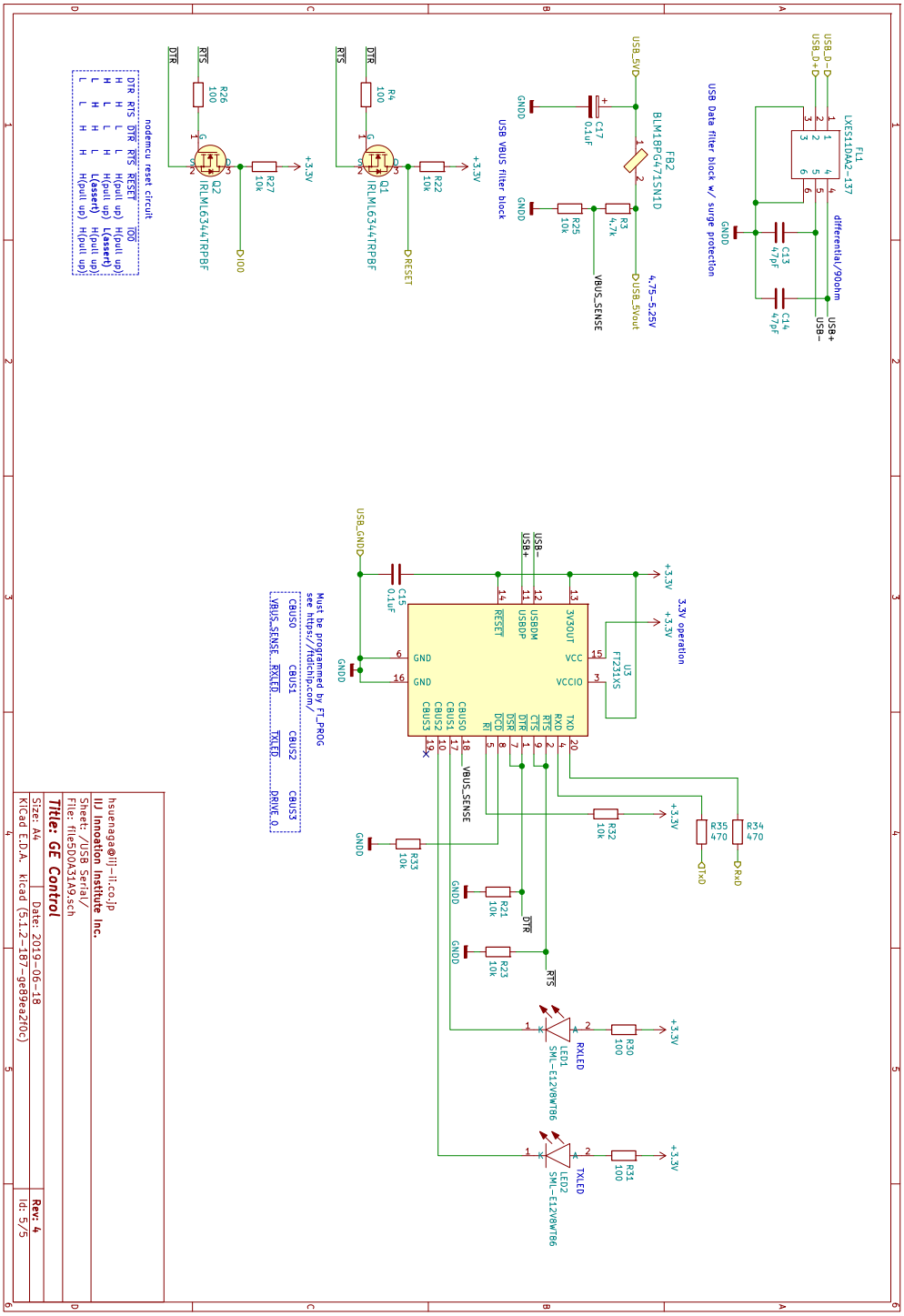



**図 5 USB シリアル回路**

大まかなブロック構成としては、ESP-WROOM-02 本体、電源回路、スイッチ入力回路、リレー出力回路、USB シリアル回路の 6 つです。

ESP-WROOM-02

## GPIO を割り当てる

今回はボタン入力が 3 つとリレー出力を 3 つの 3 チャンネル制御という仕様にしました。最終的な制御対象のマルチエフェクタが 3 ボタンだから、というのがもっとも大きな理由です。Wi-Fi 経由で操作するのでボタン入力はなくても良いのですが、デバッグのしやすさを考えて Wi-Fi なしでも操作できるようにしています。

ボタンとリレーは大きな部品なので基板上の占有面積が大きくなります。そうすると余白もでてくることになるので、ついでにピンヘッダをつけてなるべく多くの GPIO を外だしするようにしました。さらに USB シリアル変換チップも搭載して Arduino IDE でお手軽に開発できる環境を作っています。

ボタン入力とリレーについては GPIO の割り当てを決めておく必要があります。ESP-WROOM-02 は GPIO にあまり余裕がありません。また、配線に制限のある GPIO がいくつかあります。特に重要なのは、IO0、IO2、IO15 の 3 つです。これらは起動時に INPUT 端子として動作し、SoC である ESP-8266 の起動モードを決定します(ESP8266 Pin List, 2018, p. Strapping)。IO2 は pull up または未接続、IO15 は pull down、IO0 は必要に応じて pull up と pull down を選択できなければなりません。

また、汎用の通信端子として UART(普通のシリアルポート)と I2C は利用できるようにしておきたいところです。SPI も使えますが、ピンを多く消費するので今回は見送りました。ADC の入力として TOUT 端子というのもあるので一応、ピンヘッダには並べておくことにします。

という条件を考えて、ピン割り当ては以下としました。

| ピン番号 | 割り当て機能 |
|---|---|
| IO0 | ファームウェア更新(pull up、ピンヘッダあり) |
| IO1(TXD) | UART0-TXD |
| IO2 | I2C-SDA (pull up) |
| IO3(RXD) | UART0-RXD |
| IO4 | リレー出力 1(pull down) |
| IO5 | リレー出力 2(pull down) |

| | |
|---|---|
| IO12 | ボタン入力 2(pull up) |
| IO13 | ボタン入力 1(pull up) |
| IO14 | I2C-SCL (pull up) |
| IO15 | リレー出力 3(pull down) |
| IO16 | ボタン入力 3(pull up) |
| TOUT | ADC 入力(未接続、ピンヘッダあり) |

こんな感じで GPIO は売り切れです。IO12 と IO13 が逆転しているのはかつて IO15 をボタン入力 2 に割りあてて設計していたら、pull up できないことに気が付き急遽 IO12 と入れ替えたためです。さらに IO16 は入力割り込みが使えないということにも気が付きましたが、基板を製造したあとだったので、そのままにしてポーリングで監視しています。

## 内蔵 ROM のメッセージ

UART は別のピンを使って SoC の内蔵 ROM のメッセージを隠すという選択肢もあるのですが、今回はそのままにしました (ESP-WROOM-02 Datasheet v2.6, 2018, p. 6) (ESP8266EX Datasheet v6.0, 2018, p. 15)。このメッセージは少しやっかいで、ESP-8266 に供給されているクロック周波数から自動的に通信速度が決まってしまい、ユーザは速度を指定できません。40MHz 駆動の場合は 115200bps となり普通なのですが、ESP-WROOM-02 の 26MHz 駆動の場合は 74880bps という中途半端な数字になってしまいます (ESP8266EX Datasheet v6.0, 2018, p. 15)。74880bps はシリアル端末ソフトの速度指定には含まれないことが多いようですが、76800bps に設定するとデコードできることが多いようです(もともと RS-232C の非同期通信はクロック信号を持たないのでタイミングのずれには寛容です)。

ESP-WROOM-02 の場合、74880bps で使っていれば通信速度は終始一定となるのですが、UART の通信速度は安全性重視で 9600bps とするか、性能重視で 115200bps とするかの 2 択を軸に考えるのが普通の感覚です。自分の書いたプログラムでこのような普通の通信速度を採用しようと思うと、SoC の起動メッセージだけが速度が違うという状況になり、文字化けします。デバッグ専用ポートとして使っている分には構いませんが、汎用の通信ポートとしては難ありですね(ROM のメッセージが読めるのは開発初期のデバッグに役にたちますので無駄ではありません)。

## TOUT は使える端子なのか？

TOUT は特殊な端子で、おまけ感が強い仕様となっています。ESP-8266 のデフォルトの動作としては、ADC は無線 LAN のパワー・アンプ用の電源(VDD3P3)の電圧監視に使われています。これにより起動時に無線 LAN 周りのキャリブレートを自動実行してくれるという仕組みです。本来このために ADC を搭載しているのでしょう。しかし、TOUT に外付けの回路がつながっていると、ADC の出力が変わってしまい、このチューニングがちゃんと実行できません。そこで、TOUT を使う場合にはファームウェアのヘッダ部分を書き換えて VDD3P3 の電圧を固定値とするように SoC に教えてあげる必要があります（ESP8266EX Datasheet v6.0, 2018, pp. 16-17)。このヘッダの調整をせずに TOUT を利用すると無線 LAN の性能が落ちるという残念な結果になりそうです。調整をしてもキャリブレートが走らなくなると微妙に損した気分になります。多分ほとんど変わりませんけど。そもそも Arduino IDE では TOUT の扱いがどうなっているのか、ちゃんと調査していないこともありピンは付けてみたものの、使いどころがない端子となってしまいました。ADC_MODE(mode)というマクロを書いておくことで ADC で電源電圧を取得するのか、TOUT の電圧を取得するのか切り替えることはできるのですが、このマクロがファームウェアのヘッダを付け替えるような仕組みにつながっているかどうかが確認できません。特にリンカに指示を出すわけではないような？

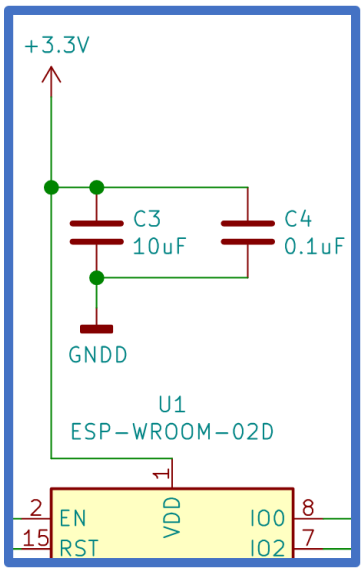
おそらく GitHub の ESP8266 core のチームも色々と悩んでいるのだろうとは思いますが、現状はあまり綺麗に使えるものではなく、Arduino 本来の analogRead()関数とのすり合わせもできていないようです。

## 欠番しているピンはなに？

IO6 から IO11 が欠番になっているのは、ESP-WROOM-02 のモジュール内部で SPI フラッシュと繋がっているためです（ESP-WROOM-02 Datasheet v2.6, 2018, p. 12)。制御しようと思えば出来てしまうかもしれませんが、SPI フラッシュへのアクアスができなくなり、まともに動作しないでしょう。他のボード用のスケッチを移植する場合は GPIO の初期化コードをきちんと確認したほうがよさそうです。

## 電源供給

下図が図 1 ESP-WROOM-2 本体周辺回路の関連部分の拡大です。

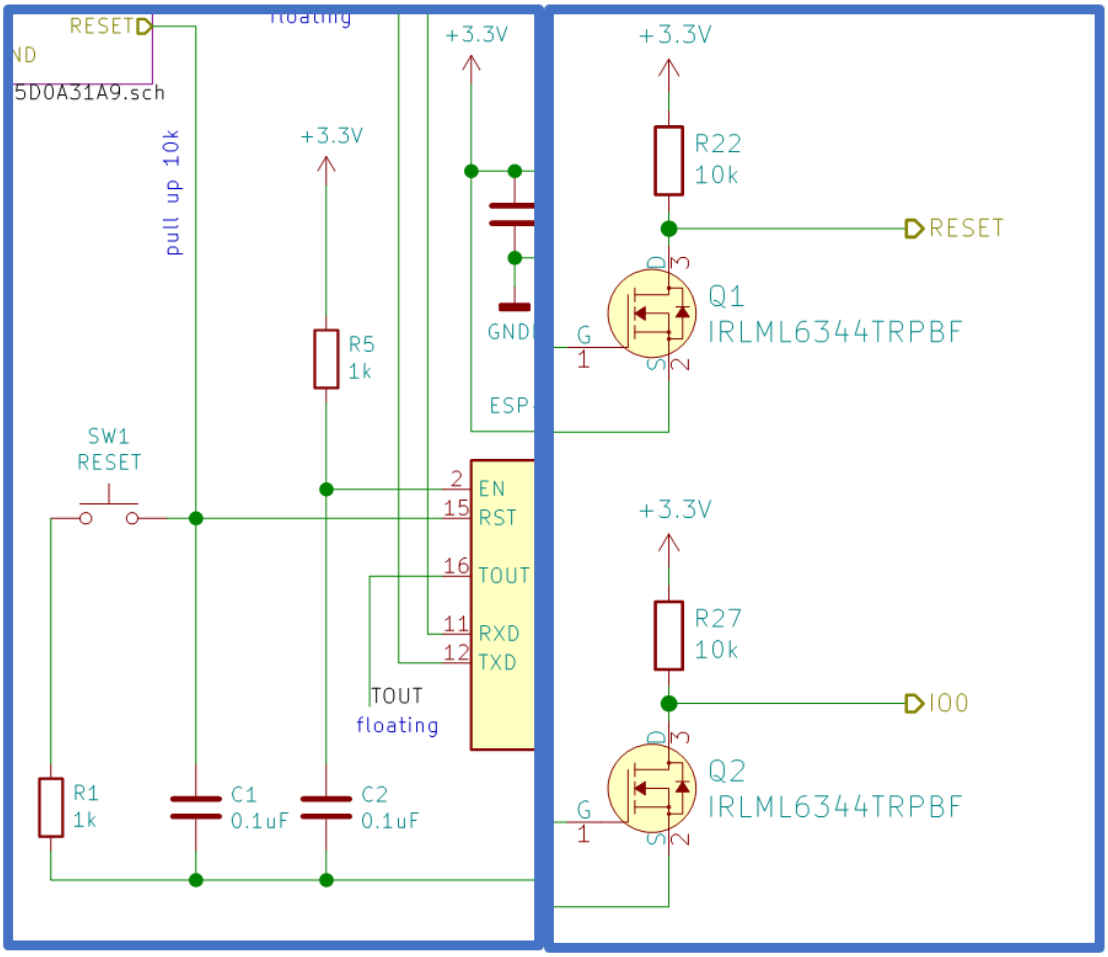



C3,C4 は電源ラインのノイズを捨てるためのバイパスコンデンサです。この組み合わせは ESP-WROOM-02 のデータシートのリファレンス回路そのものです（ESP-WROOM-02 Datasheet v2.6, 2018, p. 13)。C4 は明らかにノイズフィルタっぽいですが、C3 の 10uF はバイパスコンデンサというよりは、電源を安定させるバルクコンデンサといったほうが良いのかもしれません。設計が新しい電源回路を使っていて、負荷変動への応答性が良好な場合、10uF でも十分に安定性を確保できるでしょう。

世間的には 100uF 以上のバルクコンデンサを積んで昔ながらの電源 IC を利用している回路もよく見かけますが、データシートのリファレンス回路からは高速電源 IC を使ってくれ、というメッセージが感じられます。

### リセット回路

下図が図 1 ESP-WROOM-2 本体周辺回路と図 5 USB シリアル回路の関連部分の拡大です。




**EN 端子(Chip Enable)**

EN 端子は ESP8266 を有効にして動作を開始させるためのピンです。EN 端子が LOW になっていると ESP8266 は待機状態となり電力をほとんど消費しなくなります。他にも CPU がいるような場合には ESP8266 を必要なときだけ有効にして消費電力を抑えるという構成が可能です。今回は ESP8266 以外のコントローラは使っていませんので動作中は常に EN 端子は HIGH にしておくこととします。EN 端子を HIGH にするタイミングについては ESP8266 のデータシートに注意事項が記載されています(ESP8266 Hardware Degisn Guidelines v2.4, 2018, p. 8)。電源端子が 3.3V になってから EN 端子を HIGH にするという順番を守ること、という内容です。典型的な設計として、10kΩ/100nF で作った RC 遅延回路を使うべしということも書いてあります。ESP-WROOM-02 の回路図には RC 遅延回路は見当たりませんので、外付けで回路を組んでやる必要があるようです。CHIP Enable ピンはリセットピンの代用として使うこともあるので標準では遅延回路は入れていないということなのでしょう。

実はこのリセットシーケンスは更新されており、ESP8266 System Descriptionsv1.4という古いドキュメントでは、1kΩと 0.1uF という指定が書かれていました（ESP8266 System Description v1.4, 2016, p. 6)。手元にあったドキュメントを元に R5 に 1kΩ、C2 に 0.1uF を配置する設計にしてしまったのですが、これだと遅延が足りないようです。今のところ動作に問題はないのでそのままにしていますが、本来は抵抗の交換が必要です。回路を引くときは最新のドキュメントをチェックしましょう。

**RST 端子(Reset)**

こちらは SoC にリセットをかけるための端子です。LOW レベルに落とすとリセット状態になり、HIGH レベルにするとリセットが解除されます。リセットは EN 端子を使って実現することもでき、データシートには明示的に EN 端子をリセット端子として利用しても良いと書いてあります（ESP8266 Hardware Degisn Guidelines v2.4, 2018, p. 8)。EN 端子によるリセットのほうがハードウェアの深い部分までリセットをかけられそうに思えますが、詳細はわかりません。今回はハードウェア的におかしくなった場合は電源 OFF/ON で復帰すれば良いと考えてリセット動作は RST 端子を使うことにしました。

実はこちらも古いドキュメントでは EN 端子はリセットにも使えるとしか書かれておらず、このような設計としたのですが、現在は EN 端子の利用を推奨するという書き方になっています。RST 端子には低消費電力モードである Deep Sleep 状態から復帰するという重要な役割があります。実際には Wakeup 端子と言うべきなのでしょう。RST 端子をスイッチ経由で操作すると Deep Sleep 状態から復帰したのか、ユーザがリセットをかけたのか、分からなくなってしまうという問題が生じそうです。

RST 端子を使う場合、遅延回路があったほうが良いと書かれていますが、EN 端子用のような具体的な定数の指定はありません。RST 端子は未接続でも動作するように設計されていますから、比較的おおらかな入力端子として設計されているのでしょう。今回は R22 に 10kΩ、C1 に 0.1uF の遅延回路を作ってあります。起動時のリセット解除タイミングを遅延させるという目的と、基板上のリセットボタンを押した時にチャタリングでノイジーな信号が印加されるのを抑止するという目的の 2 点が狙いです。リセットボタンを押すと、C1 にチャージされていた電荷が R1(1kΩ)を経由して放電され緩やかに電圧が降下してリセットがかかります。ボタンを離すと R22(10kΩ)を経由して C1 が充電されて緩やかに電圧が上昇してリセットが解除されます。解除のほうが 10 倍程度時間がかかる計算ですが、数ミリ秒の範囲内なので良しとします。

R1 を省略しても動作はすると思われます。しかし、この場合スイッチを押した瞬間に C1 が短絡されることとなり、理論上は無限に大きな電流が流れます。実際には抵

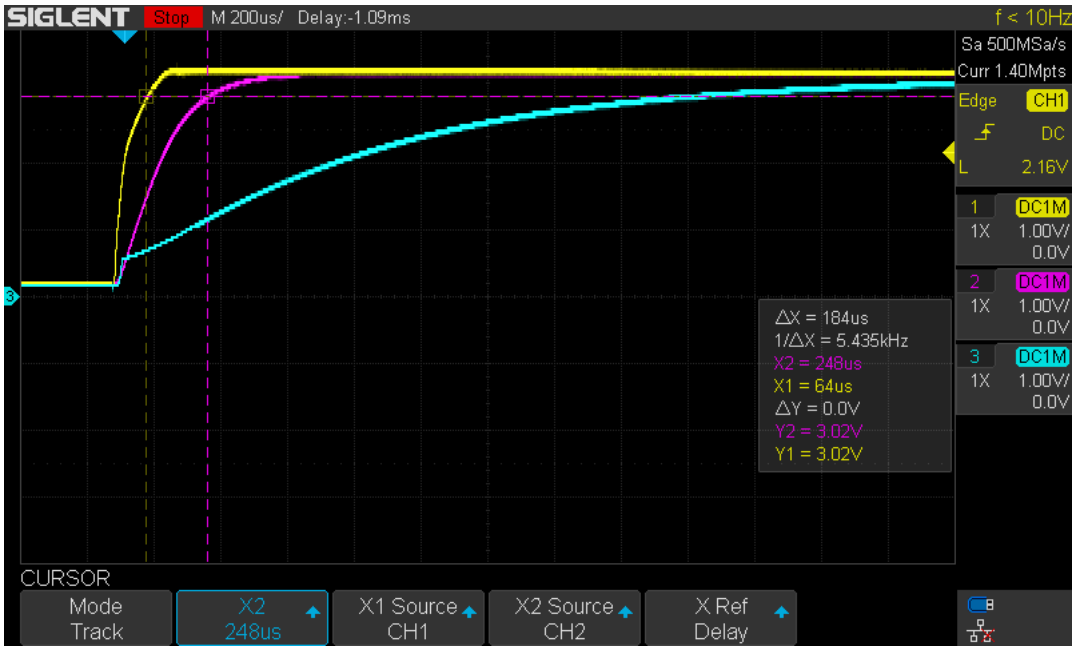
抗が 0 ということはありませんので無限とはいきませんが、スパイク状のノイズを撒き散らす原因になりかねないので R1 で電流を制限しています。扱っているのがリセット信号ですから、避けられるノイズはできるだけ避けておきたいところです。

リセットスイッチを押してコンデンサの電荷がなくなると、R22 と R1 が直接に接続された状態となります。このとき、分圧比は 10:1 になりますので、R22 で 3V の電圧降下が発生して R1 では 0.3V の電圧降下が発生する計算となります。一方 ESP8266 の HIGH レベルは  2.5V 以上、LOW レベルは 0.8V 以下という規定です。R1 にかかる電圧の 0.3V が RST 端子への入力となれば確実にリセットがかかります。

RST 端子は USB シリアル回路へもつながっています。これは Arduino IDE でプログラムを送り込む際に自動でリセットをかけるための仕掛けです。この回路は USB シリアル回路の解説で記載します。

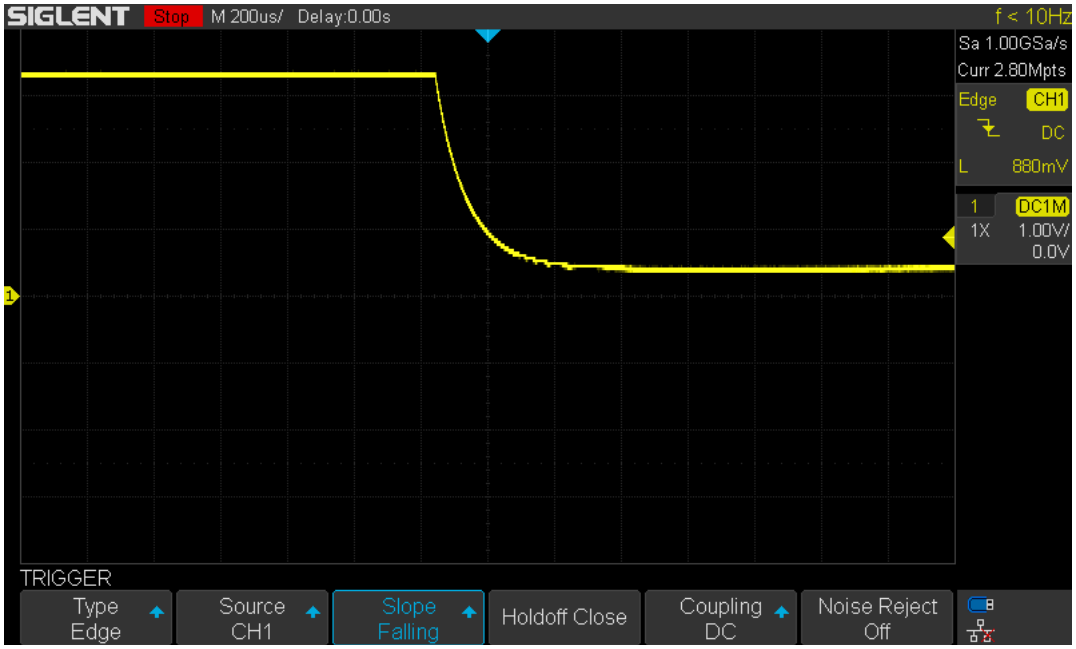
**リセット動作の動作**

この回路の電源投入直後の波形は以下のようになります。縦軸は 1 目盛 1V、横軸は 1 目盛 200 マイクロ秒です。




黄色が 3.3V の電源電圧、紫色が EN 端子にかかる電圧、水色が RST 端子にかかる電圧の時間的な変化です。3.3V 電源が立ち上がってから、180 マイクロ秒くらいあとで EN 端子が HIGH になり、さらに 1.5 ミリ秒程度たってから RST 端子が HIGH になります。ここまでくるとソフトウェアが動作を開始します。よくあるコンデンサの充電曲線ですが、ちょっとしたタイミング制御には十分ですね。

起動後に手動でリセットボタンを押した時の RST ピンの電圧を下図に示します。色が変わってしまいわかり難くなってしまいましたが、先ほどの波形では水色に相当します。

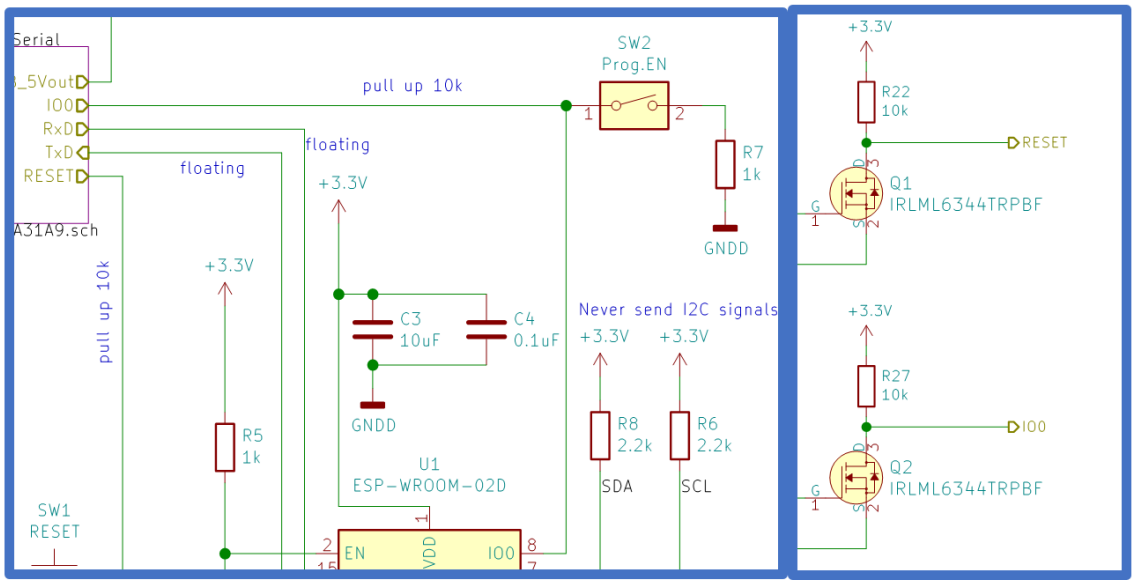



コンデンサが緩やかに放電されている様子が分かります。横軸は電源投入のグラフと同じですので、リセットをかける動作が解除する動作よりも早いことも確認できます。

スイッチのチャタリングも目立ちませんし、まぁまぁ良いのではないかと考えていました。しかし、このリセット回路には不具合がありまして、不具合については USB シリアル回路で記載します。

## ブートセレクタ

以下が図 1 ESP-WROOM-2 本体周辺回路と図 5 USB シリアル回路からの関連部分の拡大です。




ESP-WROOM-02 はリセット解除時に IO0 が LOW レベルになっていると UART 起動モードとなり、シリアル通信でファームウェアを読み込んで実行します（ESP8266 Pin List, 2018, p. Strapping)。ESP-WROOM-02 の開発環境では、この仕組みを使って SPI フラッシュへのプログラムの書き込みを実行します。このプログラム書き込みモードと実行モードの切り替えをするスイッチが SW2 となります。

SW2 が開いている場合、IO0 は R27 の 10kΩを経由して電源に接続されています。IO0 が入力端子として設定されている場合、電流はほとんど流れません(50nA 以下)から（ESP8266EX Datasheet v4.4, 2015, p. 17)、抵抗での電圧降下は生じず IO0 への入力はほぼ電源電圧に一致します。IO0 への入力は HIGH となり SPI フラッシュからの起動が選択されます。

pull up 抵抗の値にはあまり根拠はありません。経験則として 10kΩから 100kΩくらいにしておくと大抵はうまく動くことが知られています。抵抗値が小さいと、グランドと接続された場合により多くの電流が流れてしまうため消費電力が増えたり、GPIO に過電流が流れてしまったりします。IO0 は普段は入力端子ですので電流は流れませんが、ソフトウェアから pinMode()で OUTPUT を指定して digitalWrite()で LOW を出力した場合、このケースに当たります。抵抗値が大きすぎるとノイズにより微小な電流が流れた場合に大きな電圧降下が生じて電圧が不安定になることがあります。Espressif 社のドキュメントでは 1MΩを標準として、場合によっては 100kΩという雰囲気で、このくらいであれば問題ないのだと思いますが、1MΩまで大きくするとオシロスコープのプローブの内部抵抗と同じ領域になってしまい測定が不自由になることがあります。

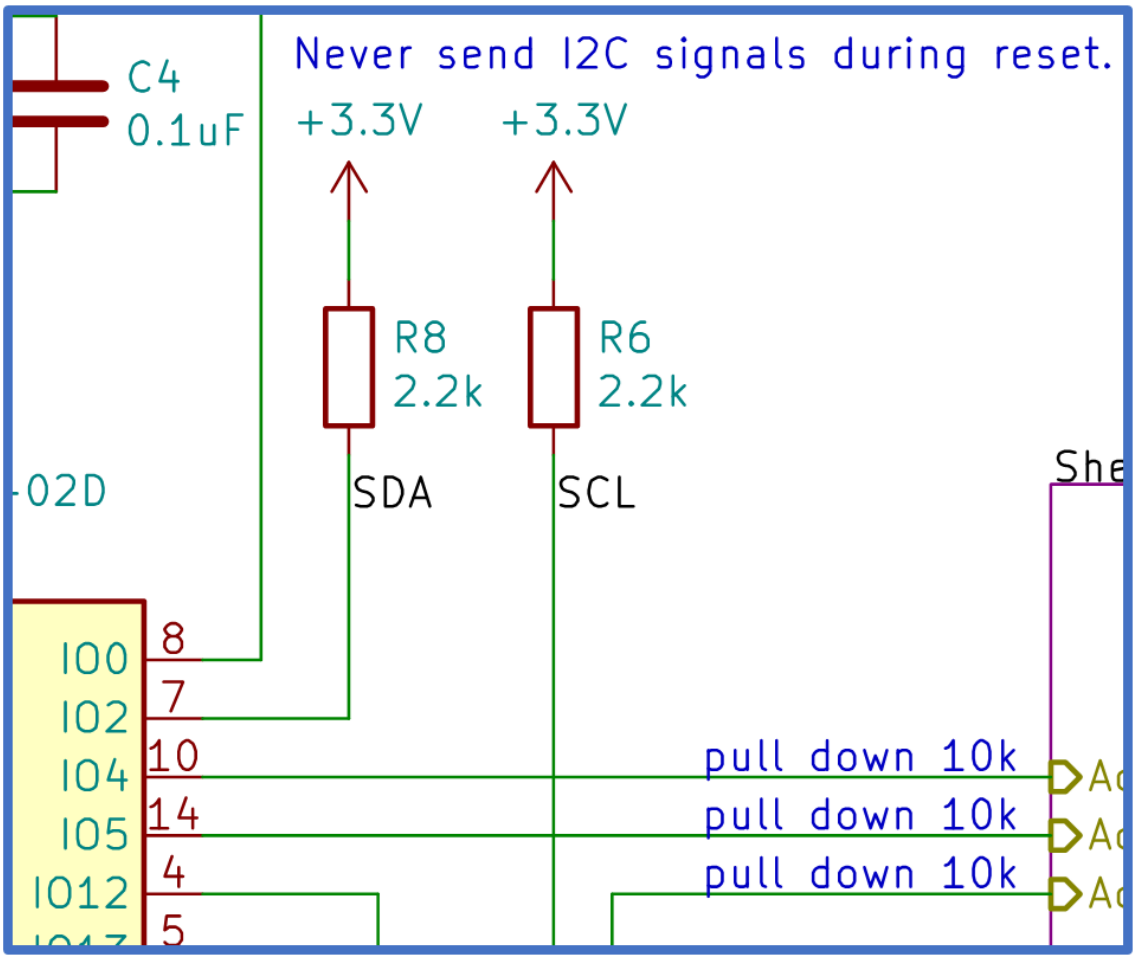
SW2 を閉じている場合、R17 の 10kΩと R7 の 1kΩが直列接続されます。IO0 はこの状態での R7 の電圧を読み取ることになります。直列接続では抵抗値の比で電源電圧が分圧され、電源電圧が 3.3V の場合ですと R17 には 3V がかかり、R7 には 0.3V がかかります。ESP8266 は電源電圧の 1/4、すなわち 0.8V 以下を LOW レベルとみなしますので IO0 への入力は LOW となりプログラム書き込みモードでの起動が選択されます。

