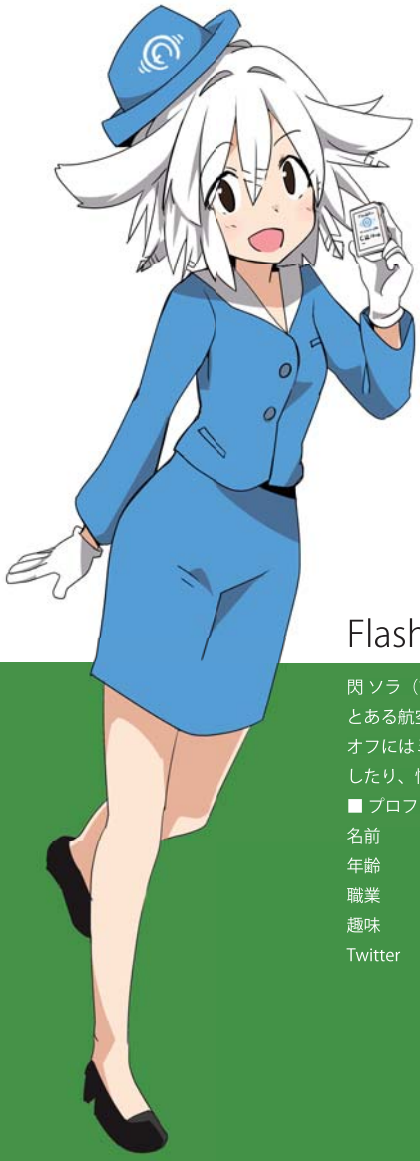
FlashAir™ Doujinshi 3

FlashAirの同人誌３

TAKE FREE ￥0

目からウロコなFlashAirの活用事例満載！
ゲーム、見守り、地図アプリ、クラウド
連携から超会議？！キミもアクセル全開だ！



FlashAir 応援キャラクター「閃ソラ」

閃 ソラ（ひらめき そら）は、FlashAir の非公式応援キャラクターです。
とある航空会社の CA をしています。フライトでたびたび不在にしますが、オフにはミラーレス一眼で写真を撮ったり、電子工作したり、アプリ開発したり、忙しい毎日を送っています。

■ プロフィール

名前	閃ソラ（ひらめき そら）
年齢	23 歳
職業	新人キャビンアテンダント
趣味	電子工作、アプリ開発
Twitter	@Hirameki_Sora

FlashAir 同人誌 3 号によせて

高田 真里

今号で FlashAir 同人誌も 3 号となりました。Maker Faire 来場者へのプレゼントで始まった同人誌ですが、今では FlashAir の情報発信媒体として欠かせない冊子となりました。FlashAir 同人誌は、FlashAir を「知ってもらうこと」、「試作に挑戦してもらうこと」、そして皆様に FlashAir に関する「情報発信をしてもらうこと」を期待して有志が手作りで発行しています。

「知ってもらうこと」については、FlashAir の機能やパートナの皆様の提供してくださる商品、さらには FlashAir にまつわる活動をご紹介します。今号は、FlashAir 向けの基板や FlashAir 向けのクラウドサービス「FlashAir IoT Hub」についてご紹介します。ニコニコ超会議への参加体験記も必見です。

「試作に挑戦してもらうこと」については、FlashAir の作例をリファレンスと共にご紹介します。今号では、SNS で情報収集した作品から複数の記事を寄稿いただきました。また、クラウドファンディングで商品開発に挑戦する記事など、ものづくりを志す皆様の Make 魂をくすぐる記事を取り揃えています。

FlashAir 同人誌の究極の期待は、皆様からの「情報発信」です。同人誌に記事を寄稿いただく形態だけでなく、同人誌を読んだ皆さまが Twitter で感想をつぶやいたり、ブログで作例を紹介したり、様々な媒体での情報発信を歓迎いたします。もちろん、FlashAir Developers サイトのフォーラムへのコメント投稿もお待ちしています。

FlashAir はイノベーターの皆様に様々な使い方を開拓いただいてきました。趣味で、試作で、会社の業務や家庭での使い方をより多くの皆様と共有し、FlashAir を便利に使っていただきたいと思います。そして、それぞれの用途でのご要望を広く受け止めて、製品をより使いやすいものにしていく。そのような取り組みをイノベーターである皆さまと共に作り上げたいと思います。そのための、私たちからの最初のアクションが同人誌です。1 年に 1 回の発行ですが、これが次の FlashAir 成長の物語につながることを信じてお届けします。社内外から寄稿いただいた記事を、是非最後までお楽しみください。

高田 真里 (@M_Takada)

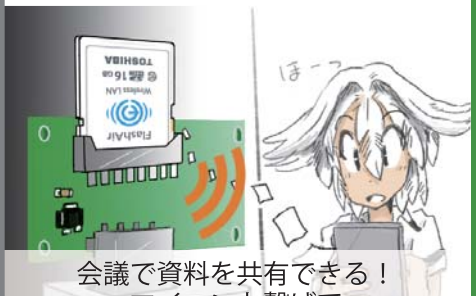
FlashAir の商品企画に携わり、現在は展示会や販促活動を担当。展示会でデモをして FlashAir を誉められると単純に嬉しいので、新しい使い方を紹介できるデモ機の開発を画策しています。

FlashAir は超ミニマイコン！

撮ったらシェア！

カメラに入れて使うことで
写真を撮ったらその場で
友人とシェアできる！

web サーバ機能搭載！



会議で資料を共有できる！
マイコンと繋げて
データロガーとしても使える！

GPIO が使える！

FlashAir の信号端子を自由に使える！
L チカからゲームまで
用途は多彩！

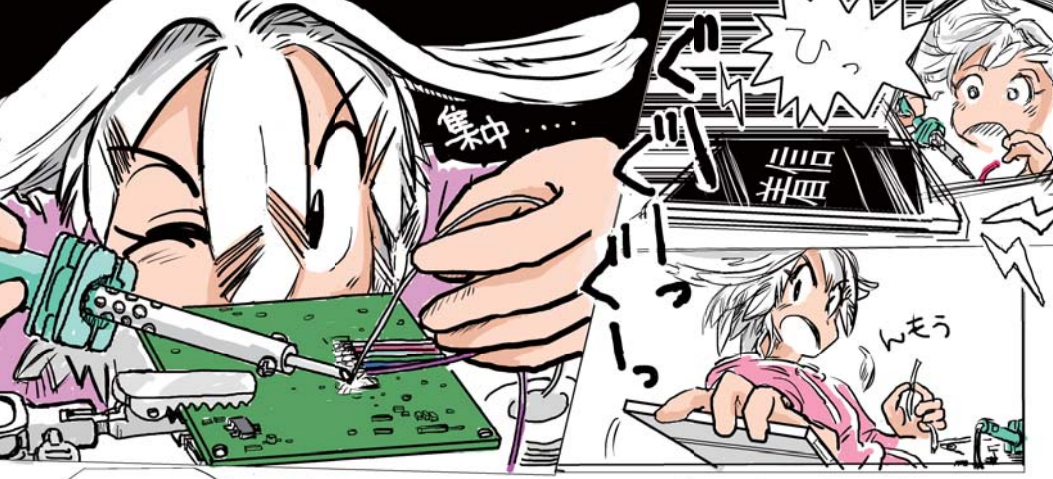
Lua スクリプトが実行可能！

FlashAir がマイコンのように使える！
メールを送信したり
ツイッターへの投稿が簡単にできる！

https://flashair-developers.com

FlashAir Developersにアクセス！

Contents



「FlashAirでライブ配信」の巻

もしもし。
……
言われたように
毎日引きこもって
ますって。

ほんと
CA の仕事も
出来ないから、
毎日退屈。

ここ最近
電子工作員の仕事が
派手になっていません？

…もちろん、コワモテ
の人に覗かれているの、
私も気づいてますって。

作じす

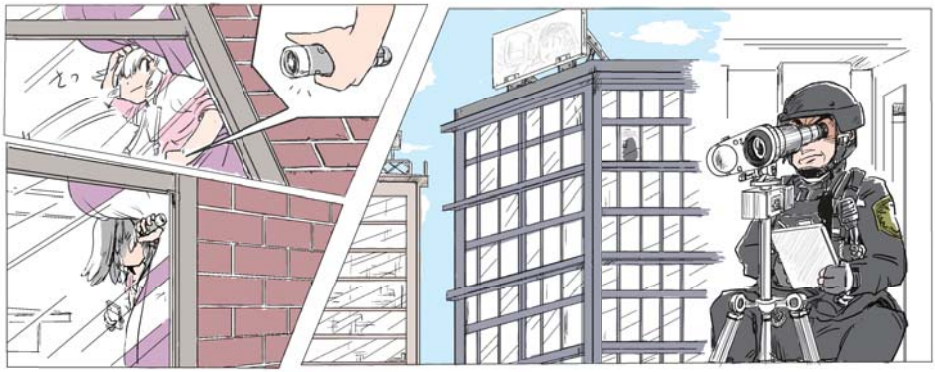
今？
Raspberry Pi の改造中。

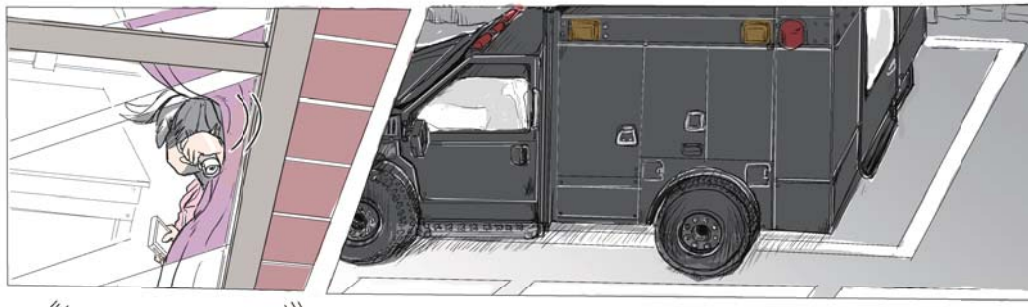
内蔵カメラ付けて
microSD ソケット外して
FlashAir を載せてた。

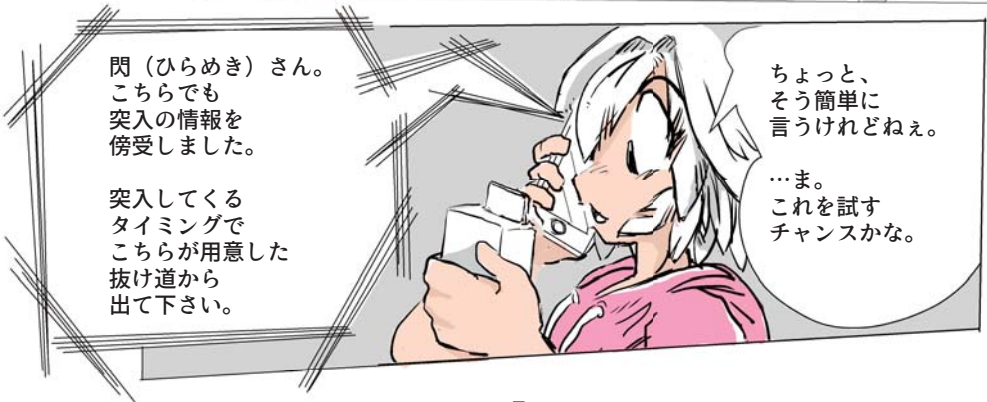
カチャ
カチャ

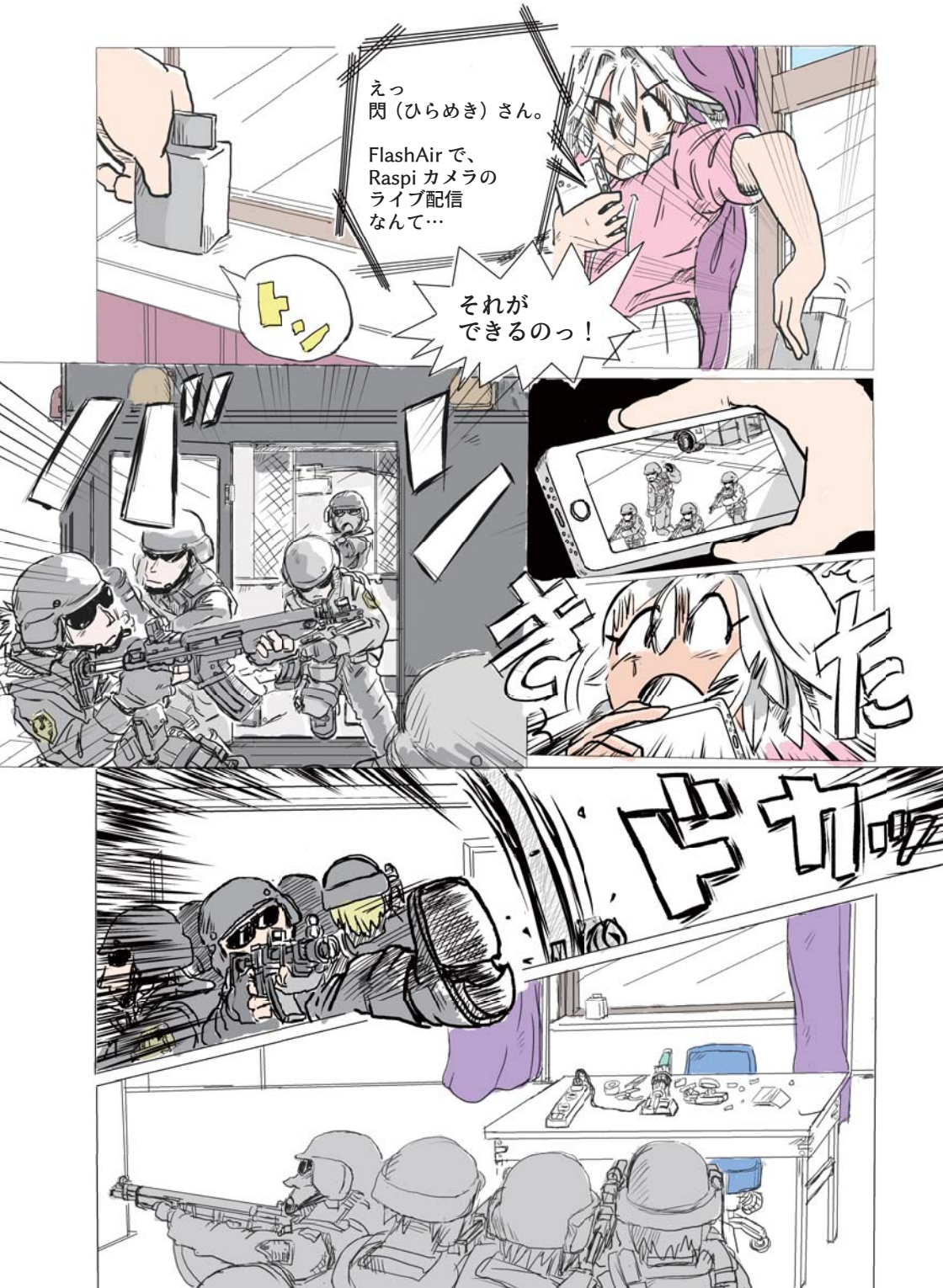
じゃーん


ラ ..Raspberry Pi !?
そんなのネットに
つないだら、
一気に攻撃を
受けますよ！
だから FlashAir だけで
遊んでいるわよ…
ん…ちょっと待って。
ダダダ






閃（ひらめき）さん。
こちらでも
突入の情報を
傍受しました。
突入してくる
タイミングで
こちらが用意した
抜け道から
出て下さい。
ちょっと、
そう簡単に
言うけれどねぇ。
…ま。
これを試す
チャンスかな。


えっ
閃（ひらめき）さん。
FlashAir で、
Raspi カメラの
ライブ配信
なんて…
トン
それが
できるのっ！
ドカッ

Flashair でライブ配信

じむ

作中にソラちゃんが作っていた FlashAir を用いたライブ配信システムですが、「FlashAir で動画ファイルを視聴する時はダウンロードしてからじゃないの?」と思われるかも知れません。しかし、"HLS" を使った方法ならライブ配信が可能なんです。"HLS" とは Apple 社が公開している「HTTP を使ったライブ配信システム」です。今回はこのライブ配信システムの製作方法を説明しようと思います。用意するものは Raspberry Pi と FlashAir です。Raspberry Pi 3、Raspberry Pi Camera Board v1.3 にて動作を確認しています。

Raspberry Pi の準備

まずは「Raspberry Pi + FlashAir」で動く環境を用意します。ライブ配信は動画を保存しますから、容量の大きい FlashAir が便利です。Raspberry Pi に実装されているのは microSD カードスロットなので、microSD を SD カードに逆変換するアダプタを使います（作中ではソラちゃんは力技で microSD カードスロットを SD カードスロットに換装していました）。Raspberry Pi で FlashAir を使うには、パーティションの編集が必要です。Ubuntu などの Linux マシンで FlashAir 内の /boot の領域（=fat32）を広げます（図 1）。その後、Raspberry Pi で "HLS" 形式の動画を作るために FFmpeg を使います。インストールの方法はここでは省略します。

