FlashAir™ Doujinshi 4

FlashAirの同人誌４

TAKE FREE ￥0

最新版の FlashAir W-04がついに登場！
FlashAir IoT HubでSDカードスロットを
搭載したすべての機器をIoT化するぞ！

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 同人誌も 04(ゼロヨン) 世代に

上岡 裕一

FlashAir同人誌も今回で第4号となりました。皆様の「Make魂」に支えられて、毎回どんどん中身が濃いものになり、今回も本当に多くの有志の方々にご協力いただきました。恐らく冊子としては過去最高の「厚み」に。(ギリギリ「薄い本」?)

FlashAir は 2012 年 3 月に第 1 世代品 (FlashAir Class6) の販売を開始。 販売開始から 5 年を経過した 2017 年 6 月には、初の「フルモデルチェンジ」となる第 4 世代品 (FlashAir W-04) の販売を開始しました。W-04 のセールスポイントは何と言っても「大幅な性能向上」です。SD メモリカードとしての性能向上として UHS-I インターフェースに対応し、4K 動画撮影にも対応可能な UHS スピードクラス 3 に対応。最大読み出し速度 90MB/s、最大書き込み速度 70MB/s を実現。デジタル一眼レフカメラなどでの使用感の大幅な向上を見込んでいます。無線通信性能については、第 3 世代品 (W-03) の約 2.9 倍の 31.4Mbps の転送スピードを実現。動画データや RAW データなどがより快適に扱える事を期待しています。また、第 4 世代品の「大幅な性能向上」をアピールするために、ラベルデザインを一新しました。

FlashAir イコール「青 (水色)」というイメージを覆すものであり、当初は開発メンバ間でも意見が分かれました。しかしインターネットでの販売を考えると商品の「顔」であるラベルデザインをひと目見ただけで「新しい FlashAir だ!」と気づいていただける様にこのデザインに落ち着きました。いかがでしょうか?

ちなみに、今回の色 (濃い目の青) は何の色かお気づきでしょうか? 実は FlashAir 同人誌 1 号のカバー色と、Maker Faire でメンバが着ている「ハッピの青」がモチーフなのです。装いも新たな FlashAir W-04 と 同人誌「04(ゼロヨン)」をよろしくお願いします!

上岡 裕一 (@YuichiKamioka)

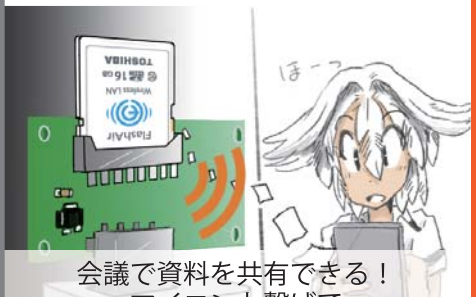
自称、FlashAir の「義理の父」。四男坊 (W-04) は商品企画から担当。可愛い息子の売込みで東奔西走中。

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！

FlashAir Developersにアクセス！

Contents

FlashAir で IoT をはじめよう！

祝！ FlashAir W-04 発売開始！

性能アップ

最大読み出し速度 90MB/s
最大書き込み速度 70MB/s
無線転送 2.9 倍

Developer 向け機能強化！

WebSocket

リアルタイム双方向
通信が可能に！

共有メモリ増強

512Byte→2048Byte

GPIO / SPI / I²C

5bit GPIO
SPI マスタ
I²C マスタ

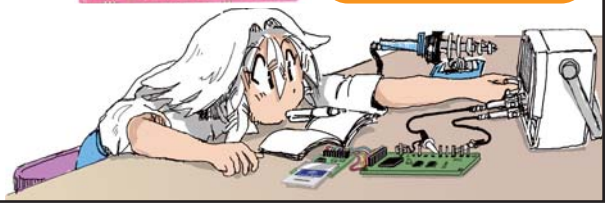
評価ボード続々登場！

GR-LYCHEE

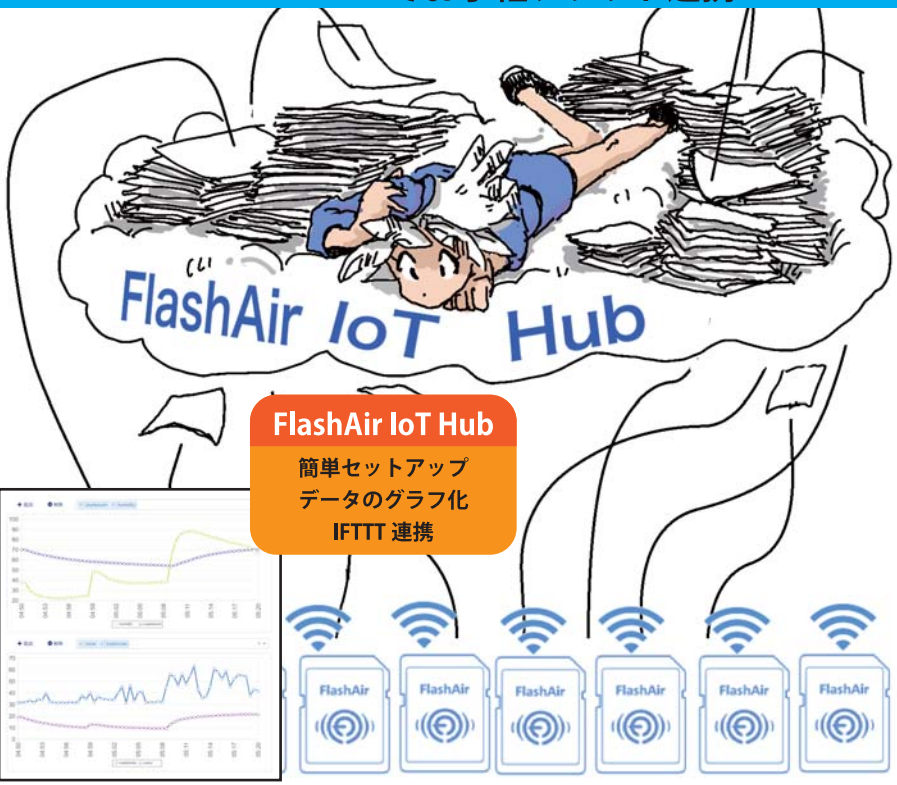
mbed 対応
カメラ I/F 付き

TOY Board

BLE- 無線 LAN
ゲートウェイ基板



FlashAir IoT Hub でお手軽クラウド連携！



FlashAir IoT Hub

簡単セットアップ
データのグラフ化
IFTTT 連携

その他作例もたくさん！

FTLE

FlashAir 用開発環境
Lua のデバッグ等に

スマートロック

遠隔でドアの開錠

AirLapse

定点撮影サービス

W-04 の開発にあたり

伊藤 晋朗

我が家には小学生の娘がいる。ちなみに今は 3 年生になっているが、小さな小学校なので入学時にはクラスに 11 人しかおらず、その中でも女の子は娘を合わせて 4 人しかいなかった。入学後しばらくして 10 月になると小学校につきものの運動会がやってきた。すると、学年のリレーの代表として男子、女子それぞれ 4 人の選手が必要になるが、足が速いとか遅いとか関係なく自動的にクラスの女の子が全員リレーの代表選手になった。しかし、娘は 2 月生まれで体も小さかった。リレーの選手というと運動会では花形だが、周りの子との体格の違いはあまりにも大きく、比較するとどうしても速くは走れなかった。

そんな小学校の運動会の前後に、FlashAir の W-04 の開発がスタートする。W-04 の企画の当初から「もっと FlashAir を速くしたい!」という声が上がったので、無線部分のコントローラの LSI を新規開発しようという話があった。W-04 開発以前にも同様に LSI 開発しようという計画があったのだが、その時には目標スペックを大体これぐらいかなと思って提案してみたところ、あっさりとトップにダメ出しされてボツになった記憶が残っている。振り返って考えてみると開発側がこれぐらいかなという目標は結局、どこかブレーキを踏んでいるので、製品の目標値としては見劣りするものになってしまっていたのだろう。

そんな企画案の失敗を一回しているので、W-04 の開発では、それまで 15~16Mbps ぐらいであった FlashAir のスループット性能に対して、新製品の目標値は 100Mbps のスループットを出しましょうという事にした。この目標値に関してはぶっちゃけると技術的な裏付けのある理由から目標を決めたわけではなく、「3 桁の数字を出せばみんな黙るだろう。」ぐらいの考えで決めたものなので、結構適当である。

開発がスタートして社内にて LSI 開発がスタートするとひとまず HW 開発者の出番となり結構な人数がかかり、結構なお金をかけて HW のサンプルを作ることになる。そのサンプルで FlashAir W-03 をベースにした FW を動かしたときに、結局 15~16Mbps しか出なかったのは記憶にある。まだ最初だったが、目標値と乖離していた数字だから「全然性能が出ていないけどこれからどうしよう。」という、焦りも感じたのを記憶している。

その後にソフトとしてのボトルネックを解消するためにファイルの読み出し方法を変更したり、内部バッファのサイズを変更したり、新しい API を作ったりしたり、色々なモジュール、スタックを改造した。それに伴いソフトアーキテクチャとしてはたくさんのレイヤーバイオレーションもした。個人的にはレイヤーバイオレーションをしないと新しい物が作れない時代になっていると思っているので、やらない方が良いという人もいるが、今は必要とされるのではとも思っている。