IO0 への入力を LOW にする目的であれば、R7 を不要としてグランドに直接接続してしまう方法も考えられます。ブレットボードでの実験などであれば、大抵は直接接続にするでしょう。この場合の問題点は IO0 が GPIO であり OUTPUT に設定される可能性があるという点です。この設定をされてしまうと、R7 を付けておかないと IO0 からグランドに対して無制限に電流が流れてしまいます。ESP8266 の GPIO ポートは最大で 12mA までしか電流を供給できませんので（ESP8266EX Datasheet v6.0, 2018, p. 18)、運が良くて誤動作、運が悪ければ故障につながります。当然、そんなプログラムを意図して書くことはありませんが、他のボード用に書いたスケッチを移植する過程ではありがちなバグでもあります。Arduino ではスケッチの再利用性の高さも魅力の一つですので、ハードウェア側でなるべく安全側に倒す設計にしています。最近は例外が多いのですが、ソフトウェアでハードウェアが壊れることはない、という計算機の原則はやはり大切だと思います。

IO0 も RST と同様に USB シリアル回路にも接続されています。これは Arduino IDE でのプログラミングのための仕組みで、RST と一緒に USB シリアル回路と一緒に解説します。

## I2C

下図が図 1 ESP-WROOM-2 本体周辺回路からの関連部分の拡大です。




今回は利用しませんでしたが、基板上の土地が余っていたので I2C の信号線を引き出しています。I2C はセンシングや表示に使えますので、使えるようにしておくとなにかと便利なインタフェースです。

R6,R8 は I2C の通信路を動かすための pull up 抵抗です。I2C は仕様上マスタ側もスレーブ側も電圧源を持たないことになっていますので、このような形で外部から電圧をかける必要があります。本当は、通信は相手あっての話ですので、実際に使う段階で通信相手の事情も考慮して pull up 抵抗をどこに追加するのか設計するのが一番だと思います。しかし、今回はお手軽さを優先してあらかじめ pull up してしまいました。

マイコンの内蔵 pull up の機能を使えるのであれば、外付け抵抗はなくても動きますし、必要に応じて切り替えられるので便利です。が、ESP-8266 の場合には残念ながら内蔵 pull up の抵抗値が 30kΩから 100kΩという範囲で（Pull up resistors, 2015)、これは I2C の駆動には大きく、配線の制約が強くなりすぎます。

I2C の駆動電流は最大で 3mA と決まっています。これを超える電流は I2C デバイスで流しきれず電圧をしっかり下げることができなくなります。I2C デバイスの入力トランジスタの発熱にもつながるでしょうから熱的にもよろしくありません。この観点から、3.3V 電源であれば 1.1kΩ以上の抵抗を付けて電流を制限する必要があることがわかります。

I2C では配線の浮遊容量も決まっており、400pF が最大値です（I2C バス仕様書 v2.1, 2000, p. 6)。一方、ESP8266 の I2C のクロック速度は 100khz が性能限界とされています（ESP8266EX Datasheet v6.0, 2018, p. 14)。そして 100kbps(標準モード)の I2C 通信のパルスの立ち上がり時間は 1000nsec=1usec と規定されています（I2C バス仕様書 v2.1, 2000, p. 30)。以上から、400pF のコンデンサを 1usec 以内に充電できれば合格であろうと分かります。単純に RC 時定数 τで信号が立ち上がると仮定すると、τ=RC ですから、R= τ/C より R=1 [usec]/400[pF]=2.5[kΩ]あたりが上限となりそうです。これより大きな抵抗をつけると電圧が上がりきりません。1 τだと電圧は 63.7%までの上昇に止まり、I2C の仕様である 70%には届きませんので実際にはもう少し低い抵抗値が必要でしょう。2kΩくらいの抵抗を付けておけば、規格通りのバスといえるでしょうか。この時点で、ESP8266 の内蔵 pull up では明らかに抵抗が大きすぎ、配線の特性に敏感になりすぎるであろうとわかります。

試しに理論値を計算してみると、1usec 経過時点で 400pF のコンデンサを 70%充電できる抵抗値は 2.08kΩ以下となりました。とはいえ、ぴったりの抵抗はなかなかありませんし、高精度抵抗を直列にして頑張るところでもありません。1.1kΩから 2kΩ付近で比較的入手しやすそうな 1.2kΩ、1.5kΩ、2.2kΩあたりが候補かと思われます。1.2kΩだと電流が過大で規格に適合しません。大抵のデバイスには余裕がありますが、動いたとしても消費電力が多めになります。電流が多ければ接触抵抗などによる電圧の変動も大きくなります。2.2kΩだと信号の立ち上がりが若干規格値に及びませんが、電流超過よりは良いでしょうか。最終候補としては、規格重視で 1.5kΩか、規格よりも消費電力を重視して 2.2kΩか、というところで今回は 2.2kΩを選びました。

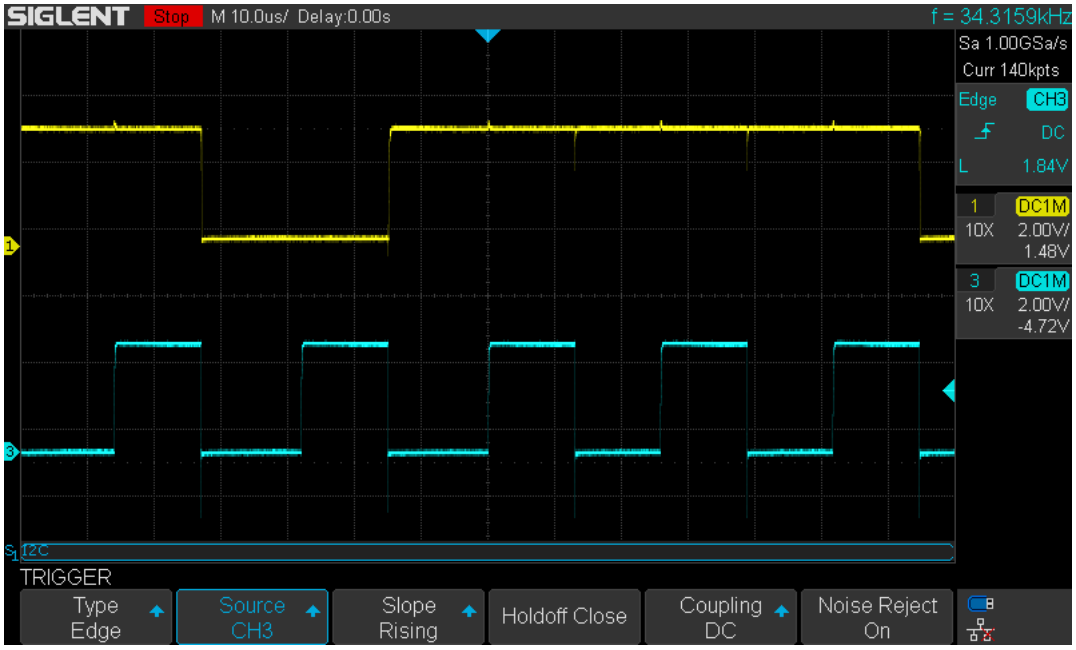
ESP-WROOM-02 で I2C を使う場合、若干の制限が生じます。データシートによると、I2C の SDA 信号は標準では IO2 を使うことになっています（ESP8266EX Datasheet v6.0, 2018, p. 6)。この IO2 はブートモードの選択にも使われるピンで、起動時には HIGH になっている必要があります。このピンを通信に使うというと厄介な問題が生じます。ESP-WROOM-02 に再起動がかかった時に、I2C で接続されたデバイスが SDA を LOW にしてしまうと起動に失敗するのです。これを防ぐためには FET を

追加するなどして、ESP-WROOM-02 のリセット動作時には I2C バスを切り離しておく真面目なリセット回路が必要となります。今回はこの回路は組んでいませんので、I2C を使っていると再起動に失敗する可能性があります。特に本機が I2C のスレーブになっている場合はいつマスタが話しかけてくるのか分かりませんので危険です。本機がマスタの場合、要求を出さなければスレーブは何もしないはずなので危険性は低いのではないかと期待できます。

細かいですが、本機がマスタであったとしても、要求を出した時点でウォッチドッグリセットによる再起動がかかったり、ユーザがリセットボタンを押したりするなどのコーナーケースは残ります。ウォッチドッグリセットがかかると起動に失敗して動作が完全に停止する、という話になったのでは本末転倒な感じです。そうすると、真面目なリセット回路を設計するとしても、ウォッチドッグリセットを外部から知る手段はあるのか？など、色々考えるべきことが見つかります。こんなコーナーケースなんて考慮する必要があるのか？と思ってしまうところかもしれません。しかし、今までの経験からすると顧客はピンポイントでコーナーケースを突いてくる存在でして、どうしても考えずにはいられません。本機は売り物ではないので、なんとなく考えるだけですが。

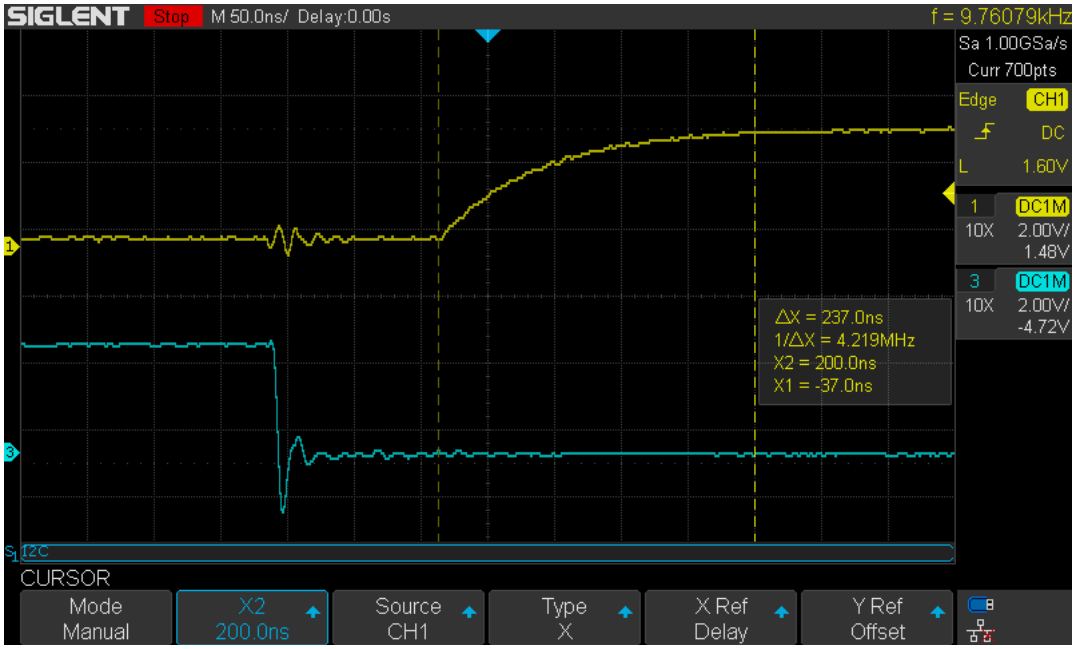
I2C インタフェースをちゃんと設計するのは案外手間がかかりますね。

I2C の波形

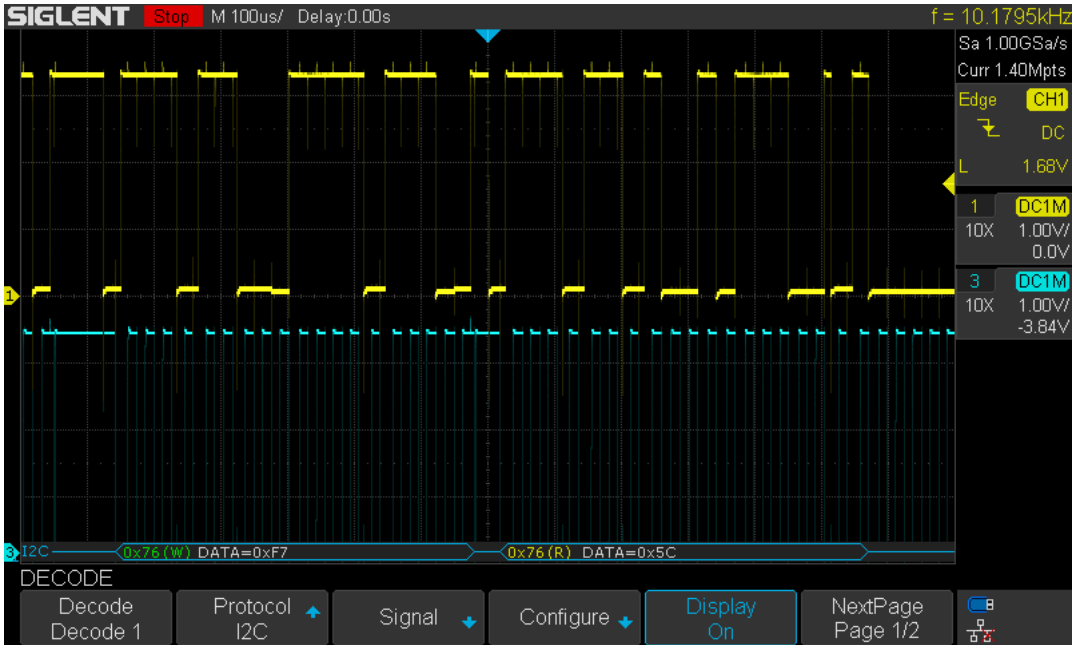



I2C インタフェースに温度・湿度・気圧センサの定番である Bosch BME280 をつないだときの波形を上図に示します。黄色のチャンネル 1 が SDA(データ)、水色のチャ

ンネル 3 が SCL(クロック)です。キャプチャ画像だとみづらくなってしまいますが、若干クロストークが見えたり、波形の立ち上がりと立ち下がりに鋭いオーバーシュートが見えたりしています。とはいえ、全体としてはそこそこ綺麗な矩形波になっています。




信号の立ち上がり、立ち下がり部分の拡大を上図に示します。クロックには割と大きなリンギングが見えますね。これは配線がコイルとして働いたものと考えることも、バスの終端で信号が反射したものと考えることもできます(同じ物理現象ですが、計算手法が 2 つあります)。真面目にインピーダンス計算をすれば軽減できるはずです。今回の配線はブレッドボードでよく使う 2.57mm ピンヘッダに刺すケーブル(約 15cm)を刺しただけですので、品質は推して知るべしという結果になりました。SDA 信号の立ち上がり時間は ΔX の値で、237ns となっています。目標は 1000ns ですので、十分に余裕がある数値です。筐体内部で配線するのであれば安心して使えるでしょう。同じ I2C でも高速モードを使うと厳しいかもしれないですね。




通信の様子がある程度わかるスケールでみると上図のようになります。縦軸が一目盛り 2V から 1V に変更されているので今までとは振幅が異なります(こうしないとオシロスコープがデコードしてくれない・・・)。こうしてみると細かいヒゲが気になってきますが、全体として均質ではあるのでバースト転送が発生しても頑張れるのではないかと思います。

## 電源回路

### 全体像

今回の回路で電源を必要とするのは、ESP-WROOM-02、USB シリアル変換チップ FT231XS、リレーの 3 種類の負荷です。ソレノイドもありますが、これは消費電力が大きいので制御基板からの電源供給にはちょっと無理があるでしょう。ESP-WROOM-02 は 3.3V 単電源で動作します。FT231XS は 5V 単電源または 3.3V 単電源で動作します。リレーは部品の選定次第で電源電圧を選べます。

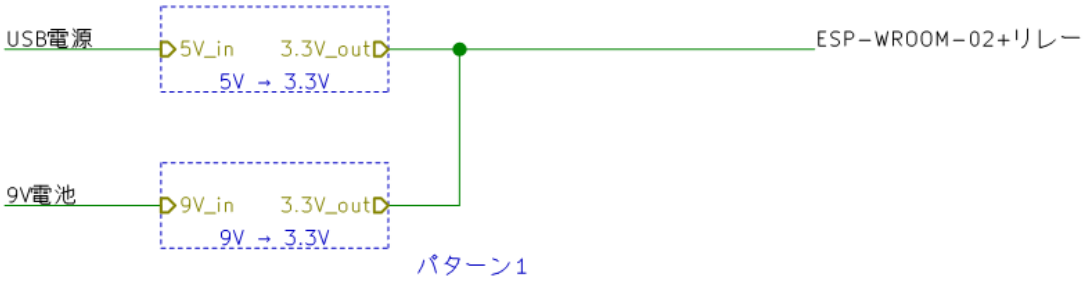
一方で外部電源として利用できるのは USB 端子から供給される 5V と、006P 電池から供給される 9V です。ESP-WROOM-02 は 3.3V が必要ですので、3.3V 電源が足りません。これは外部電源から電圧変換して生成するのが順当でしょう。

結果として、本機は 9V、5V、3.3V の 3 種類の電源をもつことになります。このうち、9V と 5V はいつでも使えるとは限りません。電池が接続されていない場合は 9V 電源

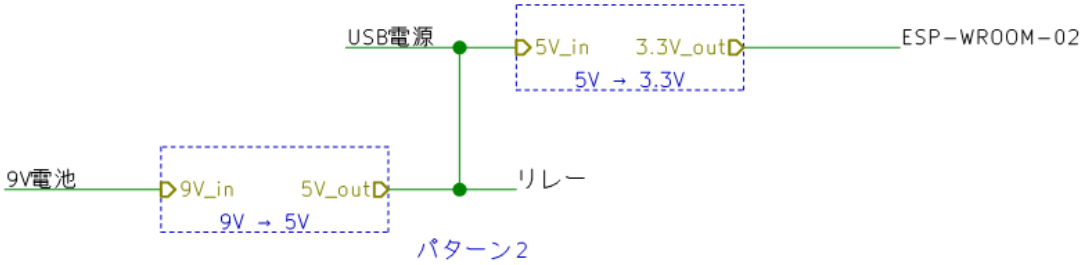
が使えませんし、USB が接続されていない場合は 5V 電源が使えません。この状態から 3.3V 電源を確保しようと思うと、

1. 9V から 3.3V の変換回路と、5V から 3.3V の変換回路を用意する
2. 9V から 5V の変換回路と、5V から 3.3V の変換回路を用意する
3. 9V から 3.3V の変換回路と、9V から 5V の変換回路と、5V から 3.3V の変換回路を用意する

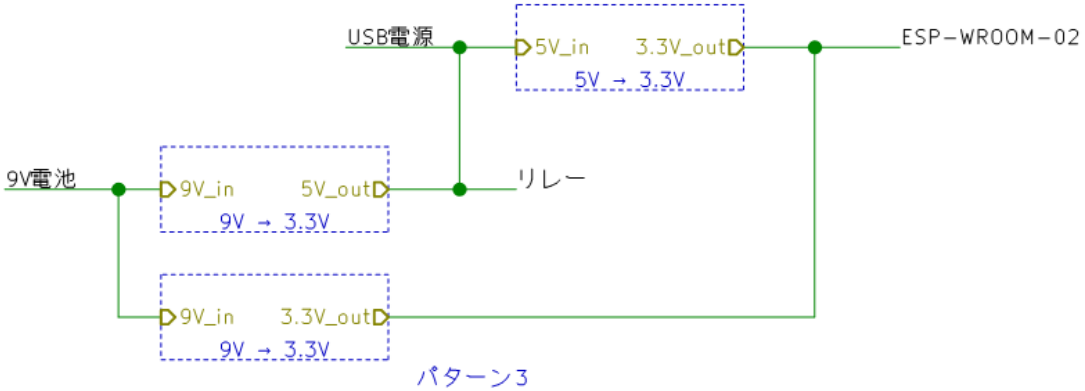
という 3 つの選択肢があります。



パターン1



パターン2



パターン3

1 は 3.3V を得るための電圧変換が常に 1 回で済むため効率が良いという利点がありますが、電池駆動時は 5V 電源が利用できないという欠点があります。この場合、全ての回路を 3.3V 駆動で設計する必要があります。

2 は常に 5V 電源を利用できるという利点がありますが、電池駆動の場合には電圧変換が 9V→5V→3.3V と 2 段階になるため効率が悪化するという欠点があります。この場合、一部の回路を 5V で設計できます。

3 は電池駆動の効率を改善しつつ 5V 電源が常に利用できるという利点がありますが、部品点数が増え回路規模が大きくなる欠点があります。これは選択肢 2 の効率改善案と言えます。

できれば 3 といいたいところですが、電源回路は部品が大きめで基板上の面積を広くとりがちです。小型の部品もカタログには存在しますが、秋葉原で売っているとは限りません。装置はギタープレーヤーに馴染みのあるコンパクトエフェクタのサイズに収めたく、5cm x 10cm 程度の基板サイズには収めたいところです。これは、ちょっと無理があるのではないかと判断しました。

1 はシンプルですが、9V からマイコンを安定して動かせるような 3.3V 電源を作り出すのは簡単ではありません。リニアレギュレータを利用すればノイズの少ない高品質な電源を作れるのですが、電位差が大きいと効率が極端に悪化します。9V から 3.3V をリニアレギュレータだけで生成した場合、効率はたったの 37%です。電力の 6 割以上が電源回路の排熱となってしまうのは流石に受け入れられません。9V から 3.3V をスイッチングレギュレータで生成するのであれば効率は 80%以上を確保できます。しかし、ノイズが大きめなのが難点です。ESP-WROOM-02 を使うのは初めてということもあり、おそらく大丈夫だろうと思いつつも、安定して動く自信は持てませんでした。リニアレギュレータとスイッチングレギュレータを組み合わせるのであれば 2 とあまり変わりません。

消去法の結果として 2 を選びました。9V から 5V を生成するスイッチングレギュレータと 5V から 3.3V を生成するリニアレギュレータを組み合わせます。リニアレギュレータの効率は 66%です。スイッチングレギュレータの効率が 80%だとすると、トータルの効率は 52%となります。正直かなり悪い数字ですが、一応 50%は超えそうです。5V 電源が 4V 程度まで低下しても良いとした場合、リニアレギュレータの効率は 82.5%まで改善します。9V からの変換でもトータルで 66%まで改善しますので、これであれば USB 駆動でも電池駆動でも効率は変わらないと言えます。この場合、5V 電源が常に

使えるとは言えないので 1 の案に近くなりますが、低圧化も念頭におきつつ各負荷の電源をどうするか検討していきます。

リニアレギュレータは 5V から 3.3V という比較的落差の少ない条件で動作する必要があるので LDO(Low DropOut regulator)と呼ばれる品種が必要です。以降ではリニアレギュレータではなく、LDO と記載します。

## ESP-WROOM-02

まず、選択の余地がない ESP-WROOM-02 の電源からみてみます。通常はマイコンというとあまり電流は消費しないものですが、ESP-WROOM-02 は Wi-Fi を搭載しているため電流消費が多くなっています。Web を検索してみるとみなさん苦労されているようです。データシートには 500mA 以上の電流を供給できること、と書いてあります。また、平均的には 80mA 程度の電流が流れると書いてあります。電流の実測をされている先人たちのページをありがたく見てみると、350mA くらいは実際に流れるようです。

350mA という大きさも問題ではあるのですが、そのタイミングも問題です。常に 350mA が流れるのであれば電源回路の制御は安定しやすいのですが、そうではありません。このような大電流が流れるのは Wi-Fi 関連の回路が動作する一瞬のみです。このように負荷が大きく変動する場合、電源回路の制御が追いつかない可能性がでてきます。対策としては大きなコンデンサを設置して電圧の安定を図る、制御が高速な LDO を選択する、の 2 点です。

実績のありそうな回路でみてみると、秋月電子の開発ボードに搭載されている LDO の新日本無線 NJM2845 は 800mA 供給可能な品種で、スイッチサイエンスの開発ボードに搭載されている LDO のトレックス・セミコンダクターXC6222 は 700mA 供給可能な品種です。開発ボードの場合、周辺回路への電源供給もある程度こなせる必要があるので、200mA から 300mA 程度の余力を持たせるという設計なのかもしれません。このくらいの余裕があれば、みんな大好き LED チカチカ回路やリレーカチカチ回路くらいは問題なく動きますね。負荷変動への対策については両者で大きくアプローチが異なっています。秋月電子の場合、大きなコンデンサを設置する代わりに LDO としてはそれほど高速ではない品種を選んでいます。スイッチサイエンスは逆で、コンデンサの容量は最低限ですが、制御の高速な高速負荷過渡応答 LDO を選んでいます。スイッチサイエンスのほうが今風の設計です。

## FT231XS

次に USB シリアルコンバータの FTDI 社製 FT231XS を考えます。このチップは 5V または 3.3V で動かせます。内部には 5V から 3.3V を作る LDO と、3.3V から 1.8V を作る LDO が入っているようで、多少品質の悪い 5V 電源でも安定して動作するようです。USB 電源をそのまま接続できる設計なのですね。3.3V 出力については外部出力ピンが用意されています。これは I/O 電圧の供給用(VCCIO)とされていますが、ちょっとした回路であれば周辺回路の電源まで全部まかなえてしまいます。といっても、FT231XS の LDO は最大で 50mA までしか流せませんので、ESP-WROOM-02 の電源としては使えません。FT231XS 専用の 3.3V 電源として使うことはできますが、I/O ピンを通じて基板上の 3.3V 電源と衝突するとわずかな電圧差により電流が逆流する可能性があります。丁寧に設計すれば良い話ではありますが、がんばってもさほど利益はなさそうなので今回は内蔵 LDO は使わない方針です。I/O 電源としては基板上の 3.3V を使います。

残るは FT231XS の主電源(VCC)を USB の 5V から取るか、3.3V から取るかという話になります。USB ケーブルを抜いたときに FT231XS の電源を落としたいのであれば 5V からとったほうがよく、FT231XS 周辺の回路をシンプルにしたいのであれば主電源も I/O 電源も 3.3V に統一したほうがよさそうです。電池の消費を考えると USB ケーブルが繋がっていないのであれば FT231XS は電源が切れていてくれたほうがわずかながら電池の持ちが向上します。ただし、ケーブル未接続時には FT231XS はサスペンド状態となり、消費電流は 125uA に抑えられますので本当にわずかです。というとこで、今回は回路をシンプルにすることを優先して FT231XS の電源は 3.3V としました。・・・が、やはり 5V にしておけばよかったと後悔していますので、次に作るとしたら 5V 電源に改修しようと思います。このへんの事情は USB シリアル回路で解説します。

## リレー駆動

リレーには電磁石で物理的にスイッチを動かす機械式リレーと、半導体間で光などを使って信号を伝達する半導体リレー(SSR: Solid State Relay)があります。今回は趣味的なこだわりから機械式リレーを使うことに決めました。

機械式リレーはスイッチを動かすためにやや大きめの電力を消費します。今回選定した HSIN DA 製のリレーの場合、150mW を消費します。電流としては 5V 駆動の

製品で 30mA、3V 駆動の製品で 50mA という計算です。ESP-WROOM-02 の平均電流が 80mA で、FT231X の平均電流が 8mA ですから、今回の負荷の中では大きい部類です。ESP-WROOM-02 の GPIO の出力は最大で 3.3V/12mA ですから、リレーを直接駆動する力は全くありません。

電源としては 5V でも 3.3V でも設計できますが、

- リレーは大きめの電力を扱うこともあり電源の効率の影響が大きくでる
- ノイズを出しやすい素子なので、少しでも ESP-WROOM-02 や FT231XS から離しておきたい

という 2 点から LDO を経由しない 5V 駆動とすることにしました。これであれば USB 電源の場合は損失なし、電池駆動であっても 80%程度の効率でリレーに電力を供給できます。

## 部品選定

部品としては 5V 出力のスイッチングレギュレータと 3.3V 出力の LDO が必要となりました。Digikey や RS コンポーネンツには選択肢が山ほどありますが、今回は秋葉原で買えることを条件としています。秋月電子とマルツ電波の在庫をみつつ、以下を選定しました。

- スイッチングレギュレータ…MINMAX M78SAR033-0.5(秋月電子)
- LDO…LINER TECHNOLOGY LT1963AEST-3.3(マルツ電波)

スイッチングレギュレータは 500mA 出力で余裕がありませんが、コイル等を一体化したモジュール製品、コンパクトで使いやすい、効率も良い、出力電圧の調整も効く、ということで M78SAR を選択しました。5V 出力なので 5V 品を買えば良いところではありますが、3.3V 品だと入力電圧を低く設定でき、いろいろと調整の余地がありそうです。外付け抵抗で 5V に調整して使います。これを 4V に調整してリレーの駆動に問題が生じないのであれば(リレーの仕様上は生じないはず)、電池駆動時の電源効率を改善できるはずです。が、結局は時間の都合で細かな調整などしなかったので、2 枚目を作るとしたら素直に 5V 品にします。基板を作り直すとしたら、1A 出力だけど出力電圧が調整できない M78AR05-1 を選択するか、別メーカも含めて再検討します。

LDO は出力に余裕があり、負荷過渡応答が高速なものとしては秋月電子で取り扱っている Analog Devices ADP3338AKCZ-3.3(1A 出力、300 円)と、マルツ電波で取り扱っている Linear Technology LT1963AEST-3.3(1.5A 出力、540 円)が目にとまりました。Linear Technology は Analog Devices に買収されましたので、結局は同じ会社の

製品です。LT1963AEST-3.3 のほうが出力に余裕があり、高速負荷過渡応答が製品の特徴のトップにきているので、こちらを選択しました。ESP-WROOM-02 と同じくらいの価格なのは問題ですが、まずは安定して動いてくれないことには展示会に間に合いません。スイッチングレギュレータが 500mA 出力ですので電池駆動のケースではバランスが悪いですが、USB 給電(USB3.0)の場合は能力を発揮できます。LDO に余裕があると開発中のフラッシュメモリへの書き込みも安心して実行できますし、電池駆動で電流不足に陥った場合でも USB モバイルバッテリーを使えば余裕をもって動かせるという保険にもなります。・・・ということを勘案してもやはり 1.5A は余裕持ちすぎですが、コストダウンは二の次ということで勘弁してください。