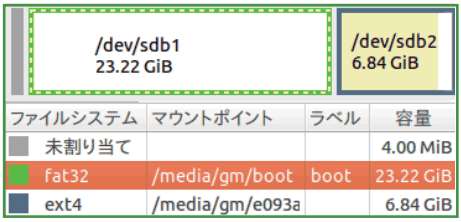


図 1: Raspberry Pi のパーティション構成

続いて Raspberry Pi 内に作業フォルダを用意します。例として "/boot/hls" というフォルダを作成します。Raspberry Pi 内のコンソールで下記のコマンドを実行してください。以降はこのフォルダにて作業していきます。

```
$>cd  /boot
$/boot>sudo mkdir  hls
```

1 https://developer.apple.com/jp/documentation/StreamingMediaGuide.pdf

HTML ファイルの作成

FlashAir 内の "HLS" ファイルをスマホのブラウザで視聴をするために HTML ファイルを作成します。下記に "/boot/hls/stream.htm" を示します。

```
<html>
<head>
  <title>HTTP Live Streaming Test</title>
</head>
<body>
    <video controls="controls" width="720" height="405" autoplay="autoplay" >
     <source src="output.m3u8" type="application/x-mpegURL" />
    </video>
</body>
</html>
```

FFmpeg を使った "HLS" 形式動画の作成

最新 FFmpeg には "HLS" ファイルを作成する機能が備わっています。今回は Rasberry Pi カメラからの映像をライブ配信するようにしました。通信速度も考慮して画角は一番小さいものにしています。下記に "/boot/hls/video.sh" を示します。

```
#!/bin/bash


sudo raspivid -n -w 400 -h 224 -fps 25 -vf -t 8640000 -b 400000 -ih -o - ¥
| ffmpeg -y ¥
    -i - ¥
    -c:v copy ¥
    -map 0:0 ¥
    -f segment ¥
    -segment_time 10 ¥
    -segment_format mpegts ¥
    -segment_list output.m3u8 ¥
    -segment_list_size 0 ¥
    -segment_list_flags live ¥
    -segment_list_type m3u8 ¥
    %05d.ts


trap "sudo rm output.m3u8 *.ts" EXIT

# vim:ts=2:sw=2:sts=2:et:ft=sh
```

上記の作業で、"/boot/hls" フォルダの下には "stream.htm" と "video.sh" が作成されました。"video.sh" を Raspberry Pi のターミナルから実行することで配信ができます。

```
$/boot/hls>sudo ./video.sh
```

スマートフォンでライブ配信を観てみよう

Raspberry Pi で作成した "HLS" ファイルは FlashAir 内に保存されており、iPhone、iPad、Android 端末からアクセスすることでライブ配信を観ることが可能です。視聴するためには FlashAir とスマートフォンを無線 LAN で接続し、chrome 等のブラウザで "http://flashair/hls/stream.htm" にアクセスします。

参考に、今回作成したファイル一式を下記に配置しました。アーカイブの解凍後、sudo で FlashAir にコピーしてください。

```
http://rdstyle.cocolog-nifty.com/flashair/hls/hls.zip
```

また、本記事のさらなる詳細は FlashAir Developers にも掲載予定です。そちらもご参照下さい。

じむ

FlashAir の職場を離れて 2 年が経ちますが、未だに FlashAir の魔力に囚われて工作ネタを模索する日々。

何回ボツになっても、ガンガン描くよー。(`・ω・´) キリッ

FlashAir のよもやま話

伊藤 晋朗

初めて自分の金で海外に行ったのは大学院生の頃だった。きっかけは 12 月の終わりに毎年恒例行事になっていたスキー合宿の初日に、スキー場のリフトに乗っている時に携帯に電話がかかってきた。相手は大学院には行かずに研究生をやっている友達からだった。会話は「ニュージーランドに行こうやあ。」「おお、いいぞ。すぐ行こう。」ぐらいの短いやり取りで全ての話が決まった。それから旅行会社に一番早いチケットを緊急発券してもらい、年明けにはニュージーランドの南島にあるクライストチャーチ空港にいた。着いたその日の宿もこれから先にどこに行くとか何も決めていなかったが、海外を行き当たりばったりで行き先を決めていくという旅をしながら、同時に日本人であるということを意識した 2 週間の旅だった。

FlashAir の仕事が始まって、最初の出張も海外だった気がする。気がついてみたら暑い国に居て、知らない人たちと打ち合わせをしていた。その時から何度も海外へ出張に行く事になるなんてその時には予想できなかった。それぐらい FlashAir は日本製の製品ではあるが、海外とのやり取りが必要になる事がとても多い。例えば無線認証を取得するためには海外が安ければそこで認証取得するし、海外で販売もしているために、現地法人向けに技術的な仕様説明や展示会に出展する事もある。また、海外で同じような苦労をしているメーカーの人にも会うこともあるし、対応アプリを開発したいという人もいるので、付き合う相手は幅広い。

ある時、カナダからお客様が来るということで、打ち合わせに参加する事があった。基本的にはいろんな会話の中から製品仕様のヒントをもらったりもするのでスケジュールが合えば参加している。打ち合わせは相手の参加者が 4 人で、日本人も 6 ～ 7 人くらいいたので、全部で 10 人を超える大きな打ち合わせだった。自己紹介で名刺交換をすると、カナダのネイティブ・アメリカンが 1 名、ニュージーランド人が 2 名、ヨーロッパから 1 名（国籍不明）という構成だったので、かなり多国籍なメンバーである。

打ち合わせを始める前に、相手のメンバーがニュージーランドのオールブラックスが試合前にやるような踊りの歌である「ハカ」を歌い出した。どうやら相手はマオリ族の人だったらしい。歌を歌いながら同時に英語で翻訳してくれているが、どう対応して良いのかわからないので、日本人サイドは黙って聞いているしかない。

次に、マオリでは親愛なる相手との握手は、「握手しながらお互いの額と鼻をくっつけるんだよ。」と教えられる。「へー、そうなんだ。」と思っていたら、「では、やってみよう。」となる。面白いことに日本人のおっさん達は素直に一列になって、額と鼻をくっつけるために並ぶ。

その後のプレゼンとしてデモビデオを見せられて、俺たちはこんなにすごい技術を持っていると主張される。でも、プレゼンのタイトルを印刷した紙を写すのからデモまで、全部ノーカットで編集無しだと、なんとなくホームビデオを見ている気持ちになる。

打ち合わせ中に勝手に会社所在地を調べて見るが、どうしてもどこにあるのかよくわからなかった。メールアドレスのドメインを調べてもドメイン取得されていない。どうやって、打ち合わせが設定されたのかというとやり取りは Hotmail だったそうだ。そんな状況でプレゼンが進んでいくが、日本人は何もコメントを言わない。

その後も打ち合わせはなんとなく進んでいく。話は契約を結びたいとか、ジョイントベンチャーをつくりたいから出資して欲しいとか色々と大きな話をしてくる。しかし、あまり FlashAir と話が結びつかない。ふと、隣の人を見ると、笑いとあきらめを混ぜたような眼差しをしている。

個人的にはとても面白いが、ツッコミができない時間が過ぎていく。

打ち合わせは2時間ぐらい過ぎたあたりで、こちら側が、「我々は、とても古い会社だ、なので色々と決めるにも、この場では決められない。」という一言で終わりになろうとしていた。そうすると向こうもわかったという感じになって、会議が終わりそうになる。その時、ネイティブ・アメリカンのカナダ人が、「我々には4つの歌がある。出会いを祝してその一つを歌おう。みんな立ってくれ。」と言った。日本人は、立ち上がれと言われたら立ち上がり、目をつぶれと言われると目をつぶった。彼の歌は、壮大な荒れた大地を目にして聞いていたとしたら、色々な事を勝手に感じて感動もしたのだろうが、残念ながらあいにくここは東京のど真ん中ともいえる港区浜松町だったので、その歌声にひたることはできずに幕を閉じた。。

FlashAir は不思議な事に色々な人を引き付けるようだ。その理由は人によって違うので色々だが、どんな理由であれ関心を持っていただけるのはとても嬉しいことである。なので、これからもいろんな事が起きるように工夫していきたいと思っている。

伊藤 晋朗 (@ikainuk)

FlashAir の " 中 " の人。関係する領域は、無線の電波伝搬、信号処理、WLAN スタック、TCP/IP、SD インターフェース、Web アプリケーション層、製品企画および技術営業。主な使用言語は Verilog-RTL、C、HTML、JavaScript、Java、Objective-C、ヒンディー語少々。休日は2児の父を演じている。

FlashAir と秋月

tnk

はじめに

はじめましてこんにちは。秋月電子の中の人 tnk です。読み物として面白いかどうかわかりませんが、弊社が FlashAir を取り扱うことになった経緯などを書かせて頂きます。

なぜか意外なものが売れ出す

いつ程かは忘れてしまいましたが、ＳＤカードスロット（と基板キット）の販売ペースが明らかに増え始めました。マイクロＳＤカードスロットＤＩＰ化キットが売れるのはわからなくも無いのですが、正直「今更マイクロでなきゃどうにもならないでしょ…」、「在庫数十年分と表示されてるのを見て見ぬふり…」などという状態から一転。これは何かあるのかと、ＳＤカード関連を調べてみると FlashAir が話題になっておりました。

もはや家電ではない？

「あれ？ FlashAir ってデジカメにさして写真共有を無線でできるよ！って**だけ**[1]の製品じゃなかったっけ？」なぜ SD カードスロットが売れるのだろうか。

さらに調べると、「写真の共有だけじゃなくて、スクリプトエンジンが載っていて、I/O が叩ける？！」なるほど納得、極小無線機能付きマイコンモジュールとして使える製品だったとは（電源を接続したり、I/O を引き出したりするのにＳＤカードスロットを使うということも知る）。これはもはや家電領域ではなく、電子部品の領域。

電子部品ならば…

「大手企業がこんなぶっ飛んだ商品[2]を世に出すなんて、新しい風の予感、楽しそう！」「電子部品ならば、うちで扱わない手はない」「仕入れてもらおう、そうしよう」

…と言ったかはさておき、さらに当時は…

無線 LAN で何かをしようと思うと、ハードルが高かった（XBee-WiFi などもありますが、本体はお高いし、周辺に必要なものも結構な出費になってしまう）ので、無線 LAN 関連商品の拡充という点からも魅力的でした。さらに付け加えれば、国内メーカというのも我々としては大きなポイントであったりします。社内でドキュメント類の日本語化をするとなると、それだけで時間がどれだけ必要か…。何か問題が生じたときに相談し易いというのも“中の人”的にはとても大きいことなのです。

と、ざっくりこのような流れでお取り扱いさせて頂くこととなりました。

1　それだけでも充分に良い製品だと思っています。念のため。

2　私の勝手な大手企業様のイメージからすると…です。

弊社初の FlashAir 専用基板　AE-FAIO-T[3]

FlashAir を販売開始し一息ついた当たりで、FlashAir 専用の基板を商品化しようと言うお話になりました。

Airio RP をベースに、ブレッドボードでの使用を想定し、8 ピン DIP ソケットにも刺さる、300 ミル幅対応のため、ヘンテコな形状としました。アートワーク担当 M 氏を煽りつつ急ピッチで進め、何とかイベントにも間に合いました。使用した SPI-I2C ブリッジ[4]が、ベースの基板と異なっていたこともあり、一時はどうなることかと思いましたが、余熱さんにアドバイス頂き、無事に製品化（図 1）。今も順調に販売数を伸ばしております。いやはや良かったです。

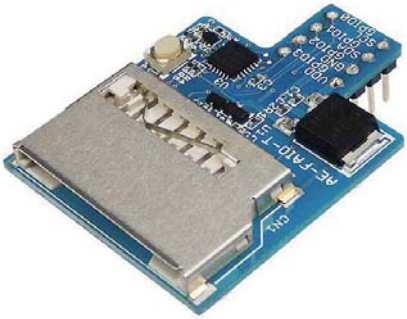


図 1: AE-FAIO-T

次回作の告知

AE-FAIO-T を販売後、もう少しシンプルな基板も欲しいとのご要望を受け、設計をすすめておりました基板 2 種が近日発売予定です。執筆時現在、既に基板の量産発注済みですので、特に問題なければ本誌配布時に販売できるかなと思います。逆を言えば、販売開始していない場合はお察しください…。

おわりに

FlashAir および関連製品の販売にこぎつけることができたのは、余熱さんを始めとする関係各位のご協力があったからこそ他ありません。この場をお借りしまして、御礼申し上げます。そして、まさかあの FlashAir 同人誌の執筆に参加させて頂けることになるとは思ってもおりませんでした。併せて、御礼申し上げます。有難う御座いました。

3　通販番号 K-10007　1 個 980 円です（2016.6 現在）。

4　SC18IS600IBS : SPI to I²C-bus interface

tnk (@tnkakzk)

学費を稼ぎに秋月八潮店バイトから始まり、電話、商品コメント、説明書作りをやりつつ基板設計、実装 (SMD) とやって今に至る。技術部所属。八潮店ガチャガチャ担当。大の猫好き。

社内レーザー加工機でいろいろ実験設計製作中！

Maker Faire Tokyo からニコニコ超会議まで

Pochio

Maker Faire Tokyo（MFT）への出展は今年で 3 回目になります。新刊の「FlashAir の同人誌」を配布するのも恒例行事となりました。今年も昨年の MFT2015 からの 1 年間、FlashAir の活動を継続できたことに感謝して、その内容を振り返ってみます。

Maker Faire Tokyo 2015 出展

MFT2015 は 2015 年 8 月 1 日、2 日に開催されました。一昨年の MFT2014 は 11 月の開催でしたが、夏休みの開催になり、お子さん連れの来場者が増えた気がします。今回は幸いにも広い通路に面してブースを構えることができましたが、お隣は（またしても）世界最大の半導体メーカーさん。圧倒的な存在感で、小型のマイコンボード「Edison」を中心とした様々な電子工作事例を展示されていましたが、ウチも負けてはいません。綾瀬ヒロさんの「ワイヤレス鉄道模型制御装置（図 1）」は、いつものように子供たちに大人気。2015 年 4 月開催の「FlashAir ハッカソン」優勝作品である光るジャグリングデバイス「NEON」や、昨年 6 月放送の「ABC ハッカソン」でサイボウズ賞を頂いた光るシュシュ「FLEURIR」など、FlashAir を活用したアイディア満載の作品を展示しました。そんな数ある展示の中で特に目立ったのが「MSX」でした。

図 1: ワイヤレス鉄道模型制御装置

図 2: 東芝製 MSX「PASOPIA IQ」

MSX とは、今から 30 年以上も前に登場した家庭用パソコンの共通規格で、現在も多くのファンに愛されています。今回 MSX に造詣の深い“せいみまさみさん”のご協力により、FlashAir を用いて製作したワイヤレスの MSX 用ゲームコントローラを展示しました。これを今では大変貴重な東芝製 MSX パソコン「PASOPIA IQ」の実機につなぎ、30 年以上前には存在しなかった大画面の液晶テレビを使って（図 2）、小さなお子さんからお父さんまで幅広い層にレトロなゲームを遊んで頂きました。このコントローラの製作記事は FlashAir 同人誌 2 に掲載されています。

ところで一昨年の MFT 2014 で FlashAir におまけ基板をつけて会場限定販売を行ったところ、あっという間に完売してしまいました。そこで味を占めた我々は再びおまけ基板を作ることにしました。今回は P 板 .com さんにご協力頂き、組み立てるとスマホスタンドになる基板と、丸いコースター基板の 2 種類をご用意（図 3）。

図 3: スマホスタンド基板とコースター基板

FlashAir を購入されたお客様にお好きな方を選んで頂きましたところ、大変好評で前回の 5 倍以上も売れました。ちなみにスマホスタンド基板は部品を実装すると、FlashAir 評価基板として販売中の Airio（えありお）と同等のものが出来上がります。またコースター基板はラジコンを作ることができます。しかしどちらも製作難度が高く、実際に作られた方は僅かだったのではないでしょうか。

図 4: FlashAir 同人誌 2

MFT2015 では前回配布した FlashAir 同人誌の続編となる第 2 号を制作し（図 4）、2000 部を無料配布しました。執筆陣を大幅に増強したところ、ページ数が前号の 2 倍になり読み応えのある一冊になりました。特に 2015 年に発売された FlashAir の最新版 W-03 ではプログラムが走るようになりましたので、遊び方の幅が大きく広がっています。FlashAir 同人誌、FlashAir 同人誌 2 はいずれも PDF 版を Flashair Developers からダウンロード頂けますので、是非この機会にご覧ください。

FlashAir が開催するイベントのあるべき姿とは

FlashAir ハッカソンを開催した縁で、各地で開催されるハッカソンイベントに API 提供企業として参加を打診されるようになりました。よい機会でしたのでイベント開催のノウハウを学ぶだけでなく、FlashAir をそのイベント参加者にご活用頂き、あわよくば新たな使い道を発見できないかと考えました。しかし、実際には FlashAir をなかなか使ってもらえませんでした。また、2 日間かけたプロトタイプ開発は傍目で見てもかなり大変で、ハッカソン参加者の顔ぶれが審査員も含めてやや固定化しつつあることに気づきました。

ハッカソンイベントでは、与えられた 3 分から 5 分程度の時間で FlashAir を紹介させて頂くのですが、いくら私が自称 FlashAir 芸人とはいえ、この短時間で「FlashAir でで

