FlashAir™ Doujinshi 5

FlashAirの同人誌 5

TAKE FREE ￥0

ここまでできる！無線 LAN搭載 SDメモリカード FlashAirのマニアックな使い方！
FW v4.00.03でさらに便利になりました！

FlashAir も独立（ひとりだち）

上岡 裕一

早いもので FlashAir 同人誌も第 5 号となりました。

前号は FlashAir の第 4 世代品（W-04）の販売開始直後の刊行ということもあり、W-04 をご購入いただいた皆様の反応が分からず期待と不安が半々であったことを思い出します。（気に入っていただけていると良いのですが）

さて、FlashAir の開発・販売元である 東芝メモリ株式会社は 2018 年 6 月の株式譲渡により 東芝グループを離れ、メモリ／ SSD 事業を営む会社として独立。新しい一歩を踏み出しました。

そして我らが FlashAir も独り立ち。

といっても大げさな話ではなく、第 3 世代（W-03）で搭載した LUA スクリプト機能が第 4 世代（W-04）で更に使いやすくなり、ホスト機器があれこれ指示しなくてもデータアップロード&ダウンロードを実現可能に。最新のファームウエア（v4.00.03）では WPA2 エンタープライズ機能や FTPS 機能に対応。Web Socket 機能も更に便利に。

本当に逞しく成長を遂げています。（←親目線）

FlashAir は、おかげ様でさまざまな機器やサービスで活用されています。定点観測カメラや工場内の測定器のデータをクラウドに送信し、WEB で遠隔監視する IoT 用途。歯医者さんで使われている口腔内写真の確認や、街中や店舗で見かけるサイネージや電子 POP コンテンツの遠隔更新用途。コンビニのレジの売上データ収集やスマホプリントサービス等々。あなたのすぐそばで FlashAir が活躍しているかもしれません。（是非見つけてみてください！）

更にマニアックな使い方は、この同人誌のページをめくっていただければと思います。

今回は、これまでよりも更に筆者の方々の気合が入りすぎて、完全に「厚い本」と化しています。（笑）

隅から隅までお楽しみください！

上岡 裕一 (@YuichiKamioka)

自称、FlashAir の「義理の父」。平昌オリンピックのスキージャンプ競技で FlashAir が活用されていたことを後日知る。

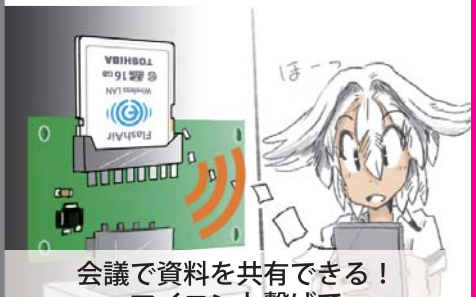
嬉しさの反面、知らずに観戦した悔しさの方が大きいスポーツ観戦オタク。東京オリンピックでの雪辱を期す。

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 スクリプトを実行可能！

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！

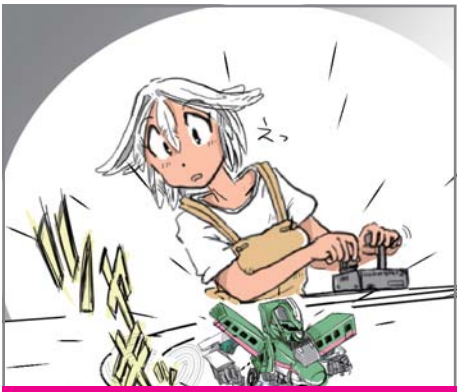
https://flashair-developers.com

FlashAir Developersにアクセス！

Contents

FlashAir 同人誌 5 も 見どころ満載

読み物記事



鉄道模型

VR

FM音源

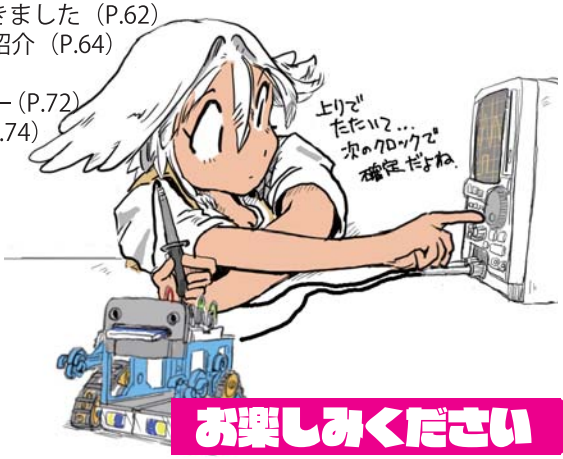
小さなゲームミュージックプレイヤー（P.36）
FlashAir リズムマシン（P.40）

カメラ

画像ファイルも FlashAir IoT Hub へ（P.50）
プログラミング不要！
FlashAir のちょっと便利な使い方（P.52）
FlashAir 用の Windows アプリを組んでみよう！（P.56）

FlashAir のディープな解析ネタも

タミヤ「カム・プログラム・ロボット」を FlashAir で制御してみた（P.14）
Lua スクリプトと I2C バスでペンプロッタを動かす（P.44）
TOY Board と SensorTag でデータの見える化！（P.48）
FlashAir W-03 と W-04 の消費電力と電波強度（P.60）
FlashAir にシリアル通信機能が付きました（P.62）
FlashTools Lua Editor v1.07b のご紹介（P.64）
FlashAir 活用テクニック（P.68）
FlashAir 活用テクニック ー実践編ー（P.72）
FlashAir ルーレット基板の設計（P.74）



お楽しみください

変わる

伊藤 晋朗

我が家には７才になる息子がいる。名前はＫくんだ。今年の４月に小学校に入ったばかりの一年生で体が細くてひょろひょろしている。そんな息子だが毎日走って小学校に行く。小学校までは子供の足で歩いて２０分くらいかかるし、途中に階段があったり坂があったり起伏のある道になっている。本人は笑いながら走っていたので気にしていなかったのだが、ある時「なんでそんなに走っているの？」と聞くと、どうやら学校に遅刻するのが心配なんだそうだ。笑っていたと思ったら引きつっていたようだ。遅刻するのが怖いらしく、心配し過ぎて早く着き過ぎて、小学校の玄関が開く前に着いてしまうらしい。

子供が毎朝走っている日常の一方で会社の関係は忙しない。赤字だの分社だの売却だの独立だの独禁法だのといろいろとあって、環境が変わってしまった。まあ、細かい内容はあまり説明しなくても良いぐらい世間にニュースが流れたのでこの話題は割愛してしまおう。そして世間的にはもう話題にもならなくなったのだが、実は今でも急ぎ足でいろいろと変わっている。おそらくこのまま行くと、もう少しで名前が変わるんだろうってほどだ。子供が新しく学校に通い始めたけど、こちらも新しい会社になったことを実感する。

そんな状況の中 FlashAir の第４世代をリリースしたが、その後も FW 開発を進めている。少しずつ対応できることや現状不満なところ変えるために FW のアップデートを既に３回リリースした。しかし、iPhone が毎年 OS をアップデートするので、他の製品もアップデートするのが普通になっている気がしている。気にしていないかもしれないけれど、アップデートによる機能追加が普通に受け入れられている。まあ、通信機能がある製品は相手の仕様が変わると、こちらも変わっていかないとあっという間に製品としてつながらなくなる可能性が出てくるので、メンテナンスが必要なのは確かである。

開発をしている中、無線 LAN の脆弱性（KRACKs）が見つかったので対応する必要が出てしまった。しかし、世間の反応が過剰すぎるように思う。無線 LAN に脆弱性が見つかりましたという話をしているのだけれども、実はどういう形で被害が発生するかを正確に説明できている記事が基本的に無い。またその脆弱性は、実際の手順として意味のあるハッキング方法では無いと思うし、その仕組を利用したツールを作れる人はいないのではないかと思う。でも、「こんな脆弱性は意味がないよ」と言ったとしても、世間的には「脆弱性」と言われてしまったので対応せざるを得ない。という事で色々と対応していたら FlashAir の仕様も時間と共に変わって来た。以前の W-03 から比べて W-04 になって大きく変わったのは以下の点。

1. スピードが早くなり、容量も増えた。
2. Lua スクリプトが使えるメモリを増量した。
3. 企業向けの無線 LAN である WPA2 Enterprise につながるようになった。
4. セカンドパーティションが使えるようになった。
5. Lua スクリプトの秘匿化もできるようになった。

Lua スクリプトの話をすると、Lua の動作は最初にスクリプトを全部読み込む。いきなり命令を一行一行読み込んで実行するわけではない。まあ、学校のテスト問題を最初に全部読んでみて、どんな問題があるのか、どれくらいの時間がかかるのかを判断するのと同じ事だろう。FlashAir が Lua スクリプトを読み込んでいく途中、関数や命令などを動的メモリから確保してメモリに積んでいくのだが、そのメモリの取得方法が実は線形的ではない。ある一定周期で断片化して取得したメモリを再構築したりするので、階段状にメモリ消費量が増えていく。そのため、もともと使えるメモリが少ない W-03 のときにはたった一行変えただけでメモリの消費量がガクッと変わる事が発生して、メモリ不足に陥ることがある。それが原因で、それまで動いていた Lua スクリプトが1行変更しただけで動かなくなるとかそういった事が発生するので、大きなスクリプトの場合には大変な仕様だったと思う。そういうわけで、W-04 ではメモリプールの量を大幅に変更した。数字的には１０倍以上。なので、それなりに大きなスクリプトも大丈夫だとは思えるぐらいに HW 仕様は変わった。

今年の５月に FW アップデートをリリースした後の事だが、家に帰ると上の子が玄関まで走ってきて、「骨折した!」と、笑いながら言ってきた。部屋に入ると右腕にギプスをはめて泣きじゃくる K くんがいた。どうやら聞くと校庭の雲梯（うんてい）で遊んでいたら落ちたらしい。しかし、痛くて泣いているのかと思ったらそうではなく、ただギプスの異物感が嫌で泣いているとのことだった。骨折もヒビが入ったぐらいで大したことがなかったのでほっとした。まあ、幼稚園に比べると小学校の遊具は大きくて子どもたちだけで遊ぶことも多いし、遊ぶ事のレベルも変わるし、危険の度合いも変わってくるのでケガをすることが多くなるのだろう。そしてケガの分だけ「成長」という名の変化をしているという事にしておこう。ついでに FlashAir も少しずつ「成長」していこう。

しかし、そんな K くんだが、困ったときに後ろ向きになって生尻攻撃してくるのはどうにも変わらない。こちらから見ると、全部見えているのでそろそろ止めてほしいのだが。。。

伊藤 晋朗（@ikainuk）

FlashAir の「中の人」とか「生みの親」とか。今年の健康診断で、1.2cm 身長が伸びた。足の裏が太ったのかしら…。

スポンサーからの陥落と 2 回目のサミット開催

Pochio

FlashAir 芸人の Pochio です。FlashAir 同人誌の新刊を今年も手にしていただき、ありがとうございます。今回初めて FlashAir 同人誌を手に取ってくださった方、はじめまして。Maker Faire Tokyo（MFT）の来場者数が毎年増え続けているとのことですので、初めてこの本をもらった方も少なくないと思います。

この同人誌は、FlashAir が MFT に初めてスポンサーとして出展した 2014 年から、毎年 FlashAir ブースにて無料配布しているものです。MFT のために毎年たくさんの方にご協力いただいて、FlashAir のマニアックな新ネタをお届けしています。というわけで、MFT 恒例の同人誌と勝手に銘打っております。おかげさまで出展 5 年目となる今年は 5 冊目の刊行となりました。暑い中、ビッグサイトまで毎年足を運んでくださる方々に感謝申し上げます。

FlashAir 同人誌はここ数年、1 日約 1000 部、2 日間で約 2000 部を無料配布しております。原則 1 人 1 部までとさせていただいていますので、ざっくりと約 2000 人の方が当ブースにお越しくださっている計算になります。冊子での配布終了後は FlashAir Developers のサイトで PDF にて配信していますので、既刊に興味をお持ちの方はぜひご覧ください。それではいつものように、この 1 年の活動をざっくり振り返ってみます。

Maker Faire Tokyo 2017 出展

新型 W-04 発売からのスポンサー陥落（汗）

FlashAir は 2017 年 3 月にめでたく販売開始 5 周年を迎えました。一方で、FlashAir を製造、販売する東芝セミコンダクター＆ストレージ社のメモリ事業部は、報道などですでにご承知の方も多いかと思われますが、2017 年 4 月に分社化して東芝メモリ（株）となりました。6 月には無線 LAN 機能や Lua 機能が充実した新型 FlashAir　W-04 が発売され、2 か月後に迫った MFT で盛り上げて行こうとしたのですが、厳しい現実が待っていました（汗）。ちょうど社員の給料やボーナスがカットされる緊縮財政の真っただ中で、MFT のスポンサー出展を断念せざるを得なくなったのです。

しかしこれまで継続的に出展してきた流れを途絶

図 1: FlashAir 同人誌 4

えさせたくないという思いから、主催者様とご相談させていただき、スポンサーではないけれど企業出展の区分で4回目の出展にこぎつけることができました。そして無事に4冊目のFlashAir同人誌の配布に至りました。皆様のご支援、ご協力に深く感謝する次第です。

さて、2017年発行のFlashAir同人誌4号（図1）は、表紙込みで3号から4ページ増の72ページとなり、発行部数は4500部と過去最高になりました。このうち2000部をMFTに割り当て、1日あたり1000部を無料配布しました。内容はFlashAir IoT Hubが無事に一般公開されたのでその紹介記事をはじめ、FlashAirによるIoT応用ネタが充実しました。

ところでスポンサー出展ではなくなりましたので、展示スペースはだいぶ減って机2本分（図2）になり、綾瀬さんの鉄道模型（ついにマスコンで操作できるようになりました！）や、せいみさんのLチカ、おなじみワイヤレスクレーンゲームといった当ブース恒例の展示物はそろえたものの、前年に比べて展示内容は限定的となりました。ちょうど当時、少女向け某雑誌に卓上小型掃除機が付録となったのが話題となり、余熱氏がこれを某社のロボット掃除機風に改造したので展示してもらいました（図3）。こちらはFlashAir搭載によりラジコンのように操作できる一方、バックさせるとそれまで集めたゴミを一気に撒き散らすという、笑いをこらえるのが相当につらい一品でした。残念ながら会場は相変わらず無線LANの電波がつながりづらく、デモで動かしてのお披露目にはいたりませんでした。

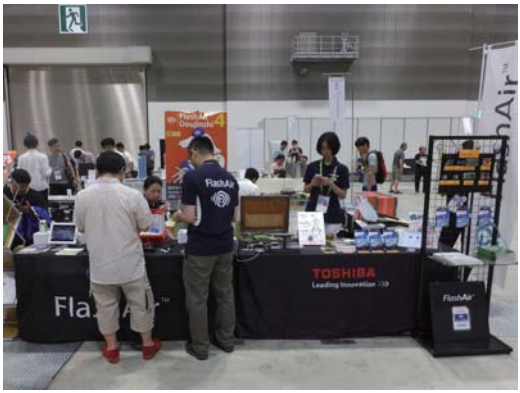


図2: 机2本分の企業展示ブース

図3: お掃除ロボット

図4: 山積みの同人誌

余談ですが、同人誌 4 号は作成した 4500 部が搬入日にブースに届き、そのうち MFT で配布する 2000 部のみを受け取って、残りの 2500 部をそのまま運送会社に預けて会社に送る手はずだったのですが、見事に失敗。4500 部の入った段ボールすべてを、前年よりも狭くなったブース内で抱える事態となりました。山積みの同人誌（図 4）の下に、返送できなかった同人誌の段ボールが山ほどしまわれていたのでした。出展するほうも何かと大変です（汗）。

FlashAir Developers Summit 2018 開催！

Yahoo さんの LODGE をお借りしました！

MFT が終わって 2 か月後の 10 月 7 日に、大阪にて FlashAir Developers Summit ハンズオンが開催されました。こちらの様子についてはクレイン電子の福屋さんがこの同人誌に寄稿してくださいましたので、ぜひお読みいただければと思います。

さて、2017 年 2 月に開催しました FlashAir Developers Summit が好評でしたので、また開催しようという話になりました。なんとなく、夏は MFT、冬は Developers Summit というサイクルができつつありますね。というわけで、昨年の FlashAir Developers Summit に引き続き、今回も FlashAir ハッカソンでお世話になった JellyWare（株）の崔さんにご協力いただきまして、2018 年 2 月 24 日に「FlashAir Developers Summit 2018」を開催する運びとなりました。

会場は Yahoo さんのご協力を得まして、紀尾井町にあります Yahoo さんのオープンコラボレーションスペース「LODGE」にて開催させていただけることになりました。ありがとうございました。午前中は組み込み系ハンズオンを開催し、同じ会場で午後から FlashAir Developers Summit 2018 を開催する運びとなりました。じむさんによる書き下ろしのソラちゃんのイラストでお客様をお出迎えさせていただきました。

図 5: サミット会場の様子

図 5: 描き下ろしソラちゃん看板

図 7: 決算資料の説明

第 2 回目のサミットのプログラムはさらに充実しまして、会場にはこの同人誌に寄稿いただいています綾瀬ヒロさん、タイプアール高瀬さん、いしかわきょーすけさん、蒼さやさん、マクニカ矢野さんにお越しいただきました。ご講演の内容の一部は今回この同人誌に寄稿いただいていますので、ぜひご覧ください。JellyWare の上田さんには、FlashAir を使ってビットコインの価格をリアルタイムでチェックできるガジェットについてお話いただきました。なお、こちらを使ったハンズオンも 2018 年 2 月 14 日に別途開催いただきました。また、DMM.make の岡島さん、野秋さんにも本イベントにてご講演いただきました。

ちなみに、FlashAir 芸人もなぜか某社の決算内容についてお話させていただきました（図 7）。また、ルネサスさん他各社からご提供いただいた景品によるじゃんけん大会も大変盛り上がりました。みなさまのご協力に改めて感謝申し上げます。ありがとうございました。

さいごに謎の予告

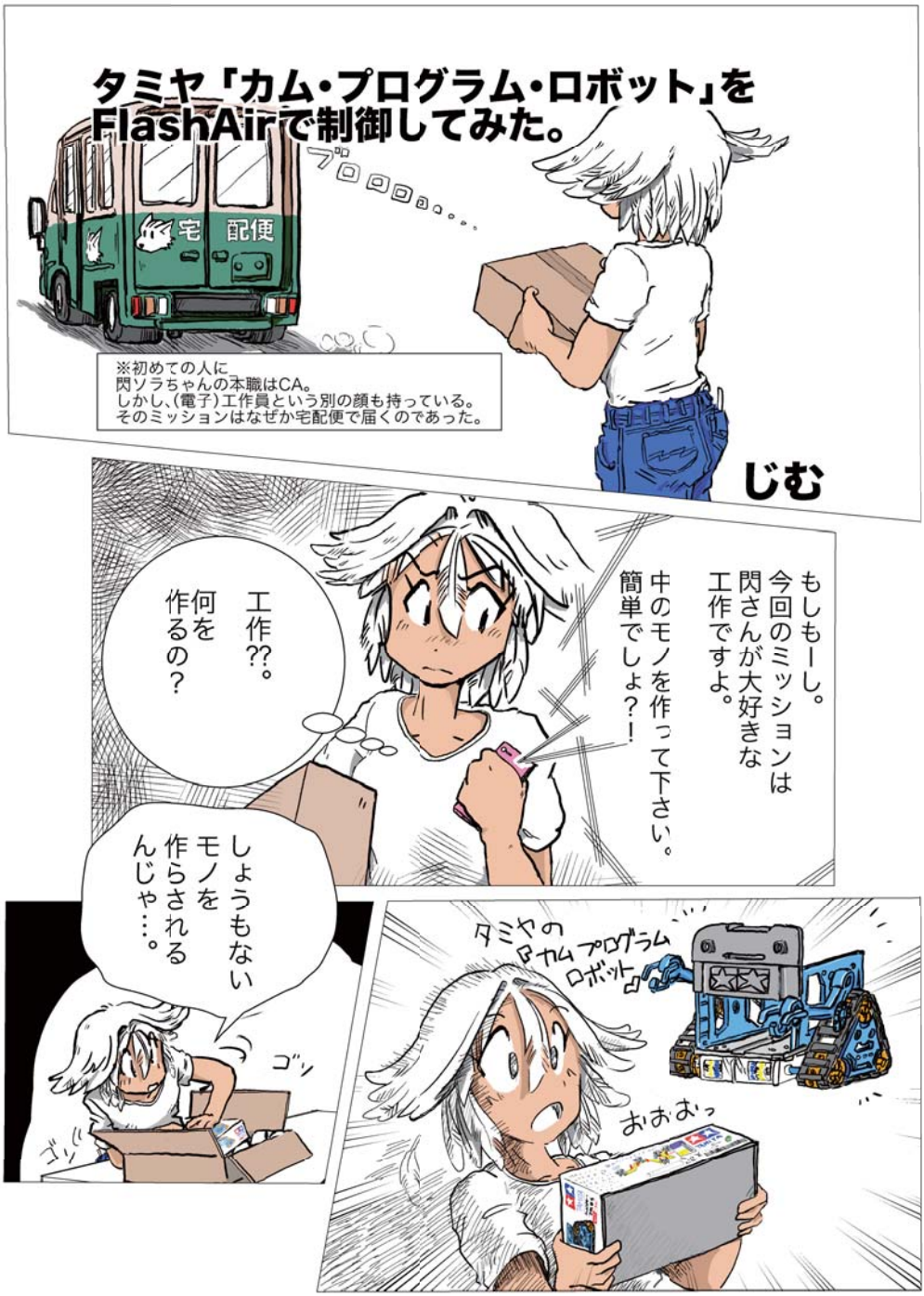
というわけで、今年の同人誌 5 号の入稿日を明日（2018 年 7 月 25 日）に控え、ただいま（25 日の午前 1 時ですが・・・汗）せっせと原稿を書いている最中です。どうして今年はこんなにギリギリの執筆なのかといいますと、実は余熱氏と一緒にある本を作成している真っただ中でして、これがかなり大忙しな状況になっています。