### 006P 電池

回路設計ではありませんが、9V 電源として想定している 006P 電池について少しふれておきます。この電池はギター界隈ではわりと見かけるものではあるのですが、あまり性能は良くありません。電圧が高いのは良いのですが、電流を流す能力も、容量も限られています。パナソニックの金色電池の 6LR61Y(XJ)について見てみると、定電流連続放電特性のグラフは 200mA 強までしか測定されていません。200mA で放電していくと、30 分ほどで 7.2V を下回るようです。この電池を基準に考えてみます。

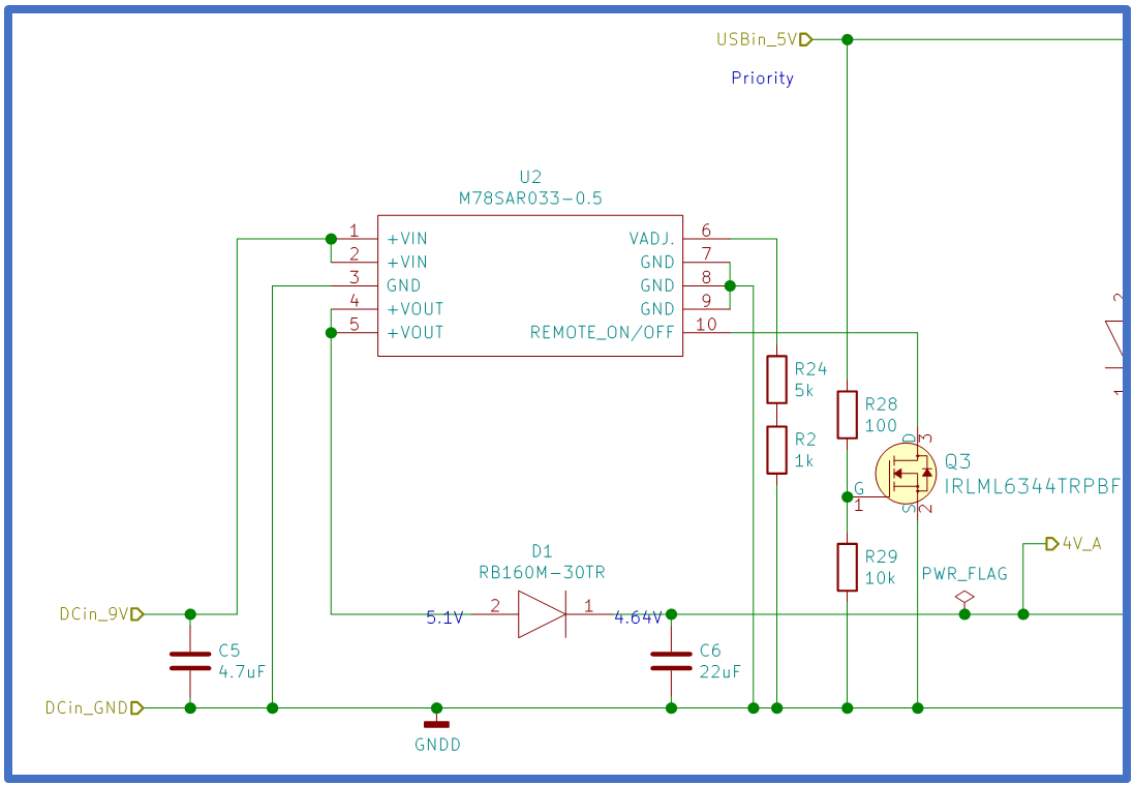
今回の回路設計では、電源の効率は 52%です。これは出力側の 2 倍程度の電力が 006P 電池から出力されなければならないことを意味します。ESP-WROOM-02 が 80mA を消費しているという条件では、3.3[V] * 88 [mA] = 264[mW]の電力が消費されます。電源の効率が 52%だとすると、電池の出力は 508[mW]です。電池が 9V を保っている状態で 56[mA]、7V まで電圧が低下した状態で 73[mA]の電流が流れる計算です。終止電圧 7[V]/出力 73[mA]という想定で 6LR61Y(XJ)の定電流連続放電のグラフを調べてみると、電池の持ちはおよそ 3 時間とわかります。実際には Wi-Fi の送信やリレーの駆動でより多くの電流が流れる、電圧が高いうちは電流が少なくて済む、7[V]以下でもレギュレータはまだ耐えられる、と色々な要因がありますから正確な稼働時間の予測は困難ですが、そう長くないのは間違いありません。

ライブで利用する場合、1 時間に 1 回程度インターバルを取って電池交換をするのが良いと思われます・・・・いや、不便だよなぁ。ESP-WROOM-02 をスリープ状態にできると飛躍的に寿命が伸びますが、今回は演奏に合わせてリアルタイムに反応してくれないと困りますのでスリープするタイミングがありません。外部電源を使うのが現実的ですね。

強い電池としては、ラジコンやドローン用の 3 セルリポバッテリーがあります。試みに検索してみると、11.1V/30C/3000mAh という品種が見つかりました。これは連続放電で 90A(!)近い電流を取り出せ、56mA で 50 時間におよぶ稼働時間を達成できるという計算になります。フルパワー出力だと性能は下がると思いますが、これなら電池一本でソレノイドまで含めて余裕で全部駆動できちゃいそうです。大きさは 13cm x 4.5cm x 2cm といったところです。ドローンをいじっていると電源設計の感覚が狂っちゃいますね。

スイッチングレギュレータ

下図がスイッチングレギュレータ周辺の回路の拡大です。




U2(M78SAR033-0.5, スイッチングレギュレータモジュール)がレギュレータ本体です。入力側には 9V 電池がつながります。入力コンデンサ C5 は入力リップル電圧やそこから生じるノイズを決めることになります。データシートには標準規格に適合する EMI フィルタの構成例が載っており、4.7uF のコンデンサとフェライトコアが並んでいます。ここに繋がるのは電池ですし、EMI など気にするだけ無駄という基板ですが、お守り代わりに一つ付けておいたのが C5 です。売り物であれば厳密に計算しないといけないところでしょうけど、これはかなり難しい話です。仕事で商品を作る場合、ハー

ドウェアは外部の専門家にお任せすることになりますが、EMI 関連の試験に関しては専門家でも苦労している印象があります。

ダイオード D1(RB160M-60TR, ショットキーバリアダイオード)は USB 電源から電流が逆流してくるのを防ぐために付けてあります。電圧降下 Vf は 0.5A 流している状態で最大 0.46V です。Vf により出力電圧は 5V を下回りますが、負荷はリレーのみで 4V 以上であれば動作しますし、LDO に必要な電位差は確保できていますので良しとします。むしろ動作に問題がないのであれば低いほうが LDO の効率が改善します。

C6 は出力コンデンサですが、これは U2 の出力リップル電圧の抑制という機能と 5V 電源が USB と電池の間で切り替わるタイミングで 5V 電源を安定させるという機能があります。基本的に大きいほどありがたいようなものですが、U2 は 220uF 以上の容量性負荷は扱えません。C5 と同様にあまり根拠はないのですが、面積を取らないチップ積層セラミックコンデンサで、秋葉原で売っており、もっとも容量が大きいものということで 22uF をつけることにしました。電源切替には耐えないかもしれませんが、切り替えながら使うものではないのでダメならあきらめます。

R24 の 5kΩ(部品の値表示は 4.99kΩです)と、R2 の 1kΩは U2 の出力電圧調整用の抵抗です。この状態で出力は 5.1V になります。わずかな違いで大きく電圧が変わってしまうため、ここは誤差の少ない金属皮膜抵抗を直列にしています。

Q3(IRLML6344TRPBF, Nch MOSFET)は USB 電源に 5V がかかっている際に U2 の動作を止めるためのスイッチとなっています。Q3 が ON になると、U2 の REMOTE ON/OFF 端子がグランドレベルに落とされ、OFF 状態となります。これにより電池の消耗を防ぐ仕組みです。Q3 が OFF の場合、REMOTE ON/OFF 端子は未接続状態となり、U2 は ON 状態となります。Q3 のゲートには R28(100Ω)と R29(10kΩ)で USB の電源電圧 5V を分圧した電圧がかかりますが、抵抗値の差が非常に大きいので実質的には 5V がそのままかかることになります。Q3 のスレッショルド電圧 Vgs(th)は最大で 1.1V ですので、5V をかければ確実に ON 状態とすることができます。

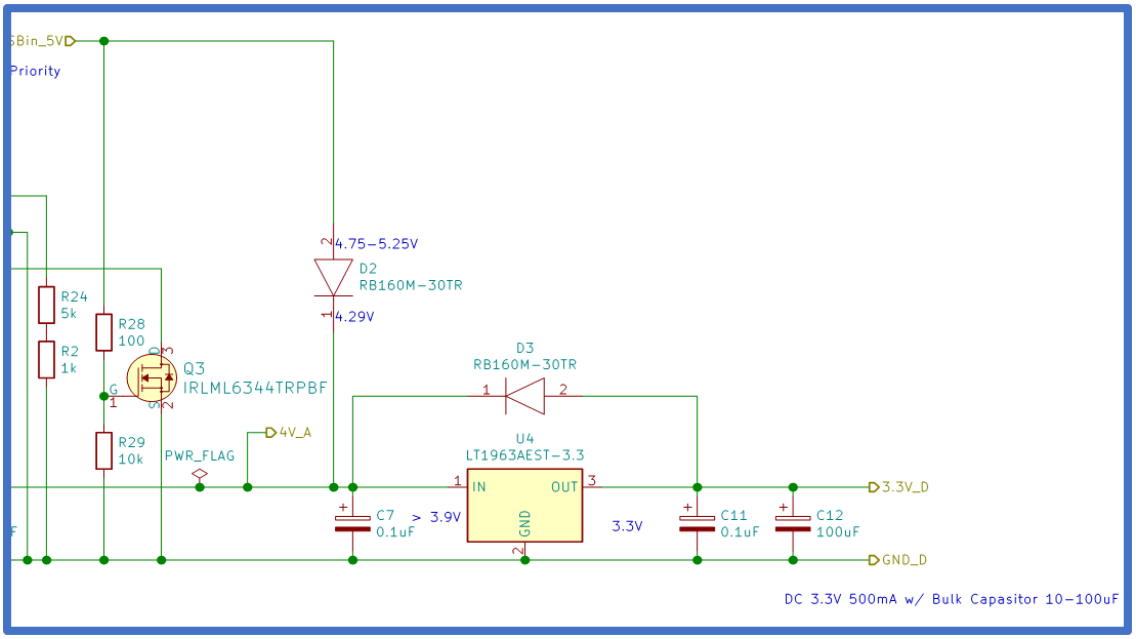
R28、R29 を使わずに USB の電源ラインを直接繋ぐだけでも多くの場合は動作します。ただし、USB が未接続の場合にノイズに弱くなるという問題があります。USB が接続されていない場合、USB 端子までの配線は、Q3 のゲートにアンテナを付けているような効果を生みます。このアンテナに外部から強いノイズが乗ってきてしまうと、Q3 がノイズに敏感に反応して ON/OFF を繰り返し U2 の動作を不安定にしてしまう可能性があります。

R28 を付けておくと Q3 のゲート容量の充電速度を遅くしてスイッチング速度を低下させるという効果と、高周波ノイズが配線上に定在波を作ってしまうような場合にエネルギーを吸収して減衰させるという効果があります。スイッチング速度が遅ければ、ノイズへの反応も鈍くなります。スイッチング速度は Q3 の仕様にあるゲート容量と R28 の値から計算することができ、速度が問題になる場合はここから最適値を求めておく必要があります。今回はスイッチング速度を追求するわけではないので定番の値として、手元の部品在庫が豊富な 100Ωを選んでいます。

R29 は R28 とは逆にゲート容量の放電に関わります。R29 がないと放電経路がなくなってしまいますので、USB ケーブルを抜いた際に Q3 のゲートに電荷が溜まったままとなり、U2 が起動されないという問題が生じます。ゲート容量はさほど高性能なコンデンサではありませんので、いつまでも溜まったままということはありませんが、安定動作のためには必要な抵抗です。こちらもスイッチング速度から値を計算する必要がありますが、今回は R28 と同様に定番の値として 10kΩを選んでいます。R28 と R29 は電源とグランドを繋いでいますが、R29 が 10kΩであれば電流は 0.3mA 以下ですので、あまり気にしなくても大丈夫そうです。

## LDO

下図が LDO 周辺回路の拡大です。




入力は先ほどのスイッチングレギュレータの出力と、USB の 5V 電源となります。D2(RB160M-30TR, ショットキーバリアダイオード)はスイッチングレギュレータから

USB への電流の逆流を防ぐためのダイオードです。USB の電源電圧は規格上は 4.75V から 5.25V までと 5%の誤差が許されています。この下限の電圧が入力された場合、D2 の Vf を差し引くと 4.29V が LDO への入力電圧の下限となります。

LDO である U4(LT1963AEST-3.3, LDO)のドロップアウト電圧は 340mV ですので、3.3V を出力するためには 3.64V 以上の入力があれば動作します。1 割程度の余裕をみても 4V あれば動作に問題はなさそうです。スイッチングレギュレータからの入力が 4.64V で、USB からの入力が 4.29V ありますから入力電圧に問題はありません。

C7 と C11 の 0.1uF のコンデンサは高周波ノイズを捨てるためのパスコンです。記号が間違っていますが、積層セラミックコンデンサを使っています。U4 のデータシートには入出力に 10uF のコンデンサがあれば安定動作すると記載されています。特にパスコンの指定はないのですが、リニアレギュレータは稀に発振が問題になることがある素子ですので保険として入れています。

入力側にはスイッチングレギュレータの回路図に出てきた C6(22uF)が接続されています。C6 は積層セラミックコンデンサですので、DC バイアスで実効容量は少なめになりますが、10uF は確保できそうです。USB 電源の場合は良いとして、電源がスイッチングレギュレータの場合、C6 は出力コンデンサとも言えるため、これだけで十分なのかの判断は困難です。正確に計算する方法も分からないので、作ってから波形を見て考える方針とします。

出力側の C12 は U4 のデータシートによれば 10uF が必要な箇所です。U4 は高速負荷過渡応答 LDO ですので、大きなコンデンサは不要なはずです。が、Web を検索するとたくさん出てくる ESP-WROOM-02 の電源問題をみていると不安になるので、一応 100uF の電解コンデンサをバルクコンデンサとして載せられるようにしてあります。

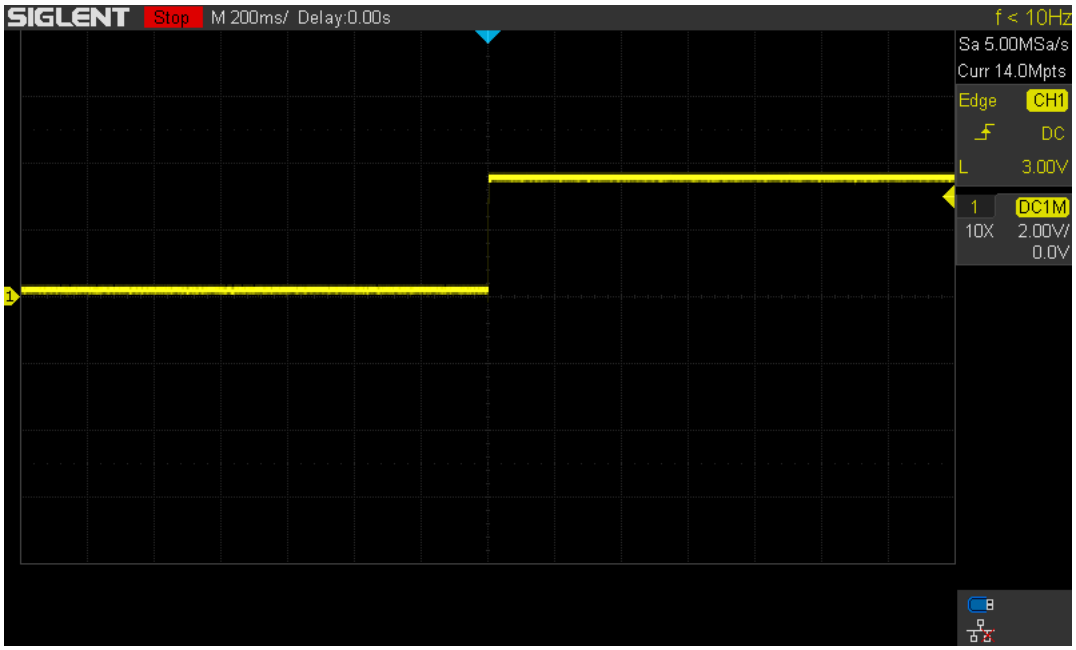
現在のところ、C11 に 10uF、C12 に 100uF を付けてあります。時間に余裕があれば C12 を外して試してみたいところです。C11 を外すと、出力側が電解コンデンサのみとなってしまい、これだと U4 が安定して動作しません。データシートではセラミックコンデンサを使うように指示されています。

D3(RB160M-30TR, ショットキーバリアダイオード)は電源を切った際に、C11 や C12 などの出力側コンデンサの電圧が U4 の出力端子にかかって入出力の電位差が逆転するのを防ぐためのダイオードです。U4 は保護機能が充実した品種なので不要かとも思いますが、念のために付けています。電源を切った際にコンデンサに溜まっていた電力は、コンデンサ自身の内部抵抗(ESR)と D3 の電圧降下で大部分が消

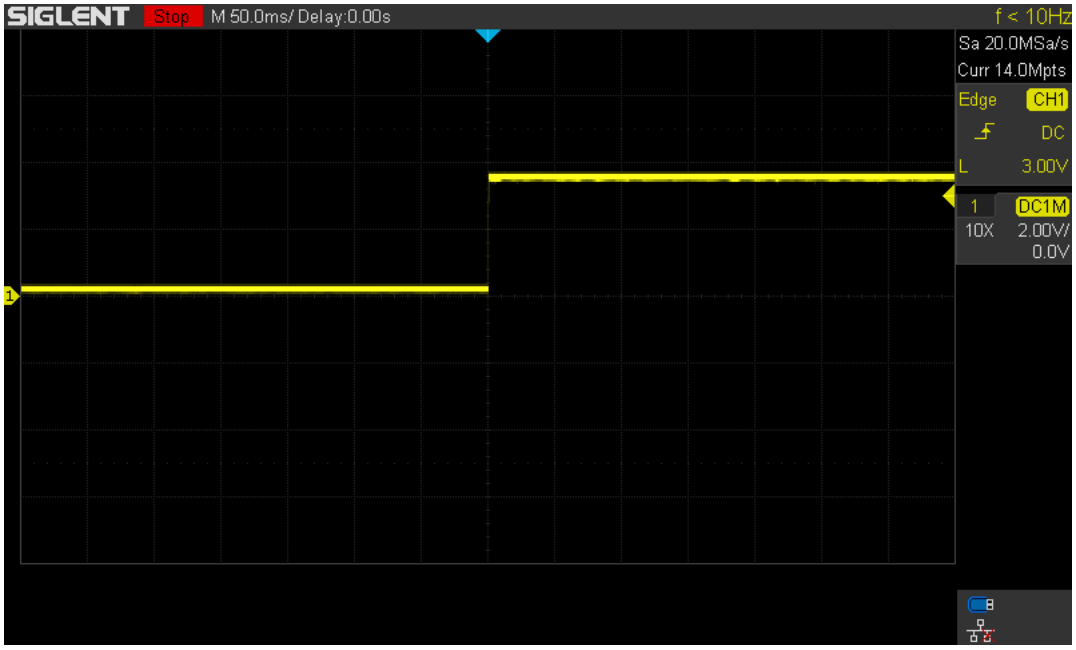
費されます。C12 の ESR の測定はしていませんが、データシートでは-25℃で 4Ωということなので、2Ω～3Ω程度はありそうです。放電電流は D3 の定格である 1A は超えないのではないかと思われます。D3 はサージ電流には 30A まで耐えますし、厳密な計算はしていませんが、壊れることはないでしょう。

### 3.3V 電源の波形(USB 電源の場合)

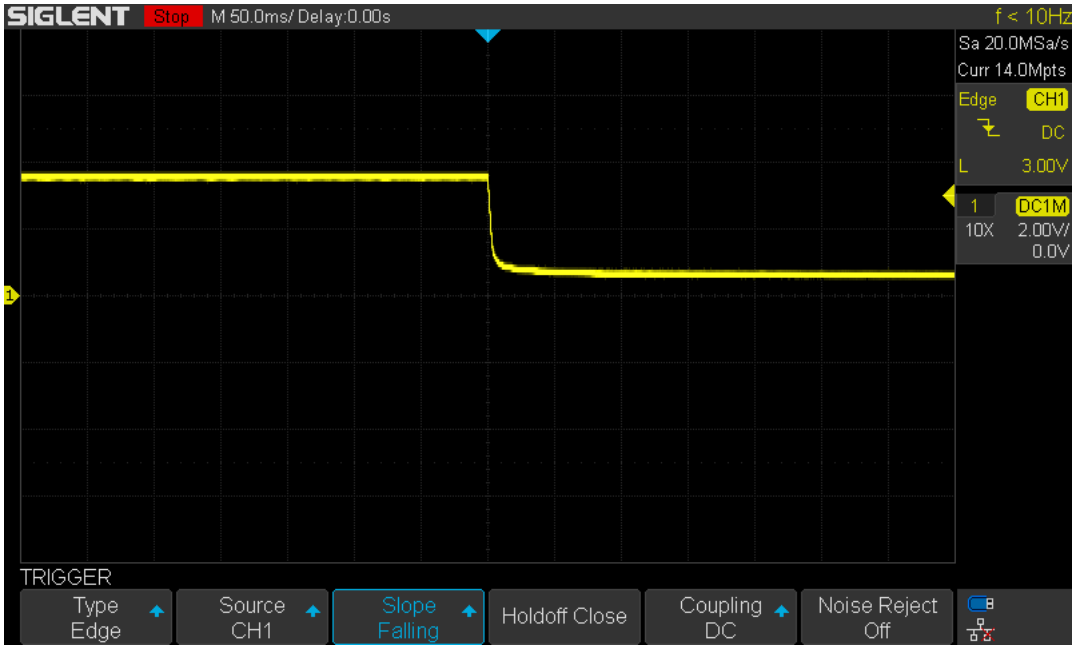
下図に 3.3V の立ち上がり波形を示します。横軸は 1 目盛 200ms です。電源投入直後に ESP-WROOM-02 が大電流を要求して電源電圧が下がりやすいというレポートが散見されるのでドキドキしながら計測しましたが、耐えているように見えます。一応、オシロの立ち下がりトリガを仕掛けたりもしましたが、トリガはかかりませんでした。




下図は時間軸を 50ms にして計り直したものですがあまり変わりません。

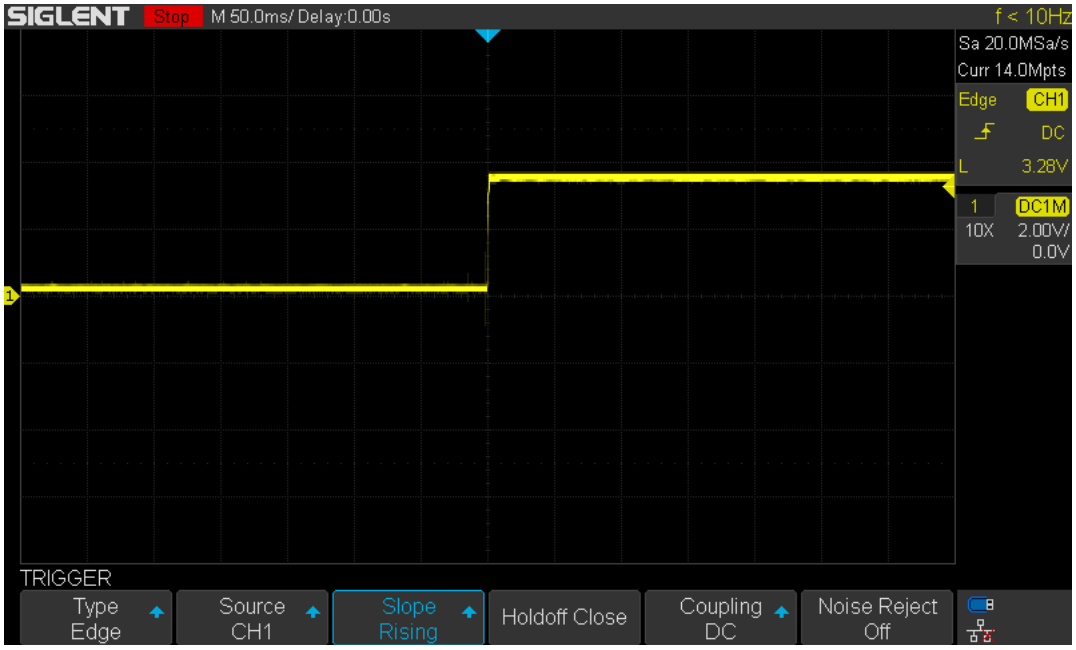



電源 OFF 時の波形も載せておきます。バルクコンデンサの放電があるため、やや緩やかな下がり方をしていることがわかります。3.3V 電源に対して 3.28V の立ち下がりトリガという挑戦的なトリガ設定ですが、電源 OFF までトリガはかかりませんでした。さすが Linear Technology 製 1.5A 出力 LDO ですね。

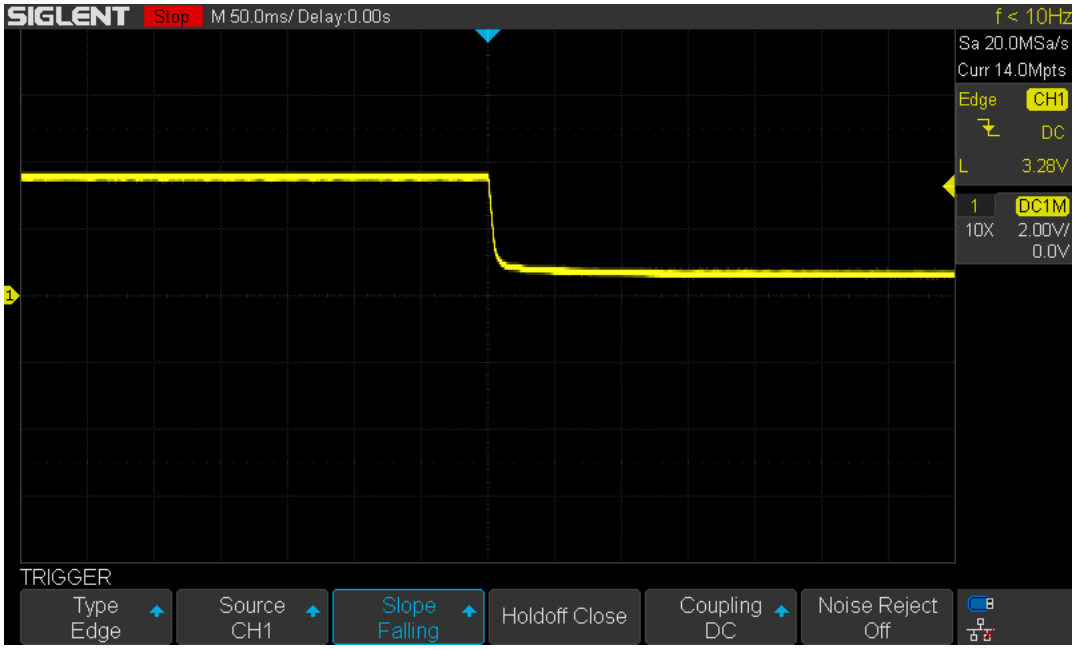



9V 電源の波形(006P 電池の場合)

以下は 9V 電池を接続した際の 3.3V ラインの波形です。USB とあまり変わりません。

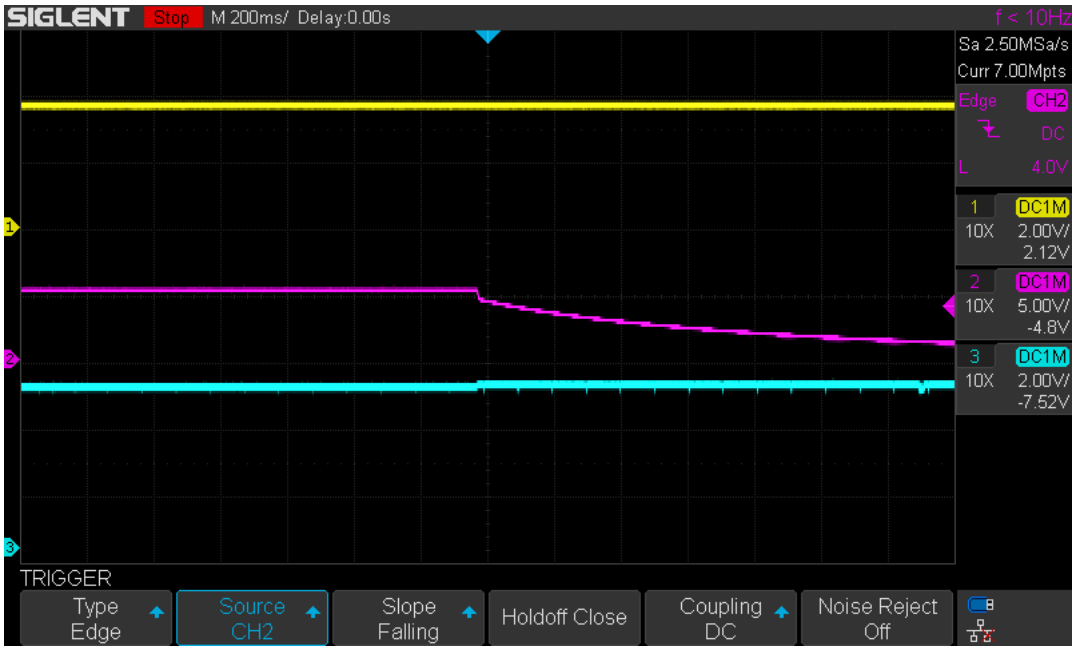



電池を外した場合も同様です。




スイッチングレギュレータの出力と LDO の出力を比較すると下図のようになりました。上が LDO、下がスイッチングレギュレータです。横軸は 1ms にしています。スイッチングレギュレータのほうがゆっくり立ち上がりますが、LDO は早い段階で安定しています。


さて、厳しい条件ですが、9V 電池で動いているボードに USB を接続して電源を切り替えた場合の動きは以下のようになりました。




黄色は LDO の 3.3V 出力、紫はスイッチングレギュレータの出力、水色は LDO への 5V 入力です。5V 入力ラインが切り替わり、微妙に電位が上昇すると同時にスイッチングレギュレータが OFF になり緩やかに電圧が低下していきます。スイッチングレギュレータが OFF の間、LDO の細かなノイズが目立ちます(よく見るとノイズは切り替わり以前から存在しています)。100ms に一回観測されていますから周波数は 10Hz というところです。スイッチング周波数よりもかなり遅いですが、なんでしょうね？測定

環境由来の外来ノイズかもしれません。LDO の出力は安定しており、電源の切り替わりの様子はわかりません。あとから USB を繋ぐ分には、ESP-WROOM-02 にはリセットがかからず継続して動作してくれそうです。これなら動作がおかしなときには USB をつないでデバッグできそうです。