色々と試行錯誤して時間がたつ中、性能は上がっていくのだがどうしても 3 桁まで数字が行かない。無線環境の問題や通信相手の性能というのもあって、社内からは、「理論値では 100Mbps 行かないよ。」という事も言われたりもした。部長あたりから「早く目標修正したら」みたいな話も出てくるが、なんとなくあとちょっとなので、終わり方としては切りが悪い。そんなときに、HW 担当の人に「この計算は 2 ビットで計算しているので、8 ビットにすると既に目標を達成しているんじゃない?」と言われた。測定環境で 12MB/s と計測されていたから計算して 96Mbps としていたが計算方法が違っていた。

12MByte = 12 x 1024 x 1024 x 8 bit = 100,663,296 bit

改めて計算すると 100Mbps 以上にはなる。「なんだ達成していたじゃない。」という結局キレの悪い結末になったが、結果は結果として割り切るしかない。実際には通信相手の性能や無線環境の影響があるので、ある特定の無線端末相手という事になってしまうので、表向きこんな性能が出るよと謳うことは無いだろう。

2017 年の日差しがきつい 5 月のある土曜日に小学校で運動会があった。今年からは学校側の都合もあり春運動会らしい。そして、クラスに転校生が入ってきたということで人数が増えたためか娘はリレーの選手にはならなかった。そのため、いつものあのリレーの時間は妙なドキドキする気持ちが無くて残念な感じである。しかし、娘が障害物競走で走った時、なんと 3 位でゴールをすることができ初めて順位に入ることができた。その姿を写真に撮ろうとカメラを持って近づくと体操着に貼られた 3 位のシールを嬉しそうに見せてくれる。こちらとしては笑顔の写真が撮れるだけでも嬉しいのだが、子供が喜んでいるのはなお嬉しい。

やっぱり「速い」って、いいことなのかもしれない。

伊藤 晋朗 (@ikainuk)

FlashAir の " 中 " の人。最近は PPT を作っている事が多い。もう何年も前から「無線とストレージの融合」という事に関して検討をしている。

Maker Faire Tokyo とサミットと分社化と

Pochio

FlashAir 芸人の Pochio です。ありがたいことに今年も Maker Faire Tokyo(MFT)で、4 冊目の FlashAir 同人誌を発行できました。まさか 4 年も続くとは思いませんでした。ご承知かとは思いますが、この 1 年の某弊社は色々な意味で下降の一途を辿っていましたから、今年の MFT への出展は相当に厳しいと思っていました。O'Reilly さんやいつもご支援いただいている方々のおかげで、こうして MFT に参加できたことをうれしく思います。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。

Make Faire Tokyo 2016 出展

FlashAir の IoT 応用事例が登場

3 度目の出展となった MFT2016（図 1）では、いくつか新しい試みをご紹介しました。綾瀬ヒロさんのおなじみ FlashAir を活用した「ワイヤレス鉄道模型制御装置」シリーズは、G ゲージとよばれる大型鉄道模型を使った新バージョンの登場です（図 2）。FlashAir を用いたワイヤレス運転や自動運転の機能だけでなく、車内に Web カメラを内蔵することで、運転台から見える迫力ある走行風景をタブレットで見ることができるようになっていました。さらに、(いつもの) スーツケースに入った鉄道模型が IoT 化されていました。FlashAir によって列車の位置や速度の情報が Microsoft のクラウドサービスである Azure IoT Hub に送信され、クラウド上でこれらを確認できるようになっていたのです。

図 1: FlashAir ブースの様子

図 2: G ゲージ鉄道模型

ところで最近よく耳にする IoT とは “Internet of Things” の略で、直訳すると「モノのインターネット」となります。様々なモノ（電子機器など）をインターネットに接続することを意味し、IoT による新しい世界やビ

ジネスを築こうとする動きが盛んになっています。今回、綾瀬さんの手によってデジタルカメラも IoT 化されました。なんとデジタルカメラで撮影した写真を、スマートフォンが音声で教えてくれるのです。名付けて「しゃべるデジカメ」です。

まずデジタルカメラで写真を撮影すると、FlashAir がその写真を Microsoft のクラウドサービスである Azure Cognitive Services に送ります。Azure Cognitive Services はその写真がどんな写真かを解析し、英語のテキストデータで回答します。例えば芝生に座っている猫の写真を撮影すると、“a cat is sitting in the grass” と返してくるのです。これだけでも驚きの技術ですが、続いてそのテキストファイルをもう一度 Azure Cognitive Services に送ると、今度は文字が合成音声に変換されて、スマートフォンから出力されるのです。様々な技術が集約されて実現された「しゃべるデジカメ」ですが、MFT 会場では無線 LAN が混雑していたせいもあってデモがうまく動かず、その凄さが伝わらなかったかもしれません。チャンスがあれば、またどこかでご紹介したいと思います。

他にも FlashAir 同人誌 3 号（図 3）に掲載された作品を展示しました。その同人誌 3 号ですが、表紙を除いて過去最大のフルカラー 66 ページとなりました。MFT2016 のために 2000 部を用意したのですが、好評につき全て配布しきってしまいました。同人誌を手に入れるためにわざわざ来てくださる方もいらっしゃいまして、本当にありがたいです。

図 3: FlashAir 同人誌 3

FlashAir を使った製作事例だけでなく漫画や読み物的な記事もある、そんな FlashAir 同人誌の 1 号から 3 号は、現在 FlashAir Developers のサイトにて PDF でご覧いただけます。この 4 号で FlashAir 同人誌をはじめて知った方は、ぜひ既刊もお読みいただければ幸いです。

秋月電子通商さんの出張販売が実現！

例年どおり MFT 会場において FlashAir を販売し、そのおまけとして新しいコースター基板を作成しました（図 4）。コースター基板とは、そのままなら飲み物を置くコースターとして使えるし、部品を実装すると何かが完成するという画期的なおまけです。MFT2015 で作成したコースター基板が好評だったので、すっかり味をしめました。前回は円形の基板でしたが今回は四角形で、部品を実装すると FlashAir でゲームを遊ぶことができるコントローラが完成します。しかも部品を買い集

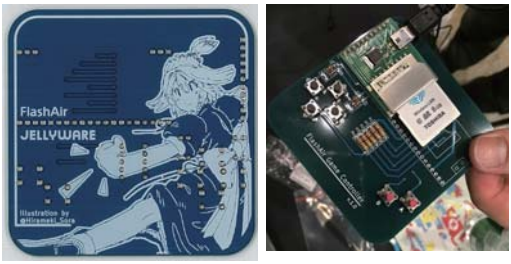


図 4: FlashAir 購入特典のコースター基板

める手間を省くため、予めほぼ全ての部品を用意して基板とセットにしました。しかし一部の部品だけは、かの秋月電子通商さんにてご購入いただく必要がありました。

そこで MFT2016 では、なんと秋月さんに FlashAir のブースにおいて FlashAir 関連部品を販売していただきました（図 5）。つまりわざわざ秋葉原に行かなくても、MFT 会場で部品がそろってしまうという素晴らしい展開になるわけです。ちなみにこのコースター基板で作ることのできるコントローラは、FlashAir 同人誌 3 号に掲載された余熱氏によるゲームコントローラ基板「Airio Play」とほぼ同等です。興味のある方は、PDF 版の同人誌 3 号をご覧ください。

図 5: 秋月電子通商さん出張販売

ところで余談ですが、MFT2016 の会場から少し離れたビッグサイトの会議棟の一室で、MFT2016 に出展できなかった方が「裏メイカー祭」というのを開催されていました。このアイディアといいますか執念といいますか、メイカーにかける思いといいますか、いずれにせよ熱い思いを感じました。ちなみに会場となった会議室の中はとても涼しかったです。

SD カードスロットを搭載したすべての機器を IoT 化するぞ！（願望）

FlashAir IoT Hub の登場！

真夏の MFT が終わり、秋になると同人誌 3 号で予告していました FlashAir の新しいサービスが始まりました。FlashAir でお手軽に IoT を試すことができる、「FlashAir IoT Hub（β版）」の公開です（図 6）。

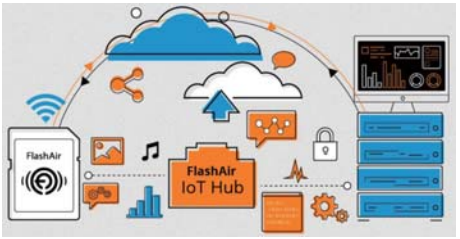


図 6: FlashAir IoT Hub のイメージ

FlashAir は SD カードなので、SD カードスロットを搭載した電子機器や家電と接続することができます。その電子機器などから FlashAir に書きこまれたデータは、インターネット経由で FlashAir からクラウドサーバなど（以下クラウド）に送信することができます。逆にクラウド側からインターネット経由で FlashAir にデータを書き込むと、電子機器はそのデータを読み込むことができます。

つまり、もともとインターネットに接続する機能を持たない電子機器でも、制約はありますが FlashAir でインターネットに接続できるようになるわけです。ということは、世界中にある SD カードスロットを搭載した電子機器が、FlashAir で IoT 化できるかもしれないのです！　しかし IoT 化のためには FlashAir をクラウドと接続する必要があります。こ