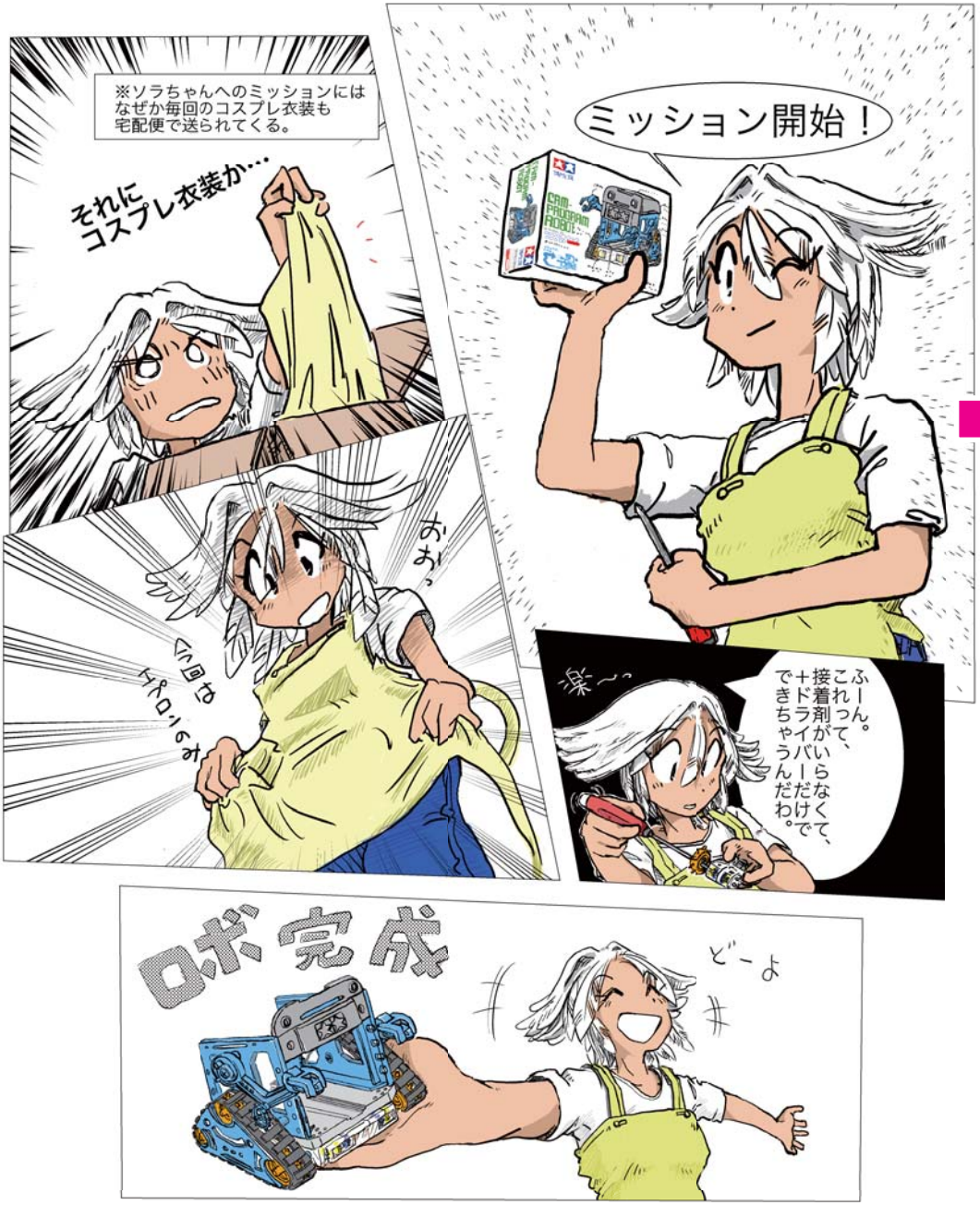
その制作中の本ですが、今年の 9 月に出版される予定となっております。どんな本かはまだ明らかにできませんが、おそらく弊社からプレスリリースなどの形で何らかの告知があるのではないか、と期待するところです。この同人誌を毎度お読みいただき、イベントなどにもお越しいただいている方でしたら、お読みいただくと「おや?」と思われること間違いなしかと思います。ちなみに FlashAir の本ではありません。残念ながら市販の予定はいまのところありませんが、図書館にてお読みいただけると思います。

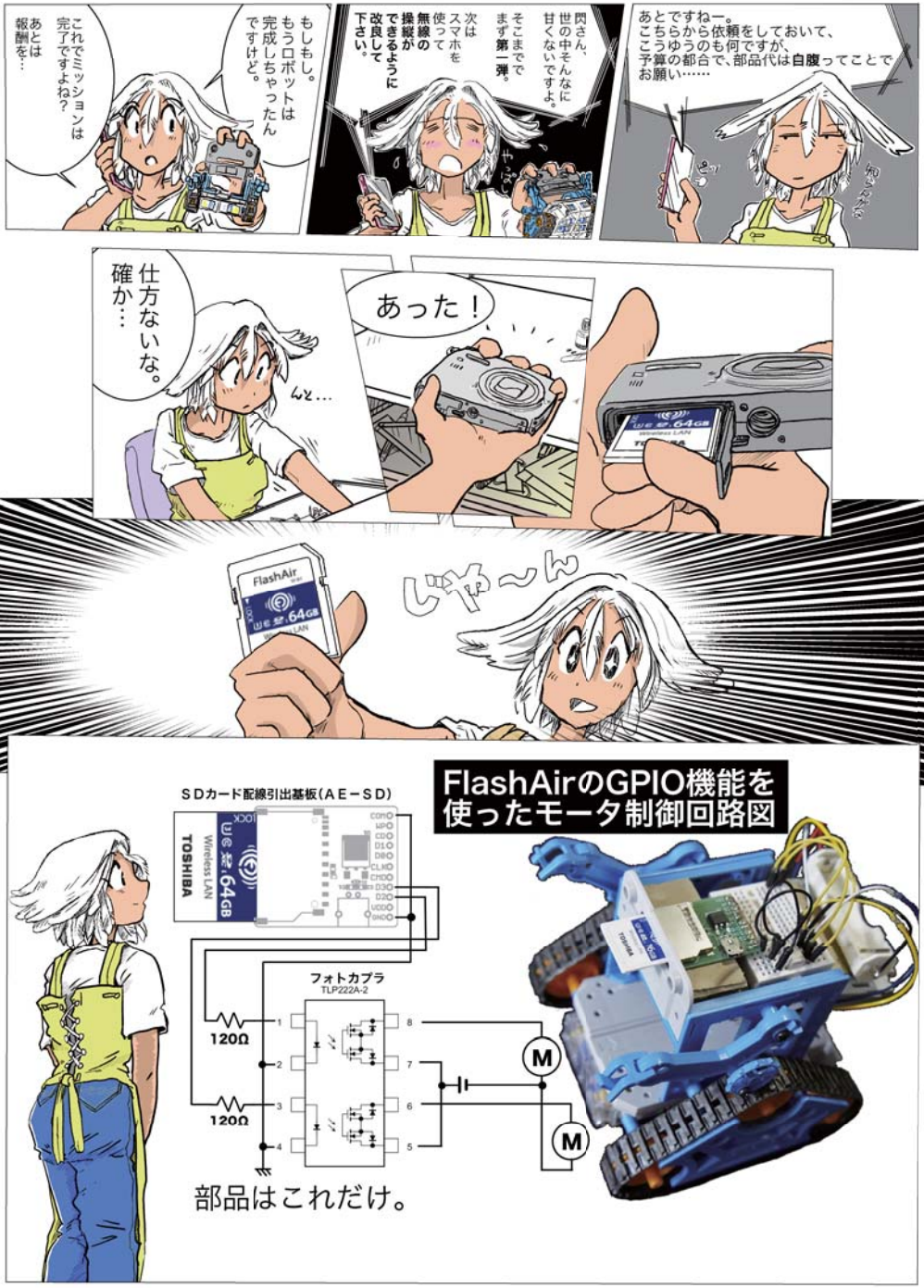
なんだかよくわからない謎の予告で恐縮ですが、情報がオープンにできるタイミングで、ツイッターなどでつぶやくと思いますので、是非ともよろしくお願いいたします！

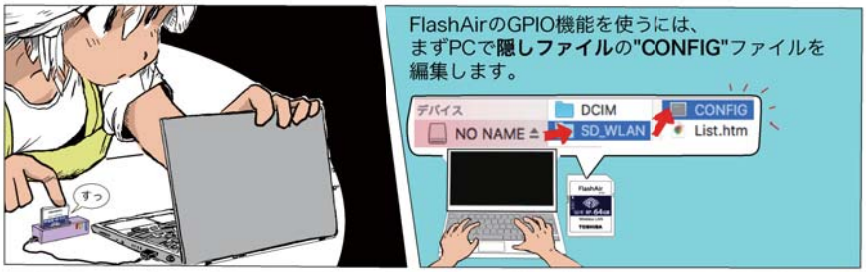
Pochio（@I_love_nintendo）

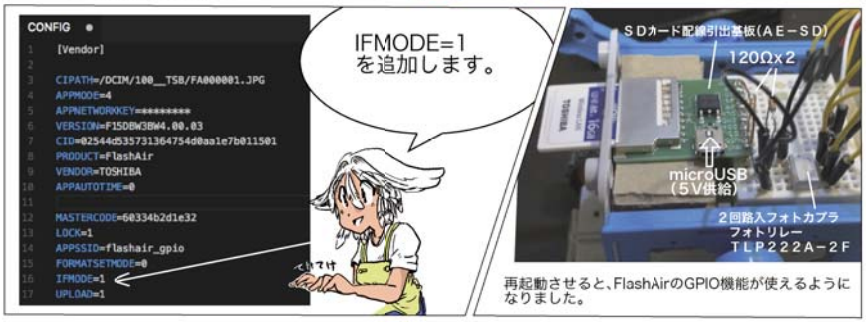
自称 FlashAir 芸人。昨年から引き続き芸人の出番があまりないのでお仕事募集中。ちなみに同人誌 1 号にソラちゃん誕生の経緯が書いてありますヨ。

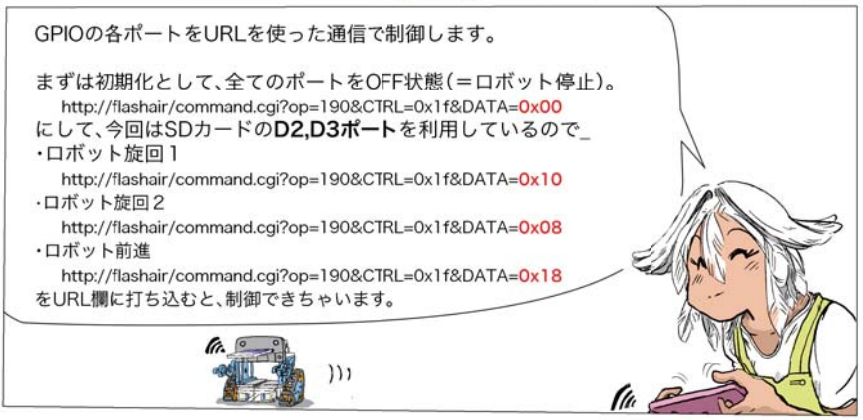

タミヤ「カム・プログラム・ロボット」を
FlashAirで制御してみた。
宅配便
プロロロロ…
※初めての人に
閃ソラちゃんの本職はCA。
しかし、(電子)工作員という別の顔も持っている。
そのミッションはなぜか宅配便で届くのであった。
じむ
もしもーし。
今回のミッションは
閃さんが大好きな
工作ですよ。
中のモノを作って下さい。
簡単でしょ?!
工作??。
何を
作るの?
しょうもない
モノを
作らされる
んじゃ…。
ゴソ
ゴソ
タミヤの
カム プログラム
ロボット
おおおっ


※ソラちゃんへのミッションには
なぜか毎回のコスプレ衣装も
宅配便で送られてくる。
それに
コスプレ衣装か…
ミッション開始！
おぉっ
今回はトイロボか
楽～
ふーん。
これって、
接着剤がいらなくて、
＋ドライバーだけで
できちゃうんだわ。
ロボ完成
どーよ


あとですねー。
ここちらから依頼をしておいて、
ここうゆうのも何ですが、
予算の都合で、部品代は自腹ってことで
お願い……
閃さん、
世の中そんなに
甘くないですよ。
そこまでで
まず第一弾。
次は
スマホを
使って
無線の
操縦が
できるように
改良して
下さい。
やっぱ
もしもし。
もうロボットは
完成しちゃったん
ですけど。
これでミッションは
完了ですよね？
あとは
報酬を…
仕方ないな。
確か…
あった！
じゃ～ん
FlashAir
SDカード配線引出基板（AE－SD）
TOSHIBA
Wireless LAN
64GB
フォトカプラ
TLP222A-2
120Ω
120Ω
M
M
FlashAirのGPIO機能を
使ったモータ制御回路図
部品はこれだけ。


すっ
FlashAirのGPIO機能を使うには、
まずPCで隠しファイルの"CONFIG"ファイルを
編集します。
デバイス
NO NAME
DCIM
SD_WLAN
CONFIG
List.htm


CONFIG
[Vendor]
CIPATH=/DCIM/100__TSB/FA000001.JPG
APPMODE=4
APPNETWORKKEY=********
VERSION=F15DBW38W4.00.03
CID=02544d535731364754d0aa1e7b011501
PRODUCT=FlashAir
VENDOR=TOSHIBA
APPAUTOTIME=0
MASTERCODE=60334b2d1e32
LOCK=1
APPSSID=flashair_gpio
FORMATSETMODE=0
IFMODE=1
UPLOAD=1
IFMODE=1
を追加します。
てりてり
SDカード配線引出基板(AE-SD)
120Ωx2
microUSB
(5V供給)
2回路入フォトカプラ
フォトリレー
TLP222A-2F
再起動させると、FlashAirのGPIO機能が使えるようになりました。


GPIOの各ポートをURLを使った通信で制御します。
まずは初期化として、全てのポートをOFF状態(=ロボット停止)。
http://flashair/command.cgi?op=190&CTRL=0x1f&DATA=0x00
にして、今回はSDカードのD2,D3ポートを利用しているので_
・ロボット旋回1
http://flashair/command.cgi?op=190&CTRL=0x1f&DATA=0x10
・ロボット旋回2
http://flashair/command.cgi?op=190&CTRL=0x1f&DATA=0x08
・ロボット前進
http://flashair/command.cgi?op=190&CTRL=0x1f&DATA=0x18
をURL欄に打ち込むと、制御できちゃいます。
)))


元に戻せば
デジカメも復活♪
どやっ
これで
ミッション
コンプリート!!


じむ
ソラちゃんも
5年目になりました。
彼女には、まだまだ
働いてもらいましょう。

ハンズオン @ 大阪と Airio-Base

（株）クレイン電子　福屋 新吾

2017 年 10 月 7 日、FlashAir Developers Summit ハンズオン @ 大阪が開催されました。関西人にとってなんとなく遠くに感じていた FlashAir 熱、今回そのアツさを目の前で、しかも関西屈指の電子部品取扱会社の共立電子産業セミナールームで行われました！参加者は配布されたセットを用いて学習しました（図1）。

図 1: セミナー開催ポスター、及び配布されたセット

L チカから FlashAir IoT Hub まで

まずは、大定番の L チカこと LED 制御。参加者の方々が第一歩を踏み出し、FlashAir の魅力へとハマっていく…のはずが、無線 LAN に接続する際、SSID リストに FlashAir がズラリと並び手間取りました（笑）。セットアップで少々時間がかかってしまったものの、ほとんどの方が L チカ、温度センサーのデータ取得を経て、IoT Hub へデータを上げることができ、無事にハンズオン終えることができました（図2）。

図 2: 会場の様子、及び終了後の集合写真

Airio-Base の誕生

今回のセミナーでは、評価・開発ボードとして、Airio と Arduino UNO を使用しました。Airio は、FlashAir の GPIO 操作を理解したり Arduino の SPI 端子に接続して拡張するのには最適で、入門者にも扱いやすいボードです。しかし、一方でもっと便利な 1 ボード構成の FlashAir 用ハードの要望もありました。そこで、セミナー準備の傍ら「FlashAir の為のベースボードを作ろう」として開発が始まり、Airio-Base（えありお・べーす）が誕生しました。Airio-Base は、以下の機能を備えた FlashAir に最適なベースボードです。

Airio-Base の特徴

- NXP セミコンダクターズ社の LPC11U35 を搭載
- arm Mbed（Mbed 2 classic 対応）を利用した開発が可能
- デバッグ用 SWD ポートを用意（オプション）
- SPI 接続された SD メモリ用カードスロット
- GPIO に接続された LED、2 つのスイッチ
- Arduino 用シールド基板が搭載可能な 3.3V IO ポート

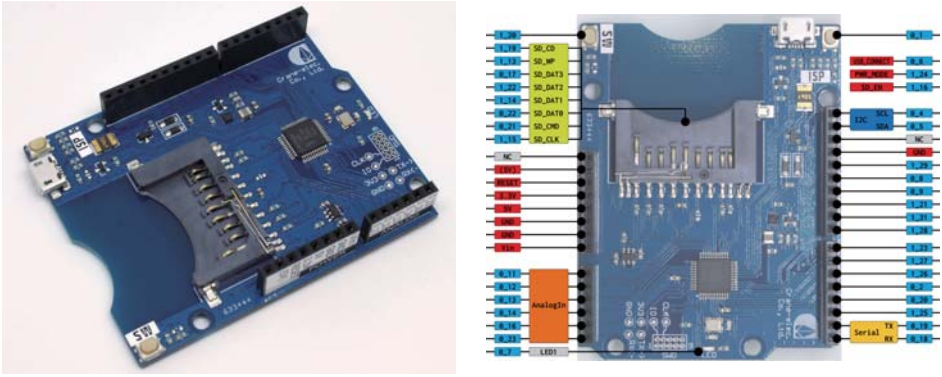


図 3: Airio-Base の外観、及びピンアウト

2018 年 2 月現在、試作である先行販売モデル（図 3）が完成し、FlashAir Developers Summit 2018 で利用されました。現在、量産モデルを開発中で Maker Faire Tokyo 2018 で発表に向けて進めています。[1]

1 http://crane-elec.co.jp/?page_id=207

株式会社クレイン電子 (@crane_elec)　福屋 新吾

技術課は私一人、社では実質何でも屋。まさか執筆するに至るとは…。余熱さんをはじめ皆様にお世話になりっぱなしです。頑張りますっ！

FlashAir と AI スピーカーで鉄道模型を動かそう！

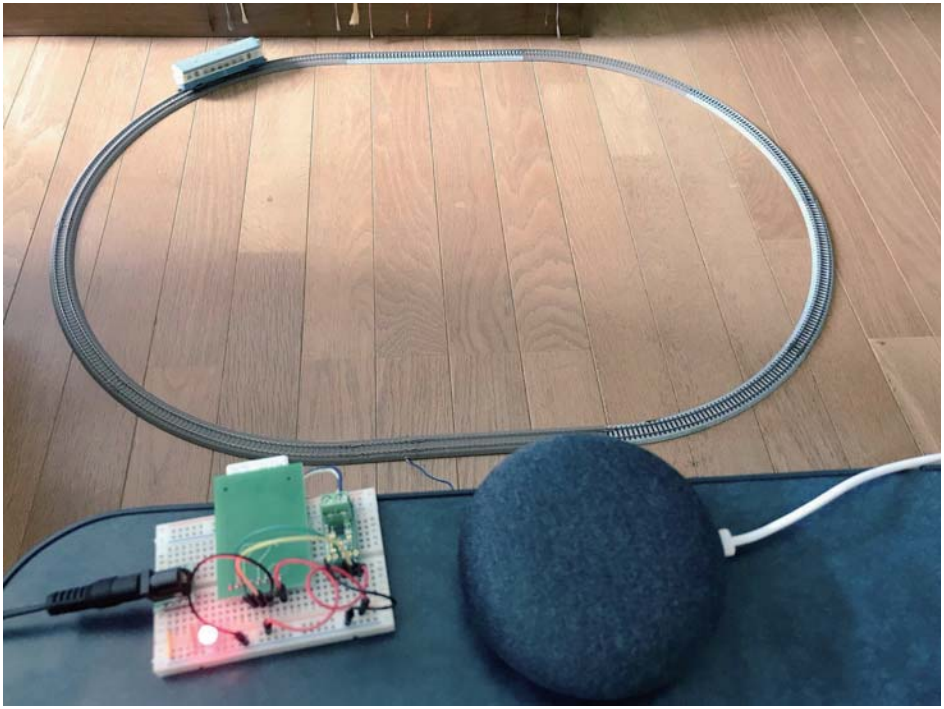
綾瀬ヒロ

FlashAir で鉄道模型をコントロールして遊んでいる綾瀬です。

さて、今回は昨今日本でも身近になりつつある AI スピーカーと FlashAir を連携させて、鉄道模型を動かしてみたいと思います。

現在、国内では Google Home、Amazon Alexa、LINE Clova などが容易に入手できますが、今回は手元にあった Google Home を利用しました。