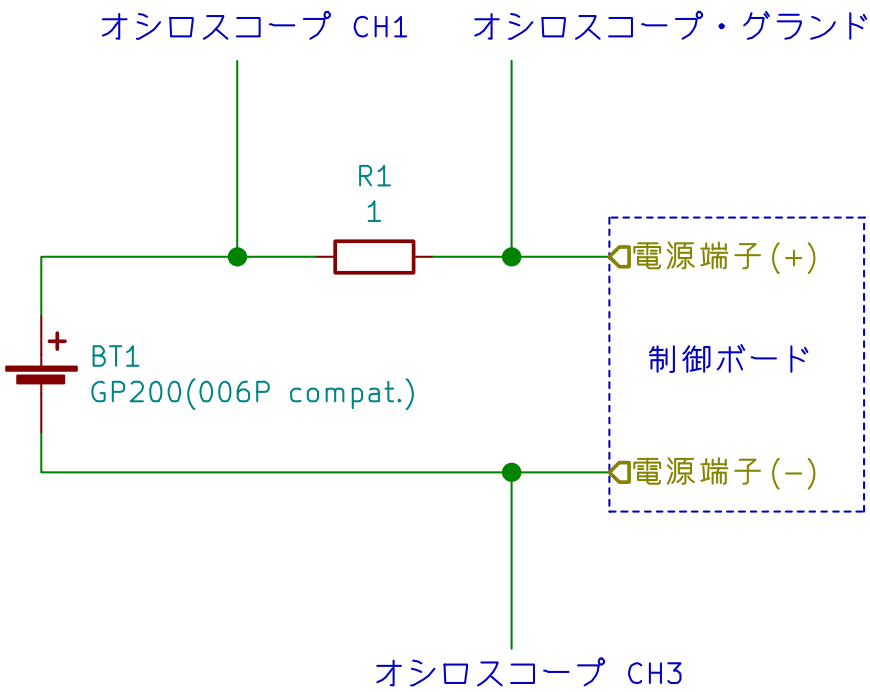
逆に USB と 9V 電池を両方接続している状態から、USB を抜いた場合の波形は以下のようになります。


黄色は LDO の 3.3V 出力、紫はスイッチングレギュレータの出力、水色は LDO への 5V 入力です。USB の電源を喪失してからスイッチングレギュレータが立ち上がるまで LDO の 3.3V 出力は維持できず、1.68V まで電圧が低下しています。時間は約 5ms です。確認していませんが、おそらくリセットがかかっているでしょう。100uF のバルクコンデンサでも足りていないということなので、これをコンデンサで救うのはかなり難しそうです。スイッチングレギュレータは 220uF までの容量性負荷しか扱えません。もともと自由に抜き差しして動くという要件は設定していませんが、USB を抜くとリセットがかかるという点は運用上注意が必要かもしれません。

USB 電源が使える状態で、スイッチングレギュレータの出力を OFF にしなければこの問題は生じませんが、それだと USB をつないでいるのに電池がどんどん消費されることになりかねません。現状で仕方ないか、というところです(9V 電源として AC アダプタを想定しているのであれば AC アダプタ優先でも良いのですが)。一応、スイッチングレギュレータの出力を USB よりも十分に低くできれば ON 状態でも電流は出力さ

れないでしょうから、電力消費はある程度は抑えられます。電源の変更に対応するとしたら、このへんが落とし所かと思います。

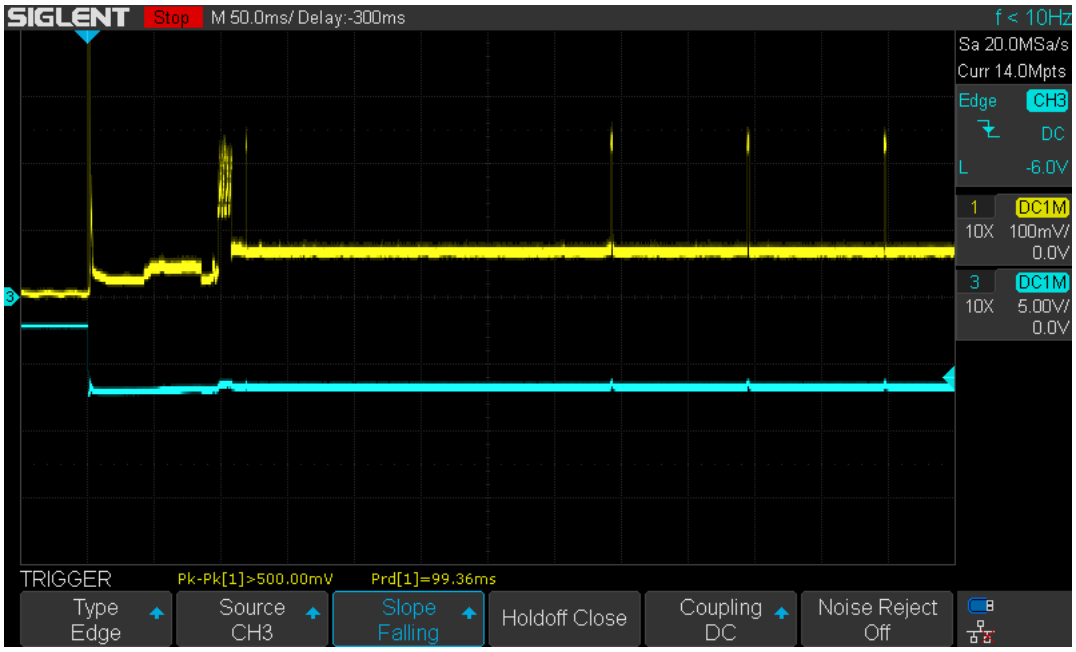
ここから電流を見ていきます。オシロスコープには電流を測定するための電流プローブというものもありますが、使いどころが少ないわりにお高いので手元にはありません。代わりに 1Ωの抵抗を電池と直接につなぎ、この抵抗の電圧降下を測定しました。ここで 1mV の電圧降下が観測されれば 1mA の電流が流れているのであろう、という原理です。利用した電池は GP200 という 006P 型のニッケル水素電池で、内部抵抗は 3Ωから 4Ω程度あるようです。1Ωの抵抗を繋ぐのは影響が大きいのですが、換算が簡単だという怠惰な理由と、これ以上小さな抵抗で微小な電圧降下を観測するとなるとオシロスコープの分解能が足りなくなりそうだという現実的な理由で 1Ωを利用しました。電流検出用の抵抗はシャント抵抗と呼ばれ、mΩオーダの製品がたくさんあります。本来はこうした抵抗と測定用のアンプ回路(またはアンプ内蔵のアクティブプローブ)を使い、測定対象への影響を軽減すべきところでしょう。




測定回路は上のようになっています。制御ボード全体を電池駆動として接地されていない状態にし、オシロスコープのグランドをシャント抵抗の直後にとっています。チャンネル 1 をシャント抵抗の手前、チャンネル 3 を制御ボードのグランドとしています。差動プローブや絶縁オシロスコープなどの機材を使うと、もっと素直な測定回路も組

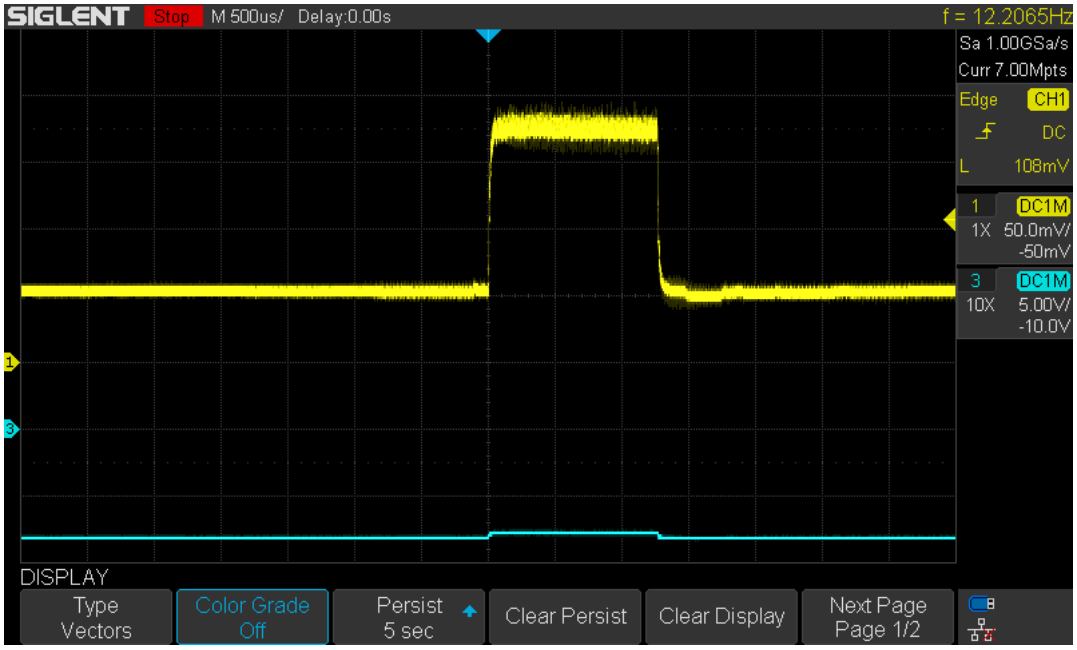
めるかもしれませんが、高価な機材ですので仕事でない限りは使う機会はないでしょう。細かいですが、チャンネルが 1 と 2 ではなく 1 と 3 になっているのは、手元のオシロスコープが ADC を 2 系統搭載していて、1/2 および 3/4 で ADC を共用しているためです。1 と 3 で測定すると各チャンネルに 1 つの ADC を割り当てることが可能です(共用するとサンプリングレートが半分になります)。

まずは電源投入時の波形をみてみます。

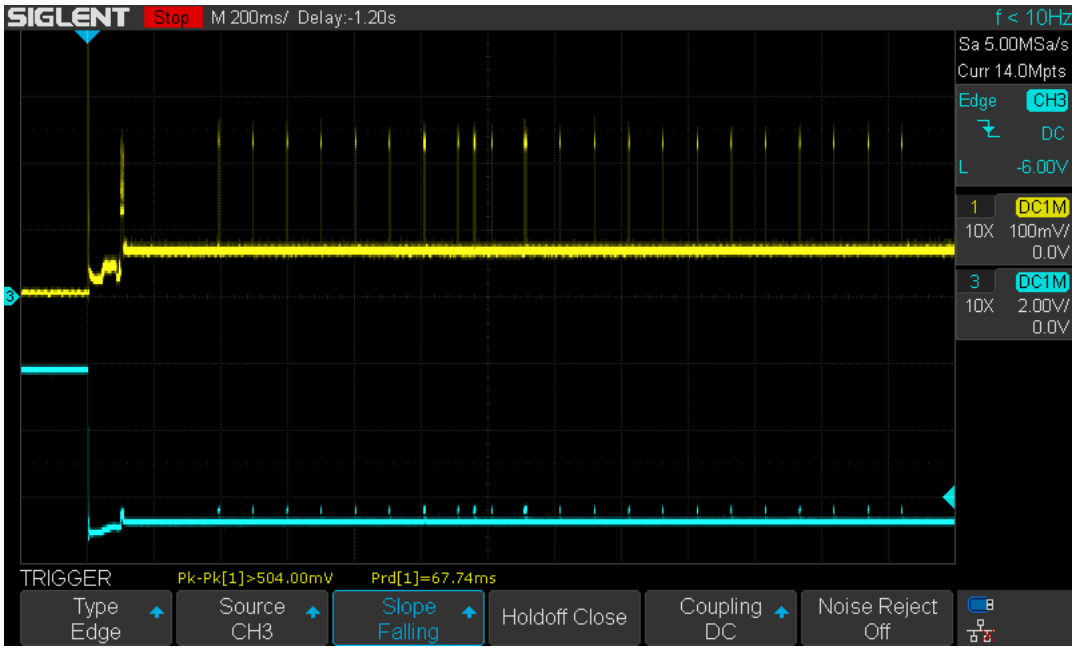



黄色がチャンネル 1 で電流を表します。横軸が 1 目盛 50ms、縦軸は 100mV(=100mA)での測定です。起動直後に電源周りのコンデンサへの突入電流が流れて測定範囲を振り切っています。100ms 程度経過した時点で 200mA を少し超えるくらいの電流が 10ms から 20ms くらい流れています。無線回路のキャリブレーションでしょうか。その後は 80mA 前後で落ち着きますが、時折 200mA 超えのスパイクが

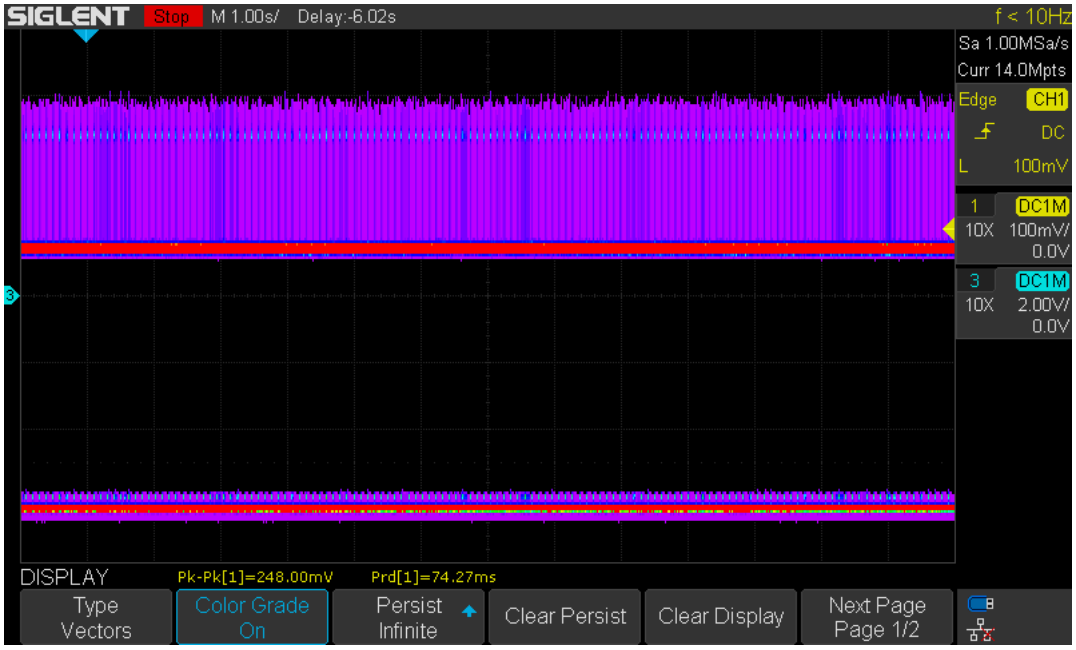
見えます。



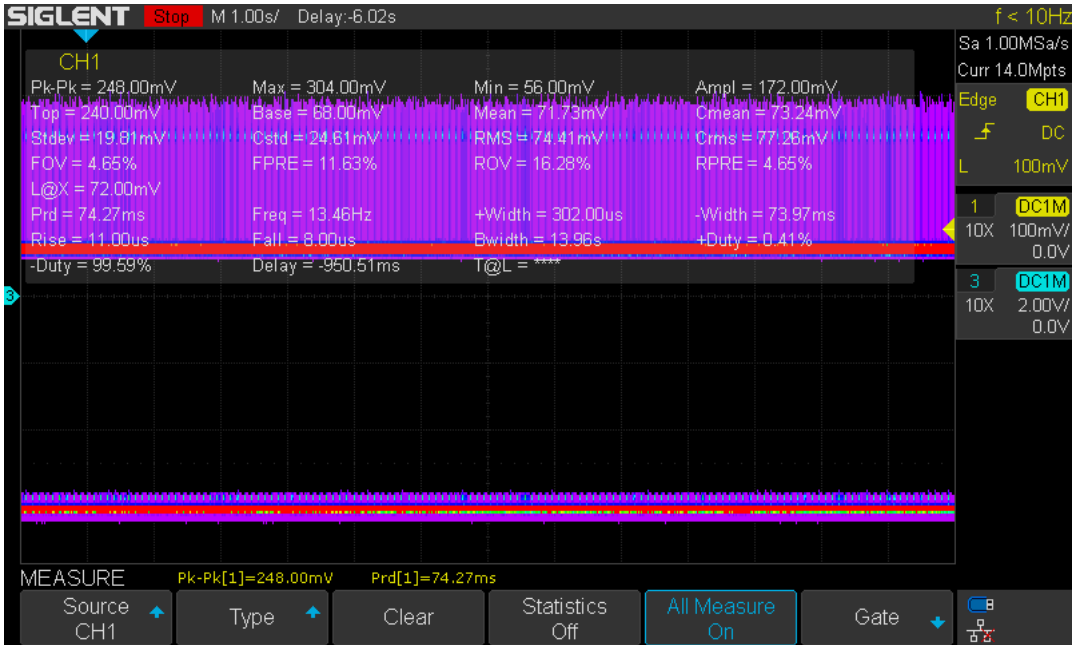

スパイク部分を一つだけ拾って拡大すると上図のようになっています。この場合、125us 程度 200mA 弱の電流が流れています。いかにも何かのスイッチが入ったような感じですね。無線用のパワーアンプ回路だと思われます。時間は処理内容によって長くなったり短くなったりするはずです。無線 LAN で巨大なパケットを送る場合にはより長くなるでしょう。



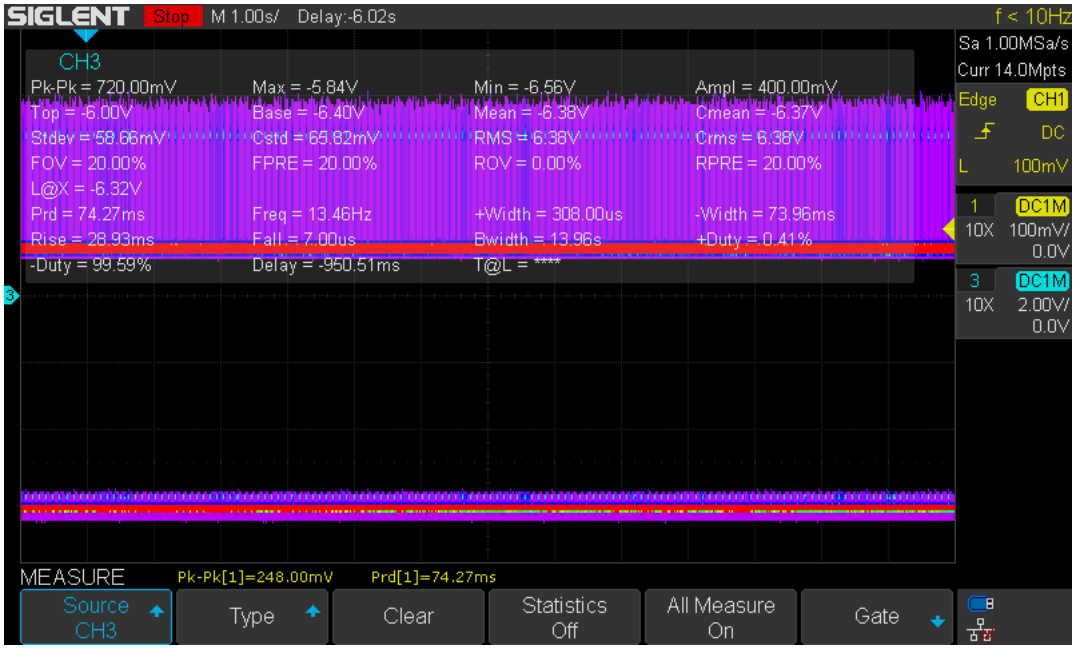
もう少し長い時間でみたものです。横軸を 1 目盛 200ms にしています。200ms から 300ms 経過後には起動処理が終わっているようです。ソフトウェアはアクセスポイント動作となっており、定期的にビーコンを送るなどの処理が実行されています。アクセスポイントとして動作している場合、省電力化は困難ですが、それでもかなりがんばっているように見受けられます。




さらに長い時間で測定したものが上図です。横軸が 1 目盛 1 秒になっています。そしてサンプリグを複数回実行してヒートマップ風の表示をさせました。ほとんどの時間帯は赤色のラインの中におさまり、時折紫色の領域になっていることもある、という見方をします。チャンネル 1 には上の方に緑色のラインが見えます。赤がアイドル状態、無線の処理中が緑、というのが基本動作で、稀に上下にはみ出すこともある(もしくは測定値にノイズがのっている)、ということでしょう。この波形から代表値を測定すると以下のようになりました。




最高で 300mA まで電流が流れるときがあり、平均だと 72mA 程度が流れているとわかります。事前の概算では電池の電圧が 7V の場合に 73mA で 3 時間の駆動と見積もりましたが、概ね一致していそうです。緑色のラインは Top として検出されており、240mV と判定されています。赤色のラインは Base で 68mV です。




チャンネル 3 の電源電圧も同様にみてみると、平均 6.4V、最低 5.84V、最高 6.56V となりました。ちょっと電池はへたり気味だったかもしれませんね。測定に利用した GP200 は定格 200mAh ですので、73mA を流していたら 3 時間もちません。いろいろと測定しているうちにかなり電力を浪費してしまったのでしょう。

両者の平均を掛け合わせて平均消費電力の概算を考えてみると、6.4V x 73mA で 467mW です。うーん、平均的にはスマホと同程度、画面を消して待ち受けになっているスマホよりもかなり多いという感じでしょうか・・・・。ピーク電力は高出力の LTE とは比較にならない低さだとは思いますが、あまり省エネとは言えませんね。電源効率が 52%ですので、ESP-WROOM-02 自体は約半分の 243mW の電力消費です。これ自体は優秀な値ですが、電源が甘すぎました。

電源を 9V に変更した場合は以下のようになりました(計測時間が少し短かったので疎になっていますが、基本的には同じ測定です)。電源は 12V の AC アダプタの出力をリニアレギュレータで 9V に落としています。


電源電圧が高いので平均で 55mA、通常時(Base)が 52mA、無線動作時(Top)が 176mA と電流値は減少しています。事前の概算では、電池の電圧が 9V のときに 56mA としていましたので、こちらも概ね予想通りとなりました。電源電圧が高くなると電流値が減っていくのはスイッチングレギュレータの効果です。これがリニアレギュレータだと、電源電圧を高くしてもレギュレータが熱くなるだけで電流値は減りません。

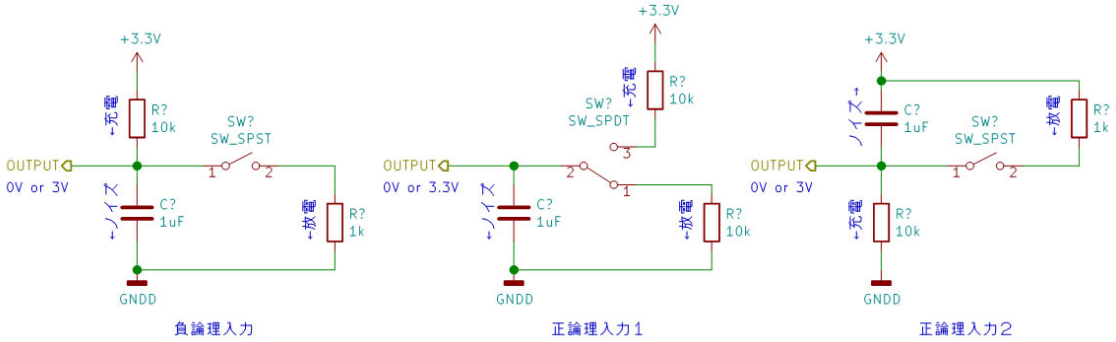
## スイッチ入力回路

### 全体像

スイッチ入力回路は 3 系統あり、各回路の構成はまったく同じです。目的は ESP-WROOM-02 の I/O ピンの電圧を切り替えて digitalRead()でスイッチの状態を読み取れるようにすることです。

I/O ピンとしては IO13,15,16 を利用しています。このうち、IO15 はブートモードの選択に利用され、起動時は LOW レベルになっている必要があります。本来は回路にリセットがかかる際には IO15 を切り離して LOW レベルに固定するようなリセット回路を設計しなければいけないところですが、今回は省略します。スイッチを押すのは人間ですので、注意書きでカバーです。外付けのスイッチ回路をつなげる場合は問題になるかもしれません。

スイッチというと気になるのはチャタリングです。ポーリングではあまり問題になりませんが、割り込み駆動にした場合には割り込みが多発してソフトウェア的なフィルタリングが必要になりちょっと面倒です。常套手段としてはコンデンサを使った LPF を付けておくことになります。



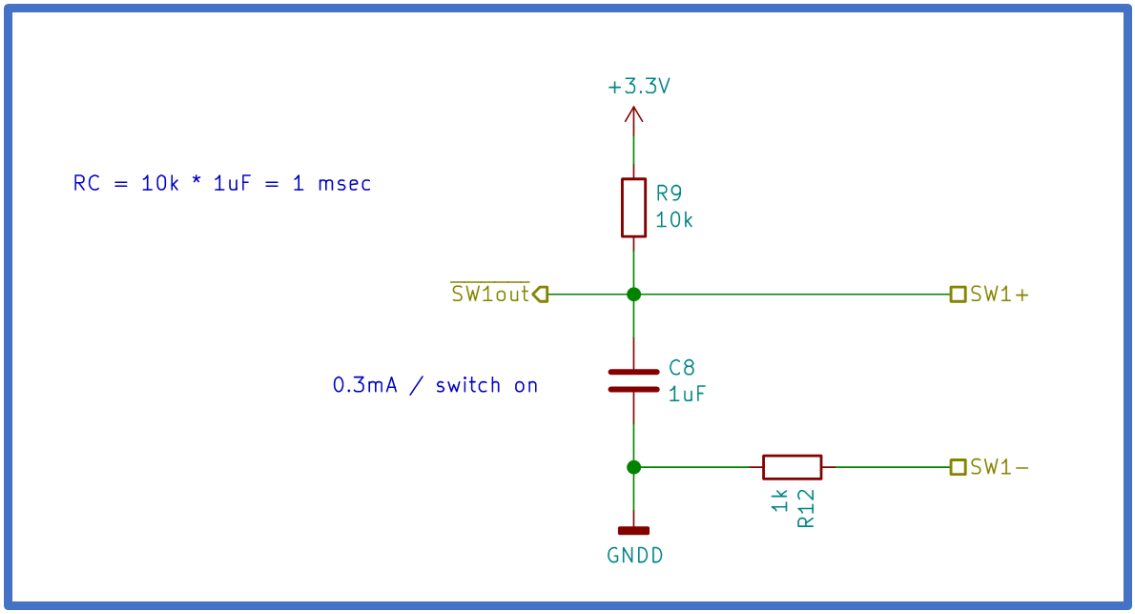

正統的な設計としては、スイッチ OFF 時にコンデンサを充電しておいて、スイッチが ON になると放電されるという負論理入力回路が考えられます。スイッチを SPDT タイプにすると、正論理入力を作ることもできます(正論理入力 1)。また、コンデンサを電源側にもってくると SPST スイッチで正論理入力を作れます(正論理入力 2)。

ソフトウェア的には正論理にしておいたほうがわずかながら分かりやすくなります。しかし、スイッチを SPDT にすると(正論理入力 1)ピンヘッダ経由で外付けスイッチを使うケースで配線が増えて不便です。SPST で正論理を作ろうと思うと(正論理入力 2)、ノイズが電源側に流れたり、逆に流れてきたりという問題が生じます。また、GPIO

を出力に設定して LOW から HIGH に遷移させると、正論理入力 2 の回路はコンデンサにたまっている 3.3V の低位側電位を持ち上げて電源ラインを 6V 前後に昇圧させてしまいます。LDO ががんばって抑え込むはずですが、あまり気持ちよくはありません。ダイオードと組み合わせれば GPIO で高圧パルスを作る回路として使える可能性はあり昇圧回路の実験としては面白いですが、今回の用途には不適と言えるでしょう。というわけで、素直に負論理入力にしました。

回路図

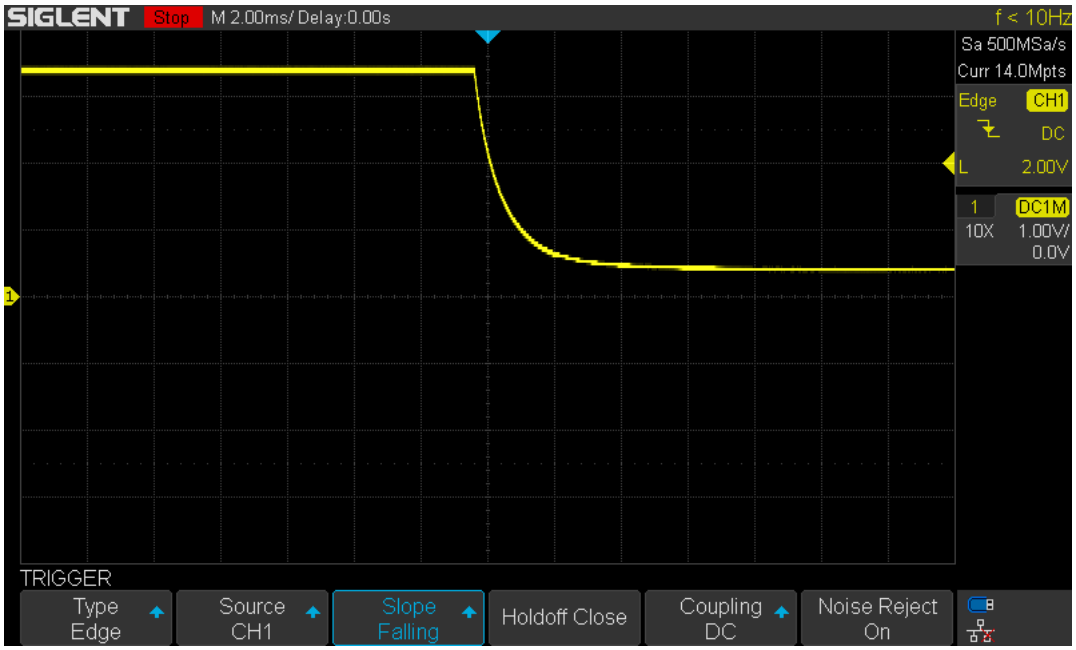
下図に回路図の拡大を示します。3 系統同じものがならんでいますので、1 系統分を抜粋しています。



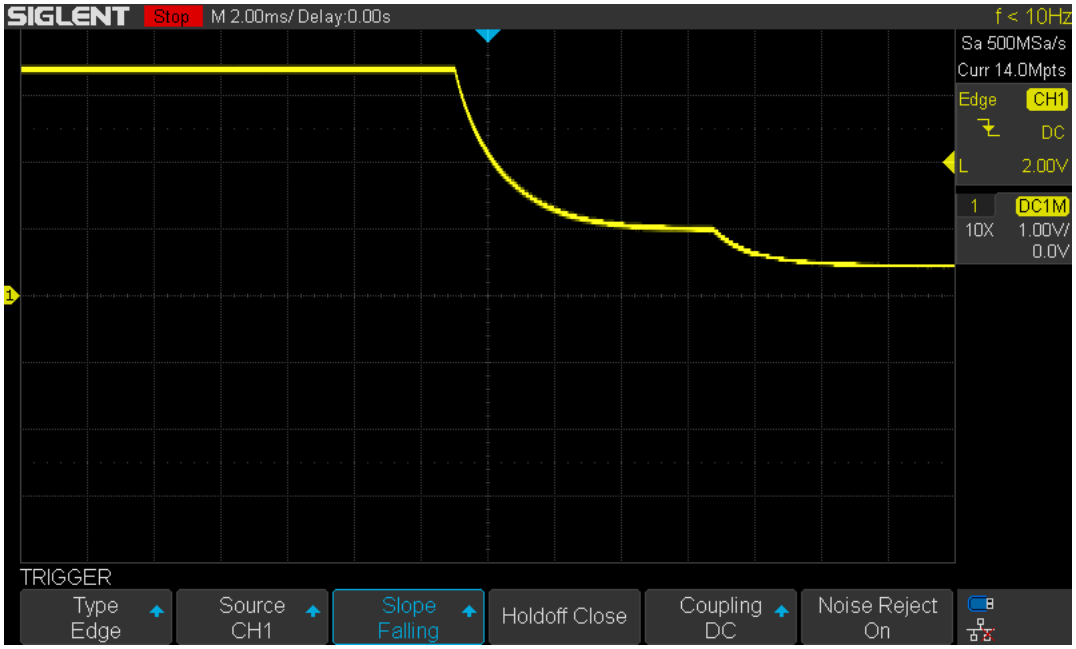

スイッチは負論理入力とし、充電側(R9,R10,R11)に 10kΩ、放電側(R12,R13,R14)に 1kΩとしました。スイッチが ON になっていると、抵抗が直列に繋がれて 10:1 に分圧されるため、GPIO には 0.3V が入力されます。コンデンサ(C8,C9,C10)はスイッチ OFF で 3.3V に充電されており、スイッチ ON で 0.3V まで放電されます。コンデンサの容量次第で LPF の効きが変わってきますが、現在は 1uF で設計しています。スイッチ ON で 0.1msec、スイッチ OFF で 1msec の動作遅延となります。