の設定が大変で、初めての人には敷居が高いのが難点でした。

そこで FlashAir IoT Hub の出番です。これを使うと結構簡単に、しかもいまのところ無料で FlashAir とクラウドとの接続を試すことができますから、初心者さんにおススメの実験環境と言えます。そんな FlashAir IoT Hub が公開されたので、ぜひ FlashAir をどんどん IoT に活用してほしいという思いから、何かイベントを開催しようという機運が高まりつつありました。

FlashAir Developers Summit の開催

というわけで、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、IoT ネタを中心にしたユーザー向けイベント「FlashAir Developers Summit（以下サミット）」を 2017 年 2 月 19 日に開催することとなりました。イベントのイメージ画像も JellyWare さんにデザインしていただきました（図 7）。会場はいつもお世話になっている綾瀬ヒロさんと、日本マイクロソフト（株）の大森さんにご協力いただきまして、日本マイクロソフト本社ビル（品川）のセミナールームをお借りしました。さらに（株）タイプ・アールの高瀬さんのご協力で、サミットの様子のライブ配信も実現しました。現在もそのライブ配信の録画を YouTube にてご覧いただくことができますので、「FlashAir Developers Summit」で検索してみてください。

図 7: FlashAir Developers Summit

図 8: サミット会場の様子

当日は「SD カードスロットを搭載したすべての機器を IoT 化するぞ！（願望）」というスローガン（？）の下、95 名の方に参加申し込みいただきまして、ライトニングトークを含めて 13 名の方にご講演いただきました（図 8）。

分社化と FlashAir の今後・・・

ところで、このサミット企画について検討していた 2016 年の年末に、東芝の米国原発事業による 7000 億円の損失を突如報道で知りました。それまで復活傾向にありましたから、青天の霹靂とはまさしくこのことです。年が明けて 2017 年 1 月 27 日に、FlashAir を開発・製造する（株）東芝ストレージ&デバイスソリューション社のメモリ事業部の分社化が発表されました。続いて 2 月 14 日に、分社会社の株式の過半を譲渡することが検

討されているとの発表がありました。この発表を受けて、イベントにご参加いただいた皆さんは FlashAir が今後どうなってしまうのか気になっていたようです。正直私も FlashAir がどうなるのか心配でしたが、それよりも今後芸人としてどうやって生きていくのかについてとても悩みました（謎）。

図 9: od さん 2000 リツイート達成！

それはさておき、実はこのサミット開催の時点で既に新しい FlashAir（W-04）の年内発売が決まっていたので、まず W-03 が近々販売終了になることをサミット冒頭でお知らせしました。参加者の皆さんには「サミットの内容を Twitter で自由にツイートしてください」とお願いしていたのですが、ここで od さんの W-03 販売終了に関するツイートが短時間のうちにどんどんリツイート[1]されて、最終的に約 2000 リツイートを達成（図 9）。驚きながらカウントの数字が増えていくさまをみていました。おそらくこれまでの報道から、「FlashAir の事業が終了するかも・・・」と思った方々が次々とリツイートされたのでしょう。ちなみにこの W-03 販売終了のお知らせの直後に、新製品の W-04 を開発中であることをお知らせし、od さんもこの情報をツイートしてくださったのですが、こちらは約 60 リツイートにとどまりました（悲）。

サミットの最後に懇親会を開催したのですが、なんと FlashAir の非公式応援キャラクター「閃ソラ」ちゃんのコスプレをしてくださっている、ハッカソンアイドルのくーらさんがたまたま東京に来られていて、忙しい合間を縫って駆けつけてくださいました。いつもご協力いただき感謝です。サミット開催のためにご協力いただいた皆さまと参加者の皆さまに御礼申し上げます。ありがとうございました！

図 10: ソラちゃん登場！

1　他の人のツイートを再びツイートすることを、「リツイート」といいます。

ハンズオンの開催と国際会議での発表

FlashAir Developers Summit ハンズオンの開催

ところで MFT2016 の FlashAir 購入特典として用意したコースター基板ですが、潤沢に在庫が残っていまして、もったいないのでこれを製作するハンズオンを開催しようということになりました。「FlashAir Developers Summit ハンズオン」と題して、かつて秋葉原の総武線高架下にありました「秋葉原ラジオストアー」のスペースにある、「ASSEMBLAGE（アセンブラージュ）」にて 3 月 5 日に開催しました。親子連れで参加して下さった方もいまして、総勢 12 名の方にご参加いただきました。半田づけが初めてという方もいらっしゃいましたが、ASSEMBLAGE の方が懇切丁寧に教えてくださるので、皆さん無事に完成させることができました。FlashAir でゲームを遊べるなんて驚きですよね。このハンズオンがとても好評だったので、できればまた開催したいと考えています。ASSEMBLAGE の皆さんには大変お世話になりました。ありがとうございました！

FlashAir の IoT 活用を国際会議で発表

最後になりますが、2017 年 6 月 18 ～ 22 日に米国テキサス州オースチンで開催された、50 年以上続く集積回路自動設計に関する国際会議「54th Design Automation Conference（通称 DAC）」に参加し、FlashAir の IoT 活用についてポスター発表をしてきました。約 5000 人が参加する大きな学会です。アメリカでもいつもの法被を着て来場者に説明しました（図 11）。この発表の様子は日経テクノロジーオンラインさんが取材してくださいまして、「東芝メモリがアピール、IoT における FlashAir の役割」というタイトルで 2017 年 6 月 28 日に記事が掲載されています。よろしければご覧ください。

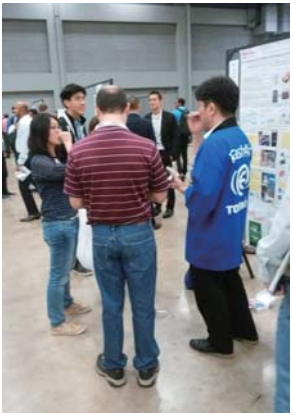
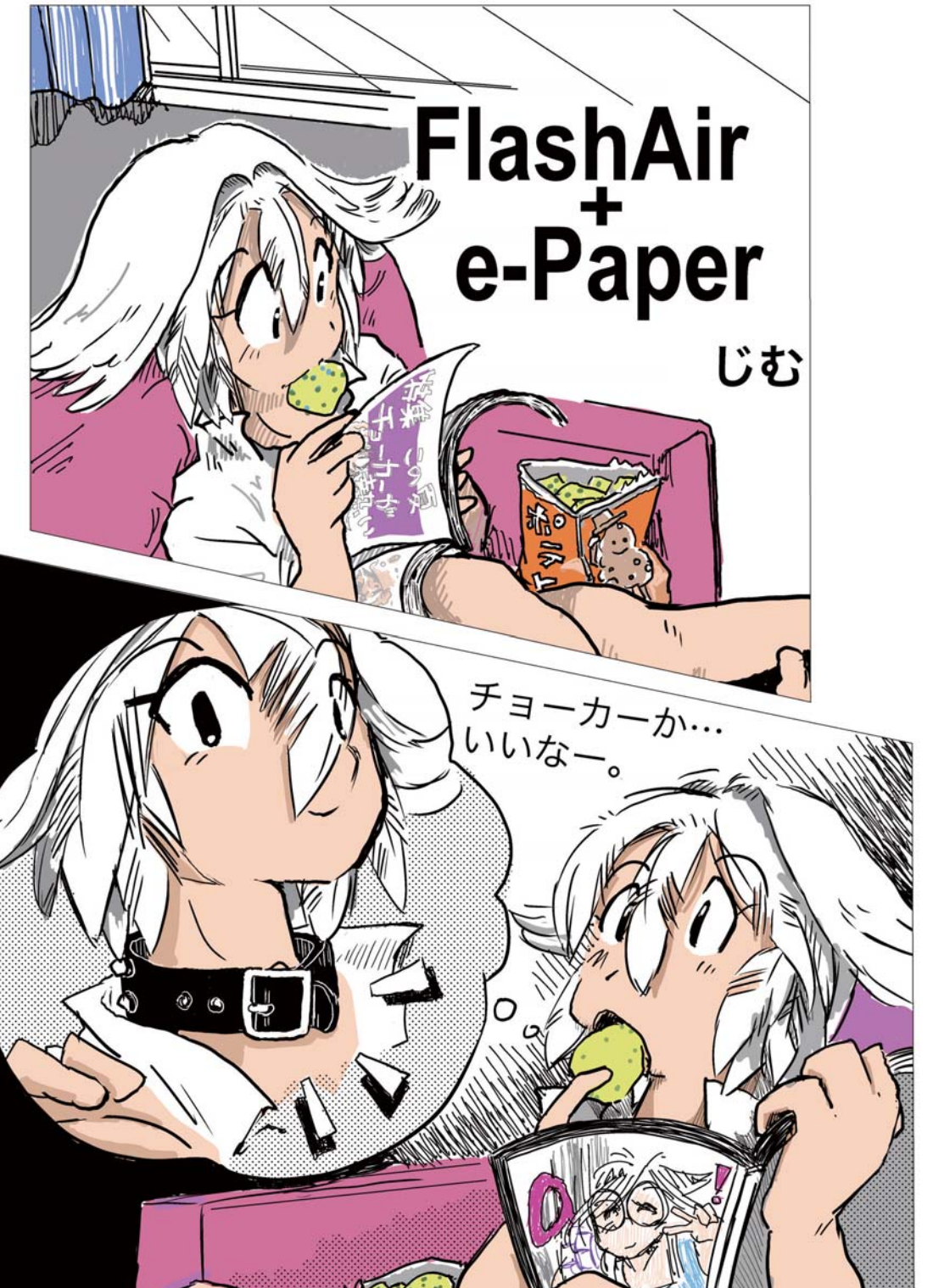