これらの AI スピーカーを、FlashAir と連携させるには、いずれも直接ローカルで連携させることができず、パブリッククラウドサービスを経由することになります。できるだけ簡単に連携できるように、今回は Web サービス同士を連携できるサービスのひとつである “IFTTT” と、Microsoft Azure のノンコーディングで連携機能を構築できるサービスのひとつである “LogicApps”、気軽に文章認識を構築できる “LUIS”（Language Understanding）を利用しました。全体では多くのサービスを連携させるので、すべてを紹介することが難しいため、本稿では、FlashAir とクラウドサービスを連携させる部分をご紹介します。



全体の概要

今回作成した FlashAir と AI スピーカーを連携させる全体の概要を図 1 に示します。このなかで、FlashAir と Azure IoT Hub の連携部分を次項以降に紹介します。FlashAir とモータードライバー DRV8830 との連携については前号「FlashAir 同人誌」第 4 号の記事「FlashAir W-04 の I2C 機能を使おう!」を参照してください[1]。

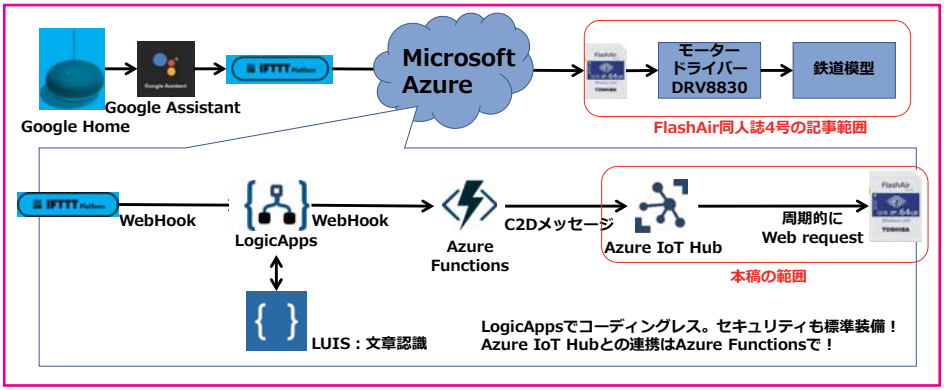


図 1: 全体概要図

Azure IoT Hub の C2D メッセージ機能

Microsoft のパブリッククラウドサービス Azure では、デバイス機器からクラウドへのデータを受信したり、クラウドからデバイス機器へデータを送信したりするための仕組みを Azure IoT Hub という機能で提供しています。Azure IoT Hub では登録したデバイス機器を管理可能であり、管理されたデバイス機器とのデータ連携が可能になっています。

Azure IoT Hub からデバイス機器にデータを送信するには、Azure IoT Hub の C2D メッセージ（クラウド to デバイス）機能にメッセージデータを登録し、デバイス機器からは以下の REST API エンドポイントを周期的に確認（HTTP GET メソッドを呼び出す）し、自デバイス向けのメッセージが届いていれば、それを受信します（表 1）。

表 1: Azure IoT Hub の C2D メッセージ（取得）

URL	https://{iotHubName}.azure-devices.net/devices/{deviceId}/messages/devicebound	
Method	GET	
Header	api-version	2016-11-14
	Authorization	SAS トークン
	Content-Type	application/json

1 https://www.flashair-developers.com/ja/documents/books/#fourth

確実に受信できたことを確認できたら、そのメッセージを指定してメッセージを消去（HTTP DELETE メソッドを呼び出す）します（表 2）。

表 2: Azure IoT Hub の C2D メッセージ（消去）

URL	https://{iotHubName}.azure-devices.net/devices/{deviceId}/messages/devicebound/{etag}	
Method	DELETE	
Header	api-version	2016-11-14
	Authorization	SAS トークン
	Content-Type	application/json

上記の {iotHubName} は、Azure IoT Hub を作成時に付与した名前、{deviceId} は Azure IoT Hub にデバイス機器（ここでは FlashAir）を登録したときに付与した名前、SAS トークンは、Azure IoT Hub で付与されたキーを用いて生成したアクセストークンになります。また etag は、メッセージを一意に示す文字列になります。

Azure IoT Hub の C2D メッセージ機能の詳細については、以下を参照ください。

https://docs.microsoft.com/ja-jp/azure/iot-hub/iot-hub-devguide-messages-c2d

FlashAir の Lua による C2D メッセージ受信機能の実装

FlashAir の Lua スクリプトで、Azure IoT Hub の C2D メッセージ機能の REST API を呼び出す部分を以下に紹介します。

C2D メッセージ機能で AI スピーカーで取得した音声認識文字列を、LUIS で速度と方向を抽出し、C2D メッセージで FlashAir に送信することで、メッセージを解釈してモータードライバーを動かし、その結果鉄道模型を動かすことができるようになります。

今回実装する上で、cocteau666 さんが GitHub（https://github.com/cocteau666）で公開されている以下のソースコードを利用させていただきました。

FaUtil.lua : URL エンコーダー関数（urlencode）を使用します。

FaAzureIoTSAS.lua : Azure IoT Hub の SAS トークンを生成する関数（sas.create）を使用します。

FaTimestamp.lua : SAS トークンを生成する際の現在時刻を NICT から取得する関数（getTimestamp）を使用します。

使用する上で、FaTimestamp.lua は、コピーしたあとに 9 行目以降のコードを削除してください。saveTime 関数で使われている os 関数は FlashAir では使えないためです。

まずは Azure IoT Hub からメッセージを取得するコードを以下に示します。

```
local b,c,h = fa.request{
  url="https://"..iotName..".azure-devices.net/devices/"..devicename..
      "/messages/deviceBound?api-version=2016-11-14",
   method  = "GET",
   headers = {["Authorization"] = auth,
              ["Content-Type"]  = "application/json"},
}
```

つぎに Azure IoT Hub からメッセージを取得した後に消去するコードを以下に示します。消去するには、さきほど取得したメッセージのヘッダーに含まれる ETag の 36 バイトの文字列を取得し、呼び出す url に含める必要があります。ここでは単純にヘッダーから Etag: の文字列を検索して、そのあとの 36 バイトの文字列を取得しています。

```
local spos, epos = string.find(h, "ETag:")
if(spos ~= nil) then
  local etag = string.sub(h, spos+7, spos+42)
  local b,c,h = fa.request{
  url="https://"..iotName..".azure-devices.net/devices/"..devicename..
      "/messages/deviceBound/" .. etag .. "?api-version=2016-11-14",
  method = "DELETE",
  headers = {["Authorization"]  = auth,
             ["Content-Type"]   = "application/json"},
   }
end
```

最後に

今回、クラウドサービスと FlashAir を連携させる方法を紹介しました。AI スピーカーとの連携もこれで可能となりますし、アイデア次第でほかにも連携できるでしょう。

ところで、FlashAir にも FlashAir IoT Hub というサービスがあります。こちらもクラウドから FlashAir の Lua スクリプトを実行する仕組みがありますが、FlashAir IoT Hub がそれ以外のサービスと連携できないため AI スピーカーとの連携には使えませんでした。IFTTT 連携など実装してくれるとより可能性が大きくなるので、期待したいと思います。

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/sample5.zip

デジタル鉄道模型コントローラ DSair

Yaasan@Desktop Station

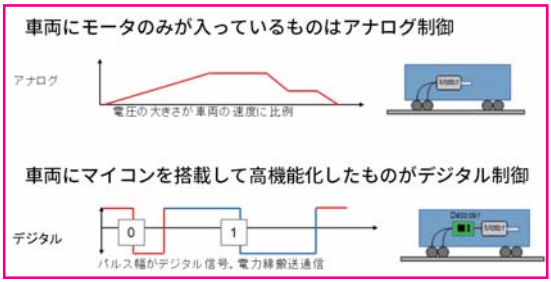


図 1: 鉄道模型の制御方式

デジタル鉄道模型には、独メルクリン社のメルクリンデジタル（Marklin Motorola/MM2 や mfx ともよばれる）と、全米鉄道模型協会（NMRA）のデジタルコマンドコントロール（DCC）という方式があります。しかしながら、日本の鉄道模型メーカーは 100 年前と何も変わらずアナログ方式に固執しており、国内ではあまり普及していませんが、欧米・オセアニアでは、既にデジタル方式が主流です。日本を除くアジアでもデジタル方式の普及が年々進んでいます。鉄道模型において、日本は世界から遅れた発展途上国なのです（図 1）。

デジタル方式の最大の特徴は、2 本の線で電力と信号を同時に送る電力線搬送通信を行う点です。模型の中にマイコン基板（デコーダとよばれます）を搭載したことで、同じ線路上で複数の車両を個別制御できること、ファンクションとよばれる様々な固有機能の制御が可能になり、音を出したり、照明を制御したり、自動運転がとても楽にできたりします。 1つの線路に1車両しか制御できないアナログ方式から大きく進化しています。

そんな状況下で、FlashAir を使って、デジタル鉄道模型をスマートフォンで制御するための機器 DSair を開発しました。メルクリンデジタルと、DCC に対応した鉄道模型の車両やポイント、信号機を制御することのできる新しいコンセプトのコントローラです。

世界標準の DCC で鉄道模型を気軽に無線で遊びたい！

Wi-Fiを使ってスマートフォン等でDCCやメルクリンの鉄道模型を運転するソリューションに限ると、世界にはいろいろな製品が既に存在します。しかしながら、それらの製品は PC を Web サーバーとして使ったり、または有線 LAN の配線を無線 LAN ルーターに繋ぐ手間が必要であったり、気軽にとは言えない商品ばかりでした。また、無線部分はオプションで、別に本体が必要なケースが多かったのです。さらに、海外メーカーの製品しかなく、日本にはそのような製品を出すメーカーもありませんでした。

そこで、簡単に持ち運びできて、電源を入れれば、スマートフォンをつなげるだけで、アプリのインストールも要らない、シンプル＆簡単な DCC を実現する無線 DCC コントローラの開発を行うことにしました。

FlashAir を選択した理由

スマートフォンを使って、複数の人が同時に、かつ、面倒なアプリのインストールが不要な方式、つまり Web アプリによって遊べることをコンセプトとして決めていました。スマートフォンには、Wi-Fi と Bluetooth が標準的に搭載されているので、この 2 方式について検討しました。

Bluetooth を使用する場合は、同時に複数人で使用するにはペアリング作業などで煩雑になることが想定されました。また、Bluetooth SIG の会員になる費用など問題がありました。このため、Wi-Fi をターゲットに絞りました。

当初、安価で入手性のよい ESP-WROOM を試しました。ESP 社のモジュールは海外の DCC メーカーで採用例が複数あります。しかし、発熱が大きく通信スピードが遅い等、欠点が目立ちました。Raspberry Pi も候補でしたが、高機能すぎて使いにくく断念しました。そこで、当時あった FlashAir W-03 を使って試作を行ったところ、まずまずの動作をしたのですが、電源投入の起動時間と Web アプリのダウンロードの遅さが気になり、無線 DCC コントローラ開発はいったん中断しました。

そして 1 ～ 2 年ほど経過して、2017 年に W-04 が登場したことを知り、再チャレンジをしたところ、W-03 で悩んでいた全ての問題が解決していて、これなら無線 DCC コントローラに問題なく使用できる！と確信し、製品化を一気に進めました。

DSair のハードウェア構成

マイコンで生成した DCC 信号を出力するドライバ IC には TB6643KQ、SD とのレベル変換用には VHC32 を選定しました。電解コンデンサや電源 IC などは日本メーカー製を使う事にこだわりました。Arduino UNO は、FlashAir を動かすには電源が弱いので、DC/DC コンバータと大きめのコンデンサを採用して電源強化している点がポイントです。

使用しているほとんどの部品は、秋月電子で買えるものにしてあります。基板形状は、あえて Arduino UNO と同じ形（図 2）にして、コネクタも位置を合わせこんでいます。ケースを新たに製造するとコストが膨大になるのと、在庫を持つ必要があるため、中国の Aliexpress で簡単に安価で大量に買える汎用 Arduino UNO ケースをそのまま使えるようにしています。

図 2: DSair

DSair の回路設計・基板製造

回路図は、水魚堂[1]の BSchV3 で設計し、アートワークも水魚堂の MBE を使っています（図 3）。

基板製造・部品実装は、中国・深センの Elecrow[2] です。基板製造だけでなく、PCB Assembly サービスも依頼しています。表面実装部品を多く使用しているため、ユーザーのはんだ付け負荷軽減を図るためです（図 4）。

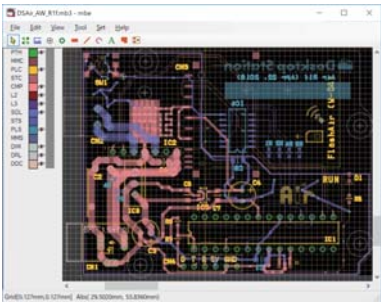


図 3: DSair のアートワーク

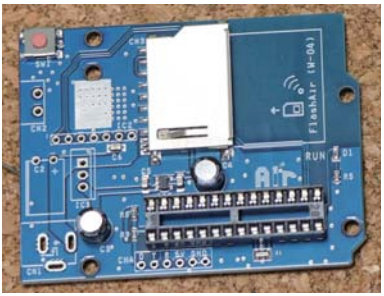


図 4: Elecrow での基板製造と部品実装

Elecrow の PCB Assembly サービスはエクセルシート、ガーバーファイルを送るだけで製造を行ってくれます。MOUSER や Digikey を指定すると Elecrow で代わりに調達してくれますが、無い部品は、日本から近所の郵便局を通じて EMS 等で Elecrow のオフィスに送ります。なお、抵抗やコンデンサ類は、Elecrow が在庫しており、部品表で指定すると調達費を抑えられます。このように基板製造だけでなく、少しの手間をかけるだけで、気軽に実装も依頼できます。

過去に多くの基板製造・実装実績があり、トラブルにもきちんと対応してくれるため、とても重宝しています。

DSair のソフトウェア設計

図 5: DSair のソフトウェア

DSair は Arduino UNO ベースで開発されており、マイクロチップ社の ATMEGA328P-PU を使っています。

よって開発環境は Arduino IDE です。DSair を Arduino UNO や SD カードシールドで実現する DS shield with FlashAir というソリューションも提供しています。ソフトは全て完全互換・共通化しているので、お財布の事情とカスタムの好みに合わせて DSair 基板か、お店で入手できる Arduino UNO+SD カードシールド +DS シールドといった構成を選べるようにしています。

FlashAir と ATMEGA328P との通信は標準の SPI を使用しています。FlashAir

1 http://www.suigyodo.com/
2 https://www.elecrow.com/

Developers で紹介されている iSDIO ドライバをそのまま使用させていただいています。このため、通信仕様のベースは、綾瀬ヒロ様のアナログ鉄道模型制御のサンプルです。FlashAir 側のソフトウェアは、いわゆるフロントエンドの Web アプリとし、SD_WLAN フォルダ内にソフト一式を入れて、アクセスしてくるスマートフォンにダウンロードさせて動かしています。

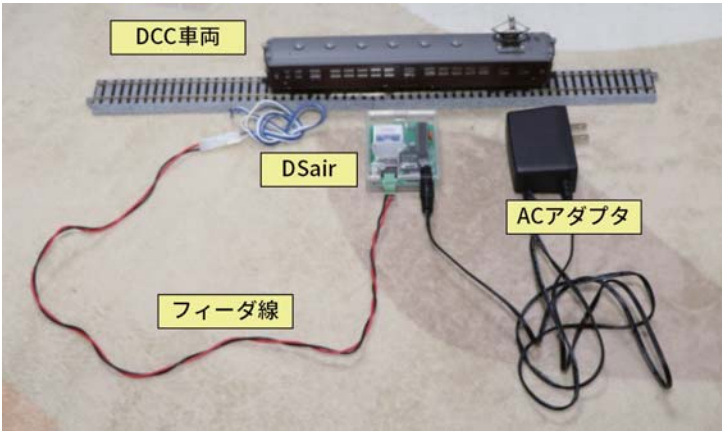
コマンドは、CGI で簡単に受け渡しできる共有メモリで ATMEGA328P と受け渡しをしています。なるべくリッチに見せるために、HTML5 Canvas を活用してスピードメーターを描画したり、jQuery UI で CSS を調整しながら作成しました（図 5）。

さいごに

DSair の air は FlashAir の air です。FlashAir が無ければ、この機器は誕生しませんでした。Raspberry Pi やいろいろな Wi-Fi モジュールを試して、パフォーマンスや安定性、取り扱いの容易性では、FlashAir がベストと考えています。私は、FlashAir の実力のほんの少ししか使ってません。これから、もっと使い倒していこうと考えております。

電子工作に興味のある方々で、IT や Web と柔軟に対応できるガジェット・デバイスを作りたい方は、FlashAir がベストな選択肢と自信を持ってお勧めできます。

DSair の Web ページ https://desktopstation.net/wiki/doku.php/dsair_jp



Yaasan（https://desktopstation.net/）

デジタル鉄道模型と電子工作が大好きな電機屋。デジタル鉄道模型に新しい風を起こすハードウェアスタートアップとして Desktop Station を立ち上げた。DCC 電子工作連合で主査を務める。

デジカメで撮った写真を VR 空間で閲覧する

南

2018 年 5 月に Facebook 傘下の Oculus 社から、スタンドアローン型 VR ヘッドセット「Oculus Go」が販売開始されました。嬉しいことに日本でも同日、消費税／送料込みで 23,800 円から購入可能となっています。お手頃な価格もさることながら、スタンドアローン型の圧倒的な手軽さが人気を集めているようです。私自身、初の VR デバイス購入となりました。

さて、この Oculus Go、標準で Web ブラウザ（Oculus Browser）が搭載されていて、WebVR という仕様に準拠しています。これにより、通常の Android アプリ開発の要領でネイティブ VR アプリを開発する以外に、Web 技術だけを使って VR アプリを開発することが可能となっています。

図 1: ライブファインダーにはなりません

WebVR アプリをスクラッチから開発することも出来なくはありませんが、対応しているフレームワークを使用するほうが一般的でしょう。軽く調べた限りでは、Facebook の React 360 と Mozilla の A-Frame が使えそうでした。今回は、個人的に馴染みのある React ファミリーの React 360 を使って、FlashAir 向け画像ビューワアプリを開発してみました。

React 360 について

もとは React VR という名前で公開されていましたが、Oculus Go の発売と同じタイミングで React 360 という名前にリブランディングされました。単に名前が変わっただけでなく、実装も大きく見直されています。

何ぶんリリースされたばかりなので、サクッと動作確認できるサンプルが少なかったり、ドキュメントに一部古い記述が残っていたり[1]、日本語が表示できなかったり[2]、Safari で表示させると何も表示されずに真っ暗になってしまったり[3]、ある日突然 Oculus Browser の VR モードですごく低い解像度で表示されるようになったり[4]、といった不具合が盛りだくさんでした。執筆時点で未だ解消されていないものも多いです。

1　README.md には Node.js v6 以降が必要と書かれているが、実際には Node.js v8 以降を使わないとリリースビルドができない

2　https://github.com/facebook/react-360/issues/463

3　https://github.com/facebook/react-360/issues/486, 修正済み

4　https://github.com/facebook/react-360/issues/508, 修正済み

しかしながら、書き慣れた React の作法で書けること（実際は React Native に近いです）や、PC やスマホのブラウザで表示させた場合でもドラッグアンドドロップや加速度センサでそれなりにそれっぽく使えることなど、メリットはそれなりに多いと感じています。

なお、Oculus Go は 3DoF とよばれるタイプで、周りを見渡すことはできますが、空間を自由に歩き回るようなことはできません。React 360 は（名前的にも）3DoF に特化していると感じます。

FlashAir IoT Hub - VR Demo

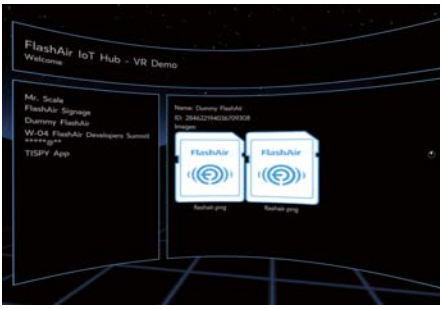


図 2: VR Demo にログインしたところ

まず手始めに、FlashAir 専用クラウドサービス「FlashAir IoT Hub」にアップロードしたデータを表示するデモアプリを書き始めました。FlashAir IoT Hub は Web API の仕様を公開しているので、誰でも連携アプリを開発することができます。今回はお手軽に開発できる、クライアントサイド（Javascript）Web アプリ（OAuth 2.0 Implicit Grant）を選択しました。React 360 は React と同じく SPA（Single Page Application）として開発できるので、生成されたファイルを静的な Web サーバに設置するだけでアプリを公開することができます。

デモページは下記にあります。FlashAir IoT Hub のアカウントを作れば誰でも試すことができます。このページは GitHub Pages を利用して無料でホスティングしてもらっています。