観測してみると基板上のタクトスイッチだとチャタリングはあまり目立たないのですが、外付けのスイッチなどで試してみると 0.4msec から 0.5msec 程度の期間発生することがあり、パルス幅が 80usec 以下という感触です。完全ではないにせよ、ある程度は抑え込めるのではないかと思います。

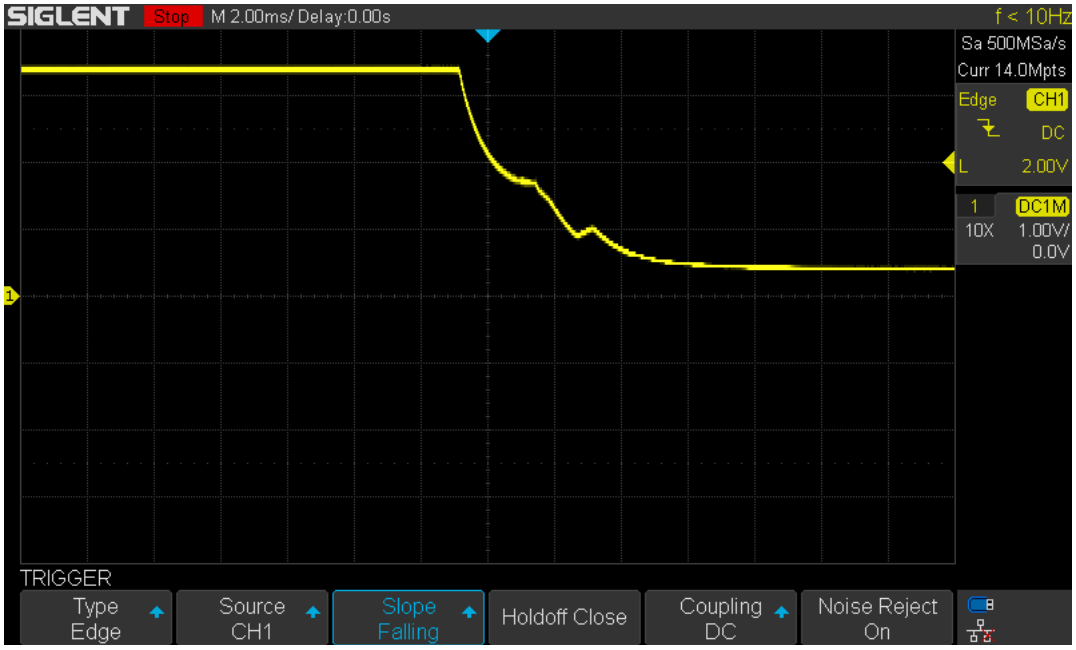
波形



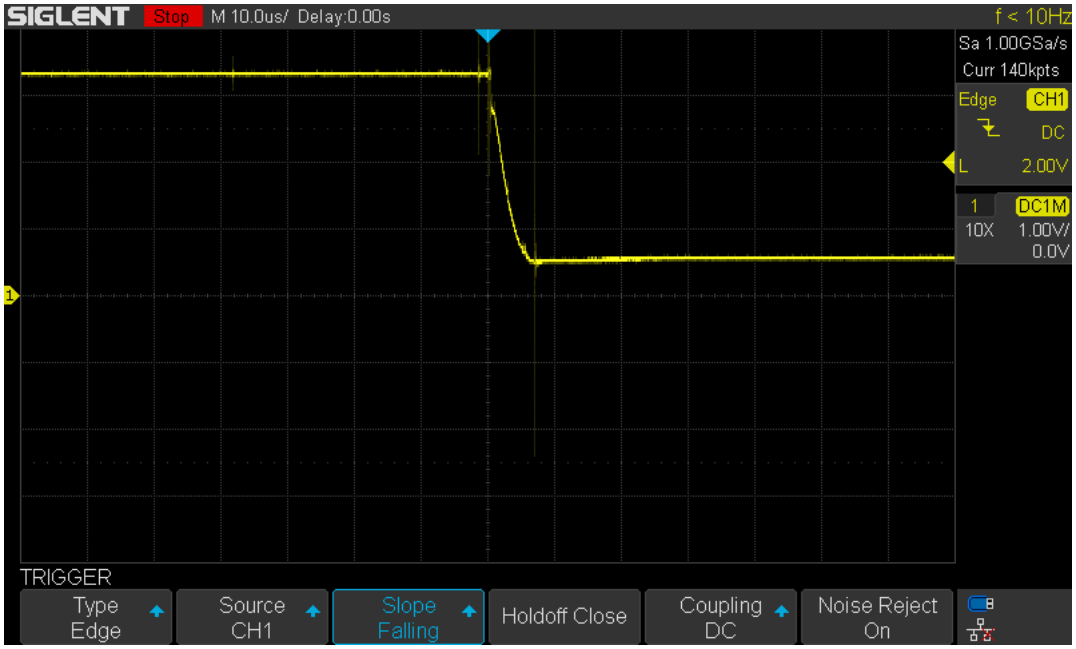

上図は基板上に実装したタクトスイッチを押した際の IO13 への入力波形です。自然な放電曲線になっており、これならチャタリングの問題は生じないと考えられます。




上図はピンヘッダにワニロクリップを噛ませて外部のスイッチを押した場合の波形です。波形に段差が付いている部分がスイッチの物理的な接触状態の兼ね合いで抵抗値が綺麗に下がらなかった箇所だと思われます。比較的悪い測定結果を拾いました。



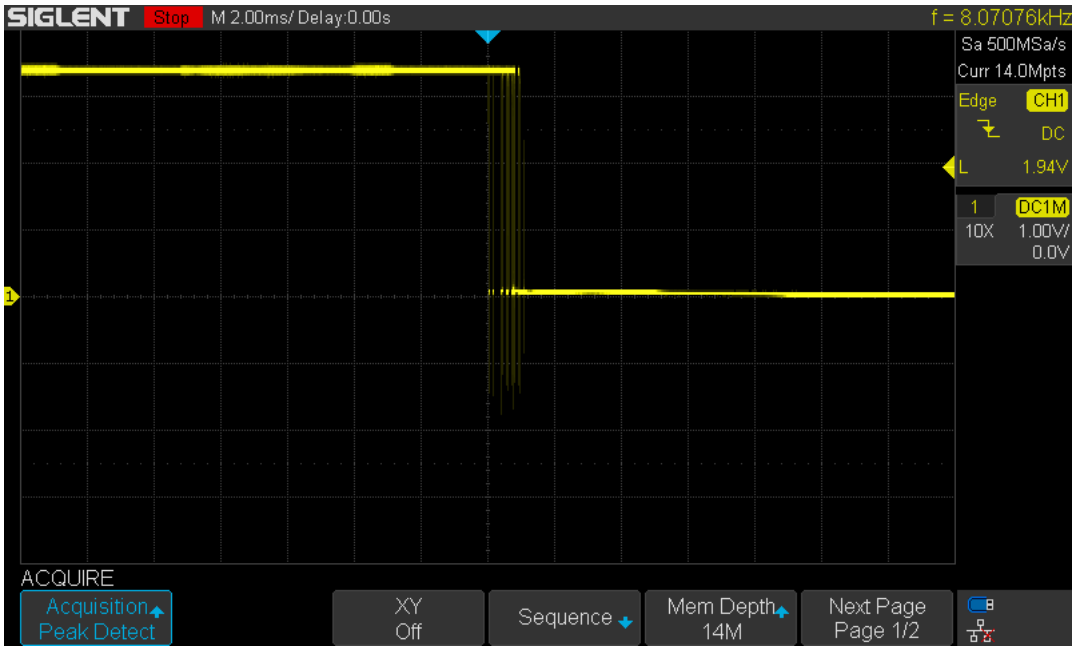

上図はピンヘッダにワニロクリップを噛ませて外部のスイッチを押したケースでもっとも悪い結果となったものです。ひどい回路を作って、中途半端な押し方をするなど、がんばらないとこのような波形は取れませんが、スイッチの付け方次第では電圧が一旦上昇してしまうケースはまだ残っていることがわかります。R9 や C8 を大きくして遅延を伸ばせばもう少し軽減できるかもしれません。




上図は C8 ではなく、GND にスイッチを直結した場合の波形です。横軸が 2ms ではなく 10us になっている点に注意してください。コンデンサがないので急峻な変化をしています。画面中央付近で上下に線が滲んで見えるところがチャタリングで電圧が

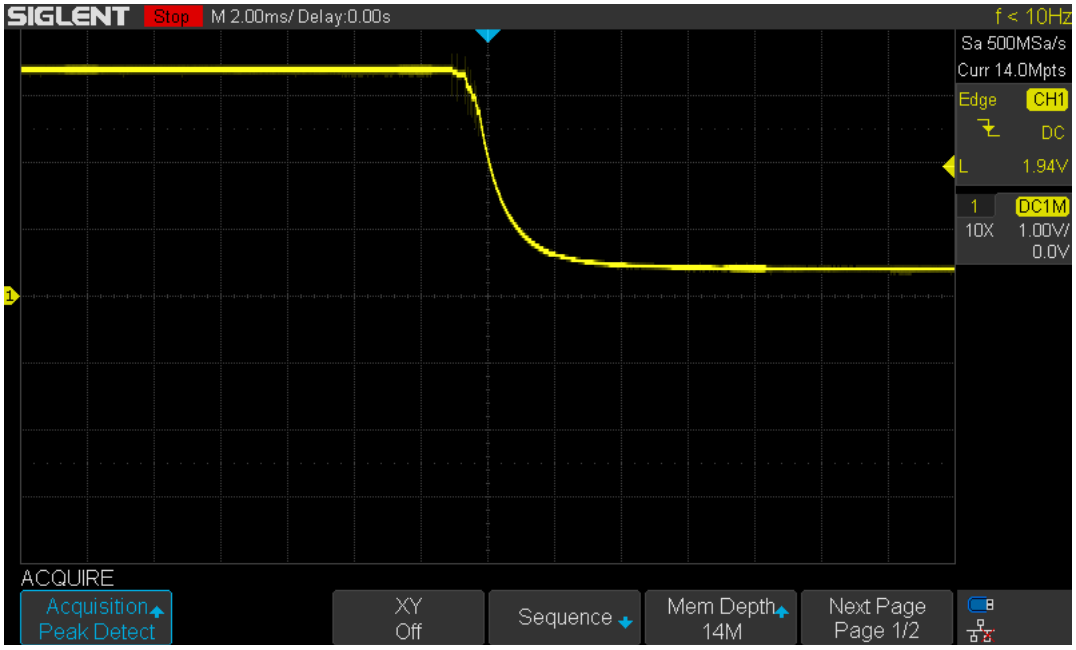
急激に上下しています。以前の C8 経由の波形と比較すると、チャタリング防止回路により電圧の急激な変動が綺麗に取り除けていることがわかります。実験に利用したスイッチは単体でも比較的優秀で、チャタリング防止はなくても動きそうでした。

基板上には低性能スイッチの代表格である機械式リレーが載っていますので、これの波形をみてみます。機械式リレーは磁石でバネに繋がれた接点を引きつけるという構造ですので、接点のバウンドが生じやすくなっています。波形からも少しバウンド感を感じられるのではないでしょうか。




横軸は 2ms です。1ms 程度の期間激しく電圧が上下しているのがわかります。リレーY14H-1C-5DS の仕様では動作時間は 5ms とありますので仕様に対しては十分に余裕のある波形です。ときどき電圧が負まで下がるのはリレー内部のインダクタンス成分(コイル)の効果でしょう。電圧が下がった後もコイルの生み出す電流が流れ続けるため、過剰な電圧降下が発生するものと考えられます。これだけ激しく電圧が上下すると、割り込みが複数回生じる可能性は高くなります。

これをチャタリング防止回路に接続した場合の波形は以下のようになりました。




横軸は同じく 2ms です。よ～くみるとリレーが動作を開始したタイミングでは電圧が揺れていますが、振幅はかなり抑えられていることがわかります。2V を下回るようにはみえませんから、誤検知の心配はほぼないでしょう。電圧が 1V を下回って、GPIO が LOW を検知するころにはリレーの動作はすっかり落ち着いており、綺麗な放電波形になっています。これであれば、リレーを使って複数の基板を連動させることもできそうです。あえて機械式のリレーで通信するような使い道は思いつきませんが、通信を音で感じ取れるという構造は Maker Faire 的にはアリかもしれませんね。40 年くらい前の電話の交換局のような心地よいリレーサウンドを聞けるかもしれません。通信速度は 100bps 程度ですけど、温度センサの値を読むくらいなら問題ないでしょう。

## リレー出力回路

### 概要

リレー出力回路は電気的な接続を避けつつ、外部の機器を ON/OFF するための回路です。リレーとして利用できる素子としては、機械的なスイッチを電磁石で操作する機械式リレーと、半導体の ON/OFF を光信号などで操作する半導体リレー(SSR)が存在します。

性能的には概ね SSR が優れており、消費電力に関してもノイズに関しても動作速度に関しても寿命に関しても SSR を選んでおいたほうが高品質な回路を作れます。秋葉原で買える汎用 SSR としてはパナソニック電工の PhotoMOS リレーAQY シリー

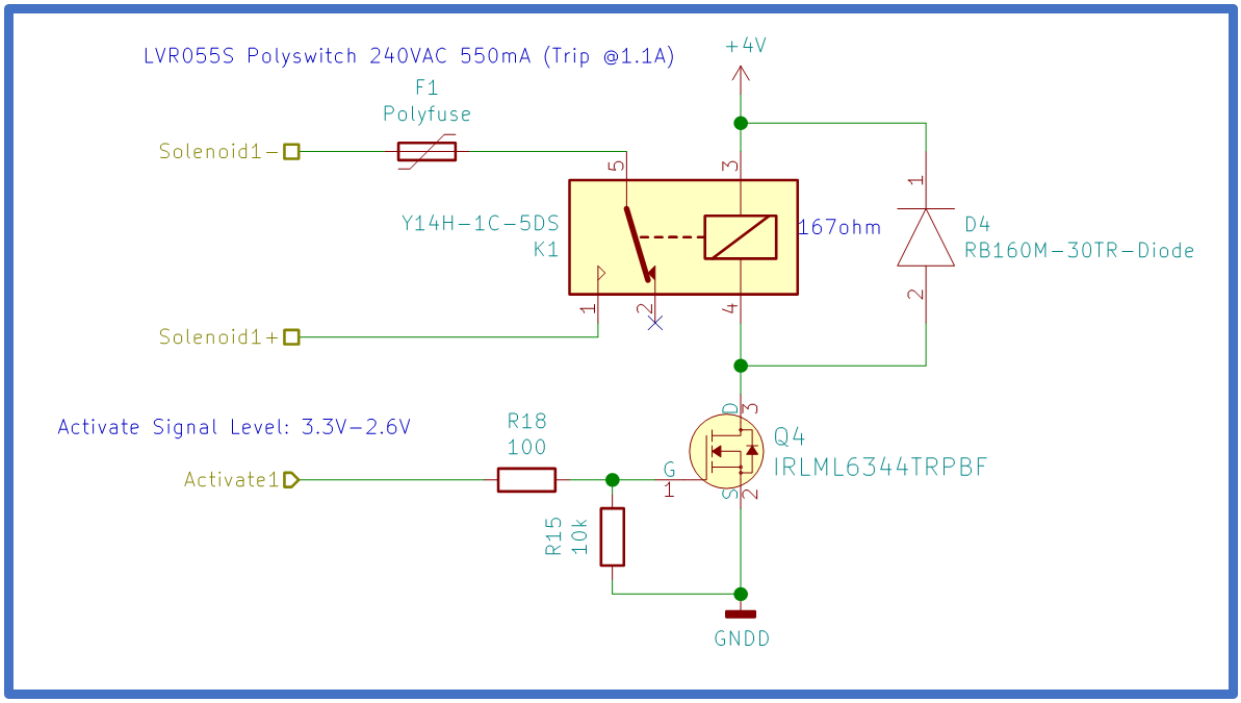
ズなどがあります。SSR には DC 専用や AC 専用という品種も多いので、ここは注意しておく必要があります。

一方の機械式リレーは性能には見るべきものは特にありませんが、出力側のシンプルさが魅力です。単なる機械式のスイッチですから。また、動作するときにカチカチ音がなるのも魅力です。人によりますが。入力側に関しては磁気回路ということもあって必ずしもシンプルとは言えません。

説得力は特にありませんが、今回はカチカチ音を評価して機械式リレーを使いました。設計のポイントとしては、電磁石が十分な力を発揮するためにはそこそこ電流を流す必要がある、電源供給を止めると電磁石は発電機として機能する、出力側はチャタリングが激しい、などの問題への対処が挙げられます。今回の負荷はソレノイドですのでチャタリングは気にしませんが、入力側については気を使う必要があります。

回路図

リレー制御回路は同じものが 3 系統載っていますので、一つを拡大します。




GPIO の出力は Activate1 に入ってきます。この信号は Q4(IRLML6344TRPF, Nch MOSFET)のゲートに接続されています。Q4 はリレーのコイル電流をグランド側で ON/OFF するローサイドスイッチとして利用しています。スレッショルド電圧 Vgs(th)は最大で 1.1V ですから、Activate1 が HIGH(約 3V)を出力すると確実に ON にできま

す。Vgs が 3V だと ON 抵抗 Rds は完全には下がりきらないのですが、データシートをみると 40mΩ以下にはなりそうです。今回はリレーK1(Y14H-1C-5DS, 5V 機械式リレー)の制御用コイルが 167Ωの抵抗を内蔵していますので Q4 の Rds は誤差の範囲といえます。

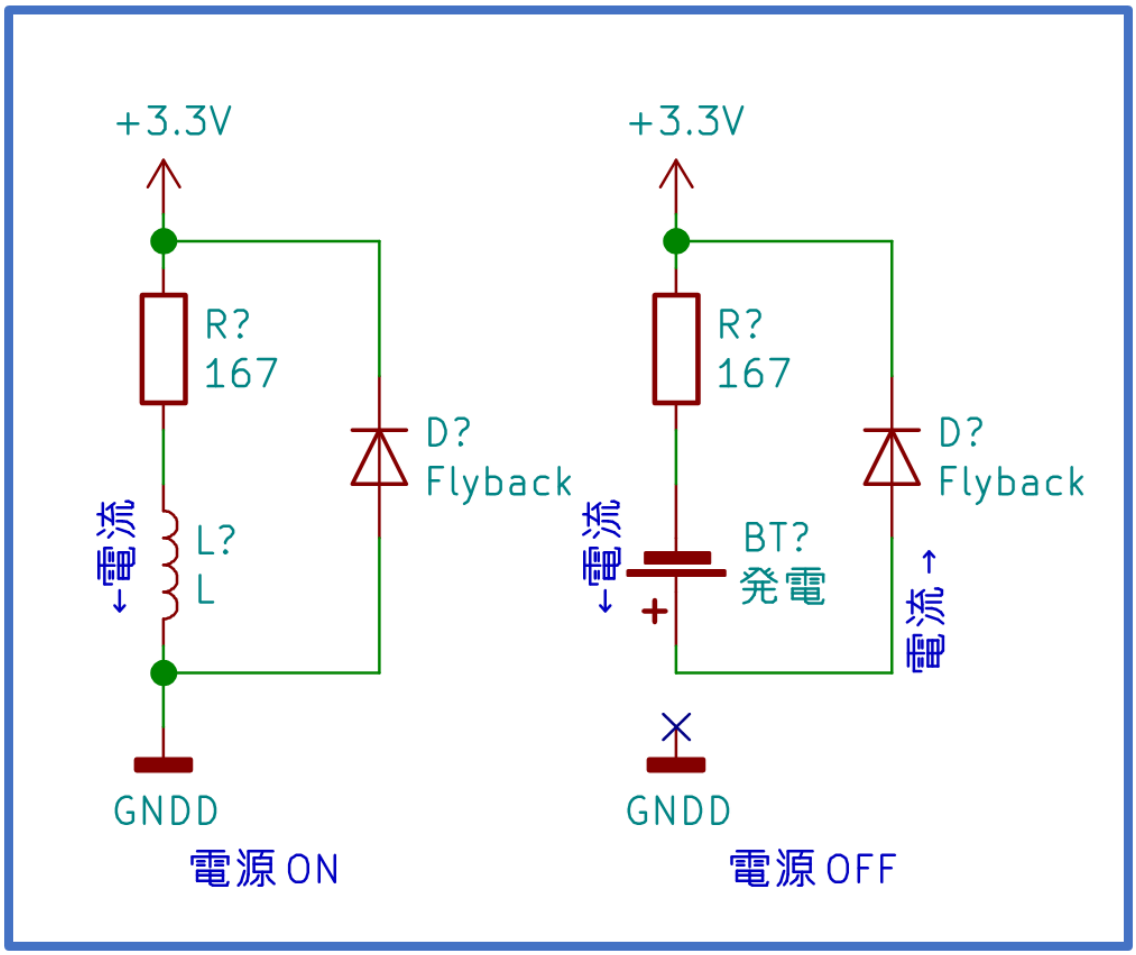
リレーの出力側に入っているヒューズ F1(LVR055S, Polyswitch)は電流制限用のヒューズです。高分子ポリマーに導電性の粉末を混ぜたポリヒューズという部品で、作動して OFF になったとしても負荷をはずしておけば自然と元に戻るという性質があります。原理としては電流が大きくなって温度があがるとポリマーが膨張して粉末の密度が下がり電流が流れにくくなるというものです。今回は 1.1A 流れると制限がかかる部品を採用しました。一旦制限がかかると電流が 550mA を切るまでは制限がかかり続けます。リレーK1 の定格が最大 1A で、そこまではぎりぎり耐えますが、1A を 10%程度超えたところで強制的に電流が遮断される仕組みです。基板としては安定して利用できるのは 500mA までという仕様にしてシルク印刷を施してあります。

R18 は Q4 のゲートのスイッチング速度を制限し、高周波ノイズの定在波を減衰させるためのゲート抵抗です。今回は定番の値として 100Ωを使っています。K1 のスイッチング速度が 5ms ですので、Q4 のスイッチング速度が問題になることはありません。

R15 は Q4 のゲートに溜まった電荷をグランドに捨てるための抵抗でこちらも定番の 10kΩを利用しています。この回路では Activate1 が GPIO の出力になりますので、通常は GPIO が LOW を出力しているのであれば電荷は GPIO 側に吸われていきます。しかし、再起動などで GPIO がハイインピーダンス状態になっている間は電荷の行き場がなくなってしまいますので R15 で確実にグランドに落として Q4 の動作を安定させるようにしています。

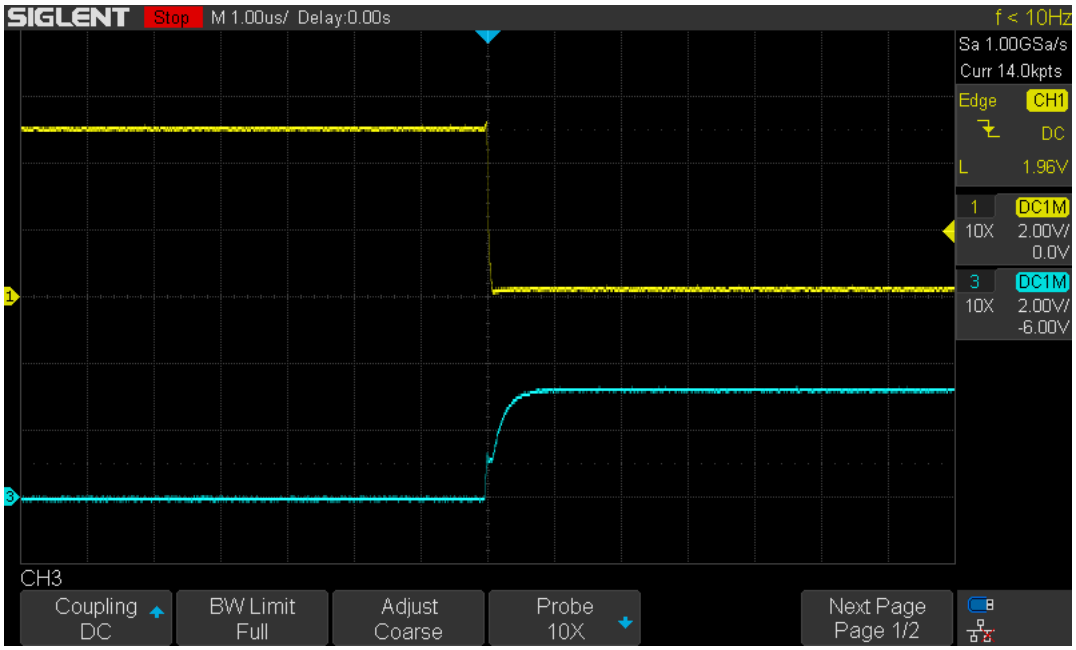
R15 にはもう一つ、ESP-WROOM-02 の GPIO の pull down という役割もあります。特に重要なのは IO15 に接続されている Activate3 です。IO15 はブートセレクタとして使われるピンの一つで、起動時には LOW にしておく必要があります。

D4(RB160M-30TR, ショットキーバリアダイオード)は、K1 のコイルへの電源供給を停止した際に働くダイオードです。下図に K1 のコイル部分の動きを示します。



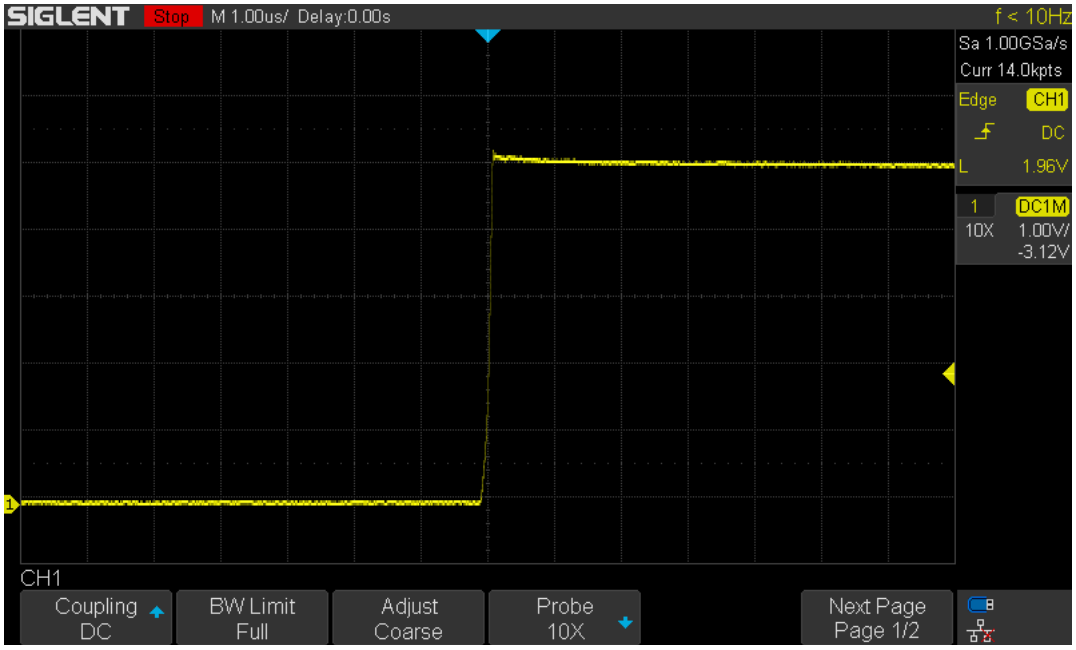

Q4 が ON の場合、コイル L には電流が流れ磁界が発生します。これによりリレーが動作します。Q4 を OFF にするとコイル L には電流が流れなくなります。このとき L 周辺の磁界も減少していきますが、これにより L は発電機として動作して電流を維持(磁界の減少に抵抗)するように働きます。ここで電流を流してあげないと、発電機の出力を開放にしている状態となり理論上無限に電圧が上昇してしまいます。機械式のスイッチであればまだしも、今回の回路では、この電圧が Q4 のドレインにかかることになり、故障やノイズの原因となります。ここで活躍するのが D4 で、コイルの両端をつないで電流を流すことで磁界のエネルギーを消費し、過電圧を防いでいます。このダイオードの呼び方は定まっていないようで、フライホイール・ダイオード、フリーホイール・ダイオード、フライバック・ダイオードという表記をよく見かけます。文脈により使い分けるべきなのかもしれませんが、よくわかりません。電源回路に関してはフライバック電圧とフライバック・ダイオードという使い方が多いようです。

波形




上図のチャンネル 1(黄色)は Q4 のドレインの電圧を観測した波形です。横軸は 1us で短時間の波形をみています。縦軸は 2V になっています。リレーは 5V 系の電源で動作していますので電圧は高めです。電源電圧は 4.5V 前後となっていますが、これは 5V の電源系に逆流防止用のダイオードが入っており、その分だけ 5V から電圧降下しているためです。FET はグランド側(ローサイド)のスイッチとなっていますので、4.5V であればリレーに電流は流れず、0V に落ちるとリレーに電流が流れるという動作をします。

チャンネル 3(水色)は同じく Q4 のゲート電圧をみています。Q4 の入力容量によりコンデンサの充電波形になっていることがわかります。ただし、Q4 のデータシート上の入力容量は 650pF に対してオシロスコープのプローブの容量は 16pF ですので、測定環境の影響を無視できません。参考程度と考えてください。ゲート側の電圧に変曲点があるのは、FET が ON になりドレイン側の電圧が降下することで、FET の入力容量の一つであるゲート-ドレイン間容量の充電状態が変化するためではないかと思います。とはいえ、測定環境の影響も考えられ、はっきりとしたことはわかりません。ドレイン側は ON のタイミングでわずかにオーバーシュート、アンダーシュートが見えますが問題になる大きさではなさそうです。




上図は逆にリレーを OFF にする際のドレイン電圧を観測した波形です。縦軸を拡大しています。大きなフライバック電圧は見当たりませんので、リレーの ON/OFF を繰り返しても FET にダメージが入ることはなさそうです。

## USB シリアル回路

### 概要

USB 端子はプログラムの書き換え、デバッグ出力、電源供給など幅広く使える端子です。

プログラムの書き換えやデバッグ出力については、Wi-Fi を使うこともできますし、ESP-WROOM-02 の UART 出力を取り出して外付けの 3.3V 対応 USB シリアル変換器を使うこともできます。基板上にレベルコンバータを設置することで汎用の RS-232C(EIA-232)接続ポートを使えるようにするという選択肢もあります。昨今は RS-232C 搭載のコンピュータは少なくなってきたので汎用とは言い難い状況ですが・・・。