きること」を十分ご理解頂くのは相当困難です。それゆえ、「FlashAir を使えばこんなことが実現できるのでは?」という発想に至る前に、参加者個々人が使い慣れた Raspberry Pi や Arduino の活用を考えるのは自然な流れと言えます。また企業目線で見ると、ハッカソンイベントで特に重要なのはプロトタイプ開発よりもアイディア出しにあると感じました。ハッカソン参加者の顔ぶれが固定化しつつあるのは、技術に自信がないとプロトタイプ開発に参加しづらいので、初心者に対して参加障壁が高いと感じさせてしまうのではないか、と推測しました。幅広いアイディアを求めるイベントでは幅広い参加者を募ることが大切です。

そこで次の FlashAir のイベントは、アイディアソンに重点を置くことにしました。また、2 日以上のイベント開催は日程や会場確保が難しくなり、参加者にも負担をかけるので、1 日で完結するイベントフォーマットを構築したいと考えました。そして我々が期待するアイディアが生まれるように、アイディアソンの前にハンズオン形式で FlashAir の機能について理解を深めて頂く時間を用意することにしました。

FlashAir × Bluemix ハンズオン＆アイディアソンの開催

最新版の FlashAir はプログラムが走るようになったので、クラウドと接続して何か面白いことをやりたいと考えていました。しかし我々はクラウドに詳しくないため、FlashAir に興味を持ってくれそうなクラウド関係の方が現れないものかと淡い期待を抱いていました。

2015 年 6 月に、大阪の朝日放送が製作した「ABC ハッカソン」が放送されました。この収録が 5 月の連休明けに朝日放送の本社スタジオで行われたのですが、そこでスポンサーとして参加されていた日本アイ・ビー・エム（以下 IBM）のみなさんとの出会いがありました。IBM のみなさんにご挨拶した際、FlashAir をサンプルとして数枚お渡しし、IBM のクラウドサービスである Bluemix に繋げたいとお話ししたところ、Bluemix エバンジェリストの木村さんがその場で早速 FlashAir から Bluemix への接続にチャレンジして下さったのでした。とても熱心に取り組んでくださることに感銘を受けまして、次回のイベントではぜひ Bluemix と連携したいと考えていました。

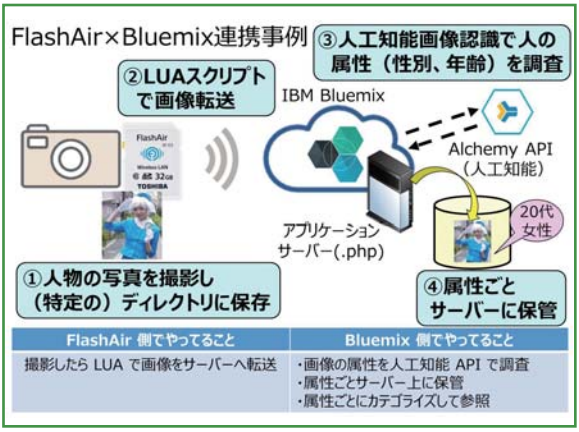


図 5: 人工知能を用いた Bluemix との連携事例

2015 年 11 月 7 日に IBM さんと「FlashAir × Bluemix ハンズオン&アイディアソン」を開催しました。イベントの後援はサムライインキュベートさんにお願いし、天王洲アイルの Samurai Startup Island を会場としてお借りしました。このイベントでは木村さんにご考案頂いた、FlashAir と Bluemix との連携事例を体験頂きました（図 5）。FlashAir が写真を Bluemix にアップロードし、人工知能が写真の人物の年齢層や性別を判定するというものです。この事例は Flashair Developers にて分かりやすいチュートリアル形式で公開しておりますので、ぜひ試してみてください。

図 6: 優勝した廣瀬賢太郎さんと審査委員長の Julie Watai さん

図 7: アイディアソンにご参加頂いた皆様

その後、FlashAir と Bluemix の連携をテーマにアイディアソンを開催しました。今回は事前に FlashAir ハッカソンでお世話になった JellyWare 代表の崔さんからアイディアソン進行のアドバイスを頂き、FlashAir 芸人がイベントを仕切らせて頂きました。ハンズオンからアイディアソンまで一通り進行してみて、とても勉強になりました。

今回のイベントではアイディアソンの審査委員長に、写真家で DJ などマルチにご活躍されている Julie Watai（ジュリ ワタイ）さんをお迎えしました。ジュリさんには FlashAir 同人誌 2 にご寄稿頂いたり、毎週月曜日に放送されているラジオ番組「Julie Watai HARDWARE GIRLS RADIO」にて FlashAir のお話をさせて頂いたりと、お世話になっております（ありがとうございます）。

FlashAir と Bluemix のハンズオンを行ったことが功を奏したのか、アイディアソンではとても面白いアイディアがいくつも出てきました。厳正な審査の結果、廣瀬賢太郎さんの、「写真がうまくなる SD カード」が最優秀賞となりました（図 6）。撮影した写真の問題を発見し、スマホがアドバイスしてくれるという優れものです。この他 3 名の方が受賞され、懇親会で参加者同士の交流を深めた後、イベントは盛大に終了しました。ご参加頂いた皆様、ご協力頂いた関係者の皆様に感謝いたします（図 7）。ありがとうございました！

ニコニコ超会議に初参加！

2015 年もあと 10 日で終わろうという頃、この同人誌の表紙などでおなじみ（？）の FlashAir 非公式応援キャラ、「閃ソラ」（以下ソラちゃん）の Twitter アカウントに突然連絡がありました。コミュニティ「プログラミング生放送」の代表の方から、一緒に 2016

年のニコニコ超会議（以下超会議）に出ませんか、とお誘い頂いたのです。実は以前に FlashAir のスタッフが超会議への出展を希望していましたが、関係者にお話を聞いたところ FlashAir での参加は難しそうだったため、あきらめていました。

ところでプログラミング生放送とは IT 勉強会を開催するコミュニティであり、「暮井慧（通称プロ生ちゃん）」というイメージキャラクターが存在します。プロ生ちゃんはコミュニティの広報的な存在で、人気声優の上坂すみれさんが CV を担当、四コマ漫画なども公開されていて界隈では有名です。そんなプロ生ちゃんとコラボさせて頂けて、さらに超会議にも参加できるなんて、こんなチャンスはめったにありません！

年明けにお話しを伺いますと、超会議に出展されるのは墨田区のタクシー会社「互助交通」さんで、何でも痛車ならぬ「痛タク」を展示されるとのことでした。いやはや、そんな斬新すぎる企画を会社に許してもらえるとは羨ましい！と思っていましたら、取締役の方がこの企画を率先されているというではありませんか（脱帽！）。互助交通さんとプロ生ちゃんは事前にコラボすることが決まっていたのですが、企画をもっと盛り上げたいとのことで、我々にお声掛け下さったそうです。大変ありがたいことです。

図 8: スマラポ！のホログラムディスプレイ

図 9: 痛タク（ラッピング装飾）

図 10: Ixy さんによるイラスト

超会議のブースでは痛タクの展示だけでなく、プロ生ちゃんのコスプレもされるということでした。これは、かつて一度だけ実現したソラちゃんのコスプレ（FlashAir 同人誌参照）を復活させる絶好のチャンスです。しかしその実現には大きな問題がありました。実は前回コスプレしてくださったタレントさんが、すでに芸能界を引退されていたのです。そこで白羽の矢が立ったのが、ＡＢＣハッカソンでお世話になったハッカソンアイドルのくーらさんでした。くーらさんはコスプレを趣味にされていらっしゃるので、二代目ソラちゃんをお願いしましたところ、大変ありがたいことにご快諾頂きました！

さらに、痛タクの前でコスプレのプロ生ちゃん＆ソラちゃんと一緒に撮った写真をプリントアウトして、プレゼントするサービスを提供することにしました。こちらは FlashAir を使って結婚式での写真共有サービスを展開されているラポ社さんにお願いし、同社のサービス「スマラポ！ Interactive」を超会議のためにカスタマイズして頂きました。撮影した写真が大型ディスプレイに表示されるだけでなく、ホログラムのように表示された写真をタッチするとプリントアウトされる近未来的なサービスです（図 8）。

超会議当日、互助交通さんのブースに登場した痛タク（図 9）はバンパーが黄金に輝き、イラストレーターの Ixy さんに描いて頂いたプロ生ちゃんとソラちゃんが、ボンネット全面に大きく描かれているではありませんか（図 10）！　実はプロの絵師さんにソラちゃんを描いて頂くのは今回が初めてです。ブースには概算で 4000 人以上の方が訪れ、コスプレのお二人と撮影したり、ラポ社さんの写真プリントサービスを楽しんだりして頂きました。くーらさんは素敵なソラちゃんになりきって、プロ生ちゃんとの素晴らしいツーショットを見せて下さいました（図 11）。

超会議に参加する機会を下さいました互助交通さんとプログラミング生放送さん、スマラポのサービスをご提供頂きましたラポ社さんに、この場を借りて感謝申し上げます。そして二代目ソラちゃんを快く引き受けて下さいましたくーらさん、本当にありがとうございました！　ご協力頂きました関係者の皆様にも感謝申し上げます。

図 11: プロ生ちゃんとソラちゃんのツーショット

次はコミケに出たいなぁとか、かなり無謀な野望を抱きつつ今後も頑張りたいと思っています。FlashAir に対する貴重なご意見や同人誌のご感想は、ぜひハッシュタグ「#FlashAir」でツイッターにてつぶやいて下さいますと大変幸いです。

Pochio (@I_love_nintendo)

自称 FlashAir 芸人。最近芸人の出番がめっきり減ったのでお仕事募集中。本業は FlashAir と直接関係のないものすごく小さなパターンの形成技術の開発で、ときどき論文書いたり学会発表したりしている。IEEE シニアメンバー、電子情報通信学会シニア会員、博士（工学）。

ソラちゃん in 超会議！

くーら

はじめまして！
ハッカソンアイドル（自称）（笑）のくーらです！

この度は、FlashAir 芸人（笑）のお誘いで、超会議で FlashAir 非公式応援キャラ「閃ソラちゃん」のコスプレをさせていただきました！
以前はハッカソンで FlashAir を使ってアイディアを出したりガジェットを作ったりしたことがありますが、ソラちゃんのコスプレというレイヤーならではのコラボも、すごく楽しみにしていました。

超会議特別仕様！ FlashAirを仕込んだひみつ道具！

超会議では、互助交通さんの「超タクシー」とプログラミング生放送の「プロ生ちゃん」と一緒に撮影会をしたり、FlashAir を入れたカメラで撮影された写真をその場で印刷してお渡しするプログラムになっていました。

ラポ株式会社さんは、ホログラム仕様に写真を映し出し、ジェスチャーで操作する装置を作ってくださいました！なんとすごい＞＜

超会議の二日間、いろんな繋がりができて、心に残るものがたくさんで 関わったすべての方に感激感謝です。 これからも、ものづくりもコスプレも頑張ります！ ありがとうございました！！

くーら (@KKKKKKKKKULA)

大阪在住。電機メーカーの UX デザイナー出身、現在はロボットの会社でデザインと企画の仕事をしています。数年前からハッカソンのご縁で IT とものづくりの世界に踏み入れて、FlashAir と出会う。普段は二次元の嫁たち（男女不問）と幸せに暮らしながら仕事を頑張っていま〜す！

「酔ったー」と「TISPY」

余熱

MFT2014 が無事に終了した 2015 年 3 月某日、Pochio さんから「酒っと」というお酒関係の同人誌即売会に出さないかと打診を受け、綾瀬さん、じむさんと「空と月[1]」という同人サークルを結成して参加することになりました。当時 Lua スクリプトが動作する FlashAir W-03 が発売になったこともあり、アルコールセンサーと FlashAir を搭載したガジェット「酔ったー」を作り、同人誌に製作記事を書きました（図 1、図 2）。

「酔ったー」はアルコールセンサーの入力値が一定以上になると「酔ったー (´・ω・`)」と twitter に投稿するガジェットです。MFT2015 などの展示会でも思いのほか評判が良く、impress さんにも記事として取り上げて頂きました[2]。

その後、社内ベンチャープログラムにて「酔ったー」を採択して頂き、本格的に企画を進めることになりました。「酔ったー」は「TISPY」と名前を変え、コンセプトも「デキるオトナのパフォーマンス管理」とカッコいいものに決まりました。サークル結成から 1 年後の 2016 年 3 月にクラウドファンディングを開始し、最終的に 1500 万円を超える支援を頂きました。

TISPY チームはデザイナーやエンジニアがバランス良く集まり、少人数であるため意思決定が非常に早いことが短期間でクラウドファンディングまでこぎつけた理由だと思います。今後もこのチームで何か面白いことができないかなぁ と画策する日々です。

図 1: 同人誌「空と月」

図 2: 酔ったー

図 3: TISPY

1　空と月 HP: http://yone2.net/sora_lua

2　FlashAir を使ったアルコールチェッカーの製作本が販売中
http://akiba-pc.watch.impress.co.jp/docs/wakiba/find/706396.html

余熱 (@yone2_net)

FlashAir 同人誌の編集担当。それから TISPY の PJL。最近、自分が何屋なのか不明になることがしばしば。

FlashAir チームに配属になってからちょうど 1 年経ちました。今後も FlashAir の拡販に力を入れていきたいと思っています！

クラウドファンディングに向けた製品続々！

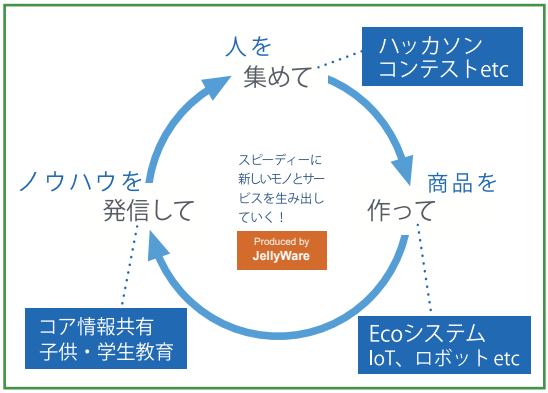
JellyWare（ゼリーウェア） だん、Hiro

はじめに

今回は、2015 年 4 月に行った FlashAir ハッカソンを経て、商品化へ向けた開発活動を続けているチームの中から生まれつつある「KittySee」と「Neon」の 2 つの製品をご紹介します。

JellyWare について

ここでは、これらの製品開発を行っている JellyWare について簡単にご説明させていただきます。JellyWare は、ハッカソン・アイデアソンを主としたオープンイノベーション活動の企画・運営及び、オープンイノベーション活動から生まれたアイデアの具現化を支援し、製品企画、設計・開発、プロモーション、販売まで行っております。



社名の JellyWare の「ゼリー」はハードとソフトの両方の質感を兼ね備えているという意味で、主に IoT 製品を得意としています。今後も続々とイベント開催や製品リリースを行ってゆきますので、WEB ページ[1]にご注目ください！

KittySee（仮称）

ハッカソンチームメンバーの中の、ラジコンヘリとネコが好きな二人のエンジニアがヘリコプターのカメラ機能って意外とすごい、と雑談をしていたことがきっかけでこのアイデアは生まれました。

ヘリ用カメラをもし猫につけたら、猫が普段何をしているか知りたいという飼い主の欲求を満たすことができるかもしれないと話題になり、そこから猫が興味を持った時には動作を止めるという習性を利用すれば、猫の気持ちが反映された面白い写真が撮れるはずだ！との発想から、このアイデアが生まれました。

1　JellyWare ウェブページ : http://jellyware.jp/

さらに、猫の飼い主同士をつなぐ、「ねこサーバ」を作って、写真や活動の様子を自動でアップロードしてみんなで共有できたら面白い、というベースコンセプトまで話が発展し、これらのアイデアは FlashAir を用いて実現できると確信し、開発を始めることにしました。

図 1: KittySee（仮称）

製品では、アイデアの基本となった猫の動作の分析を基に、飼い主が関心を持つ以下の機能の搭載を予定しています。

1. モーションセンサー搭載　→　活動量が分かる！
2. 写真撮影機能　→　猫の普段の生活を記録！
3. 声の分析　→　猫の気持ちが分かる！

今後は、Maker Faire Tokyo 2016 にて展示をした、ベースモデルに対するフィードバックを基に製品の仕様を固めて、2017 年 3 月クラウドファンディング開始を目指して製品開発をスピーディーに進めます。

Neon

Neon とは、複数の FlashAir で互いの情報を送信しあい、同期を取りながらパターン発光するスマートジャグリングボールです。大道芸で 3 個以上のボールを手元で投げる「トスジャグリング」向けのガジェットで、BGM に合わせて光り方のパータンを変えるのが特徴的です。

図 2: Neon

技術的には、FlashAir の共有メモリ機能を活用しています。

2015 年に行われた FlashAir ハッカソンで最優秀賞を獲得したチーム Neon は、ハッカソン終了後も開発と検討を続けています。

主な開発内容

- 外観の美しさと表現力を求めた、LED32 個及び LED92 個搭載の筐体
- 投げられるボールへ向けた、FlashAir 搭載可能でコンパクトな電子基板
- 楽曲のタイムラインに対してどのように光らせるかを直感的に作れるアプリ
- 加速度センサーを使って動きに合わせて光り方を変えるファームウェア

また活動面においても、Maker Faire Tokyo 2015、MUSICIANS HACKATHON 2015、CEATEC JAPAN 2015、Inter Bee 2015、GUGEN コンテスト 2015 などに出展・参加を行ってきました。