デモページ : https://kminami.github.io/FlashAirIoTHub-VR/

ソースコード : https://github.com/kminami/FlashAirIoTHub-VR/

FlashAir IoT Hub がサポートしているデータのうち、どんなデータを表示すると楽しいかなーと考えたのですが、作りやすさと見栄えのバランスで画像ファイルの表示を選択しました。なので、画像ファイルをアップロードしておかないと、実質何も表示されません。

FlashAir IoT Hub にアップロードされた画像ファイルを取得するにはアクセストークンをヘッダーに入れる必要があるので、単純に Image タグに URL を入れるだけでは表示できません。こういう場合は Object URL を使うと良いです。やり方は fetch API を使って Blob データを取得し、URL.createObjectURL に渡すだけです。ただし、React 360 でこれを単純にやるとうまくいきません。上記のソースコードを見て頂くと良いですが、Native Module を使って UI スレッド側で Object URL を作り、それを React 側で Image タグに渡すようにしてください（React 側では Blob や fetch が polyfill されているのですが、この Blob の実装が React 360 で使われているバージョンでは不完全で、fetch が Blob を

使えないものと判断した結果、.blob メソッドが undefined になってしまいます）。

Oculus Go 入手から三週間くらいかけて一通り作ってみた感想としては、VR アプリを作るなら UI 設計が大事ということでした。せっかく VR 空間に表示するのだから、VR ならではの「見せ方」というのが重要になります。UI を考えるのはあまり得意ではないので、この辺で開発の手が止まってしまった感じです。

教訓としては、あまり「枠」を作りすぎない方が良いなと思いました。React 360 では Surface という平面上に UI Component を配置していくのが基本になるのですが、必要以上に黒塗りの枠を置いてしまうと、VR モードで見たときになんだか窮屈な感じになってしまいます。概念的な「枠」は依然として存在するのですが、単にシースルーというか色を塗らないでおくだけで、ずいぶん印象が変わります。

あと、背景画像はデフォルトのものを使っていますが、個性を出すためにはやはり独自のものを使いたくなります。自作は私には難しいので、Theta や Insta360 のような 360° カメラが欲しくなりました。

FlashAir VR

FlashAir IoT Hub のデモページを一通り作った後、FlashAir のブラウザユーティリティ（ブラウザでアクセスしたときに表示されるファイルビューワ）を VR 化したほうが面白そうだということに気づきました。予め必要なファイルを書き込んでおけば、デジカメに入れた FlashAir と Oculus Go だけで簡単に VR 体験ができてしまいます。充電さえしてあれば、電源もインターネット接続も必要ありません。

図 3: PC のブラウザで表示したところ

FlashAir IoT Hub - VR Demo でハマリどころは一通り押さえていたので、FlashAir のブラウザユーティリティをVR化するのは非常に簡単でした。FlashAir Developers のチュートリアル[5]に従えば、FlashAir 上のファイル一覧を取得したりするのも簡単です。ソースコードと使い方は以下のアドレスにあります。Lua スクリプトを使用しないので、Lua に対応していない FlashAir W-02 でも動きますよ！

ソースコードと使い方 : https://github.com/kminami/FlashAir-VR

FlashAir IoT Hub の場合と異なり、FlashAir のブラウザユーティリティはユースケースが明確（ファイルを一覧し、表示するだけ）なので、UI 設計の迷いは少なかったです。また、IoT Hub のほうで学んだ教訓を活かして作り始めたので、UI も割とすっきりしたものになったと思います。

VR は写真じゃ伝わらないので、ぜひ実際に試してみてください！ 我が家では妻と一緒に娘の写真を撮りながら見せ合って楽しみました。

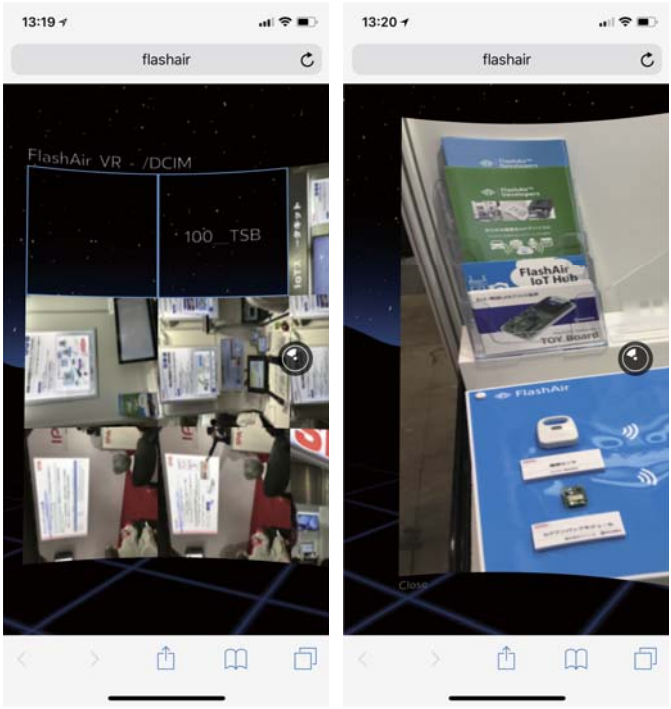


図 4: iPhone のブラウザで表示したところ

5 https://www.flashair-developers.com/ja/documents/tutorials/web/

南 (@Nang_JP)

Web 系エンジニア。FlashAir IoT Hub の中の人。

Oculus Go は、メジャーどころの VR ゲームを摘まみ食いしつつ、もっぱら Amazon プライムビデオ用のヘッドマウントディスプレイとして使用しています。VR 空間で仕事したい。

デジカメ内の画像を HoloLens で表示する

noitak

2017 年に放映されたアニメ『少女終末旅行』の劇中で、主人公のひとりが持つカメラ内の画像が空中に表示されるというシーンがありました。FlashAir と HoloLens、Vuforia を使えばこのシーンを（疑似的にですが）再現できるのではないかと考え、実装してみました（図 1）。

図 1: 空中に画像を浮かべた HoloLens の視界

HoloLens とは

Microsoft の MR[1] ヘッドマウントディスプレイです（図 2）。透過型ディスプレイで現実世界に重ねて 2D や 3D のデジタルコンテンツを表示することができます。また複数のセンサーで空間を認識し、現実の机に CG が隠れる、といった表現が可能です。

今回はこの HoloLens を使って無線 LAN 経由で FlashAir から画像を取得し、表示します。

図 2: HoloLens

1　Mixed Reality= 複合現実

Vuforia とは

Qualcomm が提供する AR[2] ライブラリです。平面や立体のマーカーを認識して、位置や角度を取得することができます。商用でなければ無料で使用できるので、手軽に AR 機能を試すことが可能です。また、後述する開発環境 Unity との親和性が高く、扱いが容易です。

「人間の操作なしで勝手に画像が表示されていく」表現をしたかったので、HoloLens 上で動作させ、デジカメ背面液晶の画像認識をトリガーにして、FlashAir の API にアクセスします。

システム構成

システム構成を図 3 に示します。

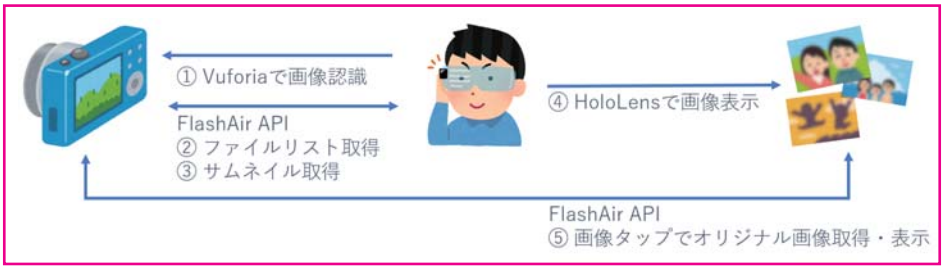


図 3: システム構成図

開発環境

HoloLens アプリを開発する方法はいくつかありますが、今回はゲームエンジン Unity と、MixedRealityToolkit-Unity（MRTK）を使用します。使用したバージョンは次のとおりです。

- Unity 2017.2.2p1
- MRTK 2017.2.1.4

Unity や MRTK は開発速度が非常に速く、うかつに最新版を使用するとビルドに失敗したりするので注意が必要です。

2　Augmented Reality= 拡張現実

Vuforia の画像認識を接続のトリガーにする

HoloLens 上で Vuforia を用いて任意の画像を認識するためには、下記の手順で行います。

1. **開発者ポータル で開発者登録**
2. **認識したい画像をポータルで登録**
3. **データベースをダウンロード**
4. **Unity にデータベースをインポート**

今回は Vuforia の以下の機能を使用しました。

図 4: デジカメ背面液晶の画像を認識中

- **認識した画像の位置にオブジェクトを表示する**
- **認識状態の変化タイミングを取得する**

デジカメ背面液晶の画像を認識したら、あらかじめ指定しておいた「接続中」文字列を画像の上方に HoloLens で表示します（図 4）。また、認識したタイミングをイベントハンドラで検知し、FlashAir への Web API アクセスを開始します。

HoloLens から FlashAir の API にアクセスする

あらかじめデジカメを起動して、HoloLens を FlashAir の無線 LAN に接続しておく必要があります。

ファイルリストの取得

最初にファイルリストを取得します。今回はディレクトリ名を決め打ちしましたが、トリガー画像によって取得先を切り替えるなど、いろいろ工夫できそうです。Unity から Web リクエストを発行するコードを以下に示します。

```
var url = "http://flashair/command.cgi?op=100&DIR=/DCIM";
var request = UnityWebRequest.Get(url);
yield return request.SendWebRequest();

if (request.isNetworkError || request.isHttpError) {
    // エラー処理
} else {
    // 正常に取得完了。レスポンスを解析・処理する
    // 省略…
}
```

正常に終了すると指定したディレクトリのファイル名リストが得られます。

サムネイルの取得

最初はいきなりオリジナル画像を読み込もうとしたのですが、通信速度の関係で表示が遅くなってしまいました。そこでまずサムネイルを表示し、その後必要に応じて高画質のオリジナル画像に差し替えることにしました。基本的なコードはファイルリストの取得と同様ですが、url は以下になります（directory と filename はファイルリストの取得結果より）。

```
var url = "http://flashair/thumbnail.cgi?" + directory + "/" + filename;
```

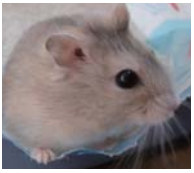
オリジナル画像の取得

サムネイル画像がタップされたらオリジナル画像を読み込みます。HoloLens はジェスチャーで操作するので、タップ操作は AirTap（立てた人差し指を前に倒す動作）になります。ちなみにカーソル移動は、正面に浮かんでいるカーソルを頭の動きで移動させます。url 指定は次のとおりです。

```
var url = "http://flashair" + directory + "/" + filename;
```

最後に

デジカメ内の画像を HoloLens で表示する方法を簡単に紹介しました。画像認識をトリガーにして「実体のカメラ」にアクセスするところ、ちょっとハイブリッド感があって良いのではないかと思っています。MR/AR/VR の世界は今どんどん面白くなっているところなので、興味を持ってもらうきっかけになれたら幸いです。



noitak（@noitak）

フリーランスのエンジニア。もうすぐ発表されそうな次世代 HoloLens が待ち遠しい、趣味の MR/AR/VR 開発者。

小さなゲームミュージックプレイヤー

高野 修一

かつてゲームを彩った FM 音源による BGM。今も心に残っている方は多いのではないでしょうか。そんな FM 音源のチップが指先サイズのモジュールとして入手できるようになりました。FlashAir と組み合わせて小さな演奏システムをつくってみましたので、ここで紹介します。

YMF825Board

YMF825 はヤマハ製の FM 音源チップで、4 オペレータ 16 チャンネルという、自分の世代にはたまらないスペックを持ったチップです。そのYMF825 を利用しやすい形にしたモジュールが YMF825Board で、スイッチサイエンスや秋月電子通商から購入することができます。

このボードが Maker Faire Tokyo 2017 で突如販売され、突然の FM 音源の復活に大変心躍るものを感じたわけですが、このチップのいいところの1つに、昔の FM 音源のような 8bit パラレルバスでの制御ではなく SPI での制御になっている点があります。そう、FlashAir の SPI 機能を使用して制御することができるわけです。実は以前から FlashAir を使って何か作ってみたかったので良い題材ができました。

プロジェクトの仕様

せっかくの 16 声も発生できる音源ですから、曲を演奏したいですよね。FlashAir のストレージに保存されているファイルを演奏できるようなものを作りましょう。曲データのフォーマットをどうするかは悩ましいところですが、MML[1] を記述したプレーンテキストとすることにします。コンバートが不要なので、ポメラなどで編集してすぐ再生ということができるかもしれません。また、FlashAir の無線 LAN 通信機能を活用しない手はありません。ということで、ファイルブラウザや簡単な MML エディタを搭載し、演奏の制御を Web アプリから行えるようなものを目指します。

1　Music Macro Language。音階や音長などをテキストのコマンドで表現する形式。

ハードウェア

YMF825Board に面倒なところは全て入っているのでとても簡単です。ただ、3.3V で駆動することになりますので、YMF825Board の 3.3V 対応加工が必要になります（詳細はメーカーページを参照して下さい）。あとは本当に電源と SPI の信号を接続するだけです。FlashAir の各ピンと YMF825Board の信号の対応は表 1 を参照してください。

表 1: FlashAir と YMF825Board の信号の対応表

FlashAir	8	7	6	5	4	3	2	1	9
	DAT1	DAT0	VSS2	CLK	VDD	VSS1	CMD	DAT3	DAT2
YMF825Board	SS	SCK	-	-	-	-	MISO	(LED)	MISO

今回は配線のベースとして、秋月電子の SD カード配線引き出し基板（AE-SD）を使用しました。SD カードのピンが不足なく引き出されており、電源入力用の Micro-USB コネクタと 3.3V のレギュレータも搭載されているので、今回の用途にはピッタリなのですが、USB からの 5V がヘッダ側に出ていないというとても惜しい基板です。今回はこの基板の上に YMF825Board を両面テープで固定し、ピン間を配線する形をとりました。この様式なら 5V はコネクタから直接取ることができますし、コンパクトに仕上げることができます。写真と回路図を図 1, 2 に示します。余っている GPIO にテスト用 LED を付けていますが演奏機能には不要です。

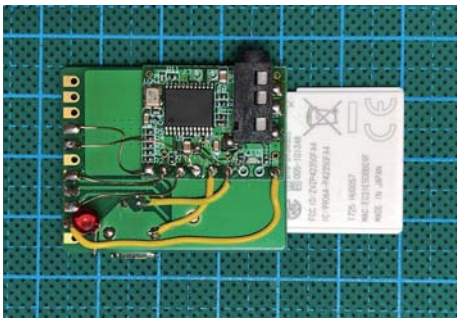


図 1: AE-SD と YMF825Board の接続写真

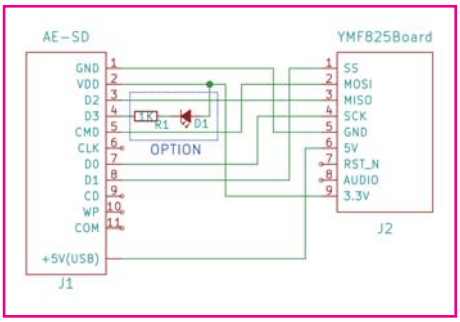


図 2: AE-SD と YMF825Board の接続回路図

FlashAir W-04

今回のプロジェクトでは W-04 を使用します。16 チャンネルの演奏データはそれなりの大きさになりますのでメモリの大きさは重要です。また、Lua でシステムの経過時間を返す関数である os.clock() を使用できること。音楽演奏にはタイマーが必須なのでこれが決め手になりました。

今回のプロジェクトでは SPI 通信と WebDAV からのファイル書き込みを使用しますので、/SD_WLAN/CONFIG に IFMODE=1 および UPLOAD=1 を設定しておきます。

MML プレイヤー

MML の仕様ですが、X68000 という PC で普及した MDX という音楽フォーマットの MML 仕様をそのまま採用することにしました。かつて慣れ親しんだフォーマットということもありますが、MDX から逆変換したデータを用いることで今すぐ様々な曲を演奏できるメリットがあります。とはいえ、同じ 4 オペレータの FM 音源でも YM2151 と YMF825 ではアルゴリズムの構成が異なり、音色パラメータにも互換性は全くないのですが、できるだけ雰囲気を合わせるような変換を行う機能も持たせます[2]。

では、このような MML を解析して演奏する Lua のプログラムを書いていきましょう。Lua を書くのは初めてで勘所がよくわからずに始めましたが、メモリの多い W-04 といえど、オブジェクト指向的にテーブルを多用して書くとあっという間に不安定になります。メモリというよりテーブルを気軽に使用しないほうが良いようです。音素などのコマンド毎にテーブルを作っていたのをやめて、チャンネルごとにコマンドバイナリ列の string を作成する構成に変更することで安定した演奏を行えるようになりました。

また、Web アプリからの演奏制御の為に、プレイヤーには共有メモリを使用した簡単なコマンドを実装しておきましょう。今回は演奏停止とボリューム設定コマンドを実装しました。

Web アプリ

FlashAir には Web サーバーが内蔵されていますので、HTML 等のファイルを入れておくだけで簡単に Web アプリを動かすことができます。今回は実装に React+Material-UI を使用してみました。Web フロントエンド開発も初めてでしたが、今時の開発環境らしく簡単にダイナミックなユーザーインターフェイスを持ったアプリが構築できました。

Lua のスクリプト実行は単純に GET リクエストを送るだけで実行できます。ファイルリストの取得と、演奏制御のための共有メモリへの書き込みには command.cgi を用います。また、内蔵する簡易 MML エディタから FlashAir へのテキストの保存には WebDAV 機能を用いています。このように、Lua, command.cgi, WebDAV といった、さまざまな手段を組み合わせて開発できるのが FlashAir の面白いところではないでしょうか。ただ、Lua のプログラムは同時に一つしか動かせないので、Lua でも Web インターフェイスでもできることは Web インターフェイスでやるほうが良いでしょう。

2　というより現時点では YM2151 からの変換ベースでの音色定義機能しか用意していません。

完成

音が鳴る機器の魅力を紙面で伝えるのは難しい[3]ですが、こんな感じのシステムが完成しました（図 3）。最近、専用基板を作成したので少しコンパクトになっています。

できることはリストから選んで曲を再生すること、MML を編集することだけの簡単なシステムですが、こんな小さな機器から往年のゲームサウンドが再生されるのはなんとも不思議で感慨深いものがあります。スマートフォン上の Web ブラウザはファイルの選択などの再生制御を行なっているだけで、実際に全ての演奏処理を行なっているのは FlashAir ですから、当然スマートフォンの電源を切っても再生は継続されます。健気ですね！

図 3: 専用基板での動作例

今後は Web アプリでのプレイリスト編集や、プレイリストに基づいたスタンドアロンでの自動再生機能などを盛り込んでみたいと思っています。

最後に

ソースコードと実行に必要なファイルを公開しています[4]。

また、開発中は GPS_NMEA_JP さんの FTLE に大変お世話になりました。これがなければとても開発できる気がしませんでした。この場をお借りしてお礼申し上げます。

3 YouTube に動作の様子をアップロードしました (https://youtu.be/DcW47RMJYJc)。

4 https://shuichitakano.github.io/ymf825playFA/

高野 修一 (@shuichi_takano)

普段から FM 音源や VDP など、レトロ PC の部品を動かして遊んでいます。高機能な小さい機器も大好きなので FlashAir には魅力を感じます。

FlashAir リズムマシン

せいみ まさみ

今回の FlashAir 同人誌 5 では FM 音源を使って遊んでみようと思います。FlashAir 同人誌 3 では PSG（Programmable Sound Generator）という音源 LSI と FlashAir W-03 のコンビで音楽を鳴らしてみましたが、いかんせん W-03 では外部モジュールへのアクセスが遅いのと、一定の間隔で処理を行うことができないため、1/60 秒ごとに PSG アクセスを行うような昔のデータをそのまま利用して音楽を演奏させるということは実はできませんでした。

しかし今回の自分には FlashAir W-04 があります。W-04 は基本的な動作速度が上がり、W-03 よりも高速に他のモジュールへのアクセスができるようになりました。更に隠し命令ですが、内部のタイマーを参照できるようになったため、一定の間隔で処理を行うことができるようになりました。今回は FlashAir 同人誌 3 の記事をバージョンアップさせて、ちょっと諦めていた演奏をさせてみようと思います。

FM 音源とは

図 1: YM2413（OPLL）

今回の記事で演奏を行う音源 LSI は FM 音源形式のものを使用します。FM 音源の詳しいことについては、Wikipedia などを見ていただければと思いますが、80 年台のパソコンやゲームセンターのゲームの音に使われていたり、2000 年台にはケータイ電話の着信音楽のために使われていました。イエロー・マジック・オーケストラの音楽も DX7 というヤマハの FM 音源シンセサイザーを使っていたりと、40 代から 50 代の人たちにはとても馴染み深い音源と言えるでしょう。

パソコンやゲームに使用されている 80 年台に生産された FM 音源は前回使用した PSG と同様に基本的にはパラレル通信でアクセスを行いますが、ケータイ電話に使用されているものはシリアル通信でアクセスが行えるものも生産されています。昨年から秋月電子さんなどで発売されている「ヤマハ製 YMF825 使用 FM 音源 LSI モジュール」は、SPI でのシリアルアクセスができる FM 音源です。