図 11: DAC での発表

というわけで色々なことが起こった 1 年でしたが W-04 が発売されましたので、また FlashAir を盛り上げていければと思っています。イベント開催が決まりましたら FlashAir Developers のサイトや、Twitter などで告知しますので、ぜひお気軽にご参加頂ければ幸いです。

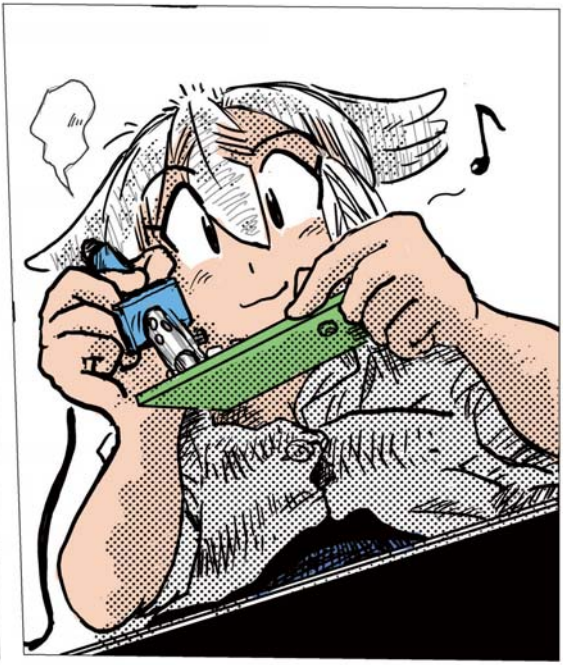
Pochio (@I_love_nintendo)

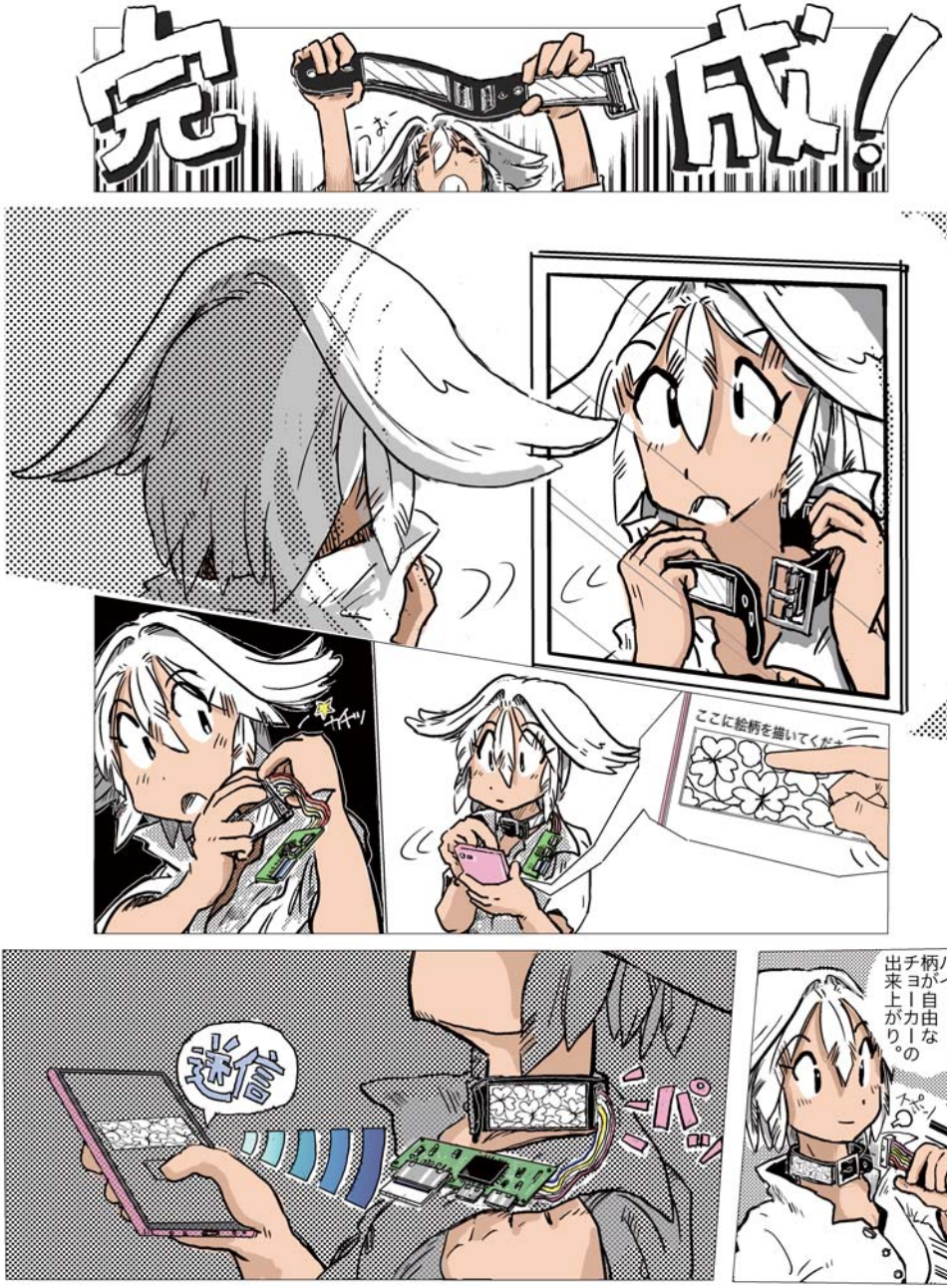
自称 FlashAir 芸人。最近芸人の出番がめっきり減ったのでお仕事募集中。今年は記事を書くのに苦心したので、来年の執筆のためにもなにか面白いことをやりたいですなぁ。


FlashAir
+
e-Paper
じむ
チョーカーか…
いいなー。


まずは
部品の買い出し！
電子部品売り場
これこれ。
あった。


カメラ
W-04
ゲット




完成！
ここに絵柄を描いてくだ
通信
ハイ。柄が自由なチョーカーの出来上がり。

FlashAir + e-Paper

じむ

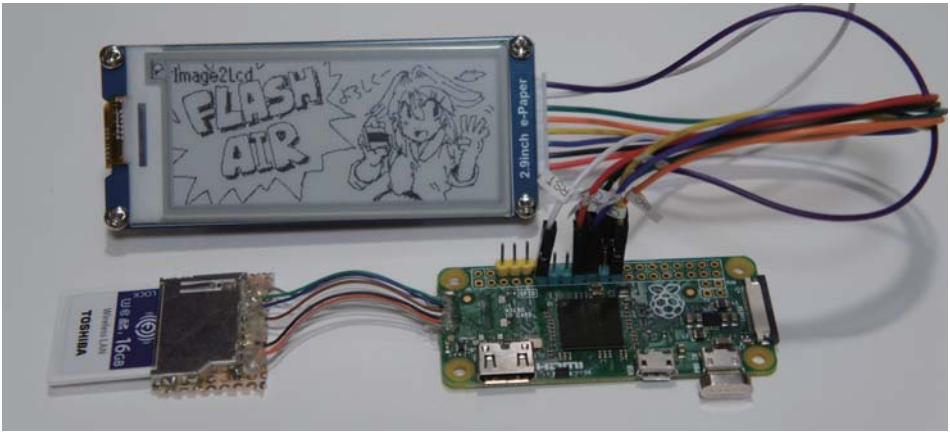
電子書籍とか、スーパーのタグで見かける「e-Paper」。これのいいところは**電源を切ってから、本領発揮**するところですね。

電源がいらないということは、ずーっと使えるってことです。メモ用紙代わりにもなるし、ちょっとしたサイネージにもなります。そして、そこに **FlashAir の登場**です。

中身を書き換えるときは電源を入れて、FlashAir に対して無線でデータを送信してサッとディスプレイを書き換えれば、あとは電源いらず。

"IoT" なんてかたっ苦しいことは置いといて、ソラちゃんはアクセサリーに応用して、チョーカーを作ってました。LED で光るチョーカーだったら絶えず電源の確保が必要ですが、「e-Paper」ならそれも不要。この参考の写真も、ディスプレイにはもう**電源が入っていない**ところがミソなんです。

元が白黒のデータですから、データ量も小さく FlashAir にもいっぱい入ります。色々切り替えて楽しんでは如何でしょうか。



じむ

エンジニアだけど、連載４年目に突入〜っ！（笑）

FlashAir W-04 の進化点

GPS_NMEA (@Seg_faul)

FlashAir W-04 の進化した点を、マイコン的な遊び方で触っている人目線で、ざざっとご紹介します。あくまで概要ですので、詳細は FlashAir Developers や、FlashAir 公式サイトでご確認ください。