現在は Maker Faire Tokyo 2016 へ向けた展示品の作り込みを行っており、その後はクラウドファンディングへ向けて Neon の開発をスピードアップさせます。

今後に向けて

2016 年 6 月に、SAMURAI ISLAND EXPO 2016 出展しました。

大好評かつたくさんのご意見を頂きましたので、開発にフィードバックしてゆきます。

さらに Maker Faire Tokyo 2016 では、よりバージョンアップしたプロトタイプを展示します。クラウドファンディングでリリースした際は、皆さんぜひチェックしてみて下さい！

図 3: SAMURAI ISLAND EXPO 2016 の様子

だん (@dandan001) / Chief IoT Imagineer@JellyWare

夢を形に変える皇子。組込ソフト開発、電気設計、Azure 構築業を経験。ディズニーが好きすぎて、イマジニアを名乗るべく、今年 4 月に JellyWare に移籍。趣味でも仕事でもハッカソンに参加中。

Hiro / Chief Technology Officer @ JellyWare

電子工作好きのアルパカ。大手電機メーカーにて電気設計担当として活動していたが、ハッカソンに夢中になり過ぎて、昨年 12 月に JellyWare に移籍。現在はソフト開発や PL、イベント運営なども行い、放牧な日々を過ごす。

FlashAir との出会いと自作ツールのお話

GPS_NMEA

FlashAir との出逢い

最初知った時には正直あまり魅力を感じなかった

私が FlashAir を初めて知ったのは twitter 上で、2014 年の 11 月ごろのようだ。当時、フォロワーの人が FlashAir の使い道を楽しげに考えているつぶやきをしていたのと、どこかのニュース記事で出ていたじむ氏の「ワイヤレスクレーンゲーム」を見たのを覚えている。当時発売されていた FlashAir W-02 への私の印象は「ブラウザから JavaScript で I/O 制御ができ、ついでにマイコンから無線 LAN モジュールとしても扱える " だけ " の特殊な SD カード」というものだった。というのも当時は、ブラウザから I/O ポートを操作できることにも、マイコンから使えるということにも興味がなかった。ただ、SD カード型のインテリジェントな機器としての魅力は感じており、「自作のプログラムが走ればなぁ」と思ったのを覚えている。

それからしばらく忘れ去り、2015 年初頭。海外のニュースにて、近々発売する FlashAir が Lua スクリプトに対応する、という記述があったのだ。幸い、Lua には何度か触れたことがあったため、「こりゃ面白そうだぞ」という印象に変わった。

「8G で 4000 円するお高い SD カード買ったぞ」

実際に手にとったのは 2015 年 3 月 25 日。家電量販店で購入後の車内にて、初めてのつぶやきが、見出しのそれである。

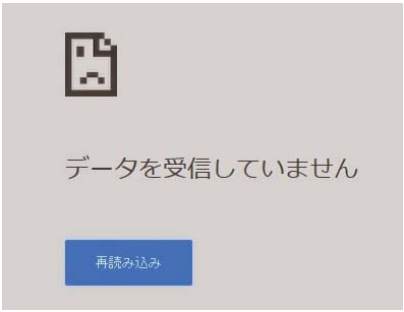


図 1: 出力なしエラー

FlashAir の Lua 環境には初日から苦労した。Lua では通常、文法エラーなどはコンソールに出力されるが、FlashAir にはコンソールがない。ブラウザへも出力されない。そのため、エラーが起きた時にはなにも出てこない（図 1）。動くか、動かないか、それだけだった。それで困り果てて作成したのが debug.lua である。これは Lua の保護実行機能を使って、本来捨てられてしまう様々な Lua のエラー情報をキャッチし、ブラウザへ流すものだ。これで FlashAir 上でのデバッグが飛躍的に楽になった。

そうして一番初めに完成したのが、ポメラ上で Lua スクリプトを編集し実行するための環境だった。ポメラはテキスト編集端末であり、プログラムを実行できない。FlashAir はちょっと手を加えればテキストとして保存されたスクリプトを実行できる。これを組み合わせると、見かけ上ポメラをプログラマブルな端末に見せかけられた。この FlashAir とポメ

ラの組み合わせから生まれる無限の可能性を、いち早く知ってもらいたいと思い、動画も作成した（図 2）。

次にやろうとしたのは、テキストブラウザの作成だった。ポメラでの文書作成中に、とっさに調べ物をしたくなった時に使えそうだと思ったのだ。しかし通信機能を本格的に使い始めると、深い泥沼へと沈み込むこととなった。当時の FlashAir のファームウェア W3.00.00 には、様々な不具合があった。例えば、関数をある数以上ネストするとフリーズするというものや、fa.HTTPGetFile が破損データを出力するというものなどだ。この辺の不具合を知った時期から、自身のサイトに情報を上げるようになった。当時 W-03 は出たばかりの商品。Web 上を探してもほとんど情報はなく、自分の試行錯誤の記録も兼ねていた。

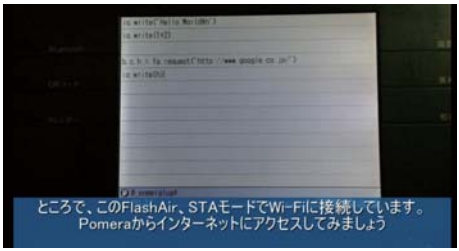


図 2: 最初に作成した動画

図 3: 解説動画

そうこうしながら、現行の不具合は小細工ではどうにもならないと判断。ポメラ以外での使いみちを探るようになった。W-02 からの定石であれば、マイコンから無線 LAN モジュールとして使うと便利なことはネットを見ていてよくわかったのだが、それでは W-02 を使うのとほとんど何も変わらない。「であれば W-03 で搭載された Lua スクリプト機能を電子工作で活用すればいいじゃないか」という考えに至り、それまで使っていなかった I/O ポート機能を扱うようになった。

やったことといえば、スマホに通知を投げてみたり、Arduino 風の IO 操作ライブラリを作ってみたり、ソフト SPI を実装してみたり、I2C マスタを移植して液晶を制御してみたり。

超高級言語が使え、書き込み器も不要なマイコンとしての FlashAir は使い慣れると非常に便利なのだが、twitter を見ていると、未だ多くの人の印象は " 単に写真が共有出来るだけの SD カード " のまま。そこで私はマイコンとしての FlashAir の存在を広めようと、いくつか動画も作成した（図 3）。

そのうち、私はカバンに FlashAir を常に忍ばせるようになった。思い立った時にいつでもスクリプトを組み、試せるようにと。それだけ、FlashAir は私にとって手に馴染んだ存在になった。

ツールの開発経緯

ここからは話題を変えて、宣伝も兼ねてツールの開発経緯を語っておこうと思う。

FlashAir Unofficial Configurator

FlashAir の CONFIG ファイルを GUI で生成するツール（図 4）。開発の動機は、固定 IP アドレスを設定できることがわかったことだったと思う。当時はその設定すら開発サイトにない隠し機能であったが、iSDIO 簡易仕様書を漁っていたときに設定項目があるのを偶然見つけたのだった。FlashAir の設定項目は数が多く、さらに、各種モードは数値で設定する都合上、どういう意味なのかわかりにくいことも多い。その上、公式の設定ツールではそのうちのほんの僅かしか設定できない。この状況を改善したいが、解説ページを作ったとしても FlashAir　Developers の焼き直しになる。ならばいっそ、と作ったのがこのツールだった。

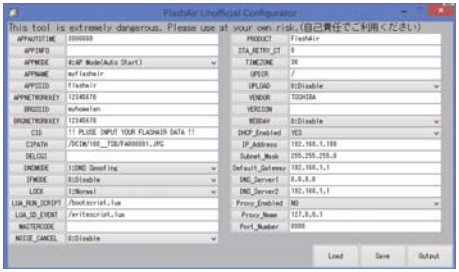


図 4: FlashAir Unofficial Configurator

FlashTools GPIO Tester & Checker

ブラウザ上のチェックボックスを操作することで I/O ポートを制御できるツール（図 5）。開発経緯はシンプルで、今は無き動画のためである。FlashAir の特徴として、ブラウザから I/O ポートの制御ができるというのがある。そのためには CGI にコマンドを投げればよいのだが、プログラムを組んでいない状態では、16 進数を計算して URL に直打ちするしか無い。Raspberry Pi なんかには WebIOPi という、ブラウザから I/O ポートの動作確認できる環境があるのに、FlashAir にはそういうのがない。これでは動画で「あなたも簡単に！」とは言いづらいだろう。しかし、比較的容易に作成できそうだと思い立ち、HTML と JS で組んだのがこれである。これを使えば、LED やモータのオンオフくらいならできるため、知人に IoT 的な制御デモをぱっと見せるのに役立っている。

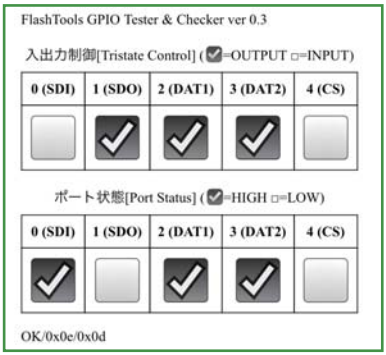


図 5: GPIO Tester

FlashTools Lua Editor (FTLE)

ブラウザで動くスクリプトエディタ。編集・アップロード・実行・デバッグまでを 1 画面で完結して行える簡易的な環境である。開発動機は、「抜き差しを繰り返し続けると PC の SD スロットがいつか壊れそう」と思ったのと、「カフェで回路組んで、タブレットからスクリプトのパラメータ変えてデモとかできたら、カッコ良さそうだ」という、単純なものであった。

初期の FTLE は FlashAir Tiny Lua Editer という名前だった（図 6）。その機能は実行とデバッグのみ、エディタは単なる入力エリアで、使い勝手の良くない単なる技術デモであった。ただ、「なにも設定していないまっさらな FlashAir に突っ込んでも開発できる」というのができれば、イベント等でも役立つと考え、1 つの Lua ファイルに FTLE と CONFIG 書き換え機能を内蔵した FTLE-All-in-One というのも作成して公開していた。今は技術上の理由で zip 配布に戻してしまったが、このコンセプトは今の FTLE の AutoSetup 機能に生きている。

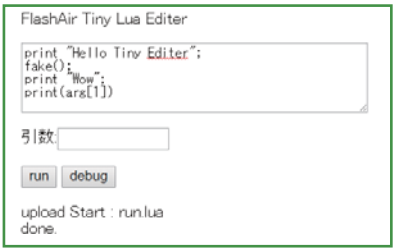


図 6: 初期の FTLE

ツールとして実用的なものとなったのは大分後の話で、きっかけは FlashAir のファームウェアアップデートだった。新しく追加された機能の分析のために FTLE を初めて本格に使用した。その時に使いにくさに気がついた。ちょうど同時期に、CodeMirror というブラウザ上で動作するリッチなコードエディタのライブラリを発見。自分の抱いていた不満なども合わさり、開発意欲に一気に火がついた。CodeMirror に対応させるだけでは飽きたらず、ボタン配置やデザインも工夫し現代的に。サンプルや IO Tester も収録した。実用的になればなるほど、自分で使うようになる。使えば不満が生まれ、さらなる改良を重ねる。そうして出来上がったのが現在の FTLE である（図 7）。

図 7: 現在の FTLE

ここに上げたものは全て拙作のサイト (https://sites.google.com/site/gpsnmeajp/) から入手・閲覧できるので、気になった方はぜひ見てみて欲しい。

GPS_NMEA(@Seg_faul)

自サイト " おまけ程度のツール置き場 " で非公式ツールなどを作って置いている人。

趣味は電子工作とツールづくり、あと小説と動画作り。

FlashAir IoT Hub

南

FlashAir を使って何か面白いものを作ったら、スマートフォンなどでどこからでも操作したくなるのが人情というものです。ここでは、そんな時の手助けになるかもしれない Web サービス「FlashAir IoT Hub」をご紹介します。

どこからでも FlashAir にアクセスしたい！

購入したばかりの FlashAir は、無線 LAN アクセスポイントモードで動作するように設定されています。アクセスポイントモードでは、FlashAir の通信圏内にあるスマートフォンから FlashAir に接続して、デジタルカメラで撮影した写真などを閲覧することが出来ます（図 1）。

図 1：アクセスポイントモードでは通信圏内に居ないとアクセスできない

「どこからでも」FlashAir にアクセスするためには、FlashAir をインターネットに接続する必要があります。そのために、FlashAir には無線 LAN ステーションモードが搭載されています（図 2）。ステーションモードの設定方法は FlashAir Developers の該当ページ[1]を参照してください。

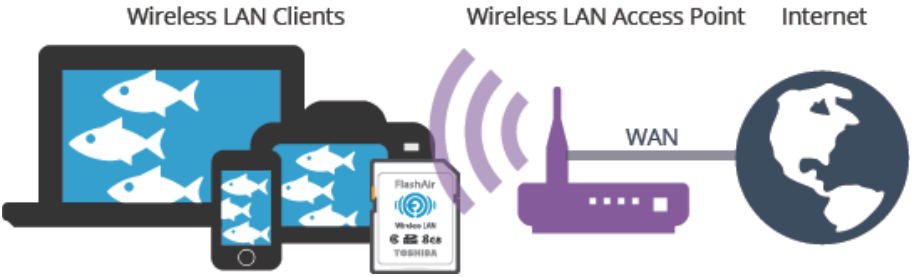


図 2：ステーションモードにすればインターネット経由でどこからでもアクセス可能

1　ステーションモードの利用 : https://flashair-developers.com/ja/documents/tutorials/advanced/1/

でも、セキュリティが心配…

ステーションモードに設定された FlashAir は、ご家庭の無線 LAN ルータなどを介してインターネットに接続することができます。しかし、一般的な無線 LAN ルータはセキュリティを守るために、インターネット側から家庭内ネットワークにアクセスできないように設定されています。このことを承知の上で無線 LAN ルータの設定を変更すればインターネット側からアクセスできるようになりますが、小さな FlashAir をインターネットの脅威にさらすことは得策ではないでしょう。

サーバを使って解決

そこでサーバを利用します[2]。インターネット上に設置されたサーバに FlashAir 側からアクセスすれば無線 LAN ルータの設定を変更しなくて済みますし、信頼できるサーバに対して適切に暗号化通信（TLS など）を使って接続すれば、セキュリティの心配もほとんどありません。

とはいえ、誰でもサーバを用意できるわけではありませんし、同じようなサーバを色々な人が作るのも効率が良いとは言えません。そこで FlashAir IoT Hub の登場というわけです。

FlashAir IoT Hub

FlashAir IoT Hub は、FlashAir のためだけに作られたサーバです。FlashAir からデータをアップロードして可視化したり（図 3）、スマートフォンから FlashAir を遠隔制御したり（図 4）[3] といったことができるようになります。

執筆時点の機能は次のとおりです。

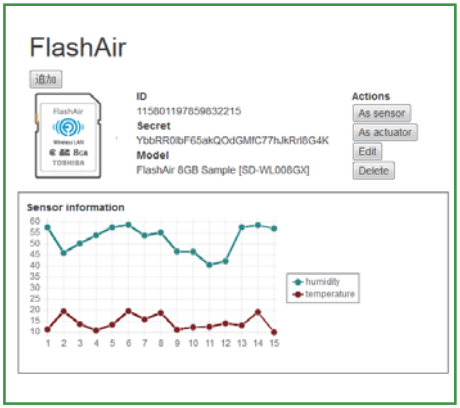


図 3: 試作段階の画面（グラフ表示）

2　無線 LAN ルータの VPN 機能などを利用する方法もありますが、ここでは説明しません。

3　PIO control と書いているのに、謎の文字列 ("abc") が入力されているのはお察しください…。

GPIO 入力

FlashAir への GPIO の入力状態を定期的にアップロードすることで、high から low、low から high といった変化イベントをサーバ側で検出することができます。

GPIO 出力

FlashAir からの GPIO の出力状態を遠隔制御で変更することができます。L チカするならこれが使えますね！

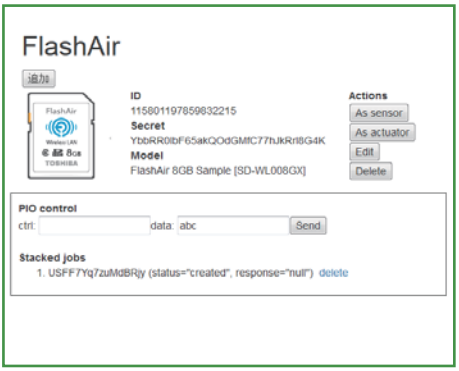


図 4: 試作段階の画面（遠隔制御）

計測値

温度センサなどから読み出した計測値をサーバへアップロードすることで、簡単にグラフ表示することができます。

スクリプト実行

FlashAir に予め保存した Lua スクリプトを遠隔制御で実行することができます。その際に任意の引数を一つだけ渡すことができます。

この他にも、FlashAir を使う上で便利な機能を順次追加していく予定です。

API 公開について

FlashAir は、API を始めとする様々な技術仕様が公開されていることが大きなポイントとなっています。FlashAir IoT Hub も例外ではなく、サーバにアップロードされたデータを取得したり、遠隔制御の指示を送ったりするための REST API 仕様を一般公開する予定です。API 公開の際には OpenID Connect や OAuth 2.0 のような標準仕様に基づいてご提供する予定ですので、楽しみにお待ちください。