ということで、今回はこのモジュールを使って制作を行いました！…と言いたいところですが、あえて今回は、80 年台のパラレル通信の FM 音源を使用してみたいと思います。使用する FM 音源は、YM2413（OPLL）とよばれるもので、セガ・マスターシステム、MSX、UFO キャッチャーやパチンコなど非常に多くの製品で使用されたものです（図 1）。

OPLL は、9 音または、6 音 + リズム音 5 音（バスドラム、スネアドラム、タムタム、トップシンバル、ハイハット）を同時に発生させることができる FM 音源です。FM 音源は普通、音色（楽器と思って下さい）自分で作る必要がありますが、OPLL はそれが内蔵されています。また、回路的にも通常は別に必要な DA コンバータが内蔵されていたりと、OPLL はソフト・ハードともにお手軽に FM 音源を楽しむことができる音源 LSI となっています。

今回のテーマの一つである、処理時間を一定に行えるようにするというのは、音楽で言えばリズムを演奏させるのと同じようなものです。この点において、OPLL でリズムを演奏させるためのリズム音モードはとても便利です。今回はこのリズム音モードを用いて、FlashAir でリズムマシンを作ってみようと思います。

動作の仕組みと回路

動作については、FlashAir 同人誌 3 の「FlashAir で音楽を鳴らそう」の記事とほぼ同等です。詳しくはそちらのページを御覧ください。FlashAir Developers には過去の同人誌が PDF で置かれていますので、そちらで見てみて下さい。

さて、OPLL は PSG と同じ様に 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。ということで、FlashAir と OPLL の接続には、PSG の時と同様に I/O エキスパンダとよばれる I/O ポート拡張 IC を介して行うことにします。I/O エキスパンダは GPIO のパラレルデータ入出力を I2C や SPI などのシリアル通信に変換してくれます。

今回の製作でも 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見立てて、OPLL とアクセスをします。PSG の時とは GPIOA, GPIOB の使い方が逆転しているので、この点は注意が必要です。

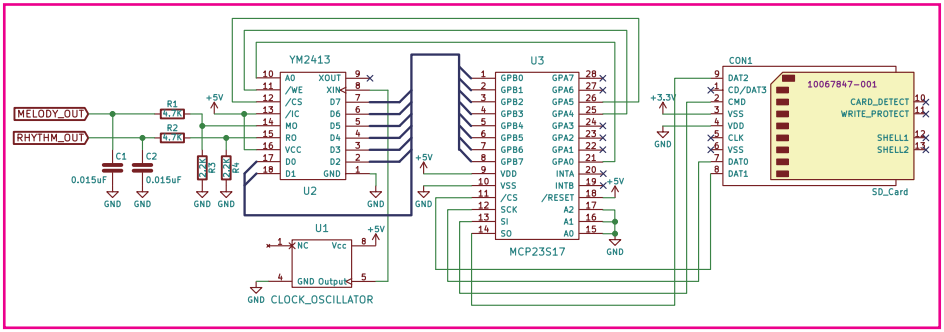


図 2: FlashAir OPLL 回路図

回路図は図 2 の様になります。こちらも PSG の時と基本的には変わりません。OPLL は音声出力後に特に積分回路とローパスフィルタ、アンプが必要になるのでお忘れなく。

プログラムのポイント

FlashAir W-04（FW バージョン v4.00.03）の Lua には FlashAir Developers では制約事項として使用できないと書かれている標準ライブラリの機能が使えたりします。この記事では特に os.clock() を使用しています。もし v4.00.03 以外の W-04 の場合では今回のプログラムは正常動作しませんのでソフトウェア更新を行って下さい。

os.clock() は、FlashAir へ電源投入後の時間の概算値を秒で返す関数です。精度はミリ秒単位で確認できるようです。Lua の処理の前後でこの関数を実行することで実際の処理にかかる時間を測ることができます。

```
start Time = os.clock()
originalFunction()
print( “Execution time(sec)” ..(os.clock()-start Time))
```

処理と処理の間の時間間隔を一定にしたいのであれば、処理の始めから終了の時間までの時間が分かれば、そこから必要な休む時間が分かります。これである意味マイコンに必須である一定時間で発生するタイマー割り込みのような使い方ができるのです。

一定間隔の時間は一度リズム音が発声してから次のリズム音が発声するまでの間隔となります。この間隔が一定ではないとリズムが狂ってしまいます。よく使われる BPM120 の曲での 16 ビートの音の間隔は 0.125 秒となります。今回のサンプルプログラムでは 0.125 秒毎にリズム音が鳴るようにしてみました。

サンプルプログラムでは必要な処理以外にランダムで無駄な処理時間が発生するようにしています。単純な sleep() ではリズムが狂ってしまいますが、os.clock() を使用することでそれを回避することができます。

最後に

2 年寝かせたネタがようやく日の目を見ることができました。この 2 年の間に同人誌 3 に書かれた「PSG は SPI でのアクセスに対応しています！」の様な FM 音源モジュールが発売されることがあるのかと大変びっくりしました。

FlashAir は W-04 になって更にマイコンとしての遊びができるようになりました。今後もマイコンとして遊んでいきたいと思います。

```
-- FlashAir & MCP23S17 & YM2413(OPLL) 一部抜粋
bpm = 120
intervalTime = 60/(bpm*4) -- 一番短い間隔（16 分音符の間隔）は bpm から計算する。
                          -- 0.125sec

function wait(startTime)  --intervalTime 以上になるまで経過時間を監視する
    Repeat
        Time = os.clock()-startTime
    until time>=intervalTime
end

playDat="33212121292121213131312129212121ff" -- 演奏データ
waitTime={}
dataIndex=1
counterCount = 0
loopCount=0

while loopCount<4 do           --4 回繰り返す
    start Time = os.clock()     -- 処理の最初の時間を記録
    data = tonumber("0x"..string.sub(playDat, dataIndex,dataIndex+1))
    if(data==0xff) then        -- 演奏データの最後まで演奏したら最初から再演奏
        dataIndex=1
        loopCount=loopcount+1
    else
        waitTime[counterCount] = math.random(50,80)    -- 無駄な待ち時間をあえて
        sleep(waitTime[counterCount])                  -- 50~80msec 作成する
        sound(0x0e, 0x20)
        sound(0x0e, data)
        dataIndex=dataIndex+2
        counterCount = counterCount +1

        wait(startTime)      -- 処理の最初から最後の時間が一定になるように
                             -- 確認処理を行う
                             -- 単純に sleep() を使ってしまうと、待ち時間を
                             -- 単純に加えただけになるので、一定間隔ではなくなる
    end
end
```

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。

FlashAir も良いけど、最近は第一子と遊んでいる時間が圧倒的に長い毎日。プログラム全部の内容が知りたいなどお問い合わせは twitter アカウントまでご連絡下さい。

Lua スクリプトと I2C バスでペンプロッタを動かす

いしかわきょーすけ

ボールペンを動かして紙に絵を描くペンプロッタを趣味で製作しています。Arduino を使ったコントローラボードを自作し、ペンプロッタをスタンドアローンで動かしたり USB 経由で PC から制御したりしています。

しかし今は 21 世紀。ペンプロッタをスマートフォンから操作してみたい、飲み会などでも今風デバイスと接続してスマートにデモしたいと思うようになりました。

無線接続に対応したコントローラボードを作るために ESP32 など無線 LAN/Bluetooth 内蔵マイコンボードを物色していた所、FlashAir W-04 で Lua スクリプトにて I2C バス制御関数が新たにサポートされたことを知りました。

更に FlashAir Doujinshi4 に掲載された、I2C バス接続のモータドライバで鉄道模型を動かす綾瀬ヒロさんの記事を拝見しました。しかも使用しているモータドライバは自分の使っているものと同じもの。これはやるしかないでしょう。

という訳で FlashAir W-04 の Lua スクリプトと I2C バスを使って自作ペンプロッタを動かしてみました。本記事は FlashAir をマイコンボードとして使用しメカトロ機器を動かす製作記です。

ペンプロッタ

まず動かす対象となる自作ペンプロッタについて説明します。

ペンプロッタは 3 軸の直動機構で構成されています。ボールペンを左右に動かし、紙を載せたテーブルを前後に動かすことで、紙の上の任意の XY 座標にボールペン先を移動させることができます。そしてボールペンを上下させることで紙の上に絵を描画します（図 1）。

ボールペンやテーブルは DC モータと長ネジを使った自作の直動機構で動かしています。ギヤ付 DC モータの軸に長ネジを取り付け、これを回転させることでネジを通したジュラコン製の四角スペーサを直線状に動かします（図 2）。

四角スペーサの動きにスライド抵抗のレバーを連動させており、スライド抵抗の電圧値を読みとることで四角スペーサの位置が分かります。テーブルやペンを四角スペーサに取り付け、四角スペーサが指定した位置に動くようにギヤ付 DC モータを正転／逆転させることでテーブルやペンを動かすことができます。

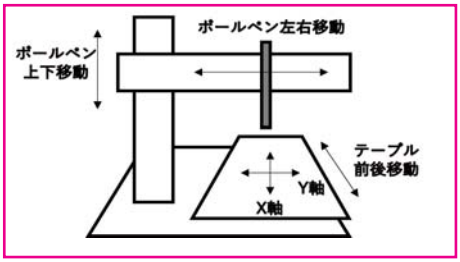


図 1: ペンプロッタ構造

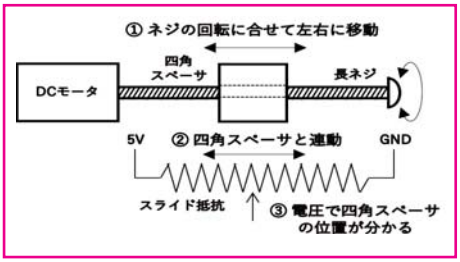


図 2: 自作直動機構

ハードウェア

続いてペンプロッタを動かすための回路について説明します。

FlashAir W-04 の他に必要なものは DC モータを動かすためのモータドライバとスライド抵抗の電圧を読み取る A/D コンバータです。両者とも I2C バスを使って FlashAir と接続します（図 3）。

モータドライバは綾瀬ヒロさんも使用されていた TI DRV8830 を使用します。モータ用電源の最少電圧が 3V と低電圧で動いてくれること、秋月でモジュールを 230 円で購入できる入手性の良さのため以前より愛用しているモータドライバです。

A/D コンバータは Adafruit の TI ADS1015 4ch 12bit A/D コンバータモジュールを使用します。Adafruit のモジュールは 1,000 円強と価格の面でやや敷居が高めなのですが、I2C バスに対応して 3ch 以上搭載している A/D コンバータの中では安価なのでこちらを使用します。Aliexpress ではもう少し安価な中華モジュールが入手できます。

製作時点では Lua スクリプトで I2C バスと GPIO が共存できなかったため、プッシュ SW などの入力装置は持たずにスマートフォンから Web 経由で操作することにします。

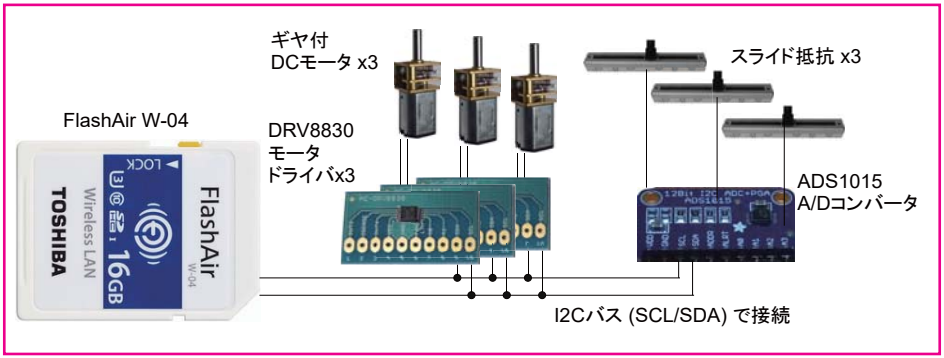


図 3: ハードウェア概要

ソフトウェア

ハードウェアに続いてソフトウェアについて説明します。

これまで Arduino でペンプロッタを動かしていた時には、配列変数の初期値として格納した画像の座標データを元に描画するか、USB 経由で PC から座標データとペン上下コマンドを受け取りペンの移動やペンの上下動を行っていました。今回はプロッタ用制御言語として一般的な HP-GL ファイルを FlashAir に格納し、これを Lua スクリプトで読みながら描画させることにします。

そして HP-GL ファイルと同じく FlashAir に格納した HTML ファイルをスマートフォンなどで表示し、ここから HP-GL ファイルを指定して Lua スクリプトを呼び出しペンプロッタを動かします（図 4）。

Lua スクリプトで実装したのは以下の 3 つの階層の処理です。

- **HP-GL ファイルのコマンドを PU,PD コマンドのみ解釈するイカサマ HP-GL パーザ**
- **PU, PD コマンドで指定された座標に応じて直線状に移動させる Line 描画処理**
- **DC モータと A/D コンバータによる位置制御**

ペンプロッタは既に Arduino で動いているため、単に HP-GL パーザを追加するだけでサクッと移植できるだろうと軽く考えていましたが甘い認識でした。Lua スクリプトは実数系変数の言語のため、これまで整数型変数に頼っていた暗黙の小数点以下切り下げが行われず少しデバッグに手間取りました。

そんな Lua スクリプトのデバッグも Segmentation Fault さんの FTLE のお蔭で何とか乗り切ることができました。スマートフォン上で Lua スクリプトが出力するメッセージや変数の値を表示することができるため、表示装置が無くモータドライバと A/D コンバータしか接続していない今回のハードウェアでも内部の挙動を確認しながらデバッグを進めることができました。Lua スクリプト開発では必須ツールだと痛感しました。

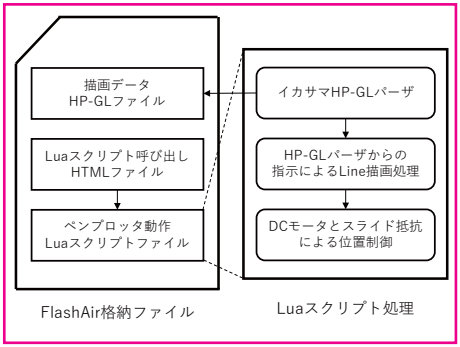


図 4: ソフトウェア概要

図 5: 動作の様子

FlashAir でペンプロッタを動かしてみて

これまで説明したハードウェアとソフトウェアで自作ペンプロッタを FlashAir の Lua スクリプトで動かすことができるようになりました。まだ挙動が読み切れていない所もありますが、FlashAir に格納した HP-GL ファイルに従って名刺用紙に絵を描画することができました（図 5）。

製作する前には気が付かなかった点について、いくつか記します。

まず FlashAir は SD カードの機能も持っているため、HP-GL ファイルを格納することに何の障壁も無かったのが印象的でした。どうやら自分は FlashAir を無線 LAN 内蔵マイコンボードとしか認識していなかったようです。

また Lua 言語やライブラリのファイルアクセスや文字列操作が充実しているため、HP-GL パーザを余り難儀せずサクッと実装できたのも大きかったです。

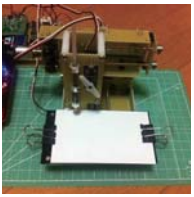
そして Web ブラウザから操作できることにより、物理的な入力装置の制約を受けないことのメリットは想定外でした。HTML ファイルでボタンやスライダーバーを追加することで、無限にスイッチや可変抵抗を増やせるのではないかという感覚を味わうことができました。

さいごに

FlashAir W-04 の Lua スクリプトと I2C バスを使い、Lua スクリプトのみで自作のペンプロッタを動かしました。FlashAir はモータドライバや A/D コンバータなどの低レイヤの制御とファイルシステムや Web ブラウザからの操作などの高レイヤの処理を両立できる 21 世紀のマイコンボードなんだと改めて感じました。

I2C バスを使用する場合でも FlashAir のピンにはまだ余裕があるので、次バージョンのファームウェアでは I2C と GPIO や UART 機能が両立されることを願っています。

本記事が FlashAir をマイコンボードとして使用するきっかけになれば幸いです。



いしかわきょーすけ（@qx5k_iskw）
趣味でマイコンや直動機構と戯れています。
「陰気な男でいいですか」という名の 20 世紀スタイルの
テキスト Web 日記を日々更新しています。
http://www.asahi-net.or.jp/~qx5k-iskw/darkside/

TOY Board と SensorTag でデータの見える化！

（株）マクニカ　矢野 均

みなさん、こんにちは！　マクニカ矢野です。FlashAir 同人誌 4 でご紹介させて頂いた TOY Board[1] が TI 社 IoT 向け開発キット SensorTag[2] と BLE（Bluetooth Low Energy）接続が可能となりました。

Maker Faire Tokyo 2018 ではマクニカブースで上記を用いてセンサデータを取得、ゲートウェイを介して CSV 化、そしてデータの見える化を行うワークショップを開催します。

ここでは FlashAir で簡単 IoT をキーワードとして SensorTag と TOY Board の接続についてご紹介させて頂きます！

TOY Board と TI 社 SensorTag との接続について

SensorTag は加速度や温湿度など 10 種類のセンサデータが取得できる IoT デモキットです。TOY Board と接続されるとセンサデータが CSV ファイルとして FlashAir に保存されます FlashAir IoT Hub を利用する事でデータの見える化を簡単に行う事が可能です。図 1 に TOY Board と SensorTag の接続例を示します。

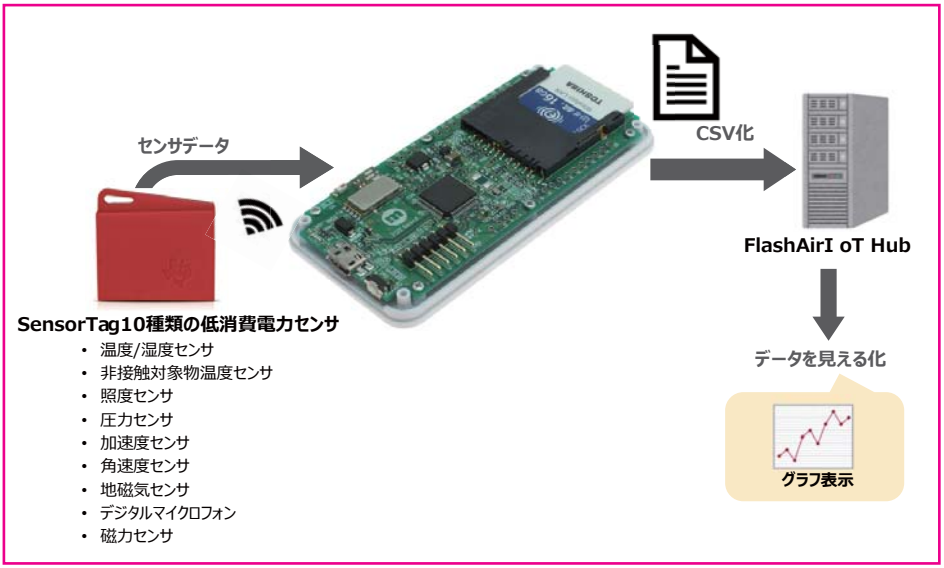


図 1: TOY Board と TI 社 SensorTag 接続例

1　FlashAir を使った小型な BLE- 無線 LAN ゲートウェイ基板
2　型番：CC2650STK

加速度は振動の測定用データとして応用されています。工場やオフィス、家庭に設置されている設備・機器に設置する事で振動や衝撃等を検出可能です。設備・機器の場合、通常動作時とは異なる振動特性が検出されると設備・機器に異常が発生している指標の一つとして活用されています。オフィスではエレベータに設置する事で稼働率や扉の開閉で人やモノの有無確認等が行えると考えます。FlashAir IoT Hub を活用し、閾値を設定することで自動的に通知される仕組み作りも可能です。

もちろん FlashAir は大容量の SD カードですので、インターネット接続ができない環境でも PC 上から CSV ファイルを加工する事で各センサデータをグラフで表示でき、問題点解決の為のツールとして応用が可能となります。

Maker Faire Tokyo 2018　ワークショップ開催！！

Maker Faire Tokyo 2017 ではオムロン環境センサ[3]と TOY Board を使用した環境データの見える化をワークショップ形式で行いました。今回の Maker Faire Tokyo 2018 では TOY Board と SensorTag を接続し、センサデータの見える化を行うワークショップを開催します。SensorTag は加速度データが取得できますので、ワークショップにあたり前述の様な使い方を想定し、実際に応用が出来ないか活用事例について意見交換を行いながら詳細を検討中です。今回も余熱さんにも全面的にご協力を頂きました！

おわりに

IoT ではセンサ設置からゲートウェイ経由で情報を集め、見える化を行い分析した結果を実行させる一連の流れを示されることが一般的です。センサ（SensorTag やオムロン環境センサ）とゲートウェイ（TOY Board）は BLE での接続が可能であり、FlashAir IoT Hub との接続や CSV がファイルの加工を行うことで見える化が可能で分析を行う事も出来ます。

誰でもまずは IoT を体験頂く事で様々な発想を持つきっかけになればと考えています。

3　型番：2JCIE-BL01　測定機能：温度 / 湿度 / 照度 / 気圧 / 音 /UV　通信方法：BLE

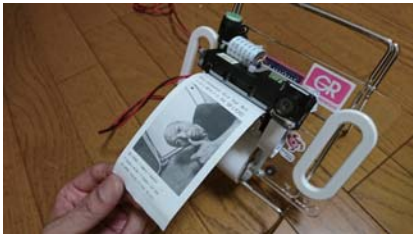
矢野 均（@kin0719）
半導体商社で営業を担当。
FlashAir 搭載可能な TOY Board を使って簡単に IoT の体験ができるように奮闘しています。FlashAir Developers Summit などでもご紹介させて頂きました。