また、ここに記載した情報は筆者が個人で測定した結果であり、環境や装置の性能によって差が出る場合があります。予めご了承ください。

読み取り・書き込み速度の向上

SD カードとしての性能がアップ！UHS にも対応しました。USB3.0 リーダーと CrystalDiskMark 5.2.0 で測定したところシーケンシャル Read/Write が実測で

W-03 では Read: 18.88MB/s Write: 11.53MB/s

W-04 では Read: 91.03MB/s Write: 78.86MB/s

となり、Read が約 4.8 倍、Write は約 6.8 倍の向上。大進化を遂げました。(図 1)

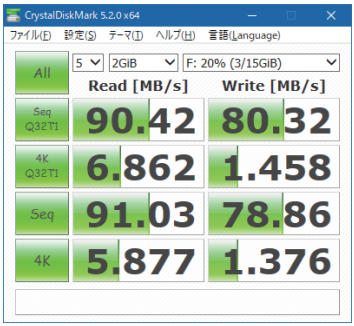


図 1: Read/Write の測定結果

無線通信速度の向上と安定化

一方、FlashAir の目玉である無線通信の速度も向上しました。FlashAir 内に 1.6GB のファイルを配置し、wget で測定したところ、W-03 は 7.7Mbps であったのに対し、W-04 は 92.8Mbps と、約 12 倍の速度に向上。

また、USB3.0 機器の近くは 2.4GHz 帯の通信が妨害されやすい悪環境ですが、その環境において W-03 では 2.7MB の写真 1 枚に 3 分近く (13.1KB/s) かかるほど性能劣化が発生したのに対し、同環境で W-04 は 0.5 秒 (5.94MB/s) までの低下で済んでいることから、ノイズ耐性も向上していることがわかります。

Lua 関数の追加とライブラリの開放

W-04 になり、Lua 関数が幾つか追加されています。またライブラリも追加されています。

- **標準 math ライブラリ**

数学計算ができるようになりました。

- **I²C マスタ fa.i2c**

I²C マスタとしてセンサ等と通信できるようになりました。

- **更新ファイル検索関数 fa.search**

Luaで行っていた最新ファイルや指定ファイル、指定の期間での最新ファイルなどの、ちょっと面倒で重い検索処理が高速かつ簡単にできるようになりました。

- **日付メモ control("time")**

32bit 整数値 (FAT の日時情報が 32bit 整数) が SD 外領域に記録できるようになりました。fa.search と組み合わせて使うことで、FAT の破損を気にすることなく、新しいファイルのみを外部にアップロードできるようになりました。

- **無線 LAN 機能のオン・オフ fa.control("fioget) / fa.control("fioset)**

無線 LAN 機能のオン・オフの取得・設定ができるようになりました。これは以前の Connect, Establish, Bridge, Disconnect とは異なり、Lua スクリプト内に無線 LAN のパスワードを記載する必要なく、CONFIG の通りに実行されます。

早いところが、画像ファイルの保護で無線 LAN をオン・オフする機能が Lua でも使えるようになったのに近いものがあります。

単にオン・オフしたいだけの場合なら、パスワードの保護を考える必要がなくなり、また設定を公式のアプリや設定ツールで行えるようになるので、使いやすさが向上します。

- **接続子機数の取得と、IP アドレス・MAC テーブルの取得　fa.ConnectedSTA**

AP モードの際、接続子機の台数が取得でき、さらに IP アドレスと MAC アドレスのリストが取得できるようになりました。これにより特定の MAC アドレスを持った機器が接続されたとき、あるいは接続されている間のみ動作するものなどが作れるようになりました。

- **その他の関数・ライブラリの追加**

Websocket 関数や、UDP 通信関数 fa.udp や、Lua 標準搭載の corutine ライブラリ、debug ライブラリ、os ライブラリなどの存在が確認されています。

メモリの増強と演算性能の向上、共有メモリの増加

まず、公式情報として、共有メモリの容量が増加しています。具体的には W-03 までは 512Byte の容量であったのが、2048Byte まで増加。

また、FlashAir 開発者向け非公式 wiki に掲載されているメモリ容量調査スクリプトによる測定の結果、W-03 では約 8KB であったユーザー変数領域が、W-04 では約 128KB まで増加。

さらに、同 wiki の Tar 展開プログラムによる処理時間の測定の結果、W-03 では 41 秒かかった処理時間が、W-04 では 17 秒まで短縮されていました。

これでもう、メモリ不足に悩まされることも少なくなります。

おわりに

いかがだったでしょうか。様々な進化を遂げた FlashAir。Lua スクリプトや電子工作で遊ぶために W-03 を使っていた方も、これを機に W-04 をぜひ入手してみてください。

W-04 の新機能 UDP 通信の活用例

GPS_NMEA (@Seg_faul)

UDP 通信機能 fa.udp

W-04 にて搭載された、UDP 通信機能 fa.udp の簡単な活用例を紹介します。fa.udp を使うと、FlashAir から UDP 通信が行えます。試験的機能のようですが、以下のようにするとメッセージの送信が行なえます。

```
fa.udp("192.168.0.255",4000,"message","Hello World")
```

また、以下のようにすると SD カード内のファイルの送信ができます。

```
fa.udp("192.168.0.255",4000,"file","/FTLE/run.lua")
```

受信処理の存在は確認していますが、うまく動作させられませんでした。

また、例示のアドレスのような方式のディレクテッドブロードキャストは動作しますが、255.255.255.255 で行う、いわゆるリミテッドブロードキャストは今のところ動作しないようです。送信はバックグラウンドで行われるため、連続送信には待ち時間を挟む必要があることに注意が必要です。UDP 通信を使うとなると、大抵の場合は独自アプリケーションを開発して使うことになりますが、今回は紹介として、開発不要で気軽に試せる例をご紹介します。

活用例 1. Scratch

昨今、プログラミング教育の義務化といった話題でよく名前の上がる Scratch[1] というソフトウェアをご存知でしょうか。このソフトにはオフラインで使われる 1.4 と、オンラインで主に使われる 2.0 があり、その中でも Scratch 1.4 には遠隔センサー接続という通信機能が搭載されています。

これはその名の通り外部のセンサーやツールと通信することができる機能で、デフォルトでは無効ですが、「センサーの値」ブロックを右クリックすることで、この機能を有効にすることができます (図 2)。この通信は TCP と UDP の両方に対応しています[2]ので、FlashAir からは以下のようなコードでアクセスすることができます。

図 1: Scratch

図 2: 遠隔センサ接続

図 3: ブロック例

1 Scratch は MIT メディア・ラボの Lifelong Kindergarten グループによって開発されました。詳細は http://scratch.mit.edu をご参照ください。

2 TCP は双方向で通信できますが、UDP の場合、Scratch は受信専用になります。

```
-- 変数 value に 1 を設定する
fa.udp("192.168.0.255",42001,"message","sensor-update value 1")
sleep(10)
--Scratch に「start」を「送る」
fa.udp("192.168.0.255",42001,"message","broadcast start")
```

図 3 のようなブロックを配置してから、FlashAir で実行してみてください。

活用例 2. IP Messenger

IP Messenger[3] はリリースから 20 周年を迎え、もはやちょっと懐かしい感じさえするソフトウェアかもしれませんが、まだまだ現役のツールです。iPhone 版などもあります。

このソフトも通信に UDP を用いていますので、FlashAir からメッセージを送ることができます。IP Messenger のプロトコルは以下のように実装できます。

```
fa.udp("192.168.0.255",2425,"message","1:1:user:host:32:Hello World")
```

実行すると図 4 のようにメッセージがポップアップします。

PC 版プッシュ通知が簡単に、しかも Web 接続も不要でできますから、接点入力と組み合わせて呼び出しボタンにしたり、撮影完了と同時に URL を知らせたりと、色々使えます。

遠隔シャットダウン機能もあるので、活用例 3 と組み合わせるとワイヤレス電源リモコンにもなります。

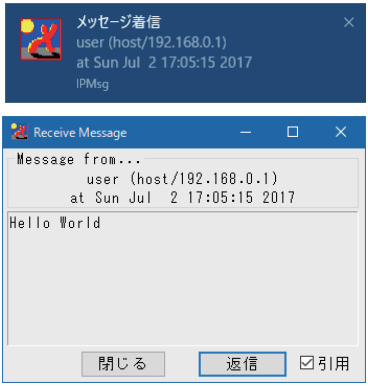


図 4: IP Messenger

活用例 3. Wake on LAN

UDP のブロードキャストが使えるということは、マジックパケットを生成できるということで、Wake on LAN(以下 WoL) も使えます。

念のために説明しておくと、WoL は LAN から信号を送り、眠っている PC を叩き起こすための仕組みです。対象の PC は有線で接続されている必要があるため、FlashAir を無線 AP に接続しておくか、あるいはイーサネットコンバータが必要です。

また、PC 側にも事前の準備が必要ですが、ハードによっても異なるためご自身でお調べください。

受信する PC 側の準備ができたら、FlashAir にスクリプトを打ち込みます。

以下の太字下線の 16 進数を、対象の PC の MAC アドレスに変えてください。

```
mac   = string.format("%c%c%c%c%c%c",0xAA,0xBB,0xCC,0xDD,0xEE,0xFF)
magic = string.format("%c%c%c%c%c%c",0xFF,0xFF,0xFF,0xFF,0xFF,0xFF)
for x=1,16 do; magic = magic .. mac; end
fa.udp("192.168.0.255",4000,"message",magic)
```