電源供給については、もともと 9V 電池を使える設計ですので、代わりに 9V 出力の AC/DC アダプタを接続することも可能です。

色々と選択肢はあるのですが、USB にまとめてしまったほうが開発の効率は確実に良くなりますので USB シリアル接続対応とすることにしています。

## 自動リセット機構

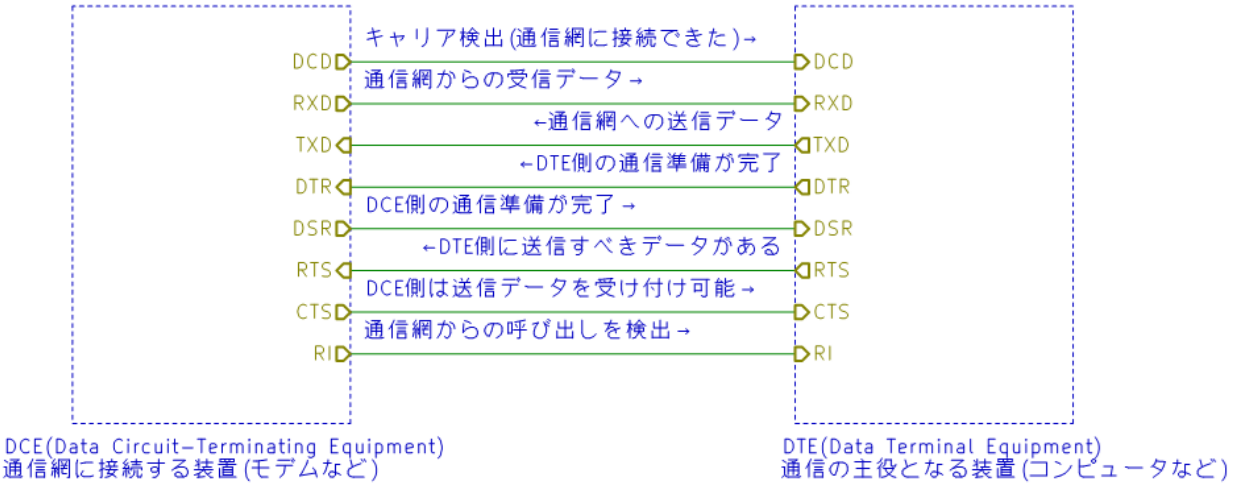
普通に UART 出力がやりとりできて、電源を供給できれば最低限の機能は果たすのですが、Arduino IDE によるプログラムの自動更新ができると開発効率が飛躍的に向上します。

ESP-WROOM-02 のプログラムを更新する手順は、以下の通りです。

1. IO0 を LOW レベルに落とす
2. リセットをかけて UART 起動の処理を開始させる
3. IO0 を HIGH レベルに戻す
4. プログラムを送信して SPI フラッシュに書き込みさせる
5. リセットをかけて書き込んだプログラムを起動する

簡易開発キットなどはディップスイッチやタクトスイッチとして IO0 と RST を操作できるように作られています。これだと毎回手作業が必要になり不便です。ESP-WROOM-02 のコミュニティでは、この作業を自動化するために RS-232C の制御線を利用した様々な黒魔術的技法が開発されています。RS-232C は古来より様々な黒魔術の舞台となってきた通信ポートで、古い技術者にとってはちょっと懐かしい話だと思います。

まず、本来の非同期通信ポートとしての RS-232C の構成をおさらいしておきます。

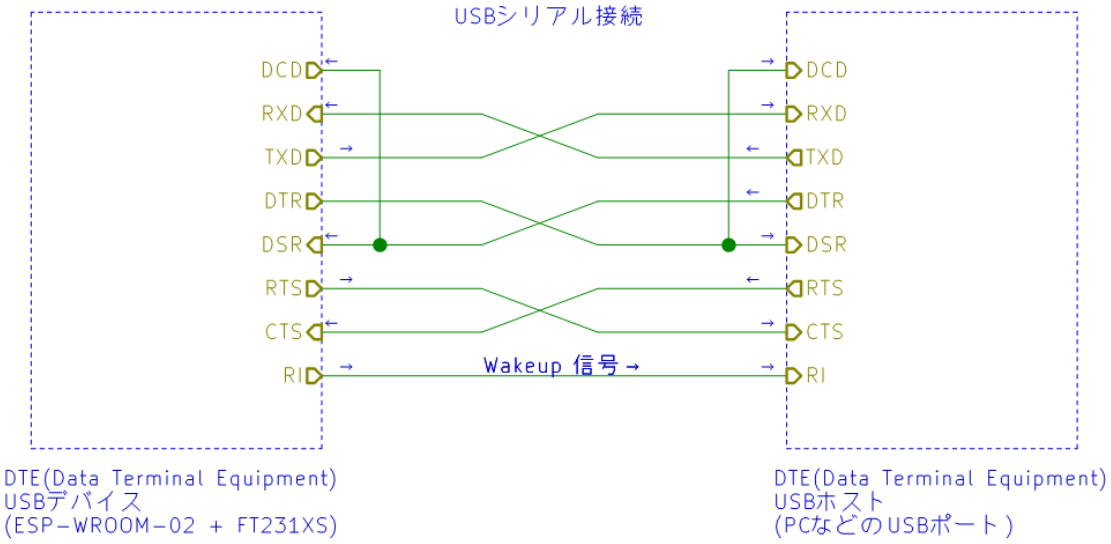



RS-232C は通信網に接続するためのポートで、非対称な制御信号を持っています。例えば、DTE は RTS で DCE に対してデータ送信の許可を求め、DCE は自身の受信バッファの空きや通信網側の輻輳状態などを確かめてから CTS により DTE に対してデータ送信の許可を与えますが、逆はできません。DTE 側の処理が溢れてい

たとしても、網側からは容赦無くデータが届けられます。したがって DTE 側では受信バッファ(FIFO)の管理を手厚くしてバッファリングでがんばるのが基本です。網からの呼び出し(着呼)を通知する RI も同様に非対称です。DTE は RI を検知すると着呼処理を実行します。要はベルがなったら受話器を取れ、という話です。対となる発呼の処理は専用線接続でない限りは一本の制御信号でできることはあまりありません。特に制御信号は使わず、最初から AT コマンドで発呼処理を実行します。

ESP-WROOM-02 は購入時点では AT コマンドで制御するモデムとして動くようにプログラムされています。この場合、ESP-WROOM-02 は(配線は異なるのですが)DCE に近い動作をしているとみなすことができます。通信網とは Wi-Fi のセグメントであり、キャリア検出とはアクセスポイントとステーション間の接続状態の検出であると言えるでしょう。・・・と言っても ESP-WROOM-02 の UART は RXD,TXD,RTS,CTS の 4 本しかありませんし、AT コマンドファームウェアは制御線の扱いはあまり気にしていないようです。

プログラムを書き換える際には、ESP-WROOM-02 は通信網に接続するための DCE ではなく、データを処理する DTE として動作します。この時点で RS-232C 本来の使い方ではなくなり、標準規格からは外れます。制御線の意味も読み換えることになるのですが、標準は存在しません。代表的な読み替え方法にしたがって通信路を作ると以下のようになります。いわゆる RS-232C クロスケーブルという結線の一つです。




RXD と TXD はそれぞれの DTE にとっての受信と送信を意味します。したがって配線はクロスします。DTR と DSR お互いの通信準備を確認できれば良いので、ここもク

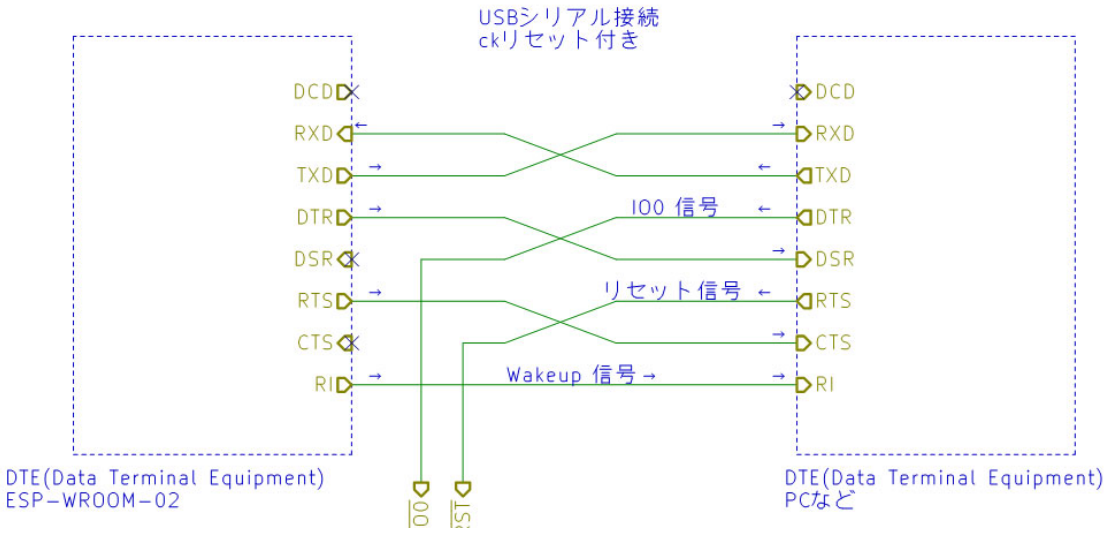
ロスにします。通信網は存在しないので DCD の定義は曖昧ですが、DSR と同じ意味とすることが多いようです。他にも DTE の電源状態を反映すると考えることもできますし、(検知できるのであれば)ケーブルの接続状態を反映すると考えることもできますね。RTS は自身の受信バッファに空きがあることを対向機器に示し、CTS は対向機器の受信バッファに空きがあることを検出します。この RTS の読み換えは危うく、かつてはケーブルごと、ソフトウェアごとに考え方が違っていて動くかどうかは運次第でした。別の流儀としてバッファの空きを通知するために DTR と DSR を利用することもあります。現在では端末ソフトでハードウェアフロー制御 OFF を指定して DTR や RTS の動作は気にしないという使い方が普通だと思います。

RI にはあまり使い道がありませんが、FTDI の場合は USB デバイスからスリープ状態の PC を復帰させるための信号線として利用できます。また、GPS 受信機や JJY 受信機で時刻を送信する際に、TXD で「ただいまより、何時何分をお知らせします」という情報を流してから、実際にその時刻になったタイミングで RI や DCD を操作して割り込みをかけることで高精度に時刻を知らせるという使い方をされることもあります。このような用途では GPIO と接続しておく必要があるかもしれません。

さて、ESP-WROOM-02 の IO0 と RST の自動制御方法を考えていたハッカーたちは、どうせ使わないであろう DTR/RTS/CTS に着目し、これらの信号線を駆使する様々な手段を編み出していきます。GitHub で公開されている esptool-ck というプログラム書き込みツールで主要なアイデアが実装されています。

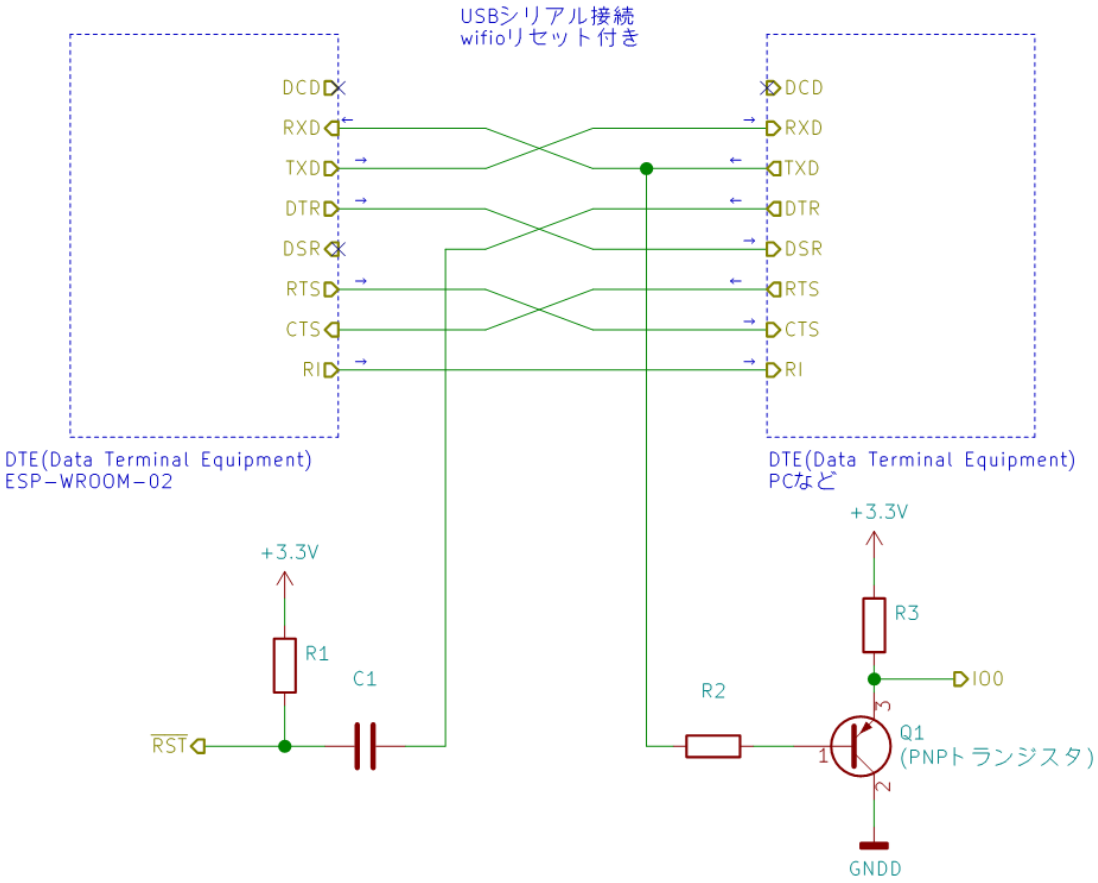
1. ck … RTS を RESET 端子に接続し、DTR を IO0 端子に接続する
2. wifio …DTR を RESET 端子に接続し、TXD の電圧を IO 端子にも反映させる
3. nodemcu … RTS と DTR の xor をとって、RESET 端子と IO0 端子を駆動する

もっともシンプルな案は ck 方式です。




DTR を HIGH にすると IO0 が LOW になり、この状態で RTS を HIGH にすれば RESET が LOW になってリセットがかかり、RTS を LOW に戻すと ESP-WROOM-02 は UART モードで起動します。FTDI だともともと DTR ピンも RTS ピンも負論理なので、接続は大変簡単です。問題は端末制御プログラムによっては起動時に DTR と RTS を HIGH に初期化することがあり(気持ちはよくわかります)、実際に意図せずリセット状態に移行してしまう事故が起こりがちだという報告がありました。なにより Arduino IDE のシリアルモニタでこれが発生するという話があり、問題は大きかったようです。

そこで考案されたのが wifio 方式です。

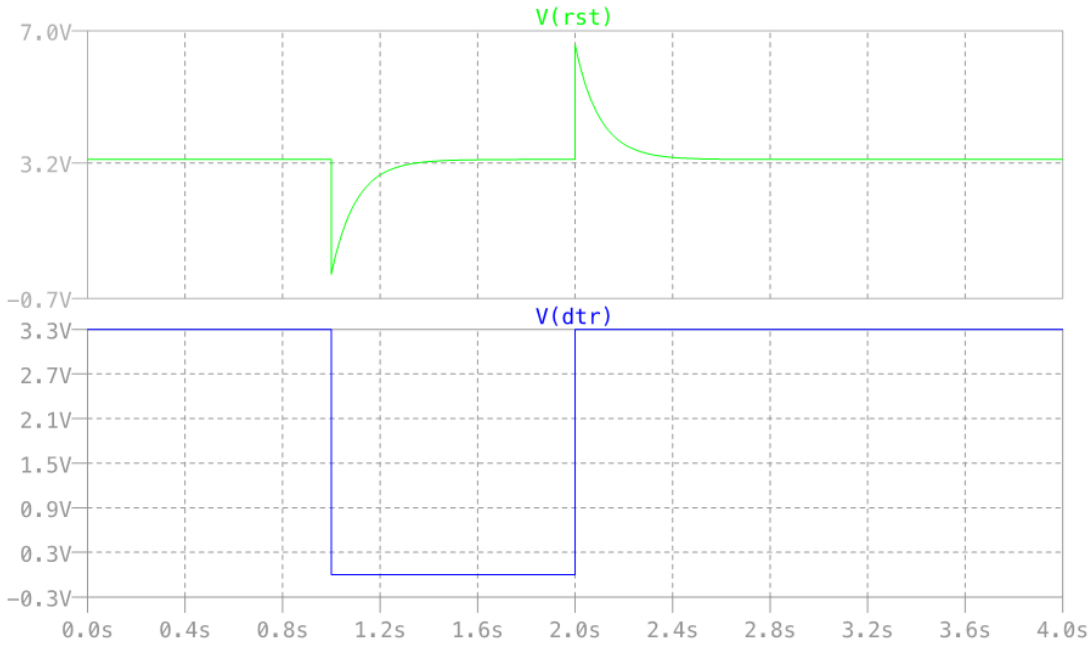



TXD を break 状態(LOW で固定)にしておくと PNP トランジスタのベースから R2 を経由して TXD へ電流が流れ込みます。これによりトランジスタは ON となり、R3 には電流が流れ電圧降下が生じます。この結果、IO0 は GNDD 近くまで電圧が下がり LOW になります。厳密にはトランジスタの Vce によりますが、LOW の範囲内でしょう。

DTR を HIGH(3.3V)にしておくと、コンデンサの両端が 3.3V となりますから、コンデンサは充電されません。$\overline{\text{RST}}$への入力も 3.3V となりますからリセットは解除状態となります。ここで、DTR を LOW に落とすと、コンデンサは R1 を通じて充電されます。この充電電流により R1 は電圧降下をおこし、充電完了までの間は$\overline{\text{RST}}$が LOW レベルとなります。このパルスによりリセットがかかります。コンデンサの充電が完了すれば$\overline{\text{RST}}$は HIGH レベルに戻ります。DTR をせわしなく上げ下げする必要はありません。リセットがかかるのは、レベルが変動するタイミングだけなので、ck 方式のようにリセット状態が維持されてしまうこともありません。この回路は一般的には CR 微分回路と呼ばれています。CR 微分回路は本家 Arduino のリセットでも使われているようですね。

PNP トランジスタ 1 個、コンデンサ 1 個、抵抗 3 個で回路が構成できるというシンプルさと、CR 微分回路を利用してパルスを作るというアイデアが光るかっこいい回路です。break 信号を使って TXD のレベルを固定するという RS-232C をよく知らなければ出てこない発想もかなりイカしています。DTR を LOW から HIGH に戻すと充電されたコンデンサが電圧を過剰に持ち上げてしまい、過電圧パルスが生じそうなところは気になるのですが・・・この回路はとても好きです。

下図に wifio 回路のリセット波形を示します。定数として R1 に 10kΩ、C1 に 10uF を入れて LTspice でシミュレーションした結果です。DTR を 1 秒間 LOW レベルに下げる制御をしています。RST の電圧が一瞬だけ低下してすぐに元に戻ること、DTR が HIGH に戻るタイミングで過電圧パルスが生じることがわかります。過電圧パルスを軽減する方法として、放電用のダイオードを入れる方法などが考えられます。




最後が nodemcu 方式です。この方式では、RTS と DTR のどちらか片方だけが HIGH になるケースは稀だ、という仮定を置いています。特に DTR が LOW であるにもかかわらず、RTS が HIGH になるという状況には無理があります。DTE-DCE 間の接続で通信準備ができていないのに送信要求を出してくるのは不条理ですし、DTE-DTE 接続で通信準備ができていないのにデータの受け入れが可能なわけがありません。そこで nodemcu 方式では、DTR が LOW かつ RTS が HIGH のとき RESET ピンを LOW にしてリセットをかけることにしています。これでリセットの誤作動の可能性はかなり減少します。逆に DTR が HIGH かつ RTS が LOW のきには IO0 を LOW にしま

す。この条件は、DTE-DCE 間接続を前提にした制御が有効になっていたり、DTE-DTE 間でハードウェアフロー制御が有効になっていたりすると普通に起こり得ます。とはいえ、昨今ではあまり見かけない条件ではありますし、IO0 のレベルが変わるだけであればリセットよりは実害は少ないと言えます。問題点としては厳密には IO0 を LOW にしつつリセットをかけるという動作ができないので、確実に UART モードで起動するとは言えません。esptool-ck ではリセットをかけた直後に IO0 を LOW にするという手順を踏んでいます。ESP-WROOM-02 の初期化が終わるより早くこの手順を実行できれば UART モードで起動するだろうということですが、実環境では問題なく動作するようです。今回はこの方式を採用していますので回路は後ほど紹介します。

Arduino IDE には他に none(制御線には触らない)と dtrset(DTR を HIGH にする)という選択肢があります。dtrset 用の回路は探し出せなかったのですが、wifio 方式のリセット部分だけ作っておいて、リセットを自動でかけられるだけでも多少楽になるという意味合いかもしれません。他の Aruduino ボードはリセットのみの自動化でなんとかなるものも多いです。ESP-WROOM-02 もブートローダの作り方次第では IO0 を使わなくても、起動時に特定のメッセージを送ることで書き込みモードに切り替える処理は可能だと思います。

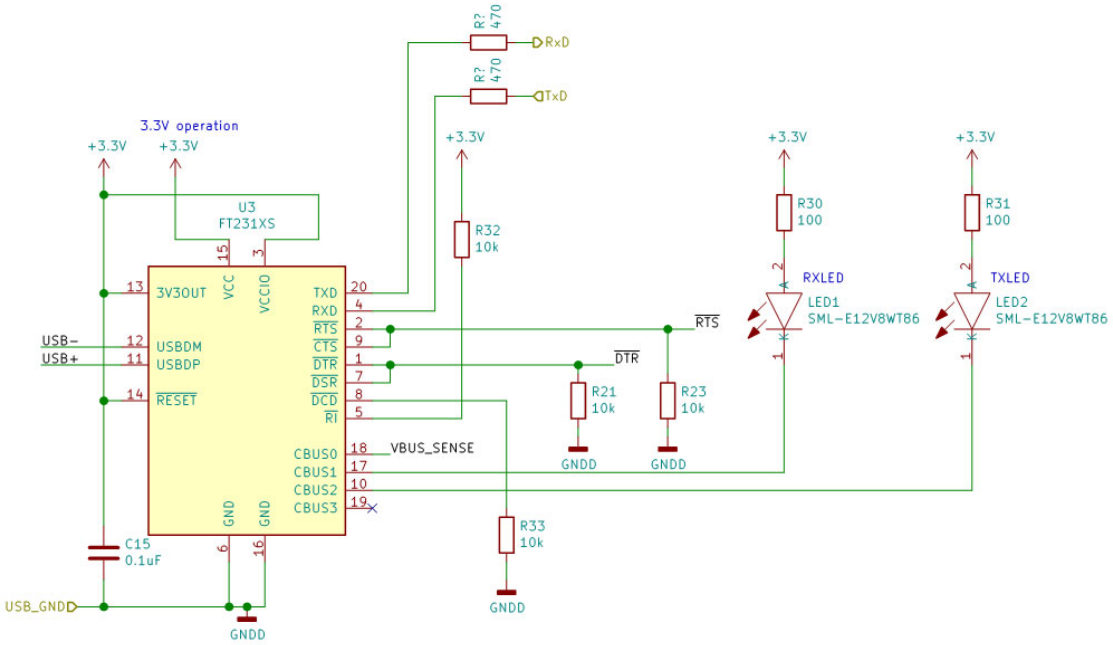
## チップ選定

USB シリアル変換 IC の供給元は、FTDI、SILICON LABS、WCH といったメーカがメジャーです。Espressif はかつて FTDI の FT232 を推奨しており（ESP8266 System Description v1.4, 2016, p. 25)、実績としても FTDI 製品は信用がおけますので今回は FTDI の USB シリアル変換 IC を選択しました。

FTDI といっても幅広い品種がありますが、半田付けの手間を考えると可能な限りピン数が少ないほうがブリッジなどのトラブルが少なく、自動リセット機構の利用を考えると DTR と RTS を扱える必要があります。ピン数からすると FT230/FT234 系統は魅力ですが、DTR の制御ができません。FT231 であれば DTR の制御ができつつ、ピン数も抑えられます。FT232 は高機能ですが、28 ピンとピン数が多く、今回の用途にはオーバースペックでしょう。ということで、秋月電子で売っている FT231XS の 20 ピン SSOP パッケージを利用することとしました。

## 回路図

下図に FT231XS 周辺の回路の拡大を示します。




電源としては 3.3V を利用しています。この場合、主電源である VCC と I/O 電源の VCCIO に 3.3V を供給するのに加え、内蔵 LDO の出力である 3V3OUT にも 3.3V を供給する必要があります。この 3 つが電源ピンです。リセット端子は利用しないので 3.3V に固定しています。データシートのリファレンス回路では VCCIO、3V3OUT、RESET の各ピンには C15(積層セラミックコンデンサ, 0.1uF)のパスコンを入れているのでそれに従っています。VCC については電源回路側にコンデンサを並べてあるのでよしとしています。本来は 0.1uF と 4.7uF を FT231XS の近くに配置しておくべきだったかもしれません。

TXD と RXD は方向がややこしいですが、FT231XS からみた TXD と RXD となりますので、FT231XS の TXD と ESP-WROOM-02 の RXD を接続し、FT231XS の RXD と ESP-WROOM-02 の TXD を接続することになります。

ESP-WROOM-02 と FT231XS の間には R34、R35 の 2 本の 470Ωの保護抵抗を入れています。保護抵抗はなくても、FT231XS と ESP-WROOM-02 が入力対出力の関係を保っている限りさほどの問題はおこらないと思います。ただし、FT231XS の TXD と ESP-WROOM-02 の RXD の関係は崩れる可能性があります。ESP-WROOM-02 の RXD は GPIO としても制御でき、OUTPUT 指定ができてしまいます。この指定をされてしまうと出力同士が衝突します。FT231XS が 3.3V 出力、ESP-WROOM-02 が 0V 出力、あるいはその逆になると、電圧を上げようとする動きと下げようとする動きが衝突して際限なく電流が流れてしまい過電流による誤動作か破壊が生じるでしょう。470

Ωの抵抗が入っている場合、抵抗による電圧降下が生じますので、7mA の電流が流れた時点で HIGH 側が 3.3V、LOW 側が 0V で安定します。ESP-WROOM-02 は 12mA、FT231XS は 22mA が出力電流に関する絶対最大定格です。保護抵抗が入っていれば、誤動作は誤動作ですがプログラミングエラーでハードウェアにダメージがいくことがなくなります。抵抗を大きくすると電流を抑えることも可能ですが、ノイズの混入による電圧の上下が大きくなってしまいますのであまり大きな値にはできません。他の設計を眺めると 470Ωを選択していることが多いようですので、素直に実績のある値を採用しています。FT231XS の RXD に関しては出力に設定することはできませんので保護抵抗は必ずしも必要ありませんが、ノイズを減衰させられるかも？という程度の意味合いで同じ抵抗を入れています。

ノイズの話はデータシートに由来するもので、ESP8266EX では ESP8266EX からみた TXD (FT231XS の RXD)に 499Ωの抵抗を挿入するように指定されています (ESP8266 Hardware Degisn Guidelines v2.4, 2018, p. 10)。80MHz の高調波を抑制するため、と書かれていますがいまいちよくわかりません。CPU クロックが 40MHz だと 80MHz の高調波ノイズは発生するだろうとは思いますが、なぜ UART の TXD でそれをことさら気にする必要があり、なぜ 499Ωが適しているのか。事情があるのだと思うのですが、調べきれませんでした。499Ωは特殊な値なので 470Ωでも概ね問題ないだろうと都合よく考えていますが、どうなのでしょうね？470Ωで良いなら最初から 470Ω を指定するところだろうという気もします。

RTS と CTS は基板上でつなげています。USB ホストが RTS を出力すると、FT231XS は即座に CTS を出力する仕組みです。これによりハードウェアフロー制御は事実上無効となります。ハードウェアフロー制御は役に立つことよりトラブルを招くことのほうが多い気がするのでこのような仕組みにしています。DTR と DSR の関係も同様です。真面目な端末ソフトは DSR を検出するまでは通信を開始しませんが、デバッグ目的ではそんなことは無視して TXD/RXD に流れている情報をみたいことのほうが多いので制御を無効化しています。

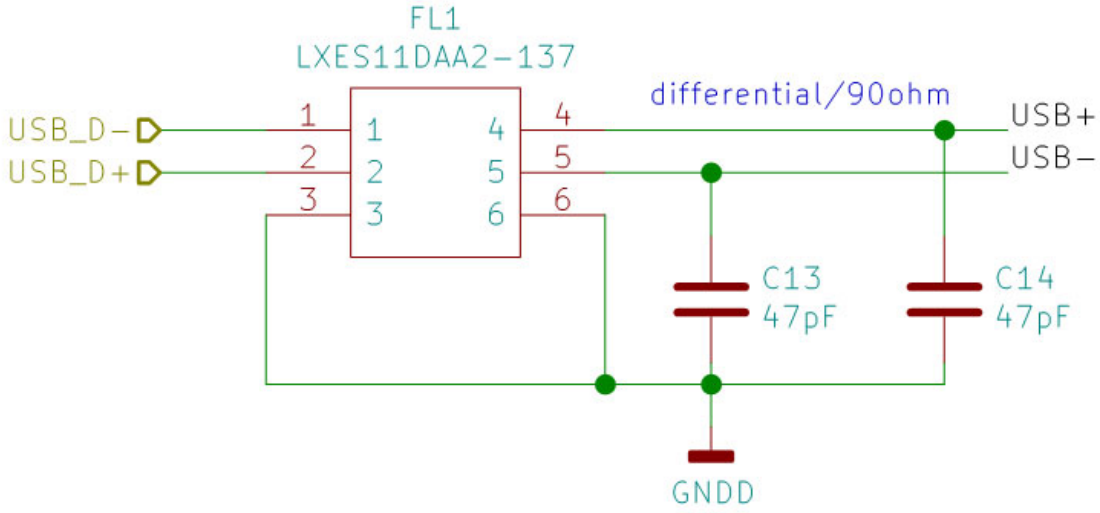
RTS と DTR は R21、R23 の 10kΩで pull down しています。これにより、FT231XS の起動が完了するまでは LOW レベルに固定されます。これは FT231XS の起動中に後述するリセット回路を安定させておくための措置です。

RI は USB ホストにスリープからの復帰を要求するために利用できますが、このために GPIO を一つ使うのもなんなので R32 で pull up して HIGH に固定しています。PC の周辺機器として利用するのであれば GPIO に接続しておくのも一案です。