画像ファイルも FlashAir IoT Hub へ

Takehiro Yamaguchi

FlashAir IoT Hub は計測値の可視化だけではないよ、というお話です。同人誌 4 に寄稿させて頂き、はや 1 年。この 1 年で製作したモノに「いぶし銀カメラ」というものがあります。簡単にご説明すると、カメラで撮影した画像をサーマルプリンタに出力することにより、誰でも、いぶし銀のように印字されるというコンセプトの自撮りカメラです。

[構成]
GR-LYCHEE（がじぇっとるねさす）
+ Camera（GR-LYCHEE 付属品）
+ OPEN CV3
+ AS-289R2 Thermal Printer Shield
+ FlashAir W-04
+ FlashAir IoT Hub



撮影した画像はマイコンボードで FlashAir に書き込みをしているのですが、画像ファイルの書き込みと同時に FlashAir IoT Hub にアップロードできる衝撃的な機能を皆様ご存知だったでしょうか？ 2018 年現在、LPWA や LTE などの無線モジュールで計測値をアップロードするのをメディアでよく見かけるのですが、FlashAir の無線 LAN 機能の利点を生かし、大きな画像ファイルのデータを FlashAir IoT Hub に初めてアップロードできた時、

「IoT Hub 半端ないって！」
「画像ファイルも扱えるなんて聞いてなかったもん！そんなん普通できひんやん！」
「メディアや全部メディアや」[1]

となってしまいました。更にアップロードされた画像ファイルは IFTTT との連携も可能ときたもんだから、夢は膨らむ、鼻息は荒くなる、大阪湾に向かって叫びたくなるといった感じです。この事実をとにかく伝えたくて、GR-LYCHEE ワークショップや mbed 祭りのデモでお話をさせて頂きました。この機会に改めて FlashAir 同人誌 5 号で訴えたいと思います。

そんな事をやっていたら 6 月の mbed 祭りで US や UK の Arm 社の方から THE ARM INNOVATOR ASIA TOUR[2] にご案内を頂き、そちらでいぶし銀カメラの LT をさせて頂きました。そこにはなんと Raspberry Pi 財団創設者の Eben Upton 氏もおられ、私の LT 中に自撮りをご体験頂き更にはお約束[3]の「Oh my God」を頂けたことは、生涯忘れられない記憶となりました。いぶし銀カメラは FlashAir を搭載したまま、東京でツアーと合流し海を渡っていきました。

1 「大迫半端ないって！」で検索！
2 Twitter ハッシュタグは #ArmInnovatorTour
3 Eben Upton の口癖

アップロードはマイコンボード側に負担無し

FlashAir IoT Hub に画像ファイルをアップロードする方法は、めっちゃ簡単です。公式ページの Lua スクリプトをダウンロードし、README に書いてあるとおりに設置するだけで、FlashAir に画像ファイルを書き込むと同時にアップロードされます。注意点があります。ダウンロードした upload_file.lua のスクリプトを見ると、ファイルの最終更新日時が最も新しいファイルを FlashAir IoT Hub にアップロードする仕組みになっているので、マイコンボードでファイルを書き込む際にタイムスタンプを付加しなければいけません。しかしマイコンボードにはバックアップ電池が付いてないため、毎回時刻を設定する必要があります。そこで電源投入時に FlashAir で NICT から現在日時を取得してしまおう、と考えました。NICT から日時を取得するライブラリ libFlashTime.lua が公開されていたので、これを利用させて頂きました。

仕組みはこうです。電源投入時に FlashAir で NICT から現在日時を取得し DateTime.txt を生成、そのファイルをマイコンボードに読み込ませて RTC に日時を伝えるというやり方にしました。

- **/SD_WLAN/CONFIG に LUA_RUN_SCRIPT=/nict.lua を追記**
- **/SD_WLAN/CONFIG に LUA_SD_EVENT=/upload_file.lua を追記**
- **libFlashTime.lua[4] を設置**
- **iothub.lua、upload_file.lua、libFlashTime.lua、nict.lua を設置**

nict.lua

```
local t = require "/libFlashTime"
while fa.WlanLink() == 0 do
      sleep(2000)
end
t.SetNICT();
local year, month, day, hour, min, sec = t.GetTime();
local file = io.open("/DateTime.txt", "w")
local datetime = string.format('%02d/%02d/%02d %02d:%02d:%02d', year, month,
day, hour, min, sec)
file:write(datetime .."¥r¥n")
file:close()
```

進化を続ける FlashAir IoT Hub を楽しみにしております。

4 FlashAir 開発者向け非公式 wiki（http://seesaawiki.jp/flashair-dev/）より

Takehiro Yamaguchi（@nada_tokki）
AS-289R2 サーマルプリンタシールドの開発者です。
最近、閃ソラさんの Twitter でまさかの個人アカウントが RT され、かなり恥ずかしい思いをしました w

プログラミング不要！ FlashAir のちょっと便利な使い方

高田

皆様の中には FlashAir Doujinshi の工作例は面白いけど自分で作るにはハードルが高いなぁ、と感じている方もいらっしゃるのではないでしょうか？　本稿ではプログラミングを知らなくても使える、FlashAir のちょっと便利な機能や使い方をご紹介したいと思います。

FlashAir アプリ（以降「アプリ」）で使える機能を中心に、後半は Developers で掲載している機能からご紹介します。

カメラ

無線 LAN「手動起動モード」で電池の消耗を減らす

FlashAir はカメラの電源を入れるとデフォルトで 5 分間無線 LAN が起動します（タイムアウト時間はアプリで変更可能）。そのためカメラの電池の消耗が普通の SD メモリカードに比べて早くなります。山登りや旅行で電池が心配！というときには手動起動モードにして使ってください。手動起動モードに設定すると、カメラで手動起動用の画像のプロテクトを OFF にした時だけ無線 LAN が起動するので電池の消耗を減らすことができます。

＜設定方法＞

FlashAir を挿入したカメラの電源を入れ、スマホと無線 LAN 接続をしてアプリを開きます。アプリの「設定」→「FlashAir 設定」→「無線 LAN 自動起動設定」を選択します。「自動で無線 LAN を起動する」を OFF にします。「完了」ボタンを押して設定を完了してください。これで手動起動モードになりました。

＜使い方＞

無線を起動する際は、FlashAir に保存されている無線 LAN 起動用の画像（/DCIM/100__TSB/FA000001.JPG）のプロテクトをカメラで OFF にします。すると無線 LAN 機能が起動します。画像のプロテクト状態の変化により無線 LAN 機能の起動／停止を行いますので、カメラの電源を入れた時点でプロテクトが OFF になっていた場合は、一度プロテクトを ON にして、再度 OFF にすることで、無線 LAN 機能を起動することができます。

SNS にスピーディーにアップロード

iPhone ではアプリの最新バージョン 4.2.0 で、撮影した写真をスピーディーに SNS にアップできるようになりました。これまでは FlashAir との無線 LAN 接続の切断後にモバイルデータ通信でアップロードされていましたが、新バージョンで FlashAir と無線 LAN 接続

したまま、モバイルデータ通信経由で写真を SNS にアップロードできるようになりました。写真を投稿する前にトリミングと回転の簡単な編集も行えます。SNS にアップする写真はスマホに保存されません。

なお、Android は仕様でこの機能が実現できませんので、iPhone と同様の手順で投稿メッセージの作成を完了したら、カメラの電源を切って FlashAir と無線 LAN 接続を切断してください。切断後にモバイルデータ通信経由で写真が SNS にアップロードされます。

IC レコーダー

音楽データも FlashAir アプリでダウンロード可能

アプリの「設定」で「音楽」を選択すると、音楽ファイルのリストが表示されます。スタジオで録音した演奏データをその場で演奏メンバーのスマホにシェアして持ち帰ることができます。帰りの電車や車の中で聞いてアレンジ曲を選択する、ということもできますね。SD カードスロットのついた大型の IC レコーダをお持ちでしたら、FlashAir で録音できるか動作確認をしてみてください。FlashAir と無線 LAN 接続中にノイズが入るケースがありますので、アプリの「設定」で自動起動のタイムアウト時間は最短の 1 分に設定し、無線が切れてから演奏を始めると良いでしょう。タイムアウト時間の変更方法は、次の「SD カードリーダ＋給電 AC アダプタ」を参照ください。

SD カードリーダ（GH-CRSDXC）＋給電 AC アダプタ

電池消耗を気にせず写真を共有

FlashAir を SD カードリーダに挿して AC アダプタで電源にさしておけば、FlashAir の中に入っている写真をカメラの電池消耗を気にすることなくシェアできます。旅行中の宿や、ダイビングツアー後のショップでの写真交換時に便利です。

この用途で利用可能な SD カードリーダはグリーンハウスの GH-CRSDXC[1] です（図 1）。

図 1: GH-CRSDXC

利用前に**無線のタイムアウト時間を「自動停止しない」に設定**しておくのがお勧めです。アプリの「設定」で「FlashAir 設定」を選択し、「無線 LAN 自動起動設定」で「自動で無線 LAN を起動する」が ON になっているのを確認し、自動起動のタイムアウト時間は「自動停止しない」を選択してください。

1　GH-CRSDXC を給電 AC アダプタに挿した時に、GH-CRSDXC の LED ランプが点灯しませんが、FlashAir の無線 LAN は起動します。この使い方は製造メーカが保証したものではありません。

カメラと同じように FlashAir アプリで写真のダウンロードや SNS へのアップができます。

どんなファイルも FlashAir アプリでダウンロード可能

FlashAir アプリの「設定」の「表示」で「フォルダー」を選択（図 2）すれば、どのような種類のファイルもファイルリストとして表示させてダウンロードすることができます。講演会等で配布したい PDF やエクセルデータを FlashAir に書き込んでおけば、CD-ROM や USB メモリを配る代わりに参加者がタブレットやスマホに資料をダウンロードすることができます。

図 2: 設定画面

ブラウザユーティリティで簡単にファイルを配布可能

参加者のパソコンに資料をダウンロードしてもらう場合や、1 回限りの講演会などで FlashAir アプリをダウンロードしてもらうことが現実的ではない場合には、ファイル配布用の専用のブラウザを FlashAir に設定することができます。来場者はパソコンやスマホから FlashAir に無線 LAN で接続し、Web ブラウザを開き、URL に http://flashair と入力することでファイルにアクセスしてダウンロードすることができます（図 3）。

図 3: 資料の配布

詳細は Developers の応用例・ビジネス利用の「ドキュメントの共有」[2] を参照ください。

デジタルフォトフレーム

スマホで撮影した写真やデコレーション写真を表示

スマホで撮った写真やデコレーションした写真をデジタルフォトフレームで表示させるには、スマホから写真をパソコンに取り込んで SD メモリカードに書き込む作業が必要で手間がかかります。FlashAir を使えば、デジタルフォトフレームに挿した FlashAir にスマホから無線 LAN 経由で直接データを書き込むことができます。

図 4: upload.cgi

図 5: アップロード成功

図 6: フォトフレーム

2 https://www.flashair-developers.com/ja/application/share-documents/

少し手が込みますが、HTML や Lua の知識は不要です。まずは FlashAir の CONFIG の設定変更をパソコンで行ってください。ここだけはパソコンでの作業が必要です。

1. **パソコンのテキストエディタ（メモ帳等）で FlashAir の SD_WLAN フォルダー内の CONFIG ファイルを開きます。SD_WLAN は隠しフォルダになっていますのでパソコンで隠しフォルダが表示されるように設定を変更してください。**
2. **CONFIG ファイルに UPLOAD=1 と APPAUTOTIME=0**[3] を書き込みます。CONFIG ファイルに記載の順番はないので、好きなところに 2 行にわけて書き込んでください。UPLOAD=1 は FlashAir への無線 LAN 経由での書き込みを有効にします。APPATUOTIME=0 はタイムアウト時間が無期限となります。
3. **FlashAir をパソコンから取り外し、デジタルフォトフレームに挿して電源を入れます。**これでスマホから FlashAir に写真を無線 LAN 経由で書き込めるようになりました。
4. **スマホの設定で FlashAir に無線 LAN で接続し、Web ブラウザを開きます。URL に「http://flashair/upload.cgi」と入力し「開く」または Enter を押します。図 4 の画面が表示されますので、アップロードする写真を選択し、Submit ボタンを押してください。アップロードが成功すると Success と表示されます（図 5）。**
5. **デジタルフォトフレームの電源を入れなおします。デジタルフォトフレームが FlashAir に新たに書き込まれた写真データを認識して表示します（図 6）。**

FlashAir 内の写真データを削除することもできます。指定方法は Developers の upload.cgi の「ファイルの削除」[4] を参照してください。

なお、デジタルフォトフレームは購入前に FlashAir の動作確認を行ってください。サンプル表示があればお店の人に頼んで、FlashAir 内の写真が表示されるか、無線 LAN が動作するかを確認しましょう。KEIAN KD71RV-W は動作しました。定格消費電力が 5W のフォトフレームは概ね動作するようですが、5W 未満のものや、5W 以上でも高機能のものは動作しない商品がありましたので注意してください。

3　本 CGI と SD メモリカードホスト機器から同時に変更を行うと FAT 不整合が起こる可能性があります。再度、カメラなど書き込み可能な機器で FlashAir を使う場合は設定ツールで初期化してください。

4　https://www.flashair-developers.com/ja/documents/api/uploadcgi/

高田（@M_Takada）
展示会やイベント、プロモーション等の拡販を担当。
バブル世代の山女。最近は皇居ランがお気に入り。

FlashAir 用の Windows アプリを組んでみよう！

品川 秀司

FlashAir って普段使いだとスマホやタブレットから専用アプリでアクセス、ホビーユースだと組込系のボード上でセンサーやカメラと組み合わせて使うケースが多いのかなって感じています。Windows からアクセスする場合は SD カードスロットに挿してエクスプローラーでファイル操作したり、ウェブブラウザからアクセスするのが普通ですよね。入出力インターフェースが豊富で色んな使い方ができる FlashAir ですが、私は「このデバイスはプログラムの学習にもってこいだ！」と感じました。そこで今回は FlashAir Developers さんで公開されている API を活用して Windows 向けに JPEG 画像ビューアを作成していきたいと思います。開発には Microsoft Visual Studio の C# 言語で Windows フォームアプリケーションを組む方法を採用したいと思います。今回作成する JPEG 画像ビューア（FlashAirViewer）には大きく分けて以下の 3 つの機能を実装します。

- **FlashAir 内の閲覧ディレクトリ（フォルダ）の移動**
- **ディレクトリ内の画像データをサムネイル形式で一覧表示**
- **選択した画像本体をダウンロードして全画面表示**

至ってシンプルな機能ではありますが、通信・UI イベント制御・画像処理などプログラム的には様々な要素を含んでいますので、Windows プログラミングの学習には最適だと思います。では、紙面の許す限り要点をかいつまんで説明していきたいと思います。

FlashAir の API を簡単に呼び出すには？

FlashAir には無線 LAN を使った HTTP 通信を用いてデータにアクセスする方法が、API（Application Programming Interface）として定義されています。今回は API を C# から簡単に呼び出す方法を紹介します。C# で HTTP 通信を行う場合、WebClient クラスを使用すると便利です。HTTP 通信の処理をメソッドとしてまとめてあるので、短いコード量で FlashAir の API 呼び出し処理が記述できます。今回は WebClient クラスの以下のメソッドを使って FlashAir の API を呼び出します。以下はファイルリストの取得 API を呼び出す時の記述例です。

```
string strResult =
DownloadString("http://flashair/command.cgi?op=100&DIR=/DCIM");
```

以下はサムネイルの取得 API を呼び出す時の記述例です。

```
Stream  stmResult  =
 OpenRead("http://flashair/thumbnail.cgi?/DCIM/100__TSB/DSC_100.JPG");
```

どうでしょう？　WebClient クラスを使用すると非常に簡単に記述できますよね。

WebClient クラスの TIMEOUT 時間を設定できるように拡張しよう！

便利な WebClient クラスですが、実は多少使いづらいところもあるのです。その一つがタイムアウトの時間が任意に設定できない事です。ここでは WebClient クラスにタイムアウト時間を指定できるプロパティを追加する方法を紹介します。まず WebClient クラスを継承した派生クラスを作成し、タイムアウトプロパティを定義します。そして GetWebRequest() メソッドをオーバーライドして指定したタイムアウト値を設定できるように書き換えます。以下は WebClient クラスを拡張した派生クラスの記述例です。

```
class WebClientEx : WebClient {
    private int timeout = 30000;

    public int TimeOut {
        get { return timeout; }
        set { timeout = value; }
    }

    protected override WebRequest GetWebRequest(Uri uri) {
        WebRequest wr = base.GetWebRequest(uri);
        wr.Timeout = timeout;
        return wr;
    }
}
```

時間のかかる処理は裏でコツコツやりましょう！

FlashAir の API の呼び出しが出来たところで、一つ問題が出てきました。FlashAir からサムネイルを取得して表示する際に FlashAir に保存されている画像ファイルの数が多いと、その数に比例して処理に時間がかかってしまいます。問題と言うのはサムネイルを取得している間、何かしらの対策をしないとビューアアプリの操作ができなくなる事です。そこでここではサムネイル取得のような時間のかかる処理を、メインの処理から切り離して別のスレッドで非同期的に実行する方法を紹介したいと思います。

Windows のフォームアプリケーションで非同期処理を実装する場合、BackgroundWorker コンポーネントを利用すると、簡単にバックグラウンドで処理を行うスレッドの記述ができます。BackgroundWorker コンポーネントは Visual Studio のツールボックスから選択して Form にドラッグする事で追加できます。以下はビューアアプリ上でディレクトリ移動操作を行った際の処理を BackgroundWorker コンポーネントを利用して記述した例です。

```
public partial class MainForm : Form {
    private void ChangeDirectory() {
        if (backgroundWorker1.IsBusy == true) {
            // BackgroundWorker で非同期処理の中断をする
            backgroundWorker1.CancelAsync();
            // バックグランド処理中間待つ
            while (backgroundWorker1.IsBusy == true) {
                System.Threading.Thread.Sleep(100);
                Application.DoEvents(); // ★★★
            }
        }
        // BackgroundWorker で非同期処理を開始する
        backgroundWorker1.RunWorkerAsync();
    }
    private void backgroundWorker1_DoWork(object sender,
DoWorkEventArgs e)  {
        BackgroundWorker worker = (BackgroundWorker)sender;
        foreach (サムネイルが個数分ループ) {
            if (worker.CancellationPending == false) {
                // サムネイルを取得
Stream stmResult = OpenRead(strFileName);
                // バックグラウンド処理の進捗をメインに通知する
                worker.ReportProgress(処理の進捗%);
            }
            else { // バックグラウンド処理がキャンセルされた時
                e.Cancel = true;
                return;
            }
        }
    }
}
```

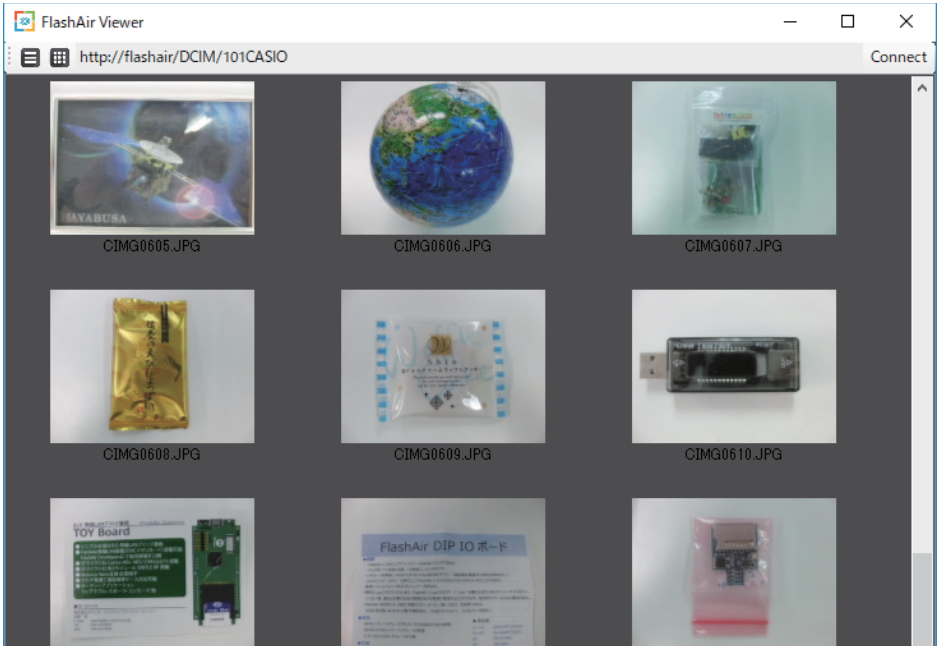
WinMain 関数のメッセージループって覚えてます？

BackgroundWorker で非同期処理の中断をする記述の中（★★★の部分）に Application.DoEvents() と言う見慣れないメソッドを呼び出しています。MSDN のドキュメントには「メッセージ キューに現在ある Windows メッセージをすべて処理します。」と書いてあります。これをみて「ははーん！」と思ったあなたは MS-DOS プロンプトから win コマンドをたたいて Windows を起動させていたバブル世代や団塊 Jr. 世代のエンジニアの方ではないでしょうか？（笑）話がそれてしまいましたが BackgroundWorker のスレッドは処理が完了（又は中断）した際にメインスレッドに処理の完了を通知します。そしてメインスレッドがこの通知を受け取ると IsBusy プロパティが false に遷移します。つまり DoEvents() を呼び出してメッセージ処理を行わないとバックグランド処理待ちのループから永遠に抜けられなくなるのです。怖いですね～。ちなみにメッセージループについて詳しく知りたい方はウィキペディアで「イベントループ」を調べてみるとよいと思います。