FlashAir 向け SDK

FlashAir から FlashAir IoT Hub に接続するためには Lua スクリプトを使用します[4]。この Lua スクリプトは基本的に開発者の皆さんに開発していただく必要がありますが、開発を簡単にするための SDK とサンプルコードをご提供する予定です。

4　第三世代（W-03）以降の FlashAir が必要です。

裏側の話を少々

突然ですが、FlashAir IoT Hub のフロントエンド（Web ブラウザで開く画面）は React[5] を使って実装しています。細かく書くと webpack + Babel (ES2015, JSX) + React + Bootstrap 3 みたいな感じです。React で書くのは結構辛かったので、次の機会があったら RIOT[6] を試してみようかな…。もし皆さんとお話しする機会があったら、フロントエンド開発のお話もしてみたいですね！ ちなみにバックエンドサーバは Go 言語で実装しています。

おわりに

FlashAir IoT Hub はこれまで、第 19 回組込みシステム開発技術展（ESEC）や Interop Tokyo 2016 において参考出展という形でご紹介してきました（図 5）。

皆さんがこれをお読みになっている頃、無事に FlashAir IoT Hub を世に送り出せていたなら、是非一度使ってみて頂き、率直なフィードバックを頂ければ幸いに思います。

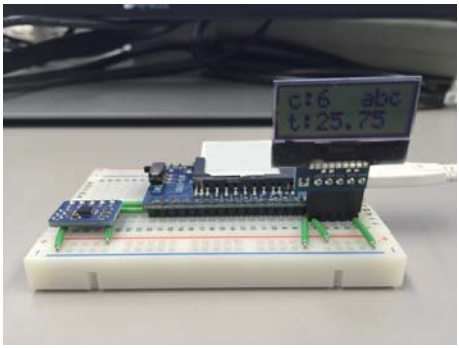


図 5: 温度計の計測値をアップロードし、文字列 ("abc") をスクリプト実行により表示するデモ

5 https://facebook.github.io/react/
6 http://riotjs.com/ja/

南 (@Nang_JP)
Web 系エンジニア。
Web サーバが動く SD カードがあると知り何かに使えないかなーと考えているうちに、L チカできるとか、Lua スクリプトが動くとか言われて目が離せなくなる。
次は何が出るのかなー。

FlashAir から Google ドライブへのファイル転送

日髙 謙太朗、寺田 賢司

今回は、Lua スクリプトを使用することにより FlashAir に保存されたファイルを Google ドライブへ転送する方法をご紹介したいと思います。この技術を応用すれば撮影した写真を自動で Google ドライブへ転送できるようになりますので、是非ご活用ください。

プロジェクト作成と認証情報の取得

まず Google アカウントを取得してください。Google アカウントを取得後、Google API Console[1]（図 1）へアクセスし、プロジェクトを作成しましょう。右上の [プロジェクトを選択] を押下しプロジェクト作成後、[Google Apps API] > [Drive API] を押下し、[有効にする] を押下します。

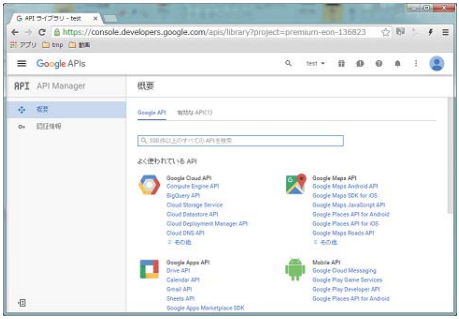


図 1：Google API Console 画面

次に、認証情報を作成します。[認証情報] > [認証情報を作成] を押下し、「OAuth クライアント ID」を選択し、アプリケーションの種類は「その他」を選択します。認証情報取得後はクライアント ID とクライアントシークレットをメモしておきましょう。

※認証情報を初めて作成する場合は、OAuth 同意画面の設定も必要となります。

デバイスコードとユーザーコードの取得

デバイスコードの取得には POST リクエストの送信が必要になります。取得方法はいくつかありますが、今回は GoogleChrome 拡張機能の [Postman]（図 2）を使用します。

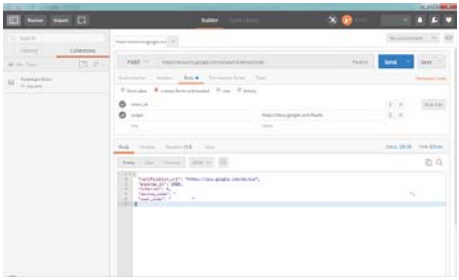


図 2：Postman 画面

Chrome ウェブストアから [Postman] を検索・インストール後、ログインを行い、送信方法を [POST]、[Authorization] > [type] は [OAuth 2.0] を選択します。次に [Body] を選択し、[application/x-www-form-urlencoded] を選択後、下記の [key] と [value] を設定します（表 1）。

1 https://console.developers.google.com/apis/

その後URLに[https://accounts.google.com/o/oauth2/device/code]を入力し、[Send]を押下するとレスポンスが返却されますので、verification_urlとデバイスコードとユーザーコードをメモしておきましょう。

デバイスコードを有効にするため、ブラウザで[verification_url]に記載されたURLへ移動し、表示された画面にユーザーコードを入力、[次へ]>[許可]を押下するとデバイスコードが有効になります。

表1：デバイスコードとユーザーコード取得のリクエスト

key	Value
client_id	{OAuth 2.0 クライアントのID}
scope	https://docs.google.com/feeds

リフレッシュトークンの取得

次にリフレッシュトークンを取得します。[デバイスとユーザーコードの取得]時の設定はそのままに、[key]と[value]のみ下記に変更します（表2）。

その後URLに[https://accounts.google.com/o/oauth2/token]を入力し、[Send]を押下するとレスポンスが返却されますので、リフレッシュトークンをメモしておきましょう。

表2：リフレッシュトークン取得のリクエスト

key	Value
client_id	{OAuth 2.0 クライアントのID}
client_secret	{OAuth 2.0 クライアントのシークレット}
code	{デバイスコード}
grant_type	http://oauth.net/grant_type/device/1.0

FlashAirの設定

転送を行うためFlashAirの設定を行います。デフォルトの設定ではFlashAirはAPモードに設定されていますが、APモードではWANに接続できないため、STAモードに変更します。FlashAirのCONFIGファイルの[APPMODE]、[APPSSID]、[APPNETWORKKEY]、[APPAUTOTIME]の値を変更します。[2]

```
[Vendor]

CIPATH=/DCIM/100__TSB/FA000001.JPG
APPMODE=5
APPSSID={接続先のSSID}
APPNETWORKKEY={接続先のSSIDのパスワード}
VERSION=FA9CAW3AW3.00.01
CID=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PRODUCT=FlashAir
VENDOR=TOSHIBA
APPAUTOTIME=0
```

変更後ファイルを保存し、FlashAirを再起動（抜き差し）し、同じ接続先（ルーター）に接続しているPC、又は端末でFlashAirに接続できれば完成です。[3]

2　詳細については FlashAir Developers(https://flashair-developers.com/ja/) をご参照ください。

3　FlashAir のIPアドレス固定方法などは FlashAir Developers(https://flashair-developers.com/ja/documents/api/config/) をご参照ください。

FlashAir の設定

転送するための Lua スクリプトを用意していきます。まずは認証部分の説明になります。ここではこれまでに取得した、クライアント ID、クライアントシークレット、リフレッシュトークンを設定します。

```
local cjson = require "cjson"
local client_id = "{クライアント ID}"
local client_secret = "{クライアントシークレット}"
local refresh_token = "{リフレッシュトークン}"
```

次にリフレッシュトークンを使用してアクセストークンを取得する関数を用意します。

```
local function getAuth()
  local mes="client_id="..client_id
  mes=mes.."&client_secret="..client_secret
  mes=mes.."&refresh_token="..refresh_token
  mes=mes.."&grant_type=refresh_token"
  local length = string.len(mes)
  print("Sending: ["..mes.."]")
  print "¥n"
  b, c, h = fa.request{
    url = "https://accounts.google.com/o/oauth2/token",
    headers = {
        ["Content-Type"] = "application/x-www-form-urlencoded",
        ["Content-Length"] = length,
    },
    method = "POST",
    body=mes,
  }
  local tempTable = {}
  tempTable = cjson.decode(b)
  local access_token = tempTable["access_token"]
  return (access_token)
end
```

その次に転送を行う関数を用意します。ここで「-- 転送するファイル」と書かれている行に、転送を行いたいファイルのパスを指定します。

```
local function uploadTest(token)
  filePath="/20160101.jpg" -転送するファイル
  local fileSize = lfs.attributes(filePath,"size")
  b, c, h = fa.request{
    url = "https://www.googleapis.com/upload/drive/v2/files",
    headers = {
      ["Content-Type"] = "image/jpeg",
      ["Content-Length"] = fileSize,
      ["authorization"] = "Bearer "..token,
      ["uploadType"]="media",
    },
    method = "POST",
    file=filePath,
    bufsize=1024*10
  }
end
```

最後に作成した関数を呼び出します。

```
access_token = getAuth()
uploadTest(access_token)
```

アップロードスクリプトファイルの実行

Lua スクリプトの作成ができましたら、ブラウザから作成した Lua スクリプトのファイルを実行 (http://192.168.0.10/upload.lua) してください。しばらくすると Google ドライブにファイルが転送（図 3）されているかと思います。

図 3：FlashAir から Google ドライブへのファイル転送

日髙 謙太朗 、寺田 賢司

FlashAir Developers の記事執筆と運営を担当しています。

皆様のお役に立てるような記事を作成してきたいと思っておりますので宜しくお願い致します。

猫でもできる！見守りシステム

大阪

全世界の猫ファンの皆さま、Hello, World! 愛しの子猫ちゃん（図 1）の様子が気になって、一歩も家の外に出られない日々をお過ごしではありませんか？ そんな悩める皆さまに代わって猫をお留守番してくれる素敵なシステムの作例をご紹介します。

図 1: 愛しの子猫ちゃん

センサとデータ

お留守番でいちばん気がかりなのは室内環境です。暑すぎても寒すぎてもいけません。次に気がかりなのは猫ちゃんたちの運動会の様子です。盛り上がりに欠けるなら、体調が悪いのかもしれません。そこで、室温と猫の動きを、温度センサと焦電センサで把握したいと思います。

データの流れはこうです。FlashAir でセンサの値を定期的に取得し、WEB 上の DB に送信します（図 2）。DB に格納されたセンサデータは時系列にまとめられ、スマホのブラウザでグラフィカルに表示されます。

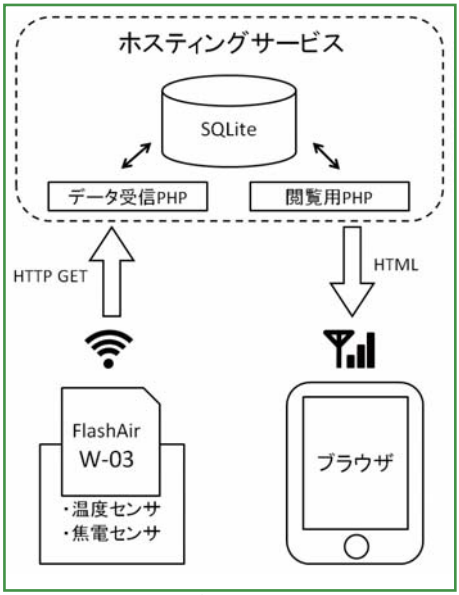


図 2: データの流れ

火事は嫌だニャー

FlashAir とセンサを搭載した「送信端末」はコンセントに固定します。ただし **100V を直接工作するのは危険を伴います。**

そこで市販の USB アダプタを利用します。コネクタで 5V を引き出して、それを電源として利用しています（図 3）。

また、各モジュールの消費電流の合計が、アダプタの定格に収まっていることを確認します。コネクタはプラス・マイナスを間違えやすいので、極性をテスタ等で調べてから、はんだ付けに取り掛かります。

見守り端末なので、安全第一なのです。

図 3: USB コネクタ

はんだ付け

送信端末は 5 つのモジュール（表 1）をリード線ではんだ付け（図 4）して作成しています。温度センサのインタフェースは SPI なので、GPIO 端子のうち 4 つを温度センサに繋ぎ、1 つを焦電センサに繋ぎます（図 5）。

図 4: 概観

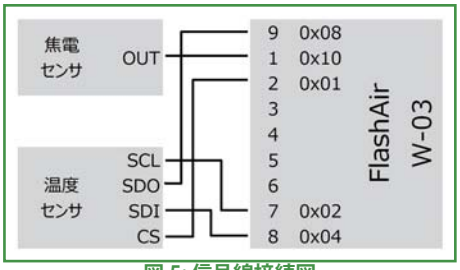


図 5: 信号線接続図

表 1: 部品一覧

部品名称	入手先	参考価格
FlashAir(W-03)	家電量販店	オープン
SD カード DIP 化モジュール	秋月電子通商	250 円
ADT7310 使用温度センサモジュール	秋月電子通商	500 円
焦電型赤外線センサモジュール	秋月電子通商	400 円
USB 電源アダプタ	家電量販店	300 円程度

Lua スクリプト

電源投入後に、無線 LAN 接続待ちのため 1 分間待機します。その後は永久ループで定期的にセンサの情報を HTTP リクエストで WEB サーバに送信します。

```
-- 温度データ送信の例
fa.request("http://***.com/sensor_temp.php?location=home&name=cat_1&stat
us=30.0625")
```

なお、関数 fa.request の送信エラーはすべて無視するという割り切りアプローチをとっています。

一般家庭では電子レンジによるジャミングなど、様々なエラー要因が考えられますが、送信端末側で回避するのは大変です。そこでエラーを前提に「下手な鉄砲数打ちゃ当たる」方式で、送信回数を増やすことで対応しています。

たとえば、DB は温度を 1 時間ごとに記録しますが、端末は 30 分ごとに送信していますので、1 回取りこぼしても問題ありません。また焦電センサは 36 秒ごとに（1 時間あたり 100 回）のサンプルを取るので、何回か取りこぼしても大丈夫です。

さあ動かそう

送信端末の電源を入れたあとは、データがたまっていくのをソワソワしながら待つのみです。さいわい動作はとても安定しており、設置以来ずっと止まることなく猫の見守りを続けています。スマホ用の HTML は PHP で動的に生成しています。グラフを見ると、どうやら夜中の運動会は盛況なようです（図 6）。……なんだか急に近隣の苦情が心配になってきました。

ね、簡単でしょう?

FlashAir の魅力は敷居の低さにあります。Windows のメモ帳でスクリプトを書くだけで、ソフト開発が完結できます。メモリ管理や通信の制御といった面倒な部分はカードの中の小人さんがやってくれるので、開発者は必要なことだけに集中できます。

ハード面の敷居はないと言えます。FlashAir は部品ではなく最終製品だからです。家庭環境（温度・湿度・電波）で、きちんと動作します。設置例は外見こそ不安ですが、動作は信頼できます（図 7）。

次は貴猫の番です!