実行すると、機器にもよりますが、スタンバイから復帰したり、シャットダウン状態から電源が入ったりします。FlashAir IoT Hub と組み合わせるのも面白いかもしれません。

3 IP Messenger https://ipmsg.org/

新 FlashTools Lua Editor v1.03 のご紹介

GPS_NMEA (@Seg_faul)

新しくなった FlashTools Lua Editor (以下 FTLE) をご紹介します。W-03 からお使いの方も、初めて知る方も、ぜひ導入と更新をご検討ください！

FlashTools Lua Editor をまだご存じない方へ

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。

FlashAir 上での Lua スクリプト開発は、素の状態で行おうとすると、エラー出力もコンパイルエラーも表示されないため、慣れと経験が要求されます。

FTLE は、FlashAir に搭載されている無線 LAN 機能と、アップロード機能、Lua インタプリタを最大限に活かし、シンプルな構造でブラウザ上開発環境を実現したものです。

FTLE を用いることで、文法エラーやプログラムエラーの確認と、プログラム出力の確認、編集、実行が 1 つの画面でできるため、トライ・アンド・エラーを手軽に、高速に行うことができます。

さらに、FTLE には FlashAir の設定を自動で行う機能 AutoSetup も内蔵。ドラッグ & ドロップで書き込んだ後は、無線 LAN 上ですべて完結します。FTLE は開発初心者・上級者を問わず、Lua スクリプト開発を楽でスマートなものにします。

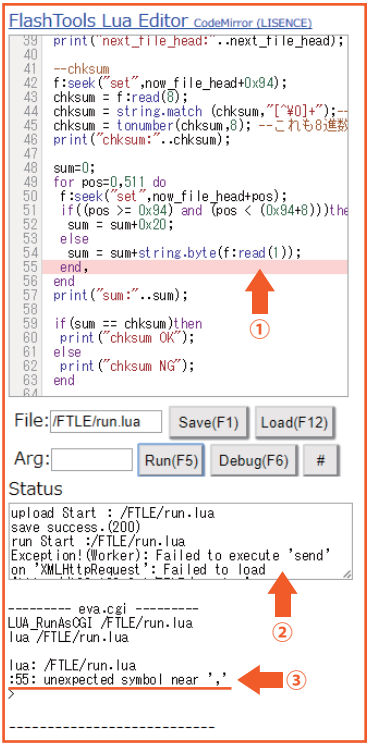


図 1: FTLE の画面

新しくなった点

それでは、既存のユーザーの方向けに新しくなった点をご紹介致します。

- **エラー行のハイライト機能を搭載 (図 1 - ①)**

以前の FTLE では、Lua スクリプトの文法の誤りがあったとき、Debug で実行しても「**attempt to call a nil value**」の一文だけが表示されていました。

そのため、一度エラーが出ると「どこが間違っているのか」、「どう間違っているのか」、「どのファイルが間違っているのか」と、延々試行錯誤して探す状態に陥りがちでした。

新しい FTLE では、非公開機能の利用と debug.lua の改善の両面からこれを解決。

Debug利用時のみならず、通常run時にもエラー情報が表示されるようになり、文法エラーの意味や行数がわかるようになりました。

また、エラー出力をJavaScript上で解析。エラー行を即座に赤色でハイライトするようになりましたので、どこが間違っているのか一目瞭然になりました。

- **JavaScriptとHTMLの全面的な再構築、Web Workerを導入(図1-②)**

新FTLEを実装するにあたり、HTMLとJavaScriptを全面的に書き直しました。見た目はさほど変わっていませんが、細かい動作などが変わっています。

またそれに合わせ、内部でスパゲッティ化していた通信関係処理を一新。Web Workerを用いて同期的に処理するようになり、処理がわかりやすくなりました。

表に見えるところでは、通信エラー発生時に詳細な状態を表示するようになり、また以前はバグで取得できなかったエラーや例外も取得・表示できるようになりました。

- **詳細エラー情報が出るように(図1-③)**

スクリプトの実行前後にeva.cgiへのアクセスを行い、常にエラー情報を取得できるようになりました。eva.cgiはFlashAirの非公開機能であり、FlashAirの内部的なエラーや状態も含めて出力されるため、開発時の諸問題を解決する強力な手助けになります。

- **不要な機能の削除と整理**

利用率の低いと思われる旧版FTLEや、古いまま放置されていたサンプルスクリプトなどを撤去。あわせて、URL指定でスクリプトの読み込みを行う機能も削除しました。

それに伴い、サンプルスクリプトは以下のサイトに統合されました。サンプルスクリプトが必要な方はこちらをご利用ください。多種多様なサンプルが揃っています。

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

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/

GPS_NMEA(@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。

なんだかんだで、FlashAirをいじり始めて2年が経過。

ついでに、アイコンもW-04仕様になりました。

新しくなった Lua で自動アップロード

寺西

例年、Maker Faire Tokyo の時期が近づくと、FlashAir 同人誌の企画が始まります。

今回は、表紙がオレンジで、「Eyefi Connected 機能を搭載したから、オレンジ?」と聞きましたが違うようです。FlashAir W-04 では、Eyefi の一部機能を取り込み、無線 LAN でのデータ転送中に、カメラから電源が切られないようになりました。しかし、FlashAir W-04 に Eyefi のような自動転送機能はサポートされていません（商品の仕様が異なるため）。

FlashAir W-04 で自動転送機能をサポートしていないのであれば、作れば良いと思い、新しく追加された Lua の機能を盛り込みながら紹介しようと思います。

新しくなった Lua 機能

FlashAir W-04 では、以下の Lua 機能[1]が追加されました。

表 1: FlashAir W-04 で追加された Lua 機能一覧

関数名	機能
fa.i2c	I2C (Master) 操作が可能
fa.search	ファイルの更新日時からファイル検索が可能 更新日時が最新であるファイルの検索が可能
fa.control(“time”)	日時メモの保存または取得が可能 （不揮発メモリ内に格納されるため、FlashAir への電源が切られた後でも保持されます）
fa.control(“fioget”)	無線 LAN の On/Off 状態が取得可能
fa.control(“fioset”)	無線 LAN の On/Off 制御が可能 （Lua スクリプト等にアクセスポイントのパスワードを平文で記述する必要がなくなりました）
fa.ConnectedSTA	FlashAir が AP モードで動作しているときの接続中の子機の数などの情報が取得可能
fa.websocket	WebSocket 通信の設定が可能 （通信のオーバーヘッドが減るため、即応性が向上します。クライアントとして動作します）

1　一部更新された機能もあります。fa.request は新たにポート番号を指定可能になり、fa.sharedmemory は使えるメモリ容量が増えています。詳しくは、FlashAir Developers の Web ページをご参照ください。

ファイルの自動アップロード

カメラで撮影した画像ファイルを自動アップロードできるようにするため、前のページに記載した Lua 機能のうち、fa.search と fa.control("time") を使って実装します。アップロード先は FlashAir IoT Hub にしています。FlashAir IoT Hub は、IFTTT と連携することも可能で、他のウェブサービスとの連携には便利だと思います。

基本的に FlashAir Developers に掲載されているサンプルなどを元に作成しました。私が書いた場所は、適当になっています（ごめんなさい）。

```
-- Change string date to FAT date format
local function StringToFatDateTime(datetime_str)
    local pattern = '(%d+)/(%d+)/(%d+)%s+(%d+):(%d+):(%d+)'
      local year, month, day, hour, min, sec = string.match(datetime_str,
pattern)
    year = year - 1980
    sec = bit32.rshift(sec, 1)
    local date_fat = bit32.bor(bit32.lshift(year, 9),
                               bit32.lshift(month, 5),
                               day)
    local time_fat = bit32.bor(bit32.lshift(hour, 11),
                               bit32.lshift(min, 5),
                               sec)
    local datetime_fat = bit32.bor(bit32.lshift(date_fat, 16),
                                   time_fat)
    return datetime_fat
end

-- Change FAT date format to string date
local function FatDateTimeToString(datetime_fat)
    local function getbits(x, from, to)
        local mask = bit32.lshift(1, to - from + 1) - 1
        local shifted = bit32.rshift(x, from)
        return bit32.band(shifted, mask)
    end
    local fatdate = bit32.rshift(datetime_fat, 16)
    local day = getbits(fatdate, 0, 4)
    local month = getbits(fatdate, 5, 8)
    local year = getbits(fatdate, 9, 15) + 1980

    local fattime = getbits(datetime_fat, 0, 15)
    local sec = getbits(fattime, 0, 4) * 2
    local min = getbits(fattime, 5, 10)
    local hour = getbits(fattime, 11, 15)

     return string.format('%02d/%02d/%02d %02d:%02d:%02d', year, month, day,
hour, min, sec)
end
```

```
-- Search non-download files
local function NonDownloadFileSearch(dl_datetime)
  local result, filelist, time = fa.search("file", "/DCIM", dl_datetime)

  return filelist, time
end

-- Last download date
local function LastDownloadDate()
  local str_date = 0
  local result = fa.control("time")

  if result == -1 then
     str_date= "0x" .. string.format("%x", StringToFatDateTime("1980/01/01
00:00:00"))
  elseif result ~= nil then
    str_date = string.format("0x%x", result)
  end

  return str_date
end

-- Upload Image file to FlashAir IoT Hub
-- FlashAir IoT Hubにある upload_image.lua を使用のため、省略。

-- Main
local str_date = LastDownloadDate()
local new_filelist, time = NonDownloadFileSearch(str_date + 1)

local t = {}
for fpath in string.gmatch(new_filelist, '(.-),') do
  table.insert(t, fpath)
end

for i = 0, 100 do
  if t[i] ~= nil then
    uploadImage(t[i])
  end
end

result, filelist, new_time = fa.search("file", "/DCIM", -1)
result = fa.control("time", new_time)
```