実は…

さて、ここまで長々とお付き合い頂きありがとうございました。ここで重大な事実を告白いたします！　実は WebClient クラスは .NET Framework 4.5 以降のバージョンから非同期処理をサポートしています。と言うわけで今回 BackgroundWorker を利用して実装した非同期処理は WebClient クラスの非同期処理用のメソッドを使用すれば、より簡潔にコードを記述する事ができると思われます。えっ?何でそれを先に言わないのかって？　そ、それは…僕にもその色々と大人の事情がありましてですね…すみません。と言うわけでこれに懲りずに是非また御付き合い下さいね！

紙面の今回作成したプログラムの実行に必要なセットアップファイルや、ソースコードは折を見て以下のブログで公開できればと考えております。



品川 秀司
横浜で小さなソフト屋さんを営んでおります。
ブログ http://www.kara-kuri.jp/blog/?p=1337

FlashAir W-03 と W-04 の消費電力と電波強度

（株）タイプ・アール　高瀬 秀樹

FlashAir W-04 が発売になり、SD インターフェースの速度比較（図 1）や、無線 LAN の速度比較（図 2）などが東芝メモリから公開されております。

ところが、消費電力や無線 LAN の電波強度の比較が見当たりませんでしたので、計測結果を 2018 年 2 月に開催された FlashAir デベロッパーサミット 2018 で発表させていただきました。今回は、そのデータをお伝えしましょう！

種類	無線LAN搭載 SDメモリカード	
	FlashAir™ SD-WEシリーズ <W-03>	FlashAir™ SD-UWAシリーズ <W-04>
最大読み出し速度	–	90MB/s
最大書き込み速度	–	70MB/s
UHSインターフェース	–	UHS-I
UHSスピードクラス	–	U3
SDスピードクラス	CLASS10	CLASS10

図 1: FlashAir W-03 と W-04 の SD I/F 速度比較

図 2: FlashAir W-03 と W-04 の無線 LAN 速度比較

FlashAir の消費電力

USB 接続の電圧・電流計を用いて、FlashAir の消費電力を USB-SD 変換器経由で計測しました。USB-SD 変換機の正確な消費電力が判りませんので、正確な値ではありません。ざっくりした値です。

	FlashAir W-03	FlashAir W-04	差
基本動作時	30mA	20mA	10mA
無線 LAN 動作時	90mA	80mA	10mA
合計	120mA	100mA	

無線 LAN の電波強度

IEEE 802.11b/g/n の 3 種類が実装されていますので、それぞれで試しました。各 3 回計測した平均値ですが、測定環境が違うので参考値として下さい。

FlashAir 近接で測定

無線 LAN 規格	FlashAir W-03	FlashAir W-04	差
802.11b	27dBm	15dBm	12dBm
802.11g	28dBm	15dBm	13dBm
802.11n	28dBm	15dBm	12dBm

スチール製防音ドア越しで距離 2m 程度で測定

無線 LAN 規格	FlashAir W-03	FlashAir W-04	差
802.11b	85dBm	73dBm	12dBm
802.11g	86dBm	74dBm	12dBm
802.11n	90dBm	73dBm	17dBm

近接・スチール製防音ドア越しの両テストとも、W-03 と W-04 の差はほぼ 12dbm となり、結構な差があることがわかりました。電波強度の単位 dBm は、電力をデシベル（dB）の値で表した単位で、1mW の電力を基準値（0dBm）とします。無線 LAN の周波数が 2.4GH z なので、多少の誤差は有りますが、簡単に書くと距離にして約 4 倍飛ぶと考えられます。

デジカメに FlashAir 入れて Wi-Fi ルータ経由で運用する場合を想定すると、これまたざっくり実用距離が、W-03 は 5m 程度で不安定になるが、W-04 は 10m 以上で安定して使用可能と言う感じだと考えられます。

と言う事で、W-04 は、W-03 に比べて省エネなのに無線 LAN は強力になって、CPU の処理速度も上がって、ファイル転送速度も向上して大幅な進化を遂げていることがわかりました！！

（株）タイプ・アール　高瀬 秀樹（@hide_tks）

小学生時代からの電子工作好き、長年放送局関係の仕事を手掛け、現在は IoT ALGYAN の運営委員も務め、Microsoft Azure と FlashAir や、Arudino・RaspberryPi をつないで IoT でビジネスしようと格闘中。カレーとハンバーガーが好物で、スキー・AV・アマチュア無線が趣味。

FlashAir にシリアル通信機能が付きました

GPS_NMEA_JP（@Seg_faul）

FlashAir W-04 に、ファームウェアバージョン v4.00.01 からシリアル通信機能が付き、v4.00.03 のアップデートでボーレートの変更ができるようになりました。シリアル通信が利用できるということは、液晶画面や PLC、各種マイコンや装置との通信手段として利用できることとなり、FlashAir の活用範囲がまた一段と広がります。

本章では、この機能の使い方について独自に調査した結果を説明させていただきます。

使い方

ハードウェア

DAT0（PIN 7）が Tx 端子、CMD（PIN 2）が Rx 端子となります。High=3.3V、Low=0V で、負論理の一般的な UART として利用できます。（I2C 機能と同時に利用すると、ピンが変化し、PIO 1 ピン、I2C 2 ピン、UART 2 ピンで利用できるようです。詳しくは 蒼 さや（@La_zlo）さんの記事をご覧ください。）

5V 系の回路と接続するには、レベルコンバータが必要です。私は、秋月電子通商の「超小型ＵＳＢシリアル変換モジュール（FT234X）」で動作確認を行いました。

初期化

以下のように初期化します。一度初期化を行うと、再起動するまで有効なようです。

ボーレートを省略すると 115200bps になります。

（v4.00.01 および v4.00.02 では 115200bps 固定です。）

```
fa.serial("init")          -- 省略した場合
fa.serial("init",9600)     -- ボーレート指定
```

この際、D0 が出力ピンに、CMD が入力ピンに設定されます。

ビット幅 8bit、ストップビット 1、ノンパリティで動作しているようです。

なお、動作確認できたボーレートは以下です。

```
300 600 1200 2400 4800 9600 12000 14400 19200
31250 38400 57600 115200 230400 460800 921600
```

実験した限り、このボーレートは、fa.spi 同様、1bps 単位で設定可能なように見えます。

開放

未初期化状態に戻します。おそらく受信処理が停止するのではないかと思います。

```
fa.serial("deinit")
```

送信

送信します。文字列、数値、テーブル（配列）が利用できます。

特にサイズ制限はないようです。

```
fa.serial("write","Hello World")
fa.serial("write","Hello World",11)
fa.serial("write",0xAA)

dat = {0x0D,0x0A}
fa.serial("write",dat)
```

受信

受信します。初期化後、自動で受信処理が動いているようで、14Byte ほどバッファリングされています。それを超えると取りこぼすようです。

1byte 単位、数値としての入力になります。（連結すれば文字列になります。）

バッファにデータがない場合は nil を返します。

```
x = fa.serial("read")
```

サンプル

シリアル通信機能を用いた送受信のサンプルを以下に示します。

ここでは、“Hello World” と送信した後、受信バッファに入っている文字列を取得してブラウザに表示します。

```
fa.serial("init")
print(fa.serial("write","Hello World"))
s=""
for i=0,16 do
   x = fa.serial("read")
   if(x ~= nil)then
      s = s .. string.format("%c",x)
   end
end
print(s)
```

おわりに

FlashAir でシリアル通信機能が使えるようになり、デバッグ手段や、既存の機器との連携の幅がまた一段と広くなりました。

私も、このシリアル通信機能を使って、Arduino のシリアルモニタのような機能をブラウザで実現した “FlashTools Serial Terminal v0.01” や、FlashAir 上で Unix 風の拡張可能なシェルのような何かを実装した “FlashTools Minimal Shell” などを作ってみました。

気になる方は、私のブログで「serial」と検索してみてください。

電子工作記録 http://gpsnmeajp.sblo.jp/

FlashTools Lua Editor v1.07b のご紹介

GPS_NMEA_JP（@Seg_faul）

新しくなった FlashTools Lua Editor（以下 FTLE）をご紹介します。前回の 1.03 から、1.07b となり、FlashAir の進化に合わせて様々な機能が追加されています。

初めて知る方はもちろん、昔からお使いの方もこの機会に新しいバージョンに更新してみてください。

はじめての方へ：FlashTools Lua Editor ってなに？

FlashTools Lua Editor は、ブラウザ上で動く FlashAir 用の簡易的な開発環境です。一旦 zip ファイルを解凍し、FlashAir に上書きコピーして導入を済ませれば、以後 SD カードスロットに抜き差しすることなく、無線 LAN 経由で開発の殆どが済みます。

FlashAir の開発ではエラーの特定が難しくなりがちですが、エラーの発生箇所や内容をわかり易く表示。また修正・編集して、その場で run ボタンを押せば実行できます。普段は PC で、出先や測定の際にはスマートフォンやタブレットからも編集できる、そんなコンセプトで作成されています（図 1）。

なお、詳しくは以前ご紹介させていただきました FlashAir 同人誌 4 をご覧ください。

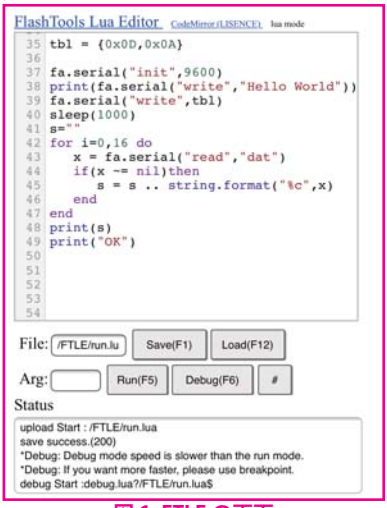


図 1: FTLE の画面

新機能のご紹介

1.03 から 1.07b にかけて進化した点をご紹介します。

簡易ファイラー機能（図 2）

「出先でスマホだけで開発する時にファイル管理もできると嬉しい」というお話を受け、FlashAir のファイル管理機能をフル活用し、ブラウザから指定ファイルの編集・削除・移動と名前変更ができるようになりました。

ファイルのアップロードやフォルダの新規作成、指定フォルダ内の全ファイル削除機能（Lua で大量のファイルを生成したときに便利）もあります。

図 2: 新機能の簡易ファイラー

ちなみに、通常時は Lua を使用せず FlashAir の CGI 機能を使ってファイル操作ができます（実行中の Lua スクリプトと干渉しません）が、Full File List にチェックを入れて Reload ボタンを押すと、隠しファイルを含めた全てのファイルを表示・操作できるようになります（図 3）。

また、このファイルリストは FlashAir の標準メニューを置き換えてしまいますが、

FTLE のメニューより、Change File List を選択すると標準のものに切り替えることができます（図 4）。（同じ操作で戻すこともできます。）

隠し機能として、画面上部の現在パスのところがリンクになっており、ここをクリックしてブックマーク登録すると、そのディレクトリをスタート地点として開くことができる機能もついています。

図 3: 隠しファイル表示モード

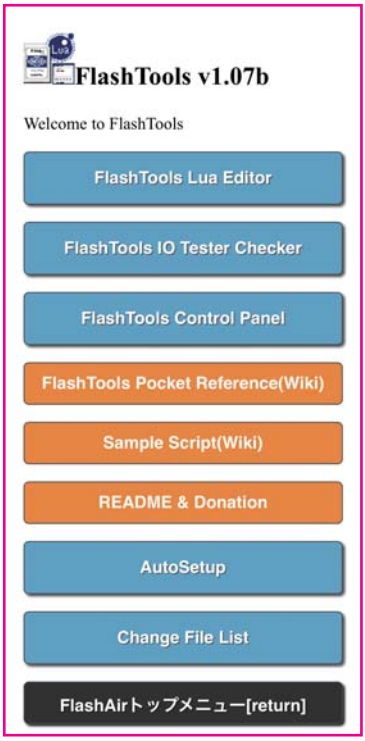


図 4: FTLE メニュー

Lazy mode 対応

ファイラー機能搭載に合わせ、使用しているエディタのライブラリである CodeMirror の Lazy mode に対応しました。

これにより、CodeMirror の対応するファイルは自動判別され、対応するシンタックスハイライトが適用されるようになりました。自動判別の結果は、ライセンス表示の横に小さく表示されます（図 5）。

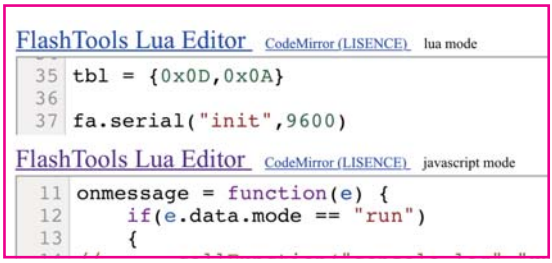


図 5: Lazy mode

出力の高速化

v4.00.01 のアップデート以降、一部のブラウザ等で、Lua の print の結果が読み込まれるのにかなりの時間がかかるようになりました。その対処として、FTLE では読み込みの際バッファリングを行うようになりました。

以前と同程度の速度で読み込みができるようになっていますが、その一方、処理の都合上、W-03 はサポート外となりました。（一応、フォールバックするようにはしてあります）

デバッグブレーク機能（v4.00.00 および v4.00.03 でのみ動作）

FTLE は FlashAir の通信機能に依存している都合上、うっかり無限ループのスクリプトを書いてしまうと、処理が戻ってこなくなり電源を再投入するしかなくなります。

そんなときのために、Debug 機能が更新されました。eva.cgi 取得機能により無意味になったエラー取得機能の代わりに、debug.sethook と共有メモリを用いたブレーク機能を搭載。Debug で実行中は Break ボタンを押すことで、ループ等の実行を止めることができます。（ただし Debug で実行中は実行速度が低下します）。

Run モードの際も

```
require "/FTLE/breakpoint"
```

としてブレークポイント機能を有効化し、

```
breakpoint()
```

をループ中に配置すれば、Break ボタンを押した際そこでブレークすることができます。

なお、v4.00.01 および v4.00.02 では debug 機能が無効化されているため利用できません。v4.00.03 では利用できますので、アップデートをお願いします。この機能は、共有メモリの先頭（0x0000）の 1byte を専有します。

デバッグメッセージ取得機能

通常、print 文の出力結果は、スクリプトの実行が終わってから見ることになります。そのため、長いループや、無限ループなスクリプトが今どんな感じで実行されているのかを確認するのは難しいものです。

FTLE では、breakpoint ライブラリに putMessage という機能を搭載しました。これは、共有メモリを用いて、スクリプトから書き出されたメッセージを FTLE で読み取ることができる仕組みです。

```
require "/FTLE/breakpoint"
```

としてブレークポイント機能を有効化し、

```
putMessage("Hello World")
```

といったように書くと、利用できます。

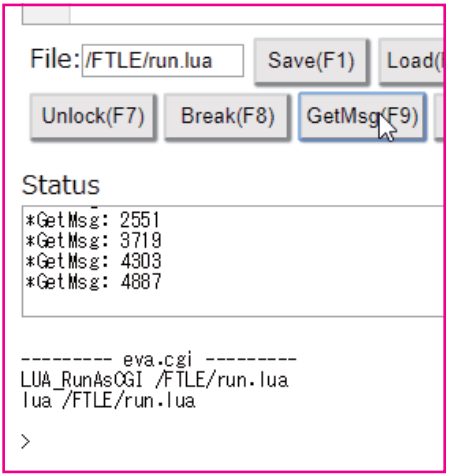


図 6: GetMsg 機能

図 6 のように、GetMsg(F9) ボタンを押すたびに現在入力されているメッセージの中身を取り出すことができます。（負荷対策のため自動では行われません）

なお、この機能は、共有メモリを 254byte 専有します。（0x0001-0x00FF）

応答時間測定機能

スクリプトを書いていると、「あれ、このスクリプトってどのくらいの時間がかかってるんだろう」と思うことありますよね。そんなときのために実行時間を測定する機能が付きました。自動で測定され、実行結果画面に表示されます（図 7）。

更新通知機能

インターネットに接続しながら FTLE のメニューを起動すると、JSONP により最新バージョンに関する案内が表示されます（図 8）。

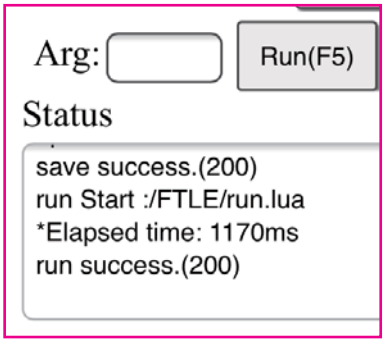


図 7: 実行時間測定

図 8: バージョン通知

配布について

FlashTools Lua Editor は、筆者サイトにて配布しています。

トップページにリンクがありますので、そちらを辿ってください。

おまけ程度のツール置き場 https://sites.google.com/site/gpsnmeajp/

サイト移行中のため、もし上記サイトにアクセスできなくなっている場合は以下のサイトへアクセスをお願いします。

Seg's Homepage https://sabowl.sakura.ne.jp/gpsnmeajp/

FlashAir に関する技術情報・FTLE の導入手順などは以下の Wiki にて公開しています。こちらもぜひよろしくお願い致します。

（FTLE の導入手順は Lua スクリプトサンプル集→ Lua: 開発の前に ... です）

FlashAir 開発者向け非公式 wiki http://seesaawiki.jp/flashair-dev/

GPS_NMEA_JP（@Seg_faul）

自サイトにてツール作ったり、非公式 wiki の管理をしていたりする人。最近はめっきり VR にハマりっぱなし。

FlashAir 活用テクニック

蒼 さや

FlashAir に部品をつないで IoT 電子工作を楽しんでみませんか？

FlashAir W-04 の最新ファームウェア（FW）、バージョン v4.00.03 を独自に調査[1]した結果を紹介します。内容は無保証ですが（お約束）、モノづくりの参考になれば幸いです。

端子と Lua 関数の関係

W-04 の汎用端子は 5 本です。最新の FW で何がつながるのか、どの機能が同時に使えるのか気になります。そこで Lua 関数と端子機能の関係を調べてみました（表1）。

濃い黄色のセルは、v4.00.03 で追加またはバグ修正された部分です。

表 1: 端子機能表

ピン番号	SD 規格		FlashAir オリジナル							備考
	SD インタフェース	電源投入直後	fa.pio	fa.spi		fa.i2c		fa.serial *5		
			GPIO	SPI	"init"	I2C	"init"	UART	"init"	端子処理
5	CLK	I	RSV	RSV	I	RSV	I	RSV	I	pull-down が望ましい
2	CMD	I	0x01	DO	O(L)	SCL	I	RX	I	
7	DAT0	I	0x02	CLK	O(*1)	SDA	I	TX	O(H)	
8	DAT1	I	0x04	CS	O(H)	PIO	*4	PIO	*6 *4	
9	DAT2	I	0x08	DI *2	I	TX *5	O(H)	RSV	*6	
1	CD/DAT3	I(pull-up)	0x10	RSV	*3	RX *5	I	RSV	*6	pull-down は避ける
4	VCC									2.7 ～ 3.6V
3	VSS1									GND
6	VSS2									GND

*1: "mode" が 0 または 1 のとき "O(L)"、"mode" が 2 または 3 のとき "O(H)"

*2: I/O が "O" の時、"write" や "read" で "L" を出力

*3: fa.pio の設定を反映、事前に fa.pio を実行してない場合は "O(H)"

*4: "setpio" の設定を反映、事前に "setpio" を実行してない場合は "O(H)"

*5: 動作しているが未公開の関数（2018/7/1 現在）

*6: fa.pio や fa.spi を実行後の初期値は "I", 事前に実行してない場合は "O(H)"

1　参考資料

- 簡易版 SD 規格仕様書 (https://www.sdcard.org/downloads/pls/index.html)
- Lua 関数リファレンス (https://www.flashair-developers.com/ja/documents/api/lua/)
- FlashAir の同人誌 (https://www.flashair-developers.com/ja/documents/books/)
- FlashAir W-04 v4.00.03 の挙動解析

端子機能が非公開かつ不明なものは RSV（リザーブ）としています。fa.pio 以外の関数では "init" を実行すると端子の入出力状態が初期化されます。そこで端子の初期状態も併せて記載しました（I: 入力、O(H): H レベル出力、O(L):L レベル出力）。

fa.serial は未公開（2018/7/1 現在）のシリアル通信機能です。基本的な使い方は本誌の GPS_NMEA_JP さんの記事をご参照ください。それ以外の Lua 関数の使い方は FlashAir Developers さんの「Lua 関数リファレンス」をご参照ください。

電源投入直後

電源電圧は SD 規格通り 2.7 ～ 3.6V に対応しています。DAT3 端子は電源投入時に IC 内部で Pull-up されます。端子機能を切り替えるとオフされますが、外部に Pull-down 抵抗をつけると誤動作の原因となるので避けたほうがよいでしょう。また、CLK 端子を SD インタフェースで使わないときは Pull-down 抵抗をつけましょう。

fa.spi と fa.pio の共存

fa.spi で "init" を実行した後の端子の入出力状態は、fa.pio で再設定することができます。fa.spi に戻す際に "init" は不要です。これを利用すると、3 線式の SPI 部品だけでなく、例えば 4 線式 SPI の TFT モジュールをつなぐことができます。また、SPI 機能で使わない端子にスイッチをつけて fa.pio でリードしたり、LED を制御したりすることができます。