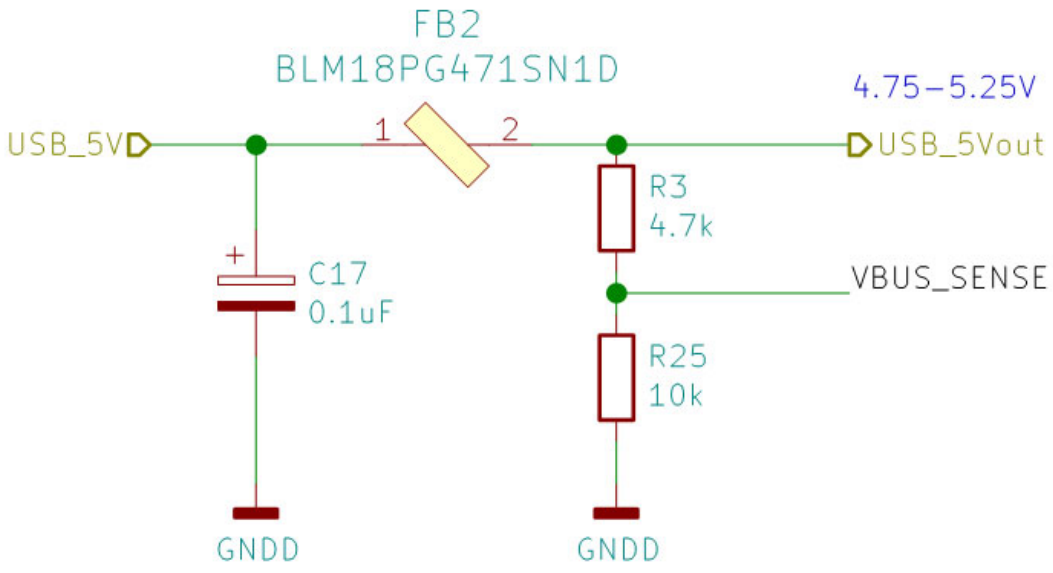
CBUS は FTDI の IC では定番のピンで、ユーザが用途を設定できるピンとなっています。設定は FT_PROG というソフトウェアを利用して USB 経由でおこないます。今回は TX と RX に反応する LED と、USB の電源状態を監視する VBUS_SENSE を利用しています。LED は ESP-WROOM-02 からみた TX と RX を表示します。USB シリアルケーブルとは逆の動作です。VBUS_SENSE を利用すると USB ケーブルが抜けている際にスリープ状態に入って消費電力を削減できます。

LED1(赤色 LED, SML-E12V8WT86)は電圧効果が 2.2V から 2.7V 程度の LED となります。電源電圧が 3.3V ですから、差し引き 0.6V から 1.1V が R30,R31 の 100Ωでの電圧降下分となり電流値は 6mA から 11mA という計算になります。20mA で最大の輝度を発揮する品種ですが、6mA で 40%程度、11mA で 60%程度の輝度を発揮できますので動作の目視確認には十分な明るさです。

下図に EMI 対策関連の回路を示します。




USB Data filter block w/ surge protection




USB VBUS filter block

USB+(USBDP)と USB−(USBDM)は差動信号の配線です。今回の基板では最も高速・高周波の信号が流れますので配線長は可能な限り短くする必要があります。リファレンス回路では、ノイズ減衰と過電流保護用と思われる 27 Ωの抵抗を入れています。これは特殊な抵抗値ですし、効果のほどもよくわからないなぁ、と思いつつ秋月電子の開発ボードやスイッチサイエンスの開発ボードの回路図を眺めてみると、秋月電子は村田製作所のコモンモードノイズフィルタと ESD フィルタを組み合わせたような素子を使っています。スイッチサイエンスは何も入れていません。ここは秋月電子に従っ

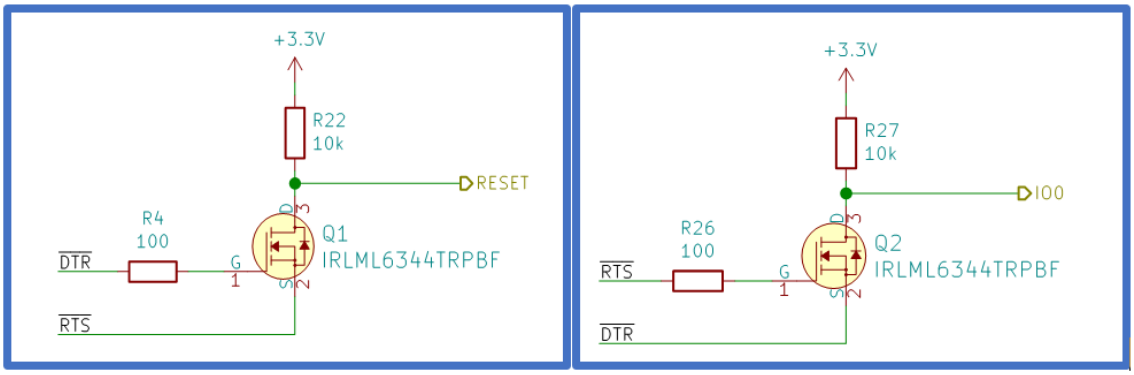
て、FL1(コモンモード ESD フィルタ, LXES11DAA2)だけを入れておくことにしました。LXES11 は特性インピーダンス 60Ωです。USB の特性インピーダンスは 90Ωですので、90Ωに対応する LXES21 を使う必要がありそうなところなのですが、売ってなさそうなので LXES11 を利用しています。正直、あってもなくても良いような部品ではありますが、手に入るのであれば入れておこうかという程度の選択です。この部品があるからといって、基板の動作に大きな改善はないでしょう。どちらかというと、自作のいい加減な基板が USB ホストとなる PC に悪さをするのを少しでも軽減するという意味のほうが強い部品です。EMI/ESD 対策部品は基本的に自分を守るというよりも、対向機器に迷惑をかけないという紳士の部品です。

電源ラインに入っている FB2(フェライトビーズ、BLM18PG471SN1D)も同じように EMI 対策の部品です。損失の多いコイルを使って電源ライン上に流れる交流成分の出入りを制限します。C17(積層セラミックコンデンサ、0.1uF)は FB2 とは逆に交流成分を積極的にグランドに流して電源ラインから排除しようという部品です。これも USB ホストに迷惑をかけないという意味合いのほうが強い部品です。

R3 の 4.7kΩと R25 の 10kΩは USB の電源電圧を分圧して FT231XS の VBUS_SENSE に入力します。これはリファレンス回路の値をそのまま採用しています。FT231XS はこの電圧が低いと USB ケーブルがつながっていないと判断して省電力動作に移行します。

## Nodemcu リセット回路

下図に nodemcu 互換のリセット回路を示します。




Q1(Nch MOSFET, IRLML6344TRPBF)と Q2(同)はスイッチとして働きます。ポイントは、ドレインは 3.3V に固定されていますが、ソースは RTS または DTR に接続されており 0V または 3.3V のどちらにもなりうるという点です。

ソース側が 3.3V になっていると、ドレインもソースも同じ電圧です。Q1、Q2 のスイッチが ON でも OFF でも電流は流れません。したがって、R22、R27 にも電流は流れず、電圧降下は発生しません。この場合出力は 3.3V(HIGH)となります。

ソース側が 0V の場合は、Q1、Q2 が ON か OFF かで出力が変わります。ON の場合、ドレインからソースに向かって電流が流れます。このとき、R22、R27 では 3.3V 分の電圧降下が発生しますので出力は 0V(LOW)となります。電流値としては 0.3mA です。OFF であれば電流は流れませんので出力は 3.3V(HIGH)です。

以上をまとめると、Q1 のソースが 0V、すなわち$\overline{\mathrm{RTS}}$が LOW(負論理なので信号としては HIGH)の場合にのみ、RESET は LOW になる可能性があります。実際に RESET が LOW になるのは Q1 が ON になる必要があり、これは$\overline{\mathrm{DTR}}$が HIGH(負論理なので信号としては LOW)の場合です。つまり、RTS が HIGH で DTR が LOW の場合のみ、RESET が LOW となり ESP-WROOM-02 にはリセットがかかります。他の条件では RESET は HIGH のままです。Q2 は信号が逆に接続されていますので、RTS が LOW で DTR が HIGH の場合のみ IO0 が LOW となります。

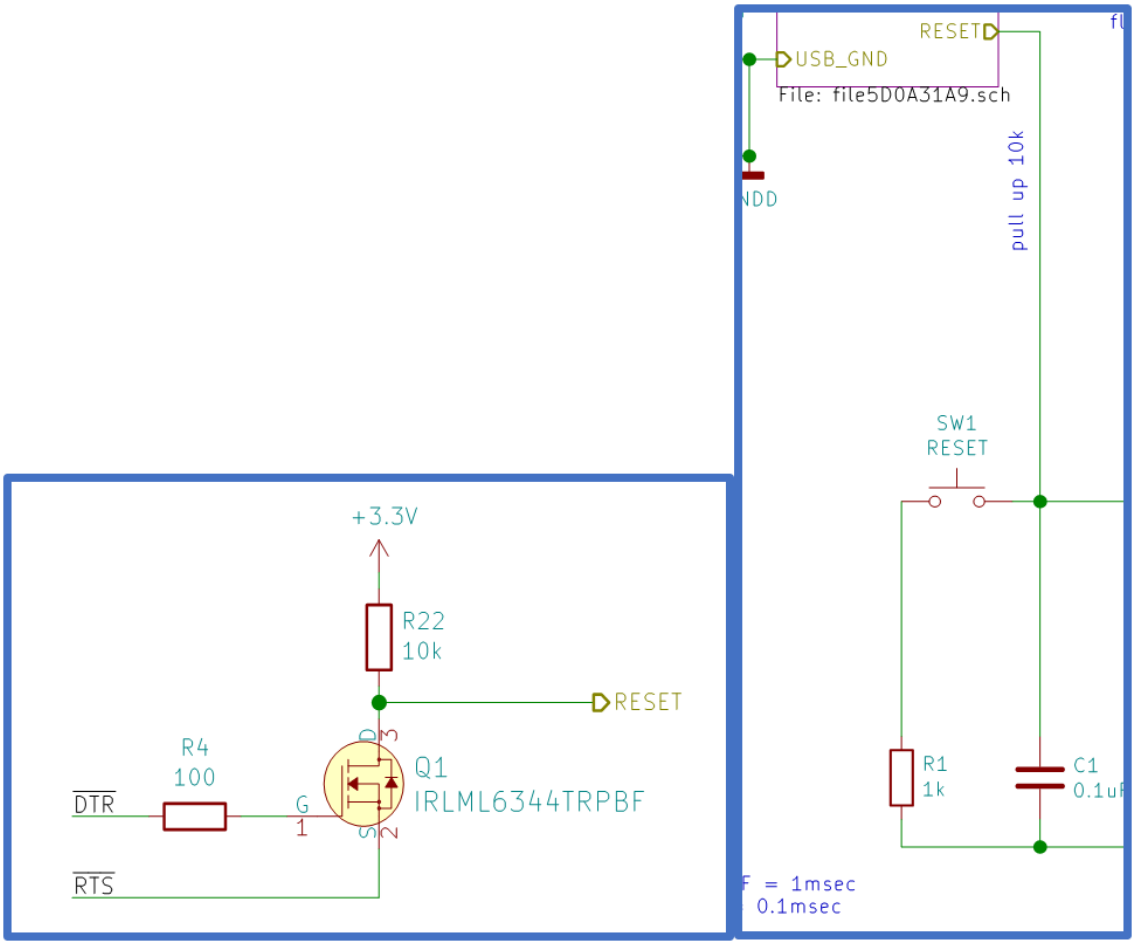
多くのシリアル端末ソフトでは、ハードウェアフロー制御を有効にしない限りは DTR と RTS を共に HIGH に初期化するか、LOW のまま放置するかという動作をしますので、リセットや IO0 の誤動作は生じにくいと言えます。Arduino IDE からプログラムの書き込みを実行する場合、DTR と RTS のどちらか一方だけが HIGH になるように制御することでリセット端子と IO0 端子を順番に操作できることになります。同時に操作することはできませんが、ESP-WROOM-02 の起動よりも高速に操作できれば、リセット後に IO0 を LOW にして UART 起動モードに設定することが可能となります。

ESP-WROOM-02 の起動時間が遅いという前提で動作する制御方式ですので、理論上は失敗する可能性もありますが、電気的には問題点の少ない優れた制御方式だと思いますのでこの回路を採用しました。

R4 と R26 は Q1、Q2 の誤動作を防ぐためのゲート抵抗です。抵抗値 100 はよく使われる値という以上の意味はありませんが、ゲート容量の充電にかかる時間を決定します。また、信号線に乗ってきたノイズのエネルギーを消費するという役割もあります。$\overline{\mathrm{RTS}}$と$\overline{\mathrm{DTR}}$は共に R21、R23 の 10kΩで pull down されています。Q1、Q2 が OFF になる場合、ゲート容量に溜まった電荷はゲート抵抗に加え、pull down 抵抗を経由してグランドに抜けていきます。また、FT231XS が何も出力していなければ、Q1、Q2 は OFF に固定されます。

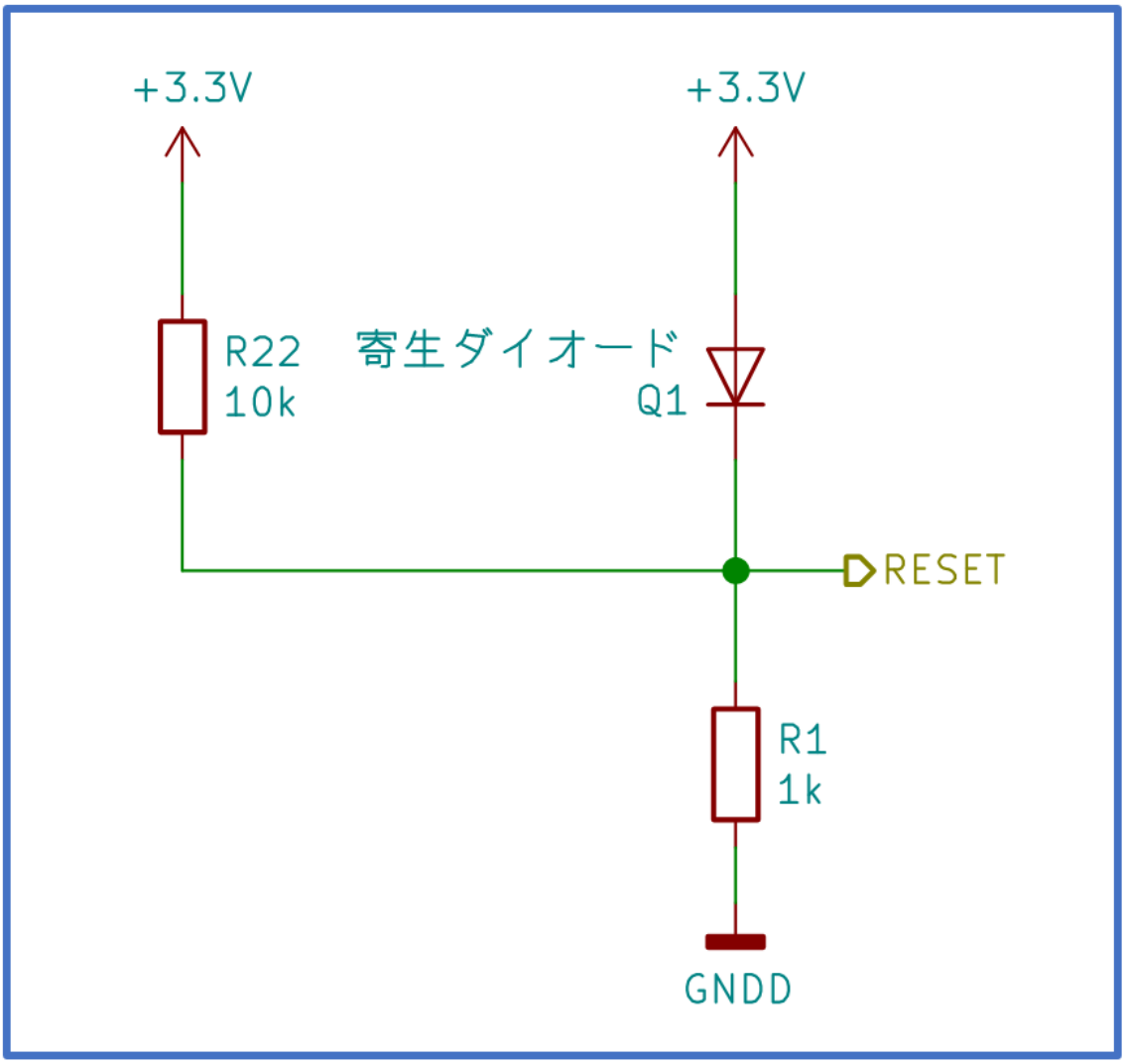
リセット問題

これでうまくいくだろう、と思ったのですが実はリセット回路には機械式のスイッチも付いているという点が問題となりました。該当箇所を下図に示します。




SW1 を ON にすると、RESET のラインは R1 の 1kΩを経由してグランドにつながります。このとき、Q1 が OFF であれば電源の 3.3V は R22 の 10kΩと R1 の 1kΩで分圧され、RESET 端子には 0.3V が入力されます。これは LOW として認識され、EPS-WROOM-02 はリセット状態になります。

ところが、$\overline{\text{RTS}}$が HIGH(信号としては LOW)の場合には問題がありました。この状態で RESET のラインを LOW に下げると、Q1 の寄生ダイオードが動作して$\overline{\text{RTS}}$から RESET へと電流が流れます。そうなると接続は以下のようになります。




とりあえず、電流は R22 にはほとんど流れず、Q1 のほうを主に流れるだろうと考えます。寄生ダイオードの電圧降下を 0.6V から 1.2V として計算すると、RESET 端子の電位は 2.7V から 2.1V くらいの範囲になりそうです。電流が微小なのでもう少し電圧降下は少なくなり RESET 端子の電位は高くなるかもしれません。また、R22 経由で流れてくる電流も RESET 端子の電位を押し上げます。これは ESP-8266 の I/O 電圧の HIGH である 2.64V に近い値で、LOW である 0.8V よりもはるかに高い値です。というわけでとても動作するとは言えない状況で、FT231XS が RTS を LOW に設定していると、手動リセットは動作しないという制限事項が生じることになりました。残念です。

問題が生じている状態でリセットボタンを押した場合の RESET 端子の波形を下図に示します。


電圧が 2.72V までしか下がっていません。差分は 0.68V というところでした。これはダイオードの順方向電圧降下(Vf)としてはありがちな数値です。

FT231XS は省電力モードに入るようにはしているものの、リセットも電源 OFF もできない仕様で回路を作っており、ときどきリセット波形がおかしくなるのはどういう条件なのか、確定するのにやや時間がかかりました。FT231XS のリセット直後の$\overline{\text{RTS}}$は LOW か Hi-Z になっているらしく、手動リセットが効くのですが一度 USB ケーブルを接続すると、それまでの通信状態次第で LOW か HIGH かわからなくなります。やはり FT231XS もリセットできるようにしておくか、電源を切れるようにしておくか、切り分けの手段は用意しておいたほうが不具合の調査にも有効でした。

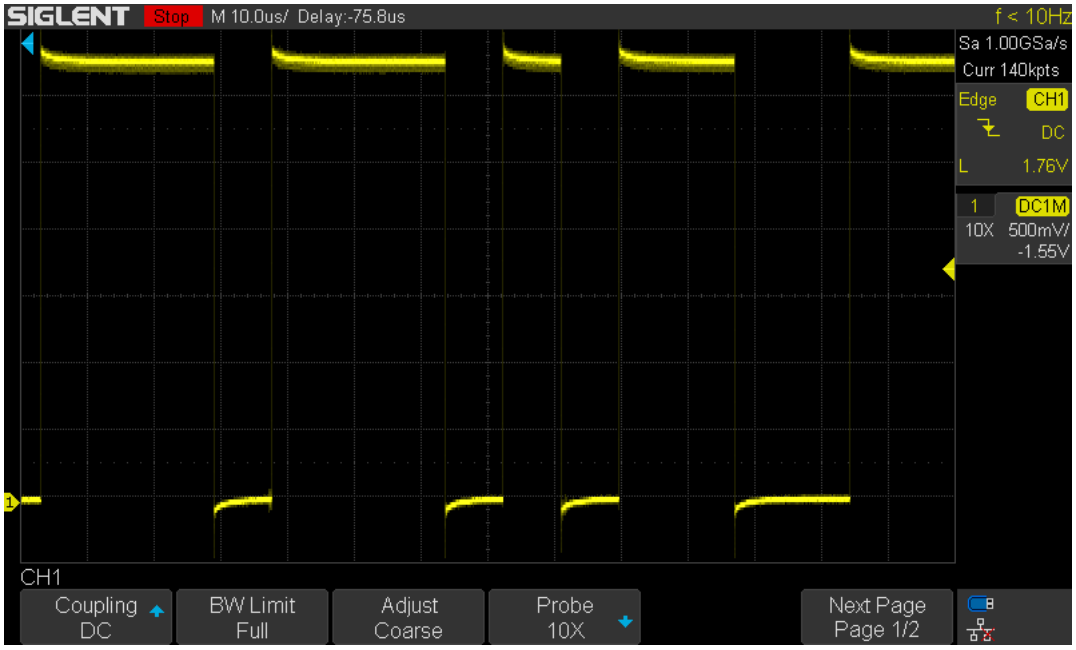
今回はトランジスタの種類を統一したかったので FET で設計しましたが、BJT を使っていれば逆接続時でも寄生ダイオードのような存在はありません。また、その状態でベース電流を流したとしても電流増幅率が低くエミッタ・コレクタ間にはさほど電流は流れません。BJT で設計しておけば動作しそうです。とはいえ BJT の逆接続は応用例があることはありますが、あまり一般的な使い方ではなくメーカも非推奨としていることが多いのでイマイチな設計ではあります。

リセット回路周りは再設計しなければならないようですが、時間切れとなりました。簡単に解決する方法としては、ESP-WROOM-02 の RST 端子は FT231XS 経由での操作のみで使うことにして、リセットボタンによる手動リセットは EN 端子を使うように変更する方法が考えられます。これであれば配線の変更だけで対処可能です。FT231XS もまとめてリセットするという方法も考えられますが、この場合はリセットボタ

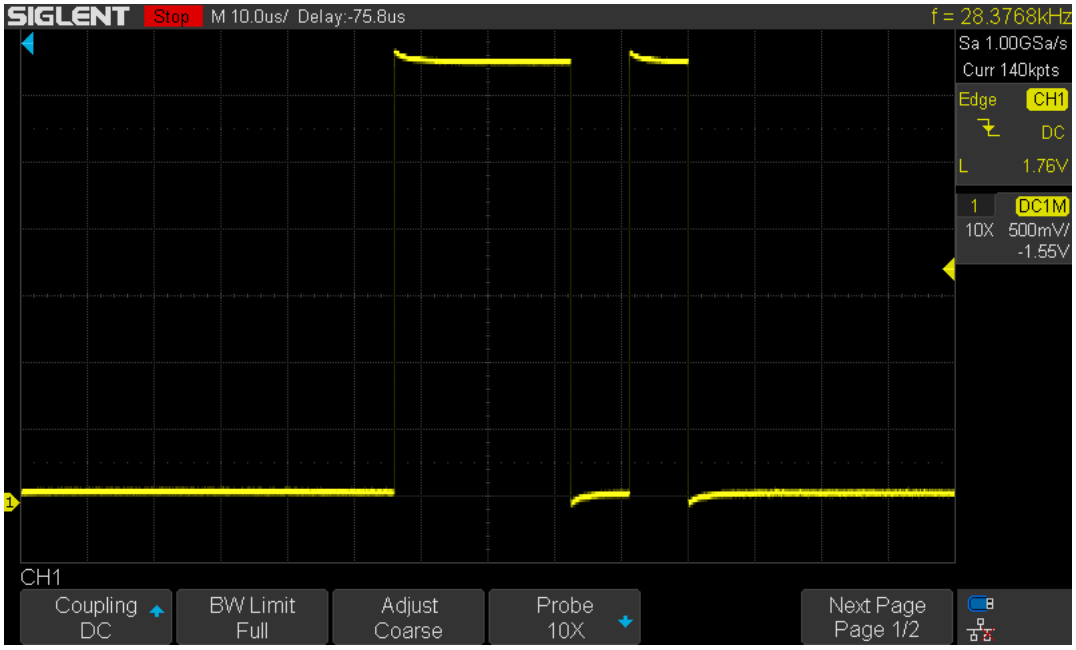
ンを押すと USB の接続が切れてしまうことになり、強力なリセット回路になる反面ちょっと不便になります。

波形

ESP-WROOM-02 からみた受信側(RX)の波形を下図に示します。

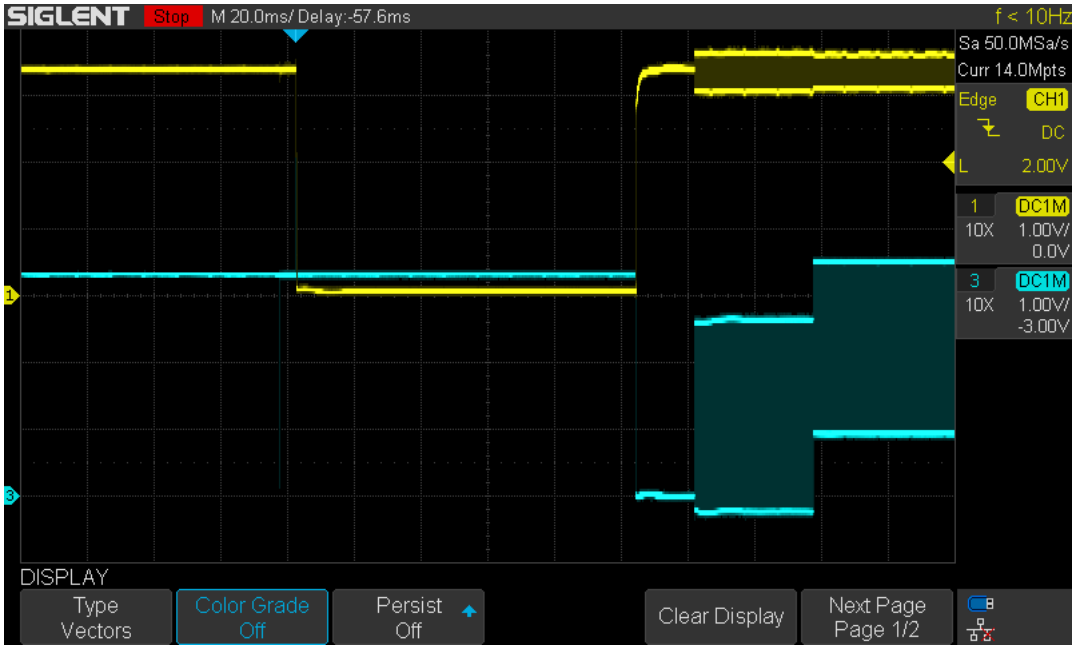



同じく送信側(TX)の波形を下図に示します。



オーバーシュート、アンダーシュートは目立ちますが、通信に問題はないレベルではないかと思います。RX のほうがノイズは多いように見えますが、通信路を駆動している FT231XS に由来するものではいかと思われます。

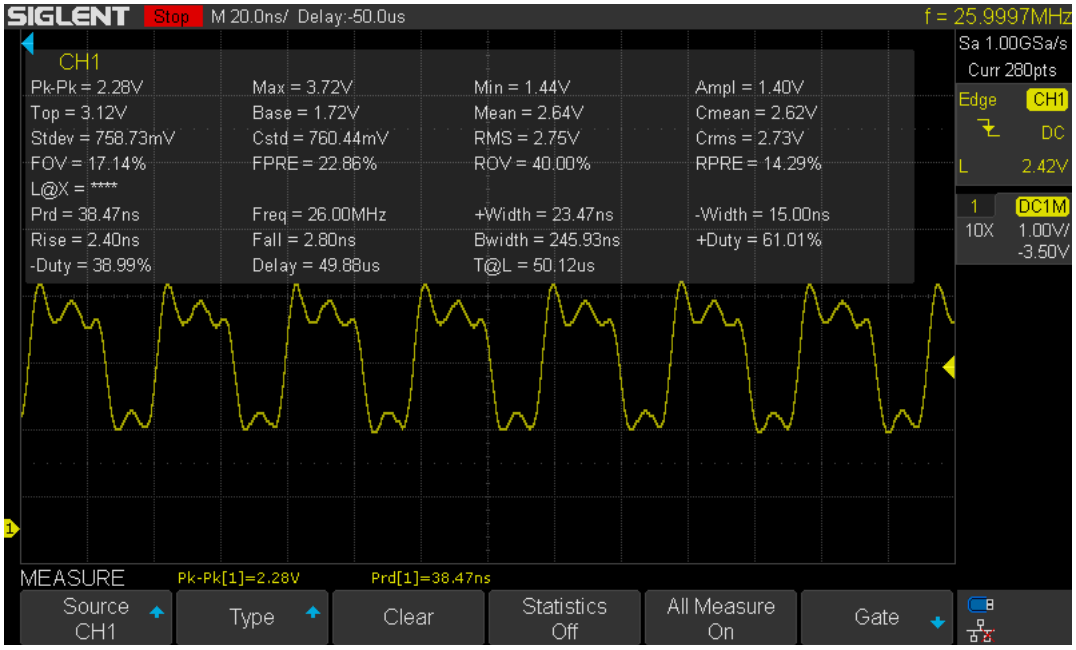
つぎに nodemcu リセット回路によるファームウェア書き込み時のリセットの様子を観測しました。




上図はチャンネル 1(黄色)がリセット端子、チャンネル 3(水色)が IO0 端子の電圧です。横軸は 1 目盛 20ms となっています。少し見難いのですが、リセット端子が 100ms 程度の間 LOW になり、リセットの解除と同時に IO0 が LOW に落ちるという動作がわかります。その後 20ms 程度経過した時点でチャンネル 1、チャンネル 3 ともに上下に激しく振動してみえているのは、FET のゲートに強いノイズが混入していることが原因のようです。

信号の上げ下げのタイミングは GitHub の esp8266/Aruduino レポジトリに含まれる esptool.py の_connect_attempt メソッドに記述されています。リセットを 0.1 秒 LOW に維持(DTR を LOW, RTS を HIGH)、IO0 を 0.05 秒 LOW に維持(DTR を HIGH、RTS を LOW)、すべての制御信号を LOW に戻す(DTR を LOW、RTS を LOW)というシーケンスになっています。Python のスクリプトですのでタイマはさほど正確ではないはずです。通信開始後に IO0 のレベルが少し変動しているのは最後に DTR を LOW に戻す処理が実行されたためと考えられます。タイミングに関しては概ね想定通りです。

問題はやたらと強いノイズですが、RTS の信号線を観測してみると下図のようになっています。




あきらかに UART の速度域ではないノイズがのっています。周波数は 26MHz で、これは CPU のクロック周波数と一致します。Pk-Pk が 2.28V ありますのでかなりの大きさです。プリント基板上では RTS は比較的長い配線となっている上、CTS と RTS を接続してループバックさせようとした結果、T 字型のダイポールアンテナになっています。なかなか優れたアンテナを設計できました・・・・いやいや、これはパターン設計上の致命的な不具合ですね。ノイズの起源が本当に CPU クロックなのか、流入経路は本当に電磁波なのか、いろいろと定かではないのですが、とにかく再設計が必要なのは間違いありません。ファームウェアの書き込みが完了すると、このノイズは観測できなくなりますので、どうやら複合的な要因で強いノイズが生じるようです。普通に使う分には問題は出ていませんが、見つけてしまったからには直したくなります。

・・・というわけで、色々調べたところ、ピンリストに気になる記載がありました ( [ESP8266 Pin List, 2018, ページ: Reg])。GPIO0 の備考に" after reset, the default is function5 to export the clock"という記載があります。どうやら、IO0 がクロック信号の出力端子になっていて、その信号が見えているということのようです。そもそもこの波形はノイズではなく、正しくクロック信号が出力されていたのでした。IO0 は Q2 を通して$\overline{\text{DTR}}$につながっています。$\overline{\text{DTR}}$が LOW(DTR が HIGH)で$\overline{\text{RTS}}$が HIGH(RTS が LOW)になっていると Q2 は ON になります。この状態で IO0 が HIGH になると保護抵抗なしで$\overline{\text{DTR}}$に電流が流れ込むことになります。本来はここに電流制限用の保護抵抗を入れるか、ダイオードで IO0 からの電流出力を遮断する必要があるところでした。また、IO0 端子に 26MHz の高周波信号が流れるのであれば、パターン設計でも

極力配線長を短くしてノイズの放射を押さえるべきでした。しかし、クロック信号の出力ピンは起動モードの選択ピンとは別にしたほうが良いんじゃないかなぁ・・・・？

同様に GPIO2(I2C_SDA)は起動時に UART0 の TX になると書いてあるので、I2C デバイスを誤動作させる可能性があるかもしれません。この GPIO0 と GPIO2 は要注意ですね。