この敷居の低いところから電子工作の世界に入り、「あっ! 動いた! 自分にもできたニャー!」という感動を味わってくれる猫が、もっともっと増えてくれたら、それが私の最高の喜びです!（おい）

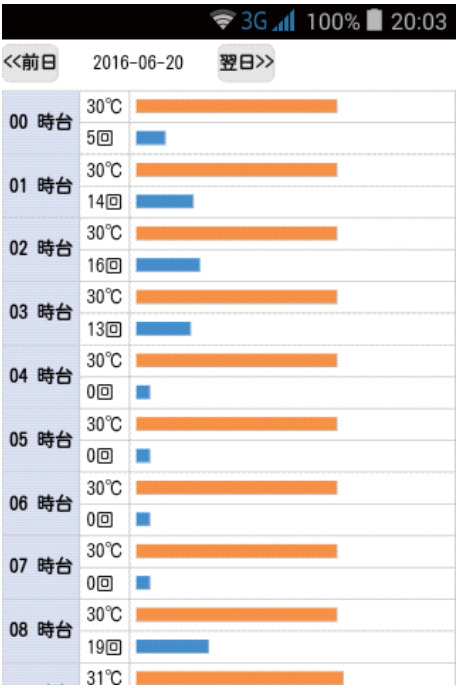


図 6: ブラウザ表示

図 7: 設置例

私も初心猫卒業を目指して、今後も精進したいと思いますニャー。（おいおい）

大阪 (@hellosaka)

人間関係が苦手でプチリタイアばかりしている不良システムエンジニアです。ことしの春に子猫を保護しました。今では猫の奴隷です。この見守りシステムを売りまくって大金持ちになりたいです。

FlashAir を使って地図アプリを作成する

清水 正行

初めまして清水と言います。普段は群馬県でデータビジュアライゼーションや GIS（地理情報システム）などをいじっています。今回は、FlashAir を使ってインターネットに繋いでいなくても利用できる地図アプリの作成方法をご紹介します。

Web 地図の仕組み

地図アプリを作成する前に、簡単ですが Web 地図の仕組みについて説明しておきます。

一般的な地図サービスでは、256x256 ピクセルに分割した地図画像（タイルと呼びます）を特定の規則にそって並べることで、フロントエンドで地図を表示します。ズームレベルに応じて必要となるタイル数が増加します（図 1）。

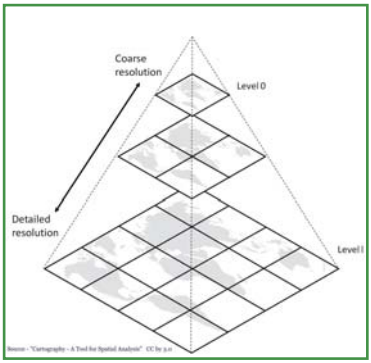


図 1: 地図タイルの埋め込み規則

地図タイルは自分で作成することもできますが、既存の地図タイルからダウンロードすることで非常に簡単に地図を表示することができます。ただし、Google Maps などほとんどの地図サービスでは地図タイルを直接ダウンロードすることは禁じられています。そこで今回は、ダウンロードが許可されている国土地理院から提供されている地理院タイル[1]を使って地図を作成します。地図タイルを表示するためのクライアントライブラリには leaflet.js[2] を使用します。

地図タイルのダウンロード

地図タイルをダウンロードするには、オープンソースの GIS ソフトである QGIS[3] を利用します。公式サイトから QGIS をインストールし「プラグイン」メニューから下記 2 種類のプラグインを追加してください。

地図をダウンロードするのに必要なプラグイン

- TileLayerPlugin – タイルを読み込むためのプラグイン
- QMetaTiles[4] – 読み込んだタイルをダウンロードするプラグイン

1 地理院タイル : http://maps.gsi.go.jp/development/ichiran.html
2 leaflet.js: http://leafletjs.com/
3 http://www.qgis.org/ja/site/
4 QMetaTile は実験的プラグインになるので、プラグイン管理メニューの「設定」で「実験的プラグインも表示する」の項目にチェックをいれて、検索してください

地図をダウンロードする手順ですが、下記リポジトリより地理院タイルの定義ファイル[5]をダウンロードし TileLayerPlugin に地理院タイルを追加します。

地理院タイルのなかから、表示したいタイルを選択し QGIS 上に表示します（図2）。必要となる地域が表示されている状態で QMeataTiles を実行するとタイル画像がズームレベル毎にローカルに保存されます（図3）。

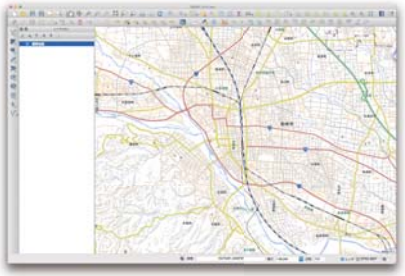


図 2: QGIS に地理院タイルを読み込こむ

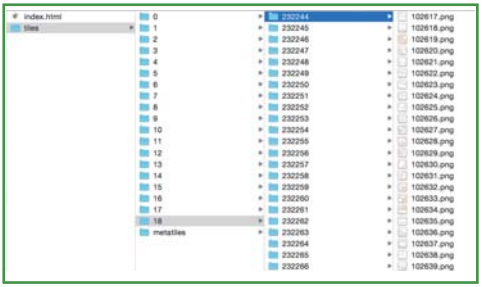


図 3: ダウンロードしたタイル画像

地図を表示する

ダウンロードしたタイル画像を FlashAir にコピーし、地図を表示するための html ページを作成します。コピーした tiles フォルダと同じ階層に index.html を作成し、leaflet.js を読み込んで地図を表示します。

```
<!DOCTYPE html>
<html>
<head>
<style>
html, body, #map {
     width: 100%;
     height: 100%;
}
</style>
<link rel="stylesheet" href="../libs/leaflet/leaflet.css" />
</head>
<body>
<div id="map"></div>
<script src="../libs/leaflet/leaflet-src.js"></script>
<script>
var map = L.map('map').setView([36.3219088  , 139.0032936], 14);

L.tileLayer(
     './tiles/{z}/{x}/{y}.png',
     {
              attribution: ' 国土地理院',
```

5　地理院ファイル定義ファイル https://gist.github.com/minorua/7654132

```
                minZoom:6,
                maxZoom: 18
        }
).addTo(map);
</script>
</body>
</html>
```

地図アプリにアクセスする

すべての作業が終わったら、FlashAir のアクセスポイントに接続して地図が正しく表示されるか確認します。

ブラウザから http://flashair/ を入力し作成した index.html にアクセスします。正しく地図が表示されてズームやパン等の操作を行っても適切にタイル画像が読み込まれれば完成です（図 4）。

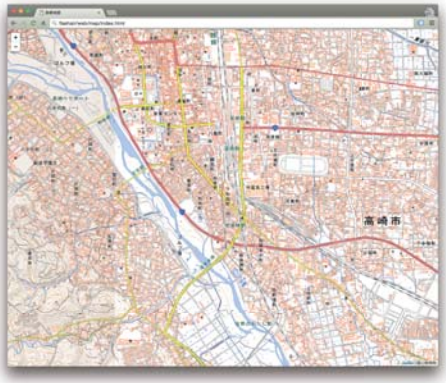


図 4: 完成した地図アプリ

作成した地図アプリを活用する

FlashAir を使うことでインターネットに接続していなくても利用できる地図アプリを簡単に作成することができました。一般的なモバイルバッテリーに接続するだけで持ち運びが可能な地図サーバーは、災害時の情報共有などに活用できるのではないかと期待しています。また、JavaScript で画像の Exif 情報を取得できる exif-reader.js などを利用すれば、撮影した写真をその場で地図上に表示して確認できるアプリなども作成できます。

そのほか、Chrome Box とモニタを使って案内板などを表示するデジタルサイネージなどアイデア次第で様々な活用方法が見つかるとおもいます。

皆さんも是非、FlashAir を使った地図アプリの作成に挑戦してみてください。

清水 正行 (@_shimizu)

群馬・東京間を行き来する出稼ぎエンジニア。ブログでは GIS(地理情報システム)・データビジュアライゼーション・オープンデータなどについて書いてます。

Lua の SPI 機能でサイドバンドつき SPI Master

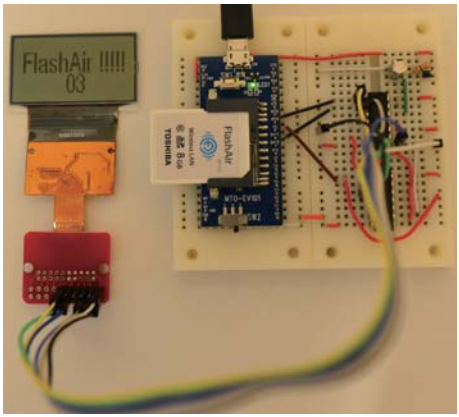
村口

はじめに

FlashAir のファームウェア 3.00.01 以降から Lua に SPI 機能が搭載され、より高速な SPI アクセスが可能になりました。

SPI スレーブデバイスでは、SPI の他に、サイドバンド信号の制御も必要になる場合が多々あります。

今回は、少しの部品追加で SPI スレーブデバイスのサイドバンド信号に対応可能な方法をご紹介します。



SPI スレーブデバイスのサイドバンド信号対応

図 1 のように標準ロジックを追加することで、SPI スレーブデバイスの CS 非選択時に Lua SPI 操作でサイドバンド信号をシフトインさせることができます。

図の LCD_RS は、LCD のサイドバンド信号です. LCD_RS は、コマンド転送かデータ転送かを区別する役割を持っています。1 つの 74HC164 で、最大 8 本までのサイドバンド信号を制御可能です。ただし、出力をラッチしないため、シフト中にサイドバンド信号が変化してしまいます。SPI デバイスのリセット信号については、LCD_RESETB のように処理してください。

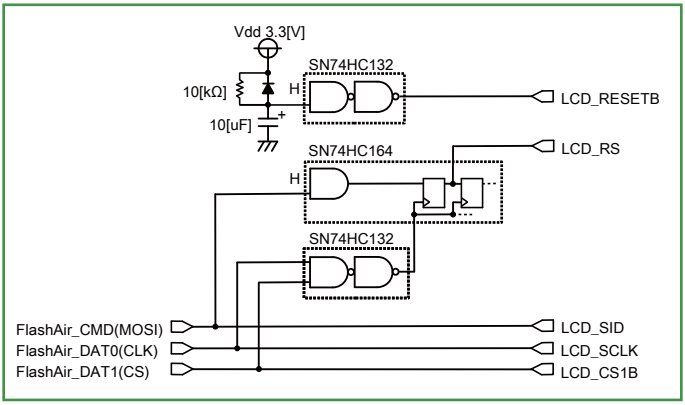


図 1: サイドバンド信号制御用の追加回路

Lua SPI 機能の効果

以前ご紹介した CGI PIO による方法や Lua PIO による方法と比較することで Lua SPI 機能の効果を示します。それぞれの方法で、100x32 ドット モノクログラフィック液晶の全面を書き換えるのに必要な処理時間は、以下のようになります。

CGI PIO では、無線 LAN 経由の PIO 制御のため、実用的とは言い難い処理時間を要していました。Lua PIO では、Flash Air 単体で Lua スクリプトが動作するようになり、大きなブレイクスルーがありました。しかし、Lua PIO では Lua スクリプトで SPI 機能を実現していたため、高速化の余地が残っていました．今回の Lua SPI では、Lua 拡張関数の中で SPI 処理を行うことで、処理時間を 1.0[秒] まで短縮しています（表 1）。

表 1: 処理時間の比較結果

	CGI PIO	Lua PIO	Lua SPI
処理時間	35[分]	7.5[秒]	1.0[秒]

さいごに

少しの部品追加でサイドバンド信号つき SPI スレーブデバイスに対応可能なこと、Lua SPI 機能により従来の 7.5 倍も高速に制御できることをご紹介しました．

さらに FlashAir の WebDAV 機能を利用すれば、LCD の表示内容を無線 LAN 経由で動作中に書き換えることも可能です．このお手軽さも FlashAir の良いところです。

村口
最近、妻と旧街道沿いの散策を楽しんでいます。

FlashAir で音楽を鳴らそう

せいみ まさみ

皆さん、FlashAir でエンジョイしていますか？ 昨年は W-02 を使用して LED 点灯（L チカ）を応用した単純な機構でしたが、ATARI 仕様 Wi-Fi コントローラーを作成しました。その延長線上で、今年も MSX で何かを考えていたのですが展示としてあまり面白いものができなさそうだったので、急遽別のものを考えてみました。

FlashAir は単独でも十分に遊べるマイコンであると考えています。その単独動作の例として視覚的に表現をする LED や LCD を直接動作させるモノはいくつも見受けられるのですが、音楽・聴覚的に表現をする例はあまりなさそうでしたので「画像が出せるなら音だっていけるはず」ということで、誌面では難しい音を出す製作をしてみたいと思います。

今回は、W-03 の機能を利用して昔のパソコンに多く使われていた音源 LSI、PSG (Programmable Sound Generator) を FlashAir でドライブして、FlashAir を電子楽器にしてしまいます（図 1）。

例によってソフト技術者が趣味でハードウエア回路図を作ったりしていますので、ハードウエア技術者から見るといろいろ危ない所があるかもしれませんが、イキオイだけのモノづくり早速スタートです！

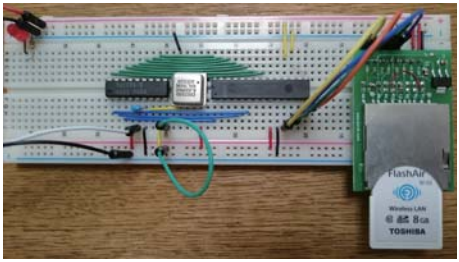


図 1: FlashAir PSG ボード

PSG とは

PSG はゼネラル・インスツルメンツ (GI) 社の AY-3-8910 という音源 LSI で、指定された周波数の音を指定された音量で最大 3 音を同時に発声させることができます。また、爆発などの効果音用にホワイトノイズも発生できるようになっています。とてもシンプルな構成の音源 LSI であり、80 年代のアーケードゲームや、パソコンなどに多く用いられていました。いわゆる 8Bit サウンドのピコピコ音はこの音源 LSI の音といっても過言ではないと思います。制御も常にアクセスするのは周波数レジスタと音量レジスタ程度なので簡単にでき、マイコンの音源 LSI としてお手軽に使うには最適でしょう。

今回使用した YMZ294（図 2）はヤマハから出ていた PSG 互換品の SSG(Software-controlled Sound Generator) と呼ばれるもので、現在でも通販などで簡単に入手できることから使ってみました。

図 2: PSG 互換品

動作の仕組み

PSG は 3 本の制御信号と、8bit データバスのパラレル通信を使用することでアクセスすることができます。しかし、FlashAir は I/O が 5 本 (5bit) しかないので、そのままでは PSG にアクセスすることができません。5bit 以上のデータを扱う手法として、SPI や I2C などのシリアル通信を用いてデータの入出力を行うというものがあります。幸い FlashAir は W-03 からの機能で Lua で簡単に SPI 通信を行うことができます。

ここで「PSG は SPI でのアクセスに対応しています!」なんてことが言えれば大変に楽ではあるのですが、流石に今から 30 年以上前に作られた音源 LSI にそれを望むのは酷なことです。ということで、何とかして SPI 通信から 3+8bit のパラレルの信号を作り出さなくてはいけません。

この様なときに使用するのが、I/O エキスパンダと呼ばれる I/O ポート拡張 IC です。I/O エキスパンダは、GPIO のポートを増設して多くのパラレルデータ入出力を行うことができるものです。マイコンからの通信方法は I2C や SPI などのシリアル通信で行うので今回のパターンにぴったりです。

今回の製作には 16bit I/O エキスパンダ MCP23S17 を使用します。マイコンとの通信は SPI で行うもので、I/O ポートとしては 8bit GPIO が A,B の 2 チャンネルあります。また扱える電圧は 1.8 ～ 5.5V と今回のような TTL レベルのレガシーなデバイスを扱うのも全く問題のない内容です。今回は MCP23S17 の GPIOB をマイコンの 8bit データバス、GPIOA をマイコンの制御信号として見立てて、PSG とのアクセスをすることにします。これで FlashAir と PSG がつながる算段が整いました。

回路図とプログラム

今回の回路図は図 3 のようになります。回路図で特に難しいところは無いと思います。部品も PSG も含めて秋月電子で全て手に入りますので部品入手もお手軽です (表 1)。

MCP23S17 と PSG の接続は GPIOA と PSG のデータバスを、GPIOB と PSG の制御信号を接続するようにしています。データバスの GPIOA のビット番号と PSG のビット番号を合わせて接続しましょう。PSG 制御信号 /CS と /WS は GPIOB の bit4,5 に、PSG の A0 は GPIOB の bit0 に接続すると、プログラムが解りやすくなります。

表 1: 部品表

部品番号	部品名	参考価格	備考
COM1	FlashAir W-03	3190 円～	
COM1	SD カードスロット DIP 化モジュール	250 円	要改造。方法については同人誌 2 を参照
U1	4MHz クリスタルオシレータ	-	U2 の付属品
U2	音源 LSI YMZ294	500 円	U1 と同一パッケージ
U3	MCP23S17-E/SP	150 円	16BIT SPI I/O エキスパンダ
	基板、線材、イヤホンなど		

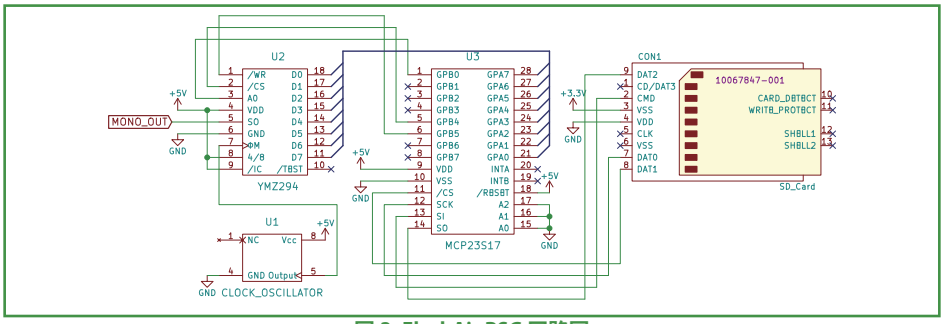


図 3: FlashAir PSG 回路図

MCP23S17 の SPI のデバイスアドレスは A0 ～ A2 のピン全てを GND に落とすことで 0x40 にしています。MCP23S17 と FlashAir の接続は SPI 関連の端子をそのまま接続します。端子の詳細については、FlashAir 同人誌 2 などの記事を参考にしてください。PSG のクロック設定は 4/6MHz のいずれかに設定できます。秋月電子で PSG を購入すると 4MHz クロックオシレータが付いてくるので、4MHz 設定（VDD 接続）にします。

電子楽器を作っているというのに、音響系のアンプ回路はここでは使用していません。回路図の MONO_OUT をイヤホンに接続して聞くという割り切った簡単仕様です。テストとして聞く分にはアンプは必要ありません。音響などにこだわるのでしたら、MONO_OUT から音響用アンプ回路を別途作成してつないでみると良いでしょう。

PSG のアクセス方法はちょっと手を抜いて SPI アクセス回数を減らします。通常では A0 を確定してから、/CS、/WR を変化させる必要があるのですが、A0、/CS、/WR をすべて同時に変化させても、問題なく鳴っているので良しとしましょう。