ILI9225 2.2 インチ TFT モジュールの接続例

ILI9225（図 1）を使った TFT モジュールの接続例を紹介します。4 線式 SPI に対応したこのモジュールは、コマンドでは RS 端子を "0" に、データでは "1" に頻繁に切り替える必要があります。一方、CS や RST はこまめに切り替える必要はありません。そこで表 2 のように接続すれば、FlashAir から効率よく制御することができます（VCC は 3.3V）。

表 2: LCD との接続

ピン番号	SD 規格	FlashAir オリジナル			部品
	SD インタフェース	fa.pio	fa.spi		TFT LCD
		GPIO	SPI	"init"	ILI9225
5	CLK	RSV	RSV	I	-
2	CMD	0x01	DO	O(L)	SDA
7	DAT0	0x02	CLK	O	CLK
8	DAT1	0x04	CS	O(H)	RS
9	DAT2	0x08	DI	I	CS
1	CD/DAT3	0x10	RSV		RST
4	VCC				VCC, LED
3	VSS1				GND
6	VSS2				-

図 1: ILI9225 TFT モジュール
2.2 インチ 176x220 TFT LCD, 4wire SPI

LUA プログラム例

```
-- 初期化関数
function ILI9225_init(mode)
fa.spi("mode", 0)
  fa.spi("init", 1)
  fa.spi("bit", 8)
  fa.pio(0x1F, 0x10) -- RST(DAT3)=1, CS(DAT2)=0, RS(DAT1)=0
  sleep(1)
  fa.pio(0x1F, 0x00) -- RST(DAT3)=0, CS(DAT2)=0, RS(DAT1)=0
  sleep(10)
  fa.pio(0x1F, 0x18) -- RST(DAT3)=1, CS(DAT2)=1, RS(DAT1)=0
  sleep(50)
end
```

```
-- データライト関数
function ILI9225_writeWord(cmd, data)
  fa.spi("cs",0)          -- RS(DAT1)=0
  fa.spi("write", cmd)    -- CS(DAT2)=0
  fa.spi("cs",1)          -- RS(DAT1)=1
  fa.spi("bit", 16)
  fa.spi("write", data)   -- CS(DAT2)=0
  fa.spi("bit", 8)
end
```

fa.i2c と fa.serial の共存

fa.i2c で "init" を実行すると、CMD,DAT0 端子で I2C 機能が使用できます[2]。また、v4.00.03 では DAT1 端子を GPIO として使えるようになりました。更に未公開ですが、DAT2, DAT3 を使ってシリアル通信が可能なことがわかりました。

初期設定手順を以下に示します。この順番で実行すると全機能が有効になります。

```
-- 端子機能の初期設定
fa.pio(0x00,0x00)
fa.i2c {mode ="setpio", data=0, ctrl=0}
fa.serial("init",9600)
fa.i2c {mode ="init", freq=400}
```

"setpio" での設定値やシリアル通信及び I2C の周波数は必要に応じて変えてください。

初期設定後は fa.i2c 関数で I2C 機能や GPIO を、fa.serial 関数でシリアル通信機能を使用できます。シリアル通信中に実行した I2C 機能は並列に実行されるようです。

2　動作メモ：freq="400" では 400kHz になりませんでした。freq=400 と書くとうまくいくようです。

fa.serial と "setpio","getpio" の共存

fa.serial の "init" を実行した後に fa.i2c の "setpio" や "getpio" を実行すると、シリアル通信機能を維持したまま DAT1 端子を GPIO として使えることがわかりました。シリアル通信中に実行した GPIO 機能は並列に実行されるようです。プログラム例を示します。

```
--DAT1 端子状態のリード
fa.serial("init",9600)
fa.i2c {mode ="setpio", data=0, ctrl=0}
res, data, ctrl = fa.i2c {mode="getpio"}
print(res, data, ctrl)
```

Lua 関数の遷移図

端子機能に関わる Lua 関数の遷移方法を図にしました（図 2）。

端子機能の切り替えには、外部からの SD カードアクセスがないことの他に、/SD_WLAN/CONFIG ファイルに IFMODE=1 が記載されていることが必要です。

なお複雑になるので fa.serial 関数は省略しています。fa.serial 関数については本文を参照ください。

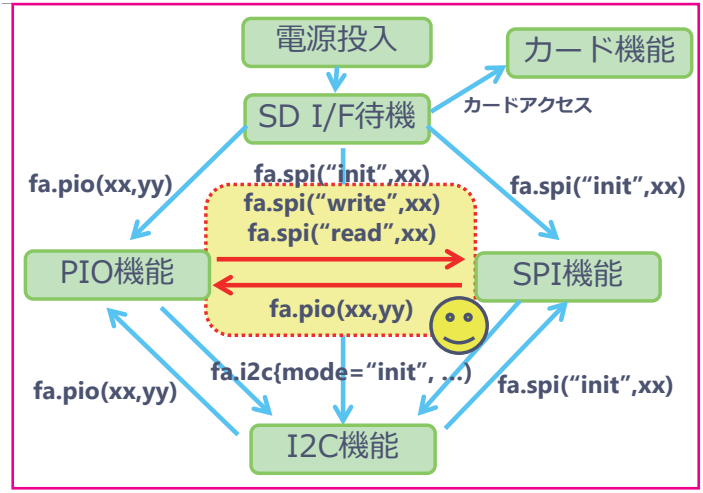


図 2: Lua 関数の遷移図

おわりに

fa.i2c と fa.serial の共存は熱いです。I2C 接続の部品や RS-232C 接続のサーマルプリンタをつないで、グラフ印刷したりネット連動したりする IoT 端末をつくってみたいです。

本記事の作成にあたり、FTLE や非公式 wiki に大変お世話になりました。便利な環境を整えてくださっている GPS_NMEA_JP（@Seg_faul）さんに深く感謝申し上げます。

FlashAir 活用テクニック －実践編－

蒼 さや

端子共有の実践編として「そらまめ 18 号」を紹介します。FlashAir W-04 の国内正規版を買うとついてくる SD カードケースで作った、光って奏でるガジェットです。

そらまめ 18 号の概略

そらまめ 18 号の外観を図1に、概略配線図を図 2 に、配線プランを表1に示します。

図 1: 外観

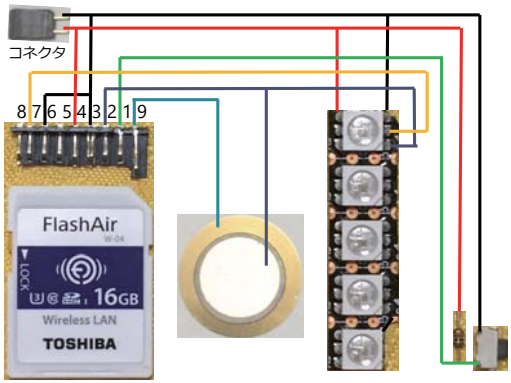


図 2: 概略配線図

表 1: 配線プラン

ピン番号	SD 規格	FlashAir オリジナル			18 号の想定 IO 設定	部品		
	SD インタフェース	fa.pio	fa.spi			LED strip	tact sw	圧電ｻｳﾝﾀﾞ
		GPIO	SPI	"init"		APA102C	TS-001	20mm Φ
5	CLK	RSV	RSV	I	-	-	-	-
2	CMD	0x01	DO	O(L)	O(L)	DI	-	◎
7	DAT0	0x02	CLK	O(*1)	O(L) or I	CI	-	-
8	DAT1	0x04	CS	O(H)	O(H)	-	-	-
9	DAT2	0x08	DI *2	I	I	-	-	○
1	CD/DAT3	0x10	RSV *3		O(L) or I	-	○	-
4	VCC				-	VCC	10k Ω	-
3	VSS1				-	GND	○	-
6	VSS2				-	-	-	-

*1: "mode" が 0 または 1 のとき "O(L)"、"mode" が 2 または 3 のとき "O(H)"
*2: I/O が "O" の時、"write" や "read" で "L" を出力
*3: fa.pio の設定を反映、事前に fa.pio を実行してない場合は "O(H)"

LUA スクリプト

LED strip を光らせる

APA102 が搭載された LED strip を 5 個分の長さに加工して使用しました。この LED strip は SPI 制御可能なので FlashAir に直結できます（図 3）。

```
s={}; s[1] = 0x00000000
for i=1, 5 do s[i+1] = <輝度+BGR>; end
s[7] = 0xFFFFFFFF
fa.spi("mode",0)
fa.spi("bit",32)
fa.spi("init",1)
fa.pio(0x07,0x04)
fa.spi("write",s)
```

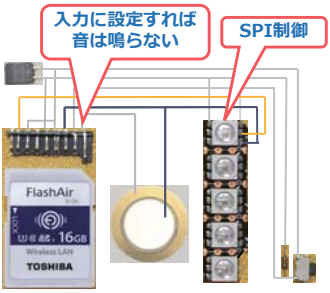


図 3: LED strip を光らせるには

ブザーを鳴らす

圧電サウンダを CMD 端子で駆動できます。この図では直結してますが、端子を保護するために 470~1k Ωの抵抗を挟んだほうがよいでしょう（図 4）。

```
s=string.rep("¥xAA", <音の長さ>)
fa.spi("mode",0)
fa.spi("bit",8)
fa.spi("init",<音程>)
fa.pio(0x0F,0x04)
fa.spi("write",s)
```

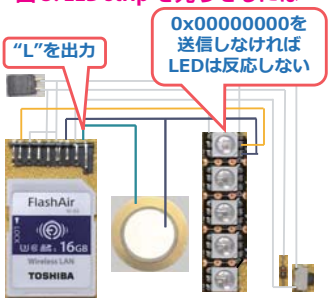


図 4: ブザーを鳴らすには

スイッチの状態を読む

DAT3 端子でスイッチの状態を読むことができます。無線 LAN の ON/OFF で使用しています（図 5）。

```
s, t = fa.pio(0x07,0x04)
k = bit32.btest(t,0x10)
print(k)
```

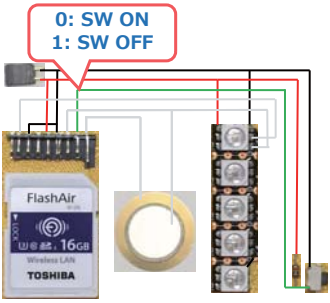


図 5: スイッチの状態を読むには

つなぐ、うごかす、あそぶ

光と電子音で遊んでみました。Twitter で LED 4 個版の動画を公開しています[1]。

1 https://twitter.com/La_zlo/status/941957599833997312

蒼（あおい） さや (@La_zlo)

本職は LSI 開発。独自の解決法を考えて形にするのが好き。
FlashAir W-04 をきっかけに電子工作を始めました。
コードや作例を公開しましたら Twitter でお知らせします。

FlashAir ルーレット基板の設計

余熱

皆さまこんにちは、余熱です。今まで、Lチカ専用基板のAirioや、ゲームコントローラのAirio-PlayなどFlashAirのGPIOを使った基板を色々と設計してきました。今回は、FlashAirを使ったルーレット基板を設計したので、それについて書こうと思います。

図 1: ルーレット基板の 3D モデル

ルーレット基板には8つのLEDと2つのスイッチ、そしてスピーカーが実装されており、単体でルーレットとして遊ぶことができます。また、FlashAirのLuaスクリプトを変更することでルーレット以外の色々な用途に応用可能です。ルーレット基板の3Dモデルを図1に示します。丸い基板の円周に沿ってLEDを配置し、左右にスイッチ、下部にスピーカーを付けています。電源は単4電池2本で、電池ホルダーとSDカードスロットは裏面に配置しています。

設計

LED デコーダ

FlashAirのGPIOは5本しかないため、8つのLEDを駆動するためにはデコード回路が必要です。今回は汎用ロジックICの74HC138を選定しました。74HC138の真理値表を表1に示します。このICを使えば、3本の入力を8本の出力にデコードすることができ、FlashAirからたくさんのLEDを制御することができます。制限として、同時に1つのLEDしか点灯させることができません。

表 1: 74HC138 の真理値表

C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
L	L	L	**L**	H	H	H	H	H	H	H
L	L	H	H	**L**	H	H	H	H	H	H
L	H	L	H	H	**L**	H	H	H	H	H
L	H	H	H	H	H	**L**	H	H	H	H
H	L	L	H	H	H	H	**L**	H	H	H
H	L	H	H	H	H	H	H	**L**	H	H
H	H	L	H	H	H	H	H	H	**L**	H
H	H	H	H	H	H	H	H	H	H	**L**

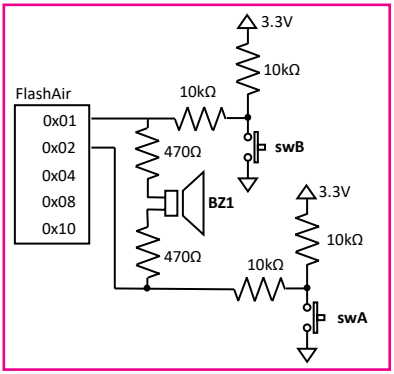


図 2: スイッチ・ブザー部分の回路図

スイッチ・ブザー

スイッチ・ブザー周りの回路図を図 2 に示します。タクトスイッチを 2 つ付けており、同じ端子にブザーも付けています。そのため、スイッチの読み取りとブザーの鳴動を同時に行うことはできません。

基板全体の最終的な信号のアサインを表 2 に示します。I2C や SPI 機能を使っていないため、W-03 でも動作させることが可能です。W-02 でも、Airio Play のように無線 LAN 経由であれば動作させることができます。

基板レイアウト

基板設計には CircuitStudio というソフトを使いました。基板サイズ直径 86mm の両面基板になりました。レイアウトを図 3 に示します。

表 2: 配線プラン

ピン番号	SD インタフェース	fa.pio GPIO	部品 74HC138	部品 tact sw	部品 ブザー
5	CLK	RSV	-	-	-
2	CMD	0x01	-	swB	○
7	DAT0	0x02	-	swA	○
8	DAT1	0x04	A	-	-
9	DAT2	0x08	B	-	-
1	CD/DAT3	0x10	C	-	-
4	VCC		VCC	pull-up	-
3	VSS1		GND	swA,swB	-
6	VSS2		-	-	-

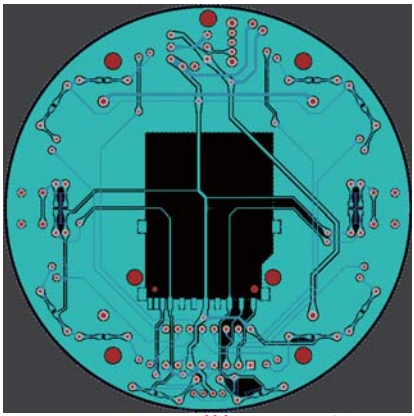


図 3: ルーレット基板のレイアウト

おわりに

Maker Faire Tokyo に出展するのも今年で 5 年目ということで、新しい試みとして小学生向けのハンズオンをやろうという話が出たのが今回のルーレット基板の企画の発端です。Lua スクリプトを入れ替えることで、色々遊べる基板になってるかと思います。

ただ、今回の基板は企画を思いつくのに時間がかかってしまった都合上、試作をすることができませんでした。この原稿を書いてる時点でも動作確認できていなかったり…。MFT 会場でこの基板を見かけなかった方は、お察し頂ければと思います。

余熱 (@yone2_net)

FlashAir 同人誌編集担当。今回のルーレット基板を設計するにあたり、さやさん、クレイン電子さん、その他沢山の方にアドバイス頂きました。ありがとうございました！

枯れた技術の水平思考とおまけ基板

Pochio

FlashAir 芸人の Pochio です。いつもは Maker Faire Tokyo（MFT）の出展やイベント開催の報告記事を書いていますが、久々に違った話を書こうと思います。おまけ基板の製作についてです。技術的な内容は設計者である余熱氏の記事に譲ることにして、こちらは読み物的な記事を書いてみようと思います。

他の出展企業さんも同じかもしれませんが、MFT の出展準備は、みなさん本業の合間を縫って対応しています。この FlashAir 同人誌の製作もそうです。ですから、何か新しい企画をするのは、時間的な制約が厳しいためそれなりに大変だったりします。

そんな中、「今年の FlashAir ブースではハンズオンを企画しよう」ということになりました。ここ数年の MFT は子供たちがたくさん来場していますから、小学生を対象にしたハンズオンがよいだろうと検討を開始しました。すると余熱氏が、あの NINTENDO　LABO に目をつけたのです。NINTENDO　LABO はご存知の方も多いと思いますが、ダンボールと Nintendo Switch とゲームソフトを組み合わせて遊ぶもので、とても任天堂らしい発想豊かなアプリケーションです。釣り竿、ピアノ、バイクのハンドルなど、様々な「コントローラ」をダンボールで自作して遊ぶことができます。

この NINTENDO LABO にリモコンカーなるものがあります。これは Nintendo Switch のコントローラ「Joy-Con」とダンボールを組み合わせて、Joy-Con に内蔵されているモーターの振動を動力にして、ラジコンのように動かすことができるというものです。実際にデモ動画を見てみると、かなりスムーズにリモコンカーが動いていることがわかります。

そこで、FlashAir で振動モーターを制御し、リモコンカーのようなものができないかと余熱氏が考えました。ハンズオンのポイントは「いかに振動を前進する力に変えるか」にあります。実際、リモコンカーもどきの足の形状を工夫しないと、振動が伝わってもなかなか前に進みません。ここに子供たちにとっての研究課題がある、と考えたのです。

さっそく業務の合間をぬって足の形を試行錯誤してみますが、これが全くうまくいきません。前に進むようにはなりましたが、NINTENDO LABO のようにスムーズには動作せず、とても一朝一夕で解決できそうな代物ではありません。いやはや、さすが任天堂さんです。小学生向けハンズオン企画は、いきなりとん挫してしまいました。

そんなこんなで今度は何にしようかと再び一から悩み始めるわけですが、既に決まっていることもありました。ハンズオンで使う基板は、FlashAir ブースにて FlashAir を購入してくださった方の「おまけ基板」にすることと、基板の形は丸形にすることです。数年前に丸形のおまけ基板を作った際、大変好評だったのがその理由です。

なかなか良いアイディアが出なくて困っていたのですが、丸形基板ということもあり、土壇場で「ルーレット、あるいは電子サイコロがいいのではないか」という話になりまし

た。ただ、小学生向けのハンズオンなのではんだ付けは大変だし、ルーレットのプログラムを作るのも敷居が高いので、おまけ基板は組み立て済みのものを用意することにしました。さらにルーレットや電子サイコロがあるのなら、すごろく盤があるべきだろうという話になりました。子供たちにすごろくのテンプレートの各マスに好きなイベントを書いてもらえば、想像力を必要とするし、楽しんでもらえるだろうと考えたのです。

というわけで前振りがだいぶ長くなってしまったのですが、実は今回のルーレット基板に面白さを感じています。LED が 8 個とタクトスイッチが 2 個だけのシンプルな基板ですから、オプションでスピーカーも付けられるとはいえ、限られた表現しかできません。でも、LED のそばに数字を書けばルーレットになりますし、大吉とか中吉とか書けばおみくじにもなります。プログラム次第では、LED をボールにみたてて、タクトスイッチを押すと打ち返すようなゲームも作れるかもしれません。このおまけ基板でどんな遊びができるかを考えるのは、とても想像力を要求されると思うのです。

そんなことを考えていたら、横井軍平さんのことを思い出しました。横井さんといえば、任天堂の開発第一部部長として「ゲーム & ウォッチ」や「ゲームボーイ」などを開発された方です。横井さんは「枯れた技術の水平思考」という考え方を製品開発の基本とされてきました。「枯れた技術」とはすでに広く使われている技術のことで、「水平思考」とは様々な視点から物事を見るということです。つまり、「既存の技術をこれまでとは違う使い方をすることで、新しい遊びを生み出そう」とされていたのです。これは Nintendo Switch にダンボールを組み合わせて新しい遊びを生み出した、NINTENDO LABO に通じるものがあるように思いました。

実は FlashAir も、「枯れた技術の水平思考」が生かされた製品だと思っています。SD メモリカードに無線 LAN と Web サーバを組み合わせたのが FlashAir で、その後 GPIO と Lua も搭載されました。個々は既存の技術ですが、組み合わせるとこの同人誌に掲載されているような、様々なアプリケーションを生み出すデバイスとなるわけです。約 5 年前のオープンソースカンファレンス東京（秋）で、このような内容のプレゼンをしたのを思い出したのですが、まだ YouTube で見ることができるようです 。

というわけで、ぜひおまけ基板の新しい遊び方を「枯れた技術の水平思考」で考えてみてください。そして新しい遊びが生まれたら、ご連絡いただけると大変うれしいです。

Pochio　(@I_love_nintendo)

自称 FlashAir 芸人。毎回 MFT のネタを考えるのはとても大変ですが、ブースにお越しいただいた方に喜んでいただけるとうれしく思います。

宮内

一昨年はゲームを作って記事を書いたが、それ以降はネタがなくて絵だけで参加。某チョコ菓子シール風のイラストも作成。当時集めてなかったのに妙に詳しくなってしまった…。

https://flashair-developers.com