CONFIG には、STA モードにする設定（APPMODE）とカメラで撮影したときに Lua が動作する設定（LUA_SD_EVENT）などを追加しています。

```
[Vendor]

APPAUTOTIME=0
APPMODE=5
APPSSID=SSID
APPNETWORKKEY=********
CIPATH=/DCIM/100__TSB/FA000001.JPG
PRODUCT=FlashAir
VENDOR=TOSHIBA
STA_RETRY_CT=0
LUA_SD_EVENT=/upload.lua
```

おわりに

今回、Eyefi のような自動アップロード機能を作ってみました。

FlashAir W-04 で Eyefi Connected 機能をサポートしたのですが、Eyefi 社の CEO であった Matt DiMaria 氏はじめ、Eyefi の皆さんと協力しながら進めてきた機能になります。Eyefi Connected 機能カメラをお使いの方は、FlashAir W-04 を是非使ってみてください。

寺西 (@soft128)

2016 年度までは FlashAir および NFC 搭載 SD メモリカード担当だったのが、2017 年度より普通の SD メモリカード、microSD メモリカードも担当。

この原稿は徹夜で作成したものです（泣）。

FlashAir W-04 の I²C 機能を使おう！

綾瀬 ヒロ

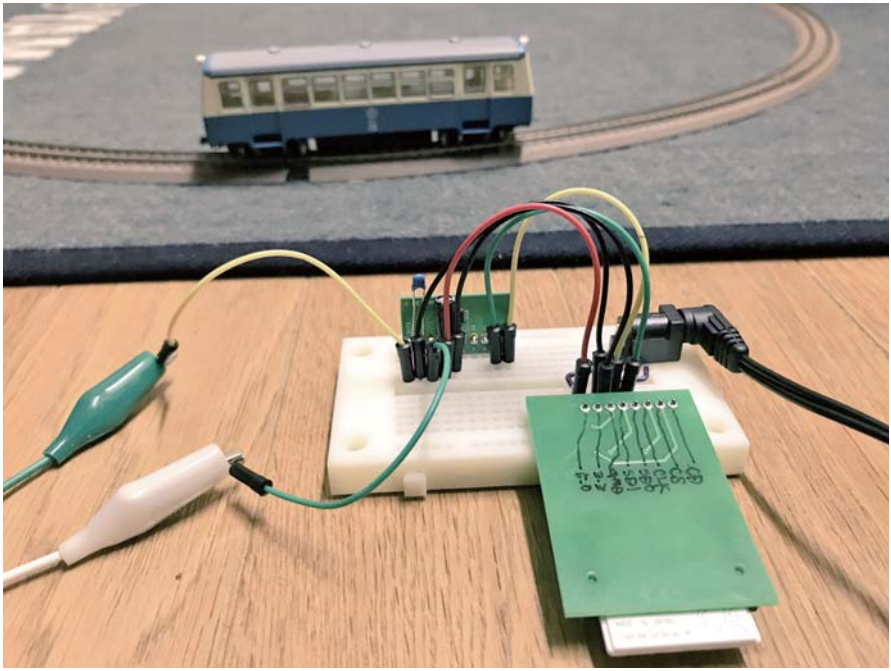
FlashAir で鉄道模型をコントロールして遊んでいる綾瀬です。

さて、今回はついに発売となった第 4 世代 FlashAirW-04 の Lua 機能に新しく追加となった I²C 機能[1]を使って、鉄道模型を運転してみます。

FlashAir の I²C 機能について

I²C（アイ・スクエアド・シー）は、２本の信号線（SCL、SDA）を用いて、デバイス同士を接続してシリアル通信するものです。FlashAir W-04 では、SD カードの 2 番ピン（CMD）が SCL、7 番ピン（DAT0）が SDA として機能します。これは FlashAir の GPIO 機能のビット割当においてそれぞれ 0x01、0x02 にアサインされているピンになりますので、I²C 機能は、内部的には GPIO 機能の 0x01 と 0x02 を使用していると推測できます。

なお、利用する場合は Lua 関数リファレンスに書かれている通り、SCL/SDA はプルアップ、CLK（5 番ピン）はプルダウンします。



1　Lua 関数リファレンス – I2C 機能
https://flashair-developers.com/ja/documents/api/lua/#i2c

I²C 対応モータードライバを接続してみる

ではさっそく、I²C で接続できるモータードライバを FlashAir W-04 に接続して、鉄道模型を運転してみようと思います。今回は、秋月電子通商の DRV8830 というテキサスインスツルメンツ社製モータードライバ 8830 を搭載したものを使います。

DRV8830 は動作電源電圧範囲が2．７５Ｖ～６．８Ｖとなっていますので、一般的な N ゲージ鉄道模型の最大対応電圧 12V に対して低い電圧になりますが今回は 5V で動作させます。5V でもモーター車両は低速で走行します。

構成を図 1 に示します。基本的には電源線と I²C の信号線を接続するだけです。DRV8830 の ISENSE 端子は、今回は GND に接続します。また FAULTn は、今回は障害通知処理を省略するため使用しません。I²C アドレスを設定する A0、A1 もオープンとして、書き込みアドレス 0xC8 と設定します。

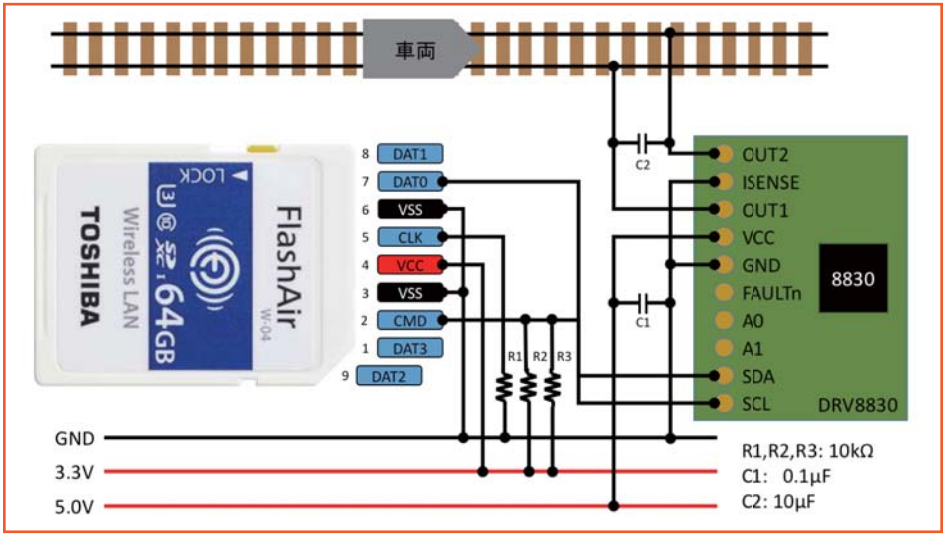


図 1: 構成

つづいて、Lua スクリプトを書いてみましょう。FlashAir W-04 では Lua 機能に I²C 機能として fa.i2c という関数が追加となりました。

fa.i2c(table) という書式となっており、table 部分にモードを記述することで、デバイス初期化、スタートコンディションの送信、リスタートコンディションの送信、接続デバイスからデータ読み出し、接続デバイスへデータ書き込み、ストップコンディションを送信、コネクションを閉じることができます。

I²C コマンドを送信する関数を以下のように記述します。

```
function write_i2c_command(addr, data1, data2)
     res = fa.i2c{ mode="start", address=addr, direction="write" }
     res = fa.i2c{ mode="write", data=data1 }
     res = fa.i2c{ mode="write", data=data2 }
     res = fa.i2c{ mode="stop" }
end
```

アドレス addr、初期送信コマンド reg、速度 vset と方向 data で DRV8830 に送信するコマンドを生成し、write_i2c_command を呼び出す関数を以下のように記述します。

```
function sendMotorDrive(addr, reg, vset, data)
     local vdata = bit32.bor(bit32.lshift(vset, 2), data)
     write_i2c_command(addr, reg, vdata)
end
```

速度 speednum と方向 tr で DRV8830 をコントロールする関数を以下のように記述します。速度は 0 ～ 200 の間で、方向は 0: 右方向または 1: 左方向として入力します。