これらを踏まえた、音を発生させるだけの Lua スクリプトは次ページのようになります。main の SOUND 文で PSG の任意のレジスタにデータを書き込むことができます。この Lua スクリプトは FlashAir の任意の場所に書いておきますが、合わせて FlashAir の設定ファイル CONFIG の変更も行いましょう。このファイルは隠しフォルダの SD_WLAN の中の隠しファイルなので注意が必要です。このファイルに「IFMODE=1」、「LUA_RUN_SCRIPT= 作成した Lua スクリプト」を追加します。これで FlashAir の起動時にすぐ Lua スクリプトが実行されるようになります。

このプログラムだけだと、決められた音がなるだけで大して面白くありません。Arduino とか Raspberry Pi で鳴らすのと差がありませんからね。なので FlashAir ならではの機能を考えて見ましょう。例えば共有メモリに音程などの情報を置き、この内容に応じて PSG にアクセスするようにして、さらに Wi-Fi 機器から共有メモリを更新できるようにするとリアルタイムに音程などが変更でき、好きなように PSG を鳴らせるようになります。たったこれだけで一つの楽器に対して、Wi-Fi 経由で多人数でも演奏することもできるという、新時代のユニークな電子楽器が誕生するのではないでしょうか。

```
-- FlashAir & MCP23S17 & PSG
function write_command(addr, data1, data2)
    fa.spi("cs",0)
    fa.spi("write",addr)
    fa.spi("write",data1)
    fa.spi("write",data2)
    fa.spi("cs",1)
end
function sound(addr, data)
    write_command(0x40, 0x13, 0x00) --/CS=0 /WR=0 A0=0
    write_command(0x40, 0x12, addr) -- アドレス出力
    write_command(0x40, 0x13, 0x30) --/CS=1 /WR=1 A0=0
    write_command(0x40, 0x13, 0x01) --/CS=0 /WR=0 A0=1
    write_command(0x40, 0x12, data) -- データ出力
    write_command(0x40, 0x13, 0x31) --/CS=1 /WR=1 A0=1
end

-- initial
fa.spi("init", 5) --SPI 出力のおまじない
fa.spi("mode", 3)
write_command(0x40, 0x00, 0x00) --GPIOA を Output
write_command(0x40, 0x01, 0x00) --GPIOB を Output
write_command(0x40, 0x12, 0xFF) --GPIOA をすべて HIGH
write_command(0x40, 0x13, 0xFF) --GPIOB をすべて HIGH

--main
sound( 7, 0x3E) --toneA のみを有効
sound( 1, 0x01) --freqA を 440Hz に設定
sound( 0, 0xC2) --
sound( 8, 0x0F) --volumeA を 15(MAX) に設定
```

さいごに

PSG に限らず MSX や 80 年代アーケードゲームで使用された OPM などの FM 音源を演奏することができると、おっさんホイホイになるのですが SPI の動作が遅くこれ以上の速度でアクセスすることができないためちょっと難しいです。FM 音源は PSG に比べて設定する内容が多いのです。あとは定期的にプログラムを実行するタイマーが存在しないのもこのような用途には地味につらいですね。

次世代の FlashAir では倍以上の速度で SPI アクセスできたり、タイマー割り込みが使用できたりするともっとマイコンとして面白くなると思います。というところで今回は終了！次回は FlashAir に何をやってもらおうかな〜。

せいみ まさみ（@masa_seimi）

MSX が好きすぎて、MSX のハードウエアを作ったメーカ（の関連会社）に入社してしまった自称ソフト技術者。MSX に FlashAir を繋いでいろいろ遊びたいが、お仕事がなかなか忙しくて遊んでいる時間が作れないのがザンネンな毎日。

ゲームコントローラ基板 Airio Play の設計

余熱

皆さまこんにちは、余熱です。FlashAir の GPIO 機能を使ったゲームコントローラ基板「Airio Play（えありお ぷれい）」を設計したので、今回はそれについて書いていこうと思います。ゲームコントローラと言っていますが、実際には FlashAir の web サーバ上にゲームが置いてあるため、なんというかもうポータブルゲーム機ですね（大げさ）。基板の取説・回路図・サンプルソフトなどは、web ページ[1]にあるので適宜参照してください。

図 1: Airio Play の試作機

Airio Play の試作機の写真を図 1 に示します。見た目はそのまんまゲームのコントローラで、FlashAir の GPIO 機能を使ったゲームをすることが可能です。また、FlashAir を用いた汎用入力装置として使うことも可能です。

企画

FlashAir 同人誌 2 で じむさんが書かれていた FlashAir と enchant.js でゲームをする漫画を読んで、「これ FlashAir の GPIO で動作させられないの?」と思ったのが企画のきっかけです。

図 2: FlashAir の GPIO で enchant.js 動作確認

FlashAir を web サーバとして動作させて、スマホからアクセスして JavaScript のゲームを動かします。JavaScript はスマホ上で動いているため、**毎フレーム FlashAir に GET リクエストを投げて** GPIO の値を確認する必要があります。GPS_NMEA(@Seg_Faul) さんの FlashTools GPIO Tester & Checker では GPIO の取得はそれなりにレスポンスが速かったのである程度大丈夫そうな予想はあったのですが、まずは Airio で動作を確認してみました（図 2）。スライドスイッチの動きに合わせてキャラクターが移動するデモを作ってみたところ、おおよそ問題なく動くことが分かったため、きっちりコントローラとして仕上げた基板を作成することにしました。

1　余熱の個人 HP: http://yone2.net/ か、
FlashAir Developers: https://flashair-developers.com/ に情報あります。

設計

電源周り

電源は USB コネクタからの給電と、乾電池からの給電をサポートすることにしました。

USB micro-B コネクタはピン間が 0.5mm で実装が面倒、かつ高価なため悩ましい部品なのですが、秋月さんから電源専用の安価なコネクタ[2]が販売になり、実装面・コスト面で大変助かりました。データシートには表面実装用のフットプリントが載っていますが、スルーホールで手実装したほうが良さそうです。

乾電池からの給電ですが、昇圧用の DC/DC コンバータや電池ホルダが必要になります。これらはオプション扱いとして、パターンだけ引いています。

5 つの GPIO で 6 個のスイッチ

FlashAir の GPIO は 5bit なので、普通に設計すると 5 個のスイッチになるわけですが、多くのレトロゲームでは十字ボタンに加え最低 2 つのボタン（A、B ボタン）を備えています。そのため、なんとか GPIO をやりくりして 6 スイッチを実現しています（図 3、表 1）。A、B ボタンはそれぞれ専用に GPIO を割り当てており、残り 3 本の GPIO に十字ボタンを割り当てています。十字ボタンと A、B ボタンは独立して押すことができます。ただし、十字ボタンは斜めに押されたことを認識することはできません。

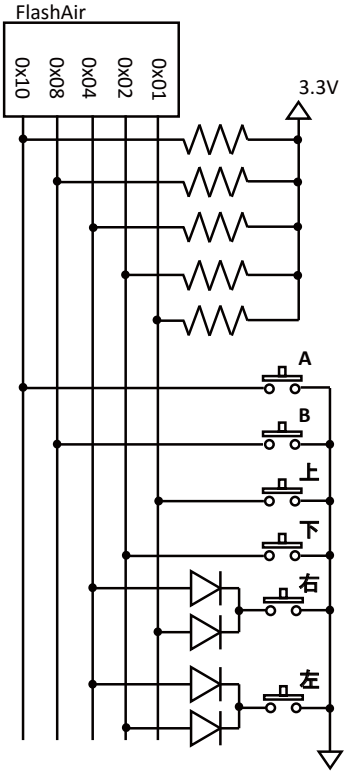


図 3: スイッチ部分の回路図

表 1: スイッチ部分の真理値表

ボタン	0x10 (DAT3)	0x08 (DAT2)	0x04 (DAT1)	0x02 (DAT0)	0x01 (CMD)	備考
A	0	x	x	x	x	
B	x	0	x	x	x	
上	x	x	1	1	0	
下	x	x	1	0	1	
右	x	x	0	1	0	
左	x	x	0	0	1	
右上	x	x	0	1	0	右上を押すと右と判定
右下	x	x	0	0	0	右下を押すと停止
左上	x	x	0	0	0	左上を押すと停止
左下	x	x	0	0	1	左下を押すと左と判定

2　基板用マイクロＵＳＢコネクタ（電源専用）http://akizukidenshi.com/catalog/g/gC-10398/

タクトスイッチとダイオードの選定

当初、十字ボタンについてはジョイスティックやマルチファンクションキーを探していたのですが、なかなか良い部品が見つからなかったためタクトスイッチを 6 つ使うことになりました。6 つ使うためなるべく安価な部品を選定する必要があり、Aliexpress なども探してみたのですが、梱包形態が不明だったため結局 Digi-Key の部品を使用することにしました。タクトスイッチは 1 回路入りのものを使用するため、2 つの GPIO を同時に切り替えるためにはダイオードが必要になります。ダイオードは汎用のもので良いのですが、「1SS184」という 2 回路入りでカソードコモンの部品があったため、この部品を選定しました。なにげに東芝製。

レイアウト

Airio 同様 KiCad で回路図・レイアウトを作成しました。基板サイズは 80x50[mm]、両面基板になりました。レイアウト図を図 4 に示します。

おわりに

今回は、ユニバーサル基板での試作の後、感光基板で Airio RP 用の子基板や最終レイアウトの動作確認などを行い、「試作の大切さ」を改めて学びました。FlashAir チームはゲーマーが多いので、最近はみんなでゲーム作って遊んでます。

今後も色々 FlashAir を使った工作を続けていきますので、FlashAir Developers や Twitter などチェックしてみてください。

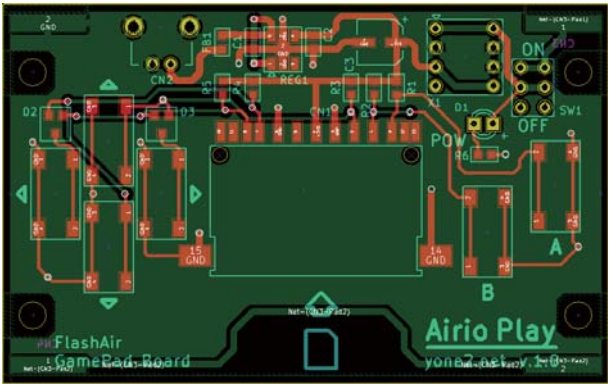


図 4: Airio Play のレイアウト

余熱 (@yone2_net)

再び余熱です。最近 FlashAir 漬けな日々が続いています…。

そういえば、2016 年 8 月号の Interface 誌に FlashAir の特集が組まれていますので、是非ご覧になってください。自分もちょっと書きました。

FlashAir でゲームを作って、遊ぶ！

寺西

FlashAir 同人誌 2 で、閃ソラちゃんが FlashAir と enchant.js 使ったゲーム機を紹介していました。当時、FlashAir を使ったゲーム機はじむ氏が試作したものしかなく、多くの方に遊んで頂けず残念に思っていました。そして我々 FlashAir（変態カード）集団としては、多くの方に遊んで頂けるよう、このゲーム機を実現すべく動き始めました[1]。

enchant.js

enchant.js[2] は、株式会社 UEI の秋葉原リサーチセンターで開発された、HTML5 ＋ JavaScript フレームワークのゲーム・エンジンです。MIT ライセンスのオープンソースで、画像素材や音楽素材を含めて無償で利用することができます。

enchant.js を使う準備

enchant.js を使うために、以下のサイトからファイルをダウンロードしてください。

```
http://enchantjs.com/ja/download-ja/
```

ダウンロードしてきた zip ファイルを解凍し、FlashAir の任意の場所にコピーしてください。enchant.js を使う準備はこれで完了です。

解凍したフォルダの中に “examples” フォルダがあります。この中にサンプルが入っていますので、これを基に FlashAir への移植方法について説明していきます。

FlashAir で動かしてみる

次に、FlashAir と Airio Play を使って、ゲームを動かせるように実装していきます。

1　一部、FlashAir 同人誌 2 の内容と重複する部分もありますがご了承願います。
2　http://enchantjs.com/ja/

CONFIG ファイル

FlashAir を PC に挿してください。FlashAir 内には “SD_WLAN” フォルダがあり、その中に CONFIG ファイルという FlashAir の動作を制御する設定ファイルがあります（Mac の方は、FlashAir 同人誌 2 の「FlashAir お悩み相談室」を参照してください）。お持ちのテキスト・エディターを使い、CONFIG ファイルを以下のように編集してください。

```
[Vendor]

CIPATH=/DCIM/100__TSB/FA000001.JPG
APPMODE=4
APPNETWORKKEY=********
VERSION=FA9CAW3AW3.00.01
CID=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
PRODUCT=FlashAir
VENDOR=TOSHIBA
LOCK=1
DHCP_Enabled=NO
Proxy_Server_Enabled=NO
APPAUTOTIME=0
IFMODE=1
```

Airio Play を使うためには、FlashAir の SD インターフェース端子を、汎用の I/O 信号端子として利用し、信号の読み書きが行えるようにする必要があります。また、コントローラの処理以外の負荷を減らすために、FlashAir の機能を一部無効化し、無線 LAN の接続タイムアウトが発生しないようにしています（詳細は、FlashAir Developers サイトの情報を参照してください）。

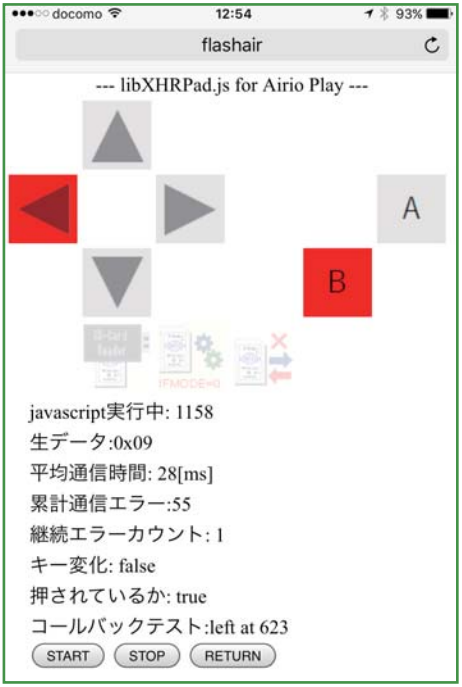


図 1: libXHRPad.js を使った
コントローラ・キー確認画面

libXHRPad.js

次に設定が正しく反映されているか確認します。

libXHRPad.js は、GPS_NMEA 氏が開発した Airio Play 向けの JavaScript ライブラリです。これを使い、コントローラのキーが正しく動いているか確認できます。

まず、libXHRPad をダウンロードし、FlashAir の任意の場所にコピーしてください。FlashAir を一度 PC から抜き、Airio

Play に挿し、スマートフォンなどで FlashAir に無線 LAN 接続してください。ブラウザを立ち上げ、http://flashair/ にアクセスし、先ほどコピーした libXHRPad のフォルダにある demo.htm を開いてください。図 1 のような画面が出てくると思います。押されたボタンの箇所が、“true” になっていれば OK です。

ゲームへの移植方法

libXHRPad.js で動作を確認できたら、次にゲームの移植をしていきます。移植するゲームは、enchant.js の “examples/beginner” フォルダ内にある Hello Bear というサンプルにします。このサンプルは、単純に画面を左から右にクマのキャラクタが移動するというものになっています。ソースコードの内容が簡単なので、すぐに移植できると思います。

“hellobear” フォルダの中には、以下のファイルが用意されています。

- index.html
- main.js
- main_noncomment.js（main.js のコメントが入っていない版）
- chara1.png（enchant.js で用意されているキャラクタ素材）

移植する場合、変更が必要なファイルは、上記のうち index.html と main.js の 2 つです。

まず、index.html から変更します。

index.html の中から、使用する JavaScript を呼んでいます。ここで libXHRPad.js を追加します。FlashAir の任意の場所に置いた libXHRPad.js を呼び出してください（以下の例では、紙面の関係上、“hellobear” フォルダにコピーしています）。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta http-equiv="x-ua-compatible" content="IE=Edge">
  <meta name="viewport" content="width=device-width, user-scalable=no">
  <meta name="apple-mobile-web-app-capable" content="yes">
  <script type="text/javascript" src="libXHRPad.js"></script>
  <script type="text/javascript" src="../../../build/enchant.js"></script>
  <script type="text/javascript" src="main.js"></script>
  <style type="text/css">
  body {
    margin: 0;
    padding: 0;
 }
  </style>
</head>
<body>
</body>
</html>
```

次に、main.js を変更します。main.js には、enchant.js の初期化やサンプルの本体が書かれています。ここに libXHRPad.js を組み込んでいきます。ソースコードの行数が多いため、変更する箇所（bear.addEventListener）を抜粋しています。

```
bear.addEventListener("enterframe", function(){
xhrPad.start();                    // libXHRPad.js の開始
    var keys = xhrPad.read();          // コントローラ・キーの読み出し

      /*
       * 方向キー入力（同時押し不可）
       */
    if( keys.up == true ) {            // 上ボタン
        this.y -= 5;
        this.frame = this.age % 2 + 6;
    } else if( keys.down == true ) {  // 下ボタン
        this.y += 5;
        this.frame = this.age % 2 + 6;
    } else if( keys.left == true ) {  // 左ボタン
        this.scaleX = - 1;               // キャラクタ反転処理
        this.x -= 5;
        this.frame = this.age % 2 + 6;
    } else if( keys.right == true ) { // 右ボタン
        this.scaleX = 1;                 // キャラクタ正転処理
        this.x += 5;
        this.frame = this.age % 2 + 6;
    }