## USB の信号波形

FT231XS の USB 通信は USB2.0 対応とされていますが、速度はフルスピード(12Mbps)までとなっています。わかりにくいのですが、ハイスピード(480Mbps)よりもフルスピードのほうが遅いというのが USB の仕様です。ビット幅はパルスの半分という計算なので、アナログ的には 6MHz の矩形波を送受信しなければなりません。

MHz 帯の信号を通すのは簡単ではありませんが、短い距離ならなんとかなるだろうという軽いのりで D+と D−で配線の長さだけ揃えるようにパターンを引いています。この矩形波を手元のアナログ帯域 100MHz のオシロスコープに 200MHz の 1:10 プローブという組み合わせで観測してみました。本来はテスト用のビットパターンで計測するのですが、時間もなかったのでファームウェアでデバッグメッセージを出力するようにして、立ち上がり立ち下がりの両エッジでトリガをかけて観測しています。大部分は SYNC メッセージの波形になるでしょう。


黄色が D+、紫が D−の波形です。ちょっとタイミングがずれてみえますが、それなりに 0 と 1 の分離は良さそうですね。案外動くものでした。

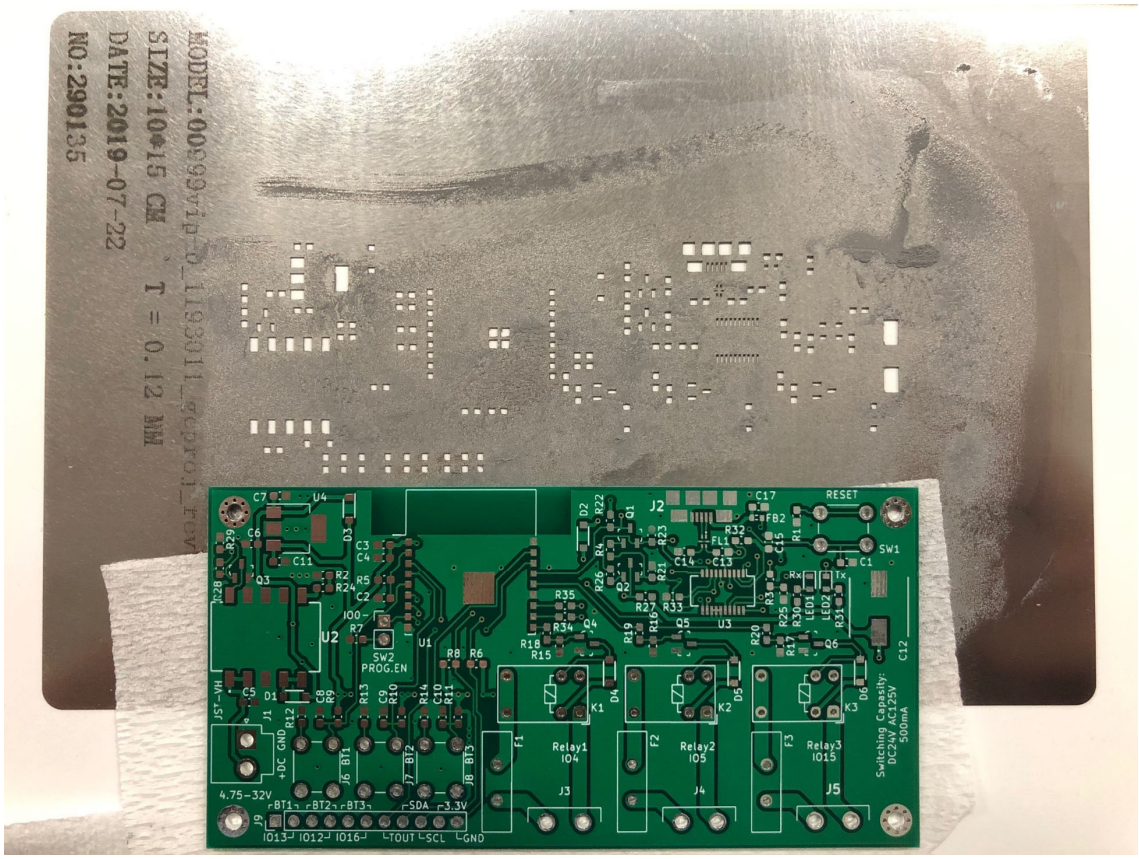
これが USB2.0 のハイスピード転送以上の速度になると一気に観測は難しくなると思われます。個人で買えるようなオシロスコープではちょっと無理です。USB3.0 はほとんど絶望的です。仕事でもこれを観測できるようなオシロスコープは使っていない人が多いでしょう。マイコンの開発・デバッグ用の USB シリアル変換で USB3.0 の速度が必要になる日はこないとは思いますが・・・。

## 製造するために

### 基板製造

この基板自体は海外の試作メーカである FusionPCB に製作を依頼しました。これも感光基板や切削基板を使うと自作できますが、昨今は試作が安いので外注してしまったほうがらくでしょう。今回発注した FusionPCB の場合、安売りキャンペーンを実施中だったようで基板を 10 枚試作して 4.9 ドルでした。これに加えてペーストハンダ用のメタルステンシルが 8.9 ドル、DHL の送料が 19.51 ドルかかりました。送料が飛び抜けて高いので、できれば仲間内でまとめて発注したいところです。

届くのは写真のようなものです。



ちょっと見難くて申し訳ありませんが、緑色の物体が部品を乗せる前の基板、上の銀色の物体がペーストハンダを乗せるためのメタルステンシルです。基板の上にメタルステンシルを乗せて、シルクスクリーンの要領でハンダを基板に塗りつけます。表面実装の部品に適した方法です。ペーストハンダというのはその名の通りペースト状のハンダです。写真は使った後に掃除していないのでステンシルにハンダペーストが残っています。

実際の作業としてはステンシルの平面を出すのが結構大変です。量産用の機械であればテンションをかけて平面にするところなのでしょうが、手作業で押さえつけながら頑張る必要があります。養生テープで仮止めしつつ、クレジットカードのようなプラスチックカードで押さえつけながらハンダを塗り込んでいく感じです。写真のステンシルは小さいのですが、より大きなステンシルで発注すると、打ち抜き加工による歪みが減少して少し作業性が改善するかもしれません。

試作メーカとしては、今回の FusionPCB のほか、Elecrow が有名です。国内では P 板.com があります。品質は圧倒的に P 板.com が優れていると言われていますが、私は使ったことがありません。

ハンダ


基板にはこんな感じのペーストハンダを塗り込みます。かつては個人ではなかなか手に入らないものだったのですが、現在はアマゾンで普通に売られています。ハンダを塗って、上に部品を置いたらリフロー炉という装置で加熱します。炉というとすごそうですが、簡単にいうと料理用オーブンの工業版です。料理用よりも細かな温度制御ができるようになっています。

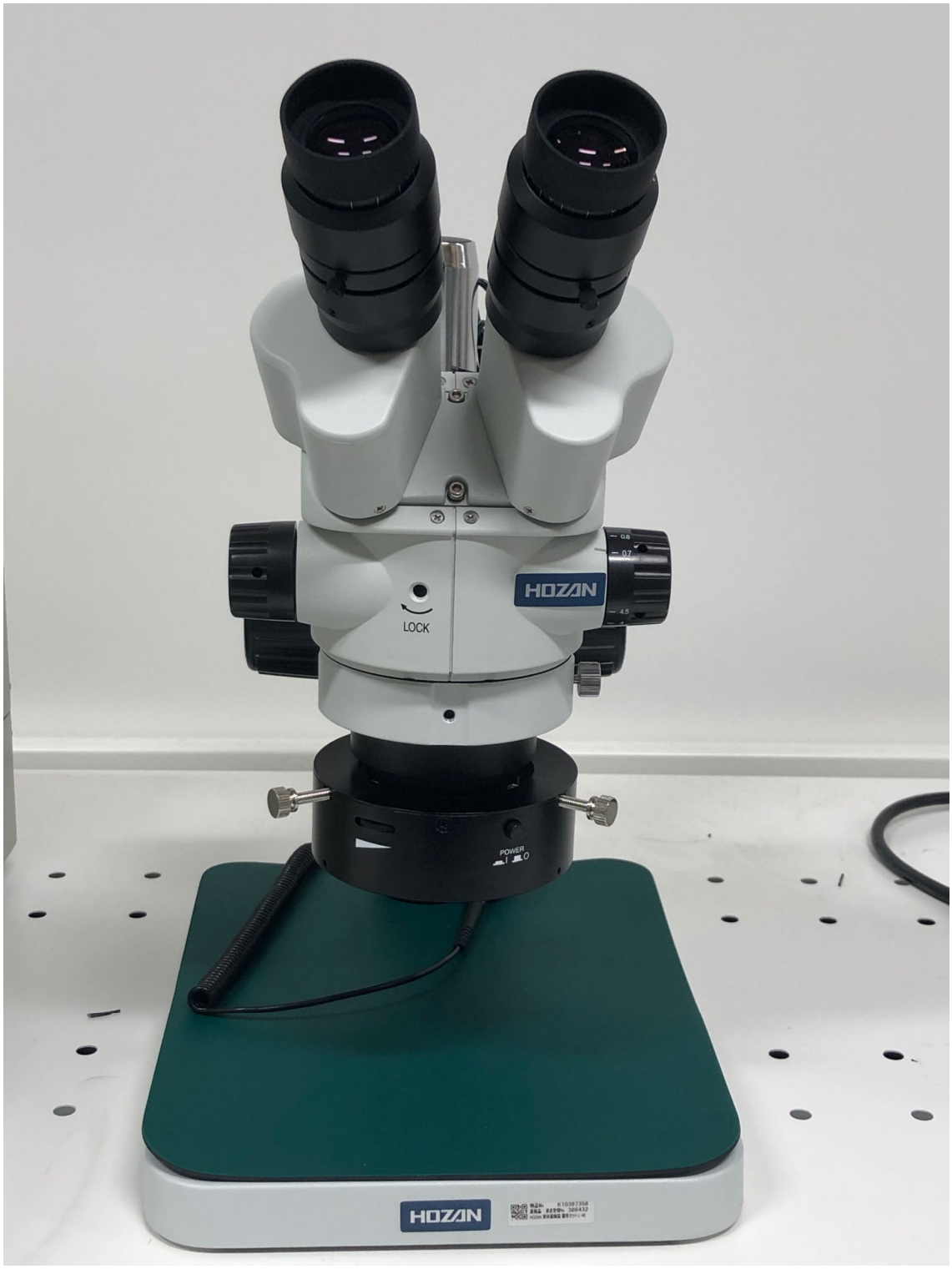
ペーストハンダには板金加工用もあり、簡単に手に入るのですが、板金用は銅を腐食させやすく電子回路には向かないとされています。注意しましょう。

リフロー炉


今回使ったのは写真のリフロー炉です。中国製の激安リフロー炉で、3 万円前後で売られています。ある程度ちゃんとした商品だと 30 万円以上はするでしょう。写真の T-962 はちゃんとしていないことで知られる製品なのですが、検索すると改造のノウハウが色々と見つかります。箱としては頑丈なので、改造を厭わない人々には愛されているようです。写真のものは未改造なのですが、実装密度が低く基板も小さいのであれば未改造でも使えないことはありません。料理用のオーブンやホットプレートを改造している人も多いですが、さすがにそれらよりは改造のベースとしてのポテンシャルは高い製品となっています。

断熱材の入れ替え、温度センサ(熱電対)の入れ替え、センサ回路へのアンプ挿入、などから始まり、制御基板から USB を生やして PC で温度制御をかけるというところまで改造している人もいます。当然、制御基板のファームウェアも独自開発したものが利用されています。改造記事を読んでいるだけで結構面白いので検索してみてください。

拡大鏡



小さな部品を乗せたり、ハンダ不良を確認したりするためには双眼実体顕微鏡が便利です。焦点深度が深く、両目で立体感をもって基板を観察できます。写真の顕微鏡は会社の備品です。個人で買うのはちょっと敷居が高いですね。大きくて場所もとりますし。今回は LXES11DAA2-137 の位置決めと、リフロー後のチェックに顕微鏡を利用しました。これは会社でないと使えない技ですが、

こういう両眼用のルーペで頑張ることも可能です。取り回しが良いので顕微鏡がある場合でも基本的にはこちらを利用するケースが多いです。写真のルーペはアマゾンで 1190 円でした。交換レンズが付いていて、1.0/1.5/2.0/2.5/3.5 倍のレンズを選択して使えます。1.0 でもちょっとだけ拡大されます。LED ライトも付いていますが、照明は別に用意したほうが使いやすいです。電池を入れると重くなりますので。

両眼ルーペの有名どころとしてはハズキルーペがありますが、倍率がちょっと足りないので物作りには今ひとつかもしれません。とはいえ、安物だとレンズの歪みで頭痛や吐き気を感じることがあります。また倍率が高いレンズだと観測対象までの距離が近くなり、焦点があう範囲も狭くなるので作業がやりにくくなります。時計職人のような高い作業台が必要です。自分に合うかどうかは実際に使ってみるしかないですね。

ハンダごて

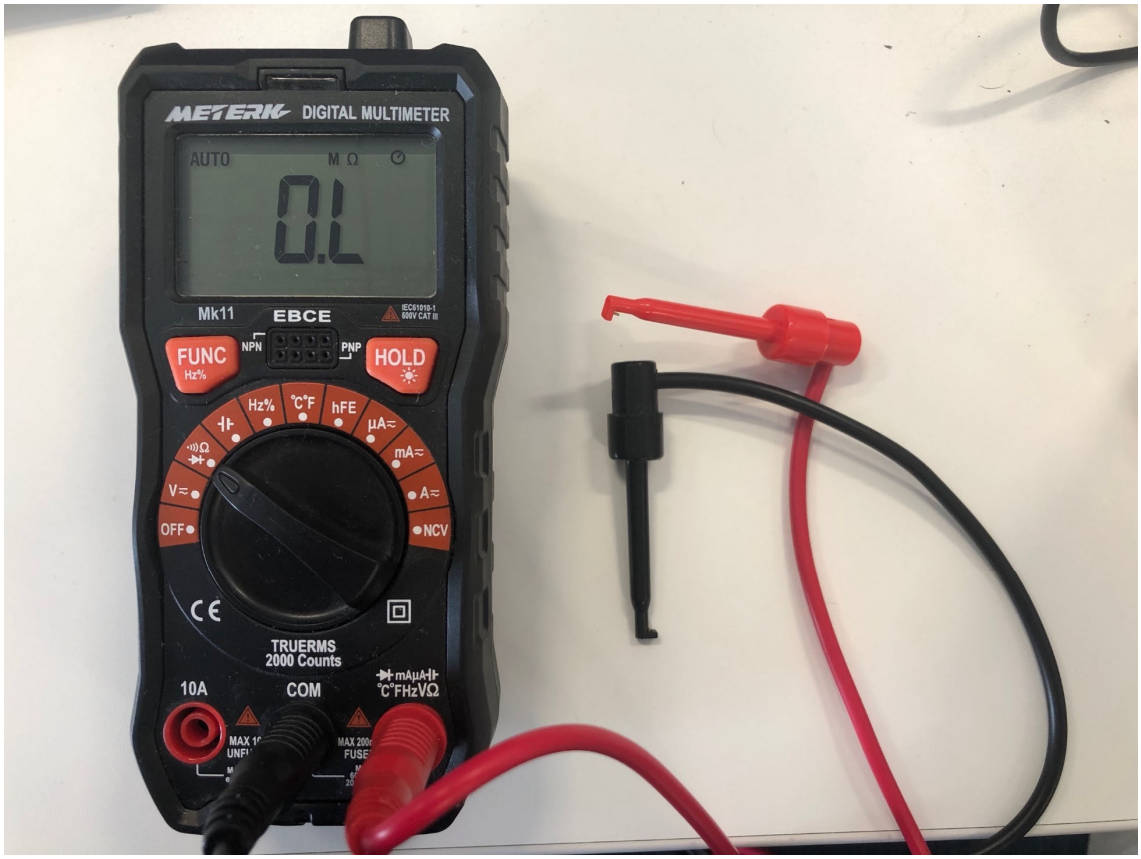

表面実装以外の部品は普通の半田ごてを使います。上の写真は会社の備品なのでやはり個人で買うには敷居が高いかもしれません。

とはいえ、温度調整機能付きのコテはあったほうが良いです。HAKKO の FX-600 が 4000 円から 5000 円くらいで買えて、ペン型で場所を取らず、かつ温度設定ができるので、個人所有のコテとしてはベストではないかと思います。

ちなみに電子部品のハンダ作業は 350 度が基本とされています。昔ながらの鉛ハンダは 180 度前後で溶けますが、温度設定が 180 度付近だとちょっと温度が下がるだけで固まってしまいますし、高すぎると酸化が進みすぎます。無鉛ハンダは組成が色々とありますが、大抵はもう少し高い温度で溶けます。しかしコテの温度は鉛ハンダと同じ 350 度設定で扱うことが多いようです。

テスタ(マルチメータ)

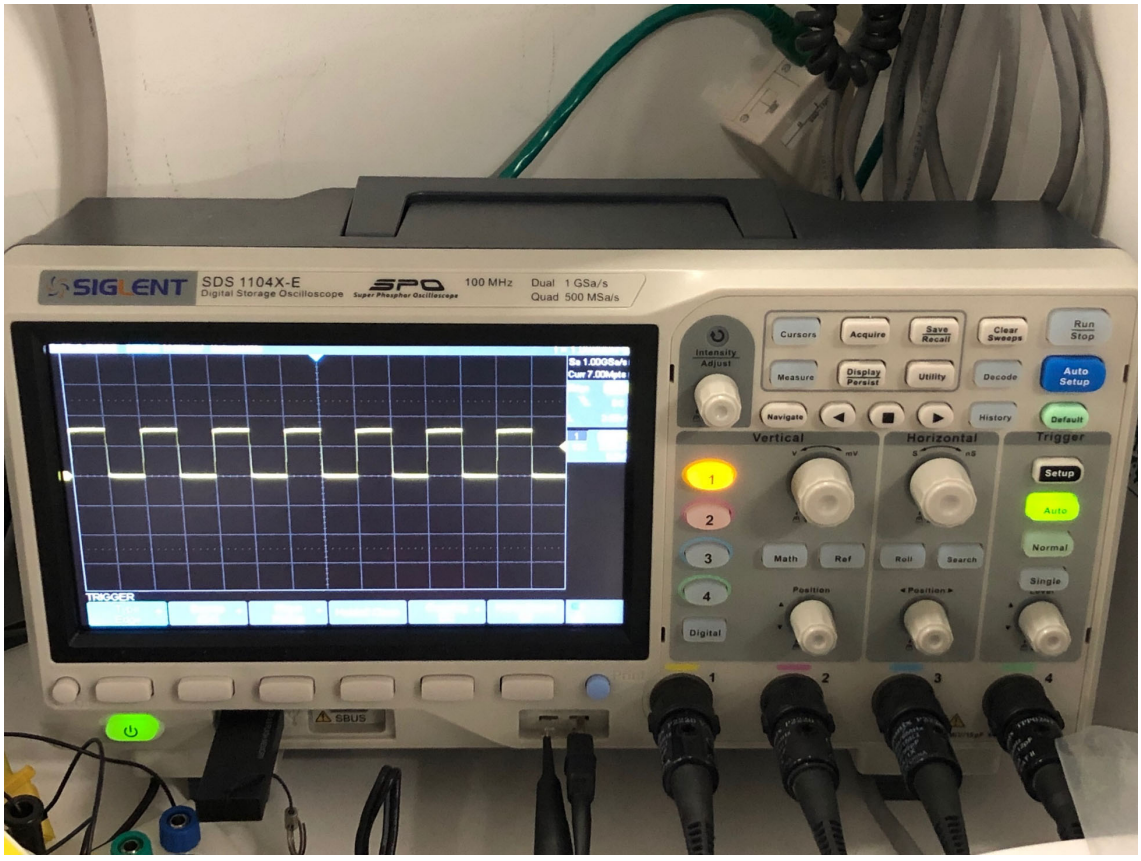

よくあるテスタ(マルチメータ)があるとハンダづけがちゃんとできているか、電源が来ているかなどの基本的な確認ができます。写真は会社の備品で１万７千円前後します。信頼はできますが、個人で買うにはちと高い製品ですね。ほとんどの場合、もっと安いもので問題ないと思います。




こちらは個人所有のテスタです。アマゾンで 1999 円でした。信頼できるのか不安になってしまいますが、導通テスト程度であれば問題はないでしょう。トランジスタの直流電流増幅率を測定できたり、温度を測定できたり、AC 電源に近づけるだけで電源が来ているか確認できたりと妙に多機能な製品になっています。普段はこれでチェックして、不思議な挙動があったときは実験室にいくという感じで作業しています。この製品は、いちいち音がなってうるさいです。特にオートパワーオフで音がなるという仕様はいまいちで、電源を切り忘れていると結構びっくりします。

テスタにはアナログ式という選択肢もあります。大学で電気工学を学んでいたときにはアナログ式が基本でした。測定器としての性能はデジタルに全面的に劣るのですが、測定誤差が大きく出るという特徴が回路技術の理解には役立つのですね。理想的な測定など決してできないのだということをしつこく叩き込まれました。真剣に電気回路を学びたい方はあえてアナログ式という選択もありかもしれません。レンジの選択からしっかり考えて端子を当てる必要がありますし、そのためには回路の挙動を事前に予測できている必要があります。単に大きいレンジから順に試していけば壊れはしませんが、それならデジタルのオートレンジで良いですね。修行を重ねれば針の震えから回路の不調を感じ取る職人技にたどり着けるかもしれません。

オシロスコープ




テスタだけでは動きがわからない箇所については、オシロスコープの出番です。あると大変便利で回路に対する理解が深まる道具なのですが、なくてもなんとかなるものでもあります。なくてもがんばれると思いつつも、なぜか欲しくなって買ってしまう機材の一つです。写真の製品は SIGLENT という会社の SDS1104X-E という機種で、国内保証付きで 8 万円を切るくらいの価格で売られています。測定器の相場からすると安いと言えるのですが、個人で買うには高価ですね。繰り返しますが、なくてもがんばれますので、欲しくなってから考えれば良いでしょう。

写真のような低価格オシロスコープは、岩通、テクトロニクス、キーサイトといったハイエンド測定器メーカと比較すると圧倒的に安いので不安になりますが、意外と使えます。高級な測定器は実験室に設置されて研究開発に使われるものですが、低価格オシロスコープは工場の工員 1 人に 1 台という意気込みで配置して製造品質を底上げするために使われるものです。出荷数が全く違いますので、性能差以上に価格差が大きく出るのですね。

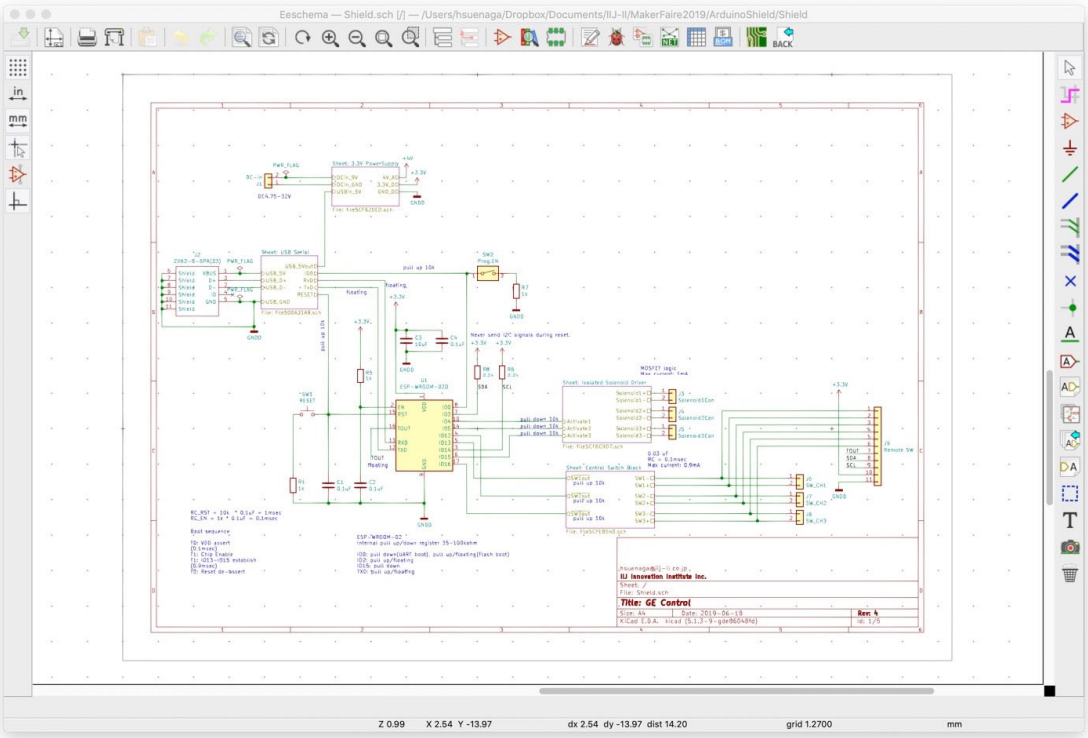
世間的には RIGOL の DS1054Z という機種が人気のようです。こちらは 6 万円を切るくらいの価格帯です。アナログ帯域幅が 50MHz と SDS1104X-E より若干狭いですが、ファームウェアで制限しているだけでハード的にはもう少し能力は高いという噂もあります。とはいえ、1 チップマイコンを中心に使う分にはアナログ帯域幅は 50MHz だろうが 200MHz だろうが大差ないでしょう。さらに安い 3 万円以下の製品もいろいろありますし、USB 接続の簡易オシロスコープなどもあります。まずは手に入れやすいもので感覚を掴むのも良いかもしれません。


オシロスコープを使う場合、アースとグランドを意識することがとても大切です。オシロスコープの測定端子についているグランド接続用端子とアース端子は全て内部で繋がっています。複数の測定端子がある場合でもグランドは共通です(共通ではない絶縁オシロスコープというものもありますが、一般的な機材ではありません)。したがって、グランド端子をどこかに接続するということは、接続先とグランドをジャンパしてショートさせることになります。接続先がグランドでなかった場合、オシロスコープを通して大電流が流れ電源周りが瞬殺されたりしますので注意しましょう。対象のグランドがアースに接続されているのか、どこにも接続されずに浮いているのかも同様に大切

です。この点、テスタは測定器側が電池駆動でアースからは常に浮いていますし、大抵は 1 入力しかありませんので事故は少なくなりますね。

　初めてオシロスコープを使う場合にはテスタと似ているようでいて違うという点を忘れないでください。

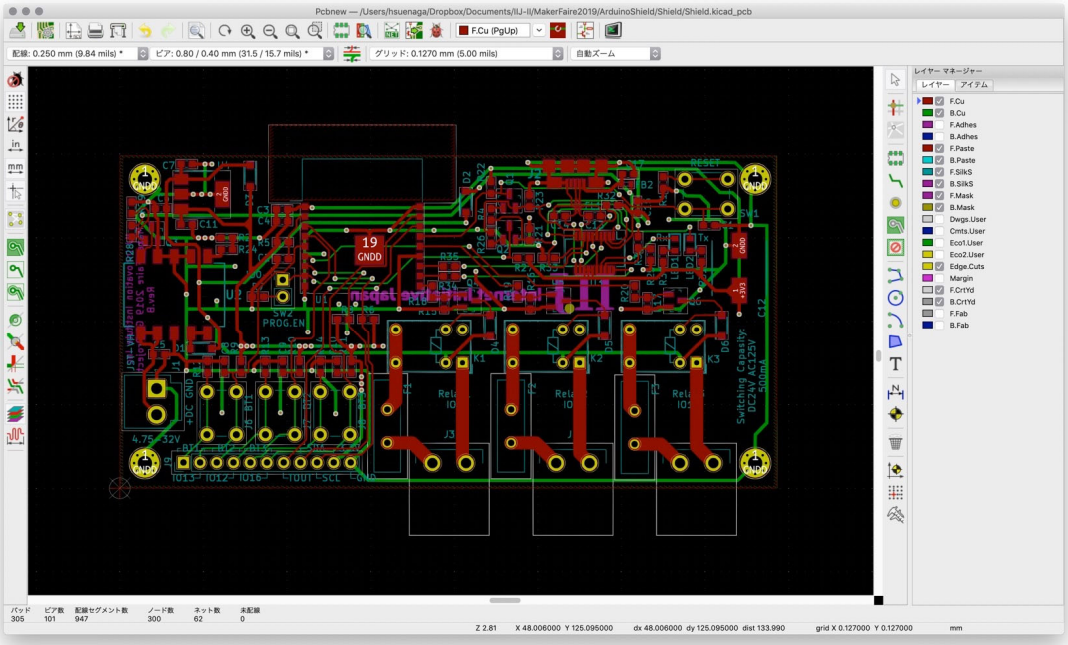
## 回路 CAD




　回路設計には KiCAD を使っています。自分で日本語ストロークフォント(KST32B)対応パッチをあててビルドしたものを使っています(方法は KiCAD.jp の wiki に書いてあります)。大抵の回路に日本語は必要ありませんが、今回のように文章に流用する場合は日本語に対応しておいたほうが便利です。

　電気回路 CAD としては Autodesk EAGLE も人気があります。CAD はどれもクセというか、流儀があるので使いやすいと感じたものを選べば良いでしょう。KiCAD は EAGLE のファイルを読むこともできますので、迷ったら EAGLE から始めた方が良いかもしれませんね。

PCB CAD




基板のパターン設計も KiCAD を使っています。今回は部品点数がそれほど多くありませんでしたので、自動配線は使わず、すべて手動で配線してしまいました。自動配線をしたいのであれば FreeRouter のような他のソフトウェアを使う必要があります。KiCAD の開発者が”NEVER trust the autorouter”と書かれた T シャツを着ている写真を見たことがありますので、自動配線をがんばるつもりはないのでしょう。

EAGLE もパターン設計に対応しています。EAGLE の場合、標準で自動配線に対応しています。電源配線など主要な配線だけ手で配線すれば、あとは自動配線で解決できるでしょう。配線の最適化は難しい問題で、大抵一発では終わりません。コンピュータがトライアンドエラーで配線していく様子を眺めるのはちょっと楽しいです。

インターネットの経路の最適化は計算が高速になるように巧妙に問題を単純化しており、理論上の最適解に実用的な速度でたどり着けます。しかし、電気回路の配線はそう上手く単純化はできません。EAGLE の最適化はかなり早いほうだと思いますが、計算が終わらなくなることもあります。人間であれば配線できるようにみえることもありますし、原理的に不可能にみえることもありますが、実際にどうなのかは判断できません。コンピュータでは計算不可能な問題なのですね。

引用文献

Espressif Inc. (2015 年 8 月 1 日). ESP8266EX Datasheet v4.4.

Espressif Inc. (2016 年 1 月). ESP8266 System Description v1.4.

Espressif Inc. (2017 年 5 月). ESP8266 Technical Reference v1.3.

Espressif Inc. (2018 年 3 月). ESP-WROOM-02 Datasheet v2.6. 21.

Espressif Inc. (2018 年 12 月). ESP8266 Hardware Degisn Guidelines v2.4.

Espressif Inc. (2018 年 12 月 13 日). ESP8266 Pin List.

Espressif Inc. (2018 年 11 月). ESP8266EX Datasheet v6.0.

Phillips Semiconductors. (2000 年 1 月). I2C バス仕様書 v2.1.

Pull up resistors. (2015 年 9 月 9 日). 参照先: ESP8266 Developer Zone: https://bbs.espressif.com/viewtopic.php?t=1079#p4097