      /*
       * AB キー入力
       */
    if( keys.a == true ) {             // A ボタン
        // A ボタンの処理
    }
    if( keys.b == true ) {             // B ボタン
        // B ボタンの処理
}

});
```

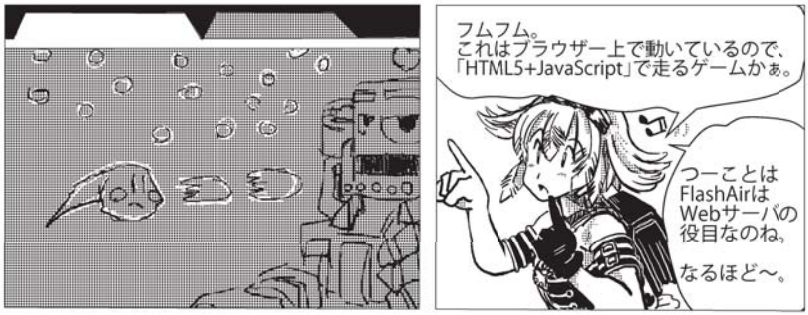
これで移植の完成です。難しくないですよね?図 2 に Hello Bear の画面イメージ を示します。

図 2: Hello Bear の画面イメージ

おわりに

今回の同人誌ではゲームを作るということになり、高校生以来のチャレンジ（禁句?）をすることになりました。本章では紙面の関係から、簡単な移植しか掲載できませんでしたが、FlashAir の違った遊び方にチャレンジ（また?）をして頂ければ幸甚です。

最後に、このようなゲーム開発ができたのも、enchant.js があったおかげです。株式会社 UEI ならびに関係者の方々に心より感謝申し上げます。また、日頃から FlashAir を応援して頂いている GPS_NMEA 氏に感謝申し上げます。



寺西 (@soft128)

FlashAir および NFC 搭載 SD メモリカードの商品企画、拡販活動、開発管理、技術サポートなどに従事。元・組込みソフト屋さん。家で電子工作（ガラクタ）を作っている。

時々、FlashAir 芸人の付き人、余熱氏のリマインダー（煽り）を担当。

FlashAir ゲームで使える小ネタ

宮内

初めまして宮内です。FlashAir のデモ用にゲームを作るということで、初心者ながら JavaScript にチャレンジしました。なんとか完成まで漕ぎ着けましたので、今回使った、ちょっと便利な小ネタを紹介させていただきます。

ゲーム概要

今回紹介するゲーム「クイズ ソラちゃんの野望」は、ランダムで出てくる 2 カ国のどちらの国土面積が大きいかを当てるクイズゲームです（図 1）。タイトルに深い意味はありません。ゲームのコンセプトは、アクション性のない、プログラミング初心者でも実現できそうなもの、です。

図 1: クイズ ソラちゃんの野望

コントローラキー割り当て

ゲーム開発は、まず PC 上で動くものを作り、その後コントローラに対応させました。コントローラ対応のやり方は、寺西さんの記事に詳しく載っています。このゲームでは、キーボードでも操作できるように、コントローラの入力とキーボードの入力を連動させています。（コントローラの各ボタンにキーボードの各キーを割り当てている）

```
game.addEventListener('enterframe', function(){
     var keys = xhrPad.read();        // コントローラキーの読み出し
     if( keys.up == true ) {          // 上ボタン
              game.input.up = true;
              game.input.down = false;
              game.input.left = false;
              game.input.right = false;
// 以下、下 / 左 / 右 /A/B ボタンと同様に続く
```

サムネイル CGI

せっかく FlashAir を使うので、「何か FlashAir ならではの要素を・・・」ということで入れたのがサムネイル表示です。ゲーム内では、ゲームオーバーになった後、最終スコアに応じてご褒美 (?) 画像が表示されます。その表示で使っているのが “thumbnail.cgi” です。これにより、FlashAir の指定のフォルダ内に入っている写真のサムネイル画像を参照することができます。

“http://flashair/thumbnail.cgi?/ 対象ディレクトリ / 対象ファイル名 ” と書きます。

```
game.preload(['card.png','kigou.png','sora.png','quiz.png',
'http://flashair/thumbnail.cgi?/DCIM/100__TSB/1.jpg',    // プリロード

Result = new Sprite(160, 120);    // 表示 (Sprite サイズは 160 × 120)
Result.x = 80;
Result.y = 100;
Result.image=game.assets['http://flashair/thumbnail.cgi?/DCIM/100__TSB/1.jpg'];
```

サムネイル CGI 使用の際には以下の注意点があります。

- 対象ファイルは EXIF 規格のサムネイル画像が格納されている JPEG であること
- EXIF 規格に準拠するため、Sprite サイズは 160 × 120 にすること
- 端末から FlashAir に無線でアクセスすること (PC から起動しても機能しません)

このようにいくつか注意点はありますが、撮影したままの巨大サイズな写真を自分で加工することなくゲームに適したサイズで使えます。

ゲームのアレンジ

このゲーム、単純なシステムと JavaScript で組んでいますので、アレンジが簡単です。各国の名前と面積の部分を変えるだけで、あらゆるものの大きさ比較ゲームを作ることができます。川の長さとか、建築物の高さとかでもいいですし、ロボットの全長なんてのもできます。ダイ○ーン 3 とエ○テバリスはどっちが大きいか?とか。。。

本ゲームの JavaScript ファイルは FlashAir Developers サイトに公開されていますので、ご自由にダウンロードしていただき、ご自由に中身をいじっちゃって下さい。

宮内

今年の春ぐらいに現れた異邦人。最初は企画を出すだけだと思っていたが、「で、誰が作るの?」となり、覚悟を決めて作ってみた。そしたら結構楽しかった。

鉄道模型も FlashAir リモコンで！

綾瀬 ヒロ

FlashAir で鉄道模型をコントロールして遊んでいる綾瀬です。最近、転職しました。

さて、今回は FlashAir のためのリモコン基板が MFT2016 に合わせて配布されるということで、これを使って、鉄道模型を遠隔運転するリモコンを作ってみます。

概要

今回のリモコン基板は、FlashAir の GPIO 端子 と接続していますので、Lua スクリプトの fa.pio 関数でボタンの押下状態を検出して、操作対象へのコマンドを送信するトリガとします。それだけでも良いのですが、リモコンボタンの数よりも操作対象の機能が多いので、FlashAir 内に HTML ファイル（init.htm）を配置し、ボタンの機能割当変更機能を JavaScript で作ってみます。ボタンの機能割当情報は FlashAir の共有メモリに書いておくことで、機能割当の変更がすぐに反映されるようにします。また電源を落とすと共有メモリの情報は消えてしまうので、FlashAir 内にテキストファイル（init.txt）としても保存

しておき、電源投入後の起動時に読み込むようにします。全体概要を図 1 に示します。また、リモコン処理本体の Lua スクリプトと、機能割当変更機能（JavaScript）の各フローチャートを図 2 に示します。

今回の操作対象である鉄道模型ミニレイアウトについては「FlashAir 同人誌 2」に詳しく解説していますので、こちらも参照ください（サウンド機能が追加になりました）。

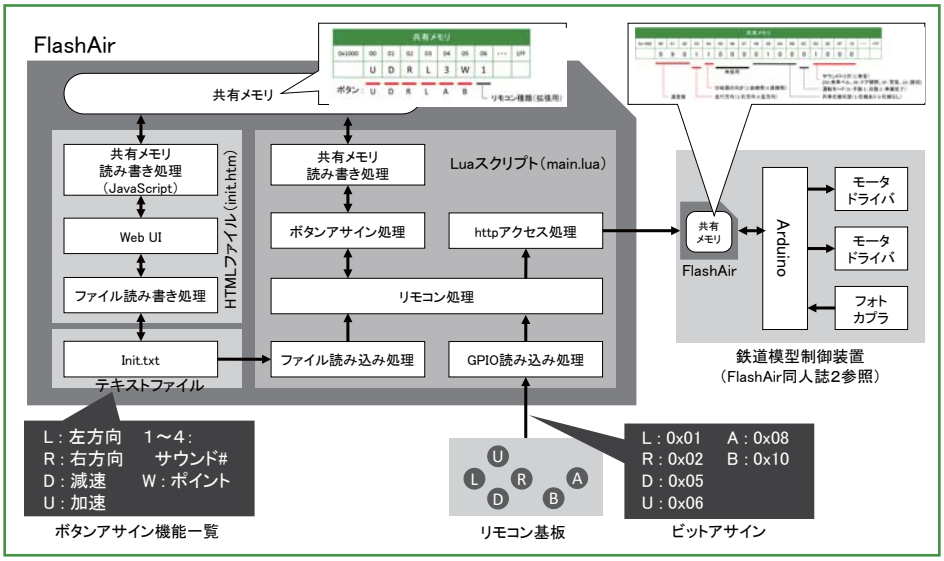


図 1: 全体概要図

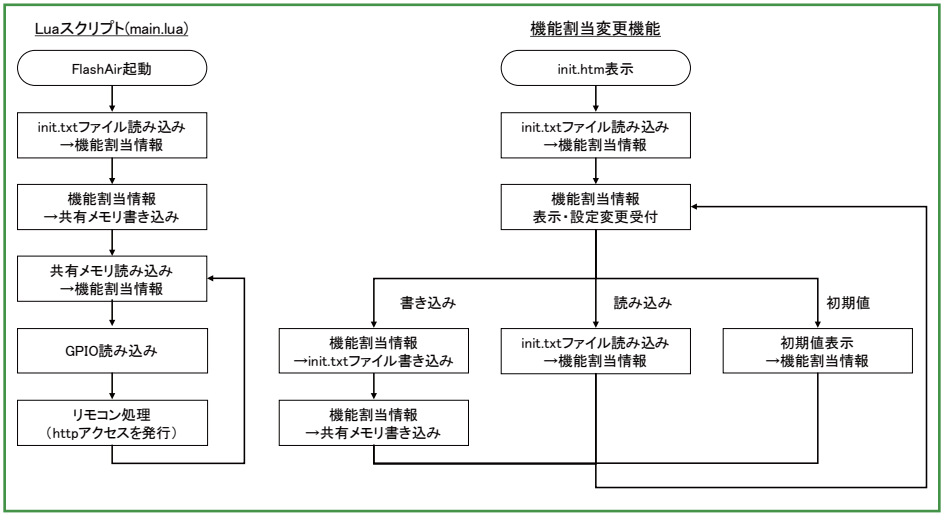


図 2: Lua スクリプトと機能割当変更機能 JavaScript のフローチャート

Lua スクリプト

Lua スクリプトでは fa.pio 関数で GPIO 端子の状態を読み込み、得られる上下右左 AB ボタンの押下状態を表すビットを判断し、それぞれのボタンに割り当てた機能を実行します（別に定義した selectFunction 関数を呼び出す）。

ここでポイントとしては、funcBtnL などの変数に、割り当てた機能を格納しておくことで、各ボタン押下時の動作を変更できるようにしておきます。

```
local s, indata = fa.pio(0x00, 0x00) --GPIO 端子の状態読み込み
if s == 1 then
 if     bit32.band(indata, 0x07) == 0x01 then -- 左ボタン
    selectFunction(funcBtnL) -- 左ボタンに割り当てた機能を実行
 elseif bit32.band(indata, 0x07) == 0x02 then -- 右ボタン
   selectFunction(funcBtnR) -- 右ボタンに割り当てた機能を実行
 elseif bit32.band(indata, 0x07) == 0x05 then -- 下ボタン
    selectFunction(funcBtnD) -- 下ボタンに割り当てた機能を実行
 elseif bit32.band(indata, 0x07) == 0x06 then -- 上ボタン
   selectFunction(funcBtnU) -- 上ボタンに割り当てた機能を実行
 elseif bit32.band(indata, 0x08) == 0x00 then --A ボタン
   selectFunction(funcBtnA) --A ボタンに割り当てた機能を実行
 elseif bit32.band(indata, 0x10) == 0x00 then --B ボタン
    selectFunction(funcBtnB) --B ボタンに割り当てた機能を実行
 end
end
```

selectFunction 関数では、引数として各ボタンに割り当てた機能を示す変数を受け取り、その内容に応じて、鉄道模型ミニレイアウトの共有メモリにコマンドを書き込むための http リクエストを発行します。

http リクエストで共有メモリに書き込む文字列は、図 3 に示したコマンドアサインのうち、0x1000 ～ 0x1007 の 8 バイト分と 0x100D ～ 0x1010 の 4 バイト分です。コード例には前者 8 バイト分の場合を示します。

```
local function selectFunction(cmd)
  strval = (cmd に応じて共有メモリに書き込む文字列を生成 )
  uri = "http://flashair/command.cgi?op=131&ADDR=0&LEN=8&DATA=" .. strval
  b, c, h = fa.request(uri)
end
```

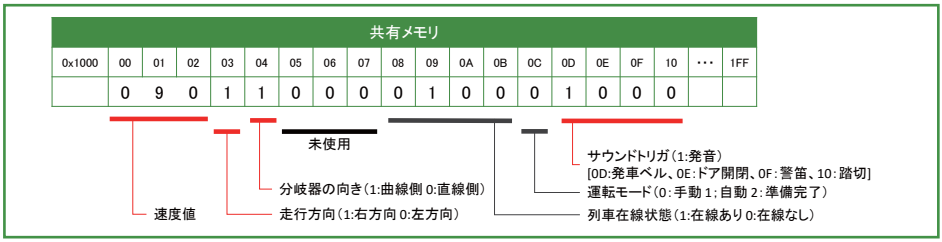


図 3: 鉄道模型ミニレイアウトの共有メモリのコマンドアサイン

リモコンボタンの機能割当変更機能

さきほど Lua スクリプトの解説で、funcBtnL などの変数に割り当てた機能を格納すると書きましたが、この格納する値を、FlashAir 上の Web ページ（init.htm）から書き換えられれば、リモコンボタンの機能割当を動的に変更することができます。そこで、Lua スクリプトでは、ボタン押下状態の判断時に、常に共有メモリ上の機能割当情報を参照するようにしておきます。よって、Web ページでは機能割当を変更したら、共有メモリ上の機能割当情報を書き換えるようにします。

共有メモリ上の機能割当情報は、0x1000 ～ 0x1005 の 6 バイト分に、上下右左 AB のボタン順に定義します。定義は以下のようにしました。7 バイト目（0x1006）は将来拡張用の部分で、今後、操作対象を増やせるようにしています。

U：加速、D：減速、R：右方向、L：左方向、S：緊急停止、W：ポイント切替
1：発車ベル、2：ドア開閉、3：警笛、4：踏切

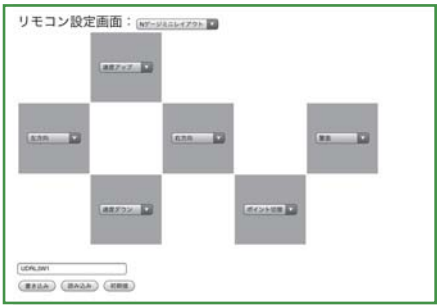


図 4: リモコン設定画面例

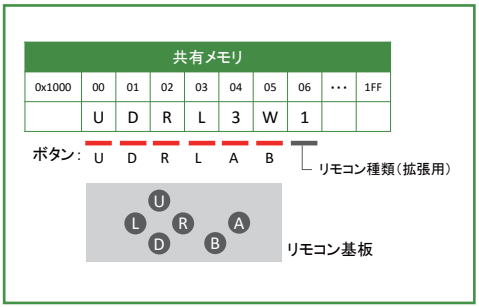


図 5: 共有メモリ上の機能割当情報

綾瀬 ヒロ（@ayasehiro）
某 IT 企業の運輸系インダストリーマネージャです。
紹介したコードは以下で配布中。自由に利用ください。
http://www14.big.or.jp/~ayase/flashair/sample3.zip

閃ソラのひみつ描き方

描いてる本人も日々変化しているので、細かい設定は殆ど無いです。

羽根化の例
※全部やるとうざくなるので適当に。

髪 ネオブロック
(ここから羽根っぽくする。)

髪 ネ2ブロック(私ぐいを乗せたかんじ)

☆髪が「2ブロックの白い羽根」とわかれば後は自由です。適当でOK!!
細かいことは気にしません。カワイければ何でもアリ。

閃ソラを描くために…

一独り言一

- 3年目にしてコツをつかめてきた。やはり特殊な髪型だけに描きにくいと思う。
- 本人も未だ髪で苦労している。w
- キャラクターを増すとそれの説明にコマを消費するので、ート芸をさせている。
- とどのつまり、カワイければアリってことで…。

■ FlashAir Doujinshi 3 - FlashAir の同人誌 3

2016 年 8 月 6 日　第 1 版第 1 刷発行

著者 : 高田 真里 / じむ / 伊藤 晋朗 / tnk / Pochio / くーら / 余熱 / だん / Hiro /
GPS_NMEA / 南 / 日髙 謙太朗 / 寺田 賢司 / 大阪 / 清水 正行 /
村口 / せいみ まさみ / 寺西 / 宮内 / 綾瀬 ヒロ
表紙・本文イラスト : じむ / Ixy / くーら
表紙デザイン : 余熱 / じむ
編集 : 余熱 / Pochio
発行 : FlashAir Developers
連絡先 : support@flashair-developers.com
印刷 : 株式会社 プリントパック

https://flashair-developers.com