速度は、0 ～ 200 を DRV8830 の仕様に合わせて 58 段階（6 ～ 63）に変換します。速度 10 以下の場合は停止させます。

DRV8830 の I²C アドレスは前述の通り 0xC8 と設定しましたが、Lua 機能の I²C 機能では 7bit アドレス表現として指定する必要があるため、1 ビット右シフトした 0x64 を指定します。

```
function controlSpeed(speednum, tr)
     local tr_reg = 0x00
     if(speednum < 0) then return end
     if(speednum > 200) then return end
     if (tr == 1) then tr_reg = 0x01
     else              tr_reg = 0x02
     end
     if(speednum < 10) then
             sendMotorDrive(0x64, 0x00, 0x00, 0x03)
     else
             sendMotorDrive(0x64, 0x00, (speednum/10)*3+3, tr_reg)
     end
end
```

最後に、I²C デバイスの初期化をして、controlSpeed 関数に任意の速度と方向を指定して呼び出すことで、鉄道模型の車両を動かすことができます。以下の例では最高速度 200 と方向 1: 左方向を指定しています。

```
res = fa.i2c{ mode="init", freq="100" }
controlSpeed(200, 1)
```

この仕組みを拡張して、例えば共有メモリの先頭4バイトに、速度 000 ～ 200 と方向 0/1 を記述して、これを常時読み込んで controlSpeed 関数を呼び出すようにすれば、共有メモリを書き換えるだけで、モータードライバを制御して、鉄道模型の車両を自由に動かすことが可能になります。

[コラム] I²C 機能と GPIO 機能の併用は可能か？

ところで、前述の通り Lua 機能の I²C 機能は、GPIO 機能のアドレス割当 0x01 と 0x02 を利用していると推測されます。一方で、FlashAir の GPIO ピンは5本が利用可能ですが、I²C 機能で使用していないと思われる残り3本の GPIO ピン（SD カードのピンは、それぞれ 0x04：8 番ピン、0x08：9 番ピン、0x10：1 番ピン）は I²C 機能と併用可能なのでしょうか？

結論から言うと、単純な併用はできないようです。

I²C 機能のデバイス初期化時に、8 番ピンと 1 番ピンは Low に、9 番ピンは High になるようです。また、デバイス初期化後に GPIO 機能を用いるとデバイス接続が保てないようなので、I²C 機能が継続利用できなくなります。再利用するにはデバイス初期化を再度行う必要があります。そのため、GPIO 機能と I²C 機能を併用する場合は、この挙動を認識した上で利用する必要があります。

[コラム] iSDIO 機能による共有メモリアクセスが廃止？

FlashAir W-04 はさまざまな機能追加が行われましたが、iSDIO 機能の共有メモリアクセス（読み書き）に用いていた CMD48、CMD49 がサポートされなくなりました。そのためマイコンなどと FlashAir を接続して iSDIO 機能で共有メモリを読み書きできなくなりました。代替方法はさまざま考えられますが、I²C 機能でマイコンと通信して共有メモリの情報をやり取りすることも1つの方法と考えられます。

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/sample4.zip

W-04 でも SPI Master

村口

はじめに

同人誌（初代）では、FlashAir W-02 の GPIO 機能を SPI Master として使用し 35[分] かけて液晶に画像を表示させました。同人誌 2 では、W-03 の Lua スクリプトの PIO 機能により、7.5[秒] に短縮しました。さらに、同人誌 3 では、Lua 組み込みの SPI 機能によって 1.0[秒] に短縮しました。今回は、FlashAir W-04 に、日本語フォントを組み込んで、W-03 と W-04 のレンダリング速度のベンチマークを行います。

構成

表示装置には、前回と同様に SPI 接続の 100x32 ドットのモノクロ液晶を使用しています。Lua 機能を使用して、W-04 内に保存した「ビットマップフォント」と「テキストファイル」を読み込んで、順次表示します。

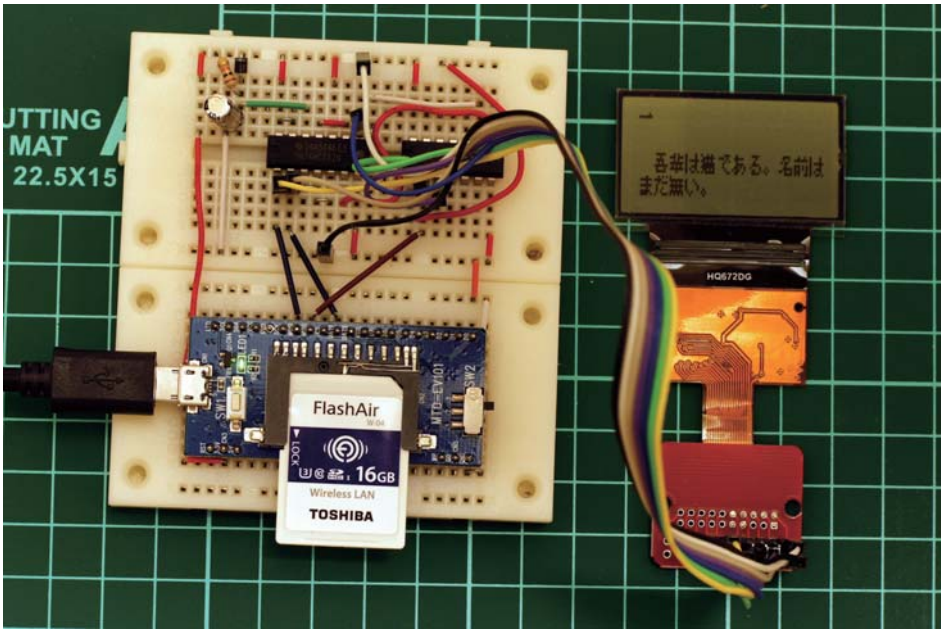


図 1: 夏目漱石の「吾輩は猫である」を表示

フォントについて

フォントは、ライセンスフリーの美咲フォントを使用しました。美咲フォントは、全角 1 文字が 8x8 ドットで表現されていますが、ほとんどの文字は 7x7 ドットに収まっています。字間や行間を詰めても、上下左右に余白ができ読みやすいという特徴があります。BDF 形式で配布されている美咲フォントは、JIS コード（ISO-2022-JP）のビットマップです。これを加工して、漢字コードの整数倍の位置にシークした位置から表示方向に合わせたビットマップデータを読み込めるように下準備しておくと Lua から扱いやすくなります。

ベンチマーク

「吾輩は猫である」の 1 章のレンダリング時間を、W-03 と W-04 で比較した結果を下記に示します。

表 1: W-03 と W-04 のレンダリング時間

	W-03	W-04
レンダリング時間	300[秒]	52[秒]

今回、作成した Lua スクリプトは、W-03 でも W-04 でも動作するため、同一条件での比較になります。W-04 は、W-03 に比べて、約 6[倍] 高速化しており、大きく改善していることがわかります。

さいごに

ハイエンド向け W-04 で、約 6[倍]（W-03 比）高速化しました。世代を重ねるごとに、どんどん高速になりますね。今後に期待です。

村口

最近、ミラーレスカメラに、古いレンズを組み合わせて、スナップ写真を撮っています。

マニュアルフォーカスで、一枚一枚に時間をかけると、不便の中にも楽しさがあることに気付きます。

電子工作の楽しさの本質も、不便を克服する過程を楽しめるところにあるのではないでしょうか。

FlashAir で WebSocket

たぞえ

FlashAir に WebSocket が搭載予定！ということで、簡単に使用例を紹介します。WebSocket はリアルタイムに双方向通信を実現するための通信規格で、ゲームやチャット、オンライントレードなどの高いレスポンス性能が必要な分野で使われています。

とりあえず Hello World

FlashAir に搭載されるのはクライアントライブラリなので、WebSocket 通信を行うためには通信相手のサーバが必要です。今回は node.js[1] のライブラリ ws[2] を利用してサーバを立てます。node.js は windows, mac, linux 等で利用可能ですが、ここでは windows を想定して進めます。

node.js の環境構築と ws のインストール

node.js の公式ページ[3] から windows 向けインストーラーをダウンロードして、案内に従ってインストールします。正常にインストールが完了するとコマンドプロンプトで npm と node コマンドが利用可能になります。

```
> npm -v
5.0.3
> node -v
 v8.1.4
```

次に開発用ディレクトリを作成し npm コマンドで ws をインストールします。

```
> mkdir ws-hello
> npm install ws
```

ws-hello 配下の node_modules フォルダに ws と依存ライブラリがインストールされます。

サーバソースコードの作成と実行

先ほど作成した ws-hello ディレクトリに下記のような内容の app.js を作成します。

```
const WebSocket = require('ws');//ws ライブラリをインポート
const wss = new WebSocket.Server({port: 8080});
wss.on('connection', function connection(ws) {
 ws.on('message', function incoming(message) {
   console.log( ' %s', message); // 受信したメッセージを表示
 });
 ws.send('hello'); // connection 確立時に hello と送信
 console.log(' hello');
});
```

1 サーバサイドで動作する JavaScript 環境

2 https://github.com/websockets/ws

3 https://nodejs.org/ja/

node コマンドでサーバを起動します。これで、ws://{windows のローカル ip}:8080 にアクセスすることで websocket 通信が可能になります。

```
> node app.js
```

(3)　FlashAir 側の Lua スクリプトを実行

FlashAir に下記のような Lua スクリプトを作成し、/SD_WLAN/CONFIG に LUA_RUN_SCRIPT パラメータ[4]を追加します。ip は各自書き換えてください。

```
sleep(60000) - WLAN 接続を待機
fa.websocket({mode="open", address="ws://{サーバ ip}:8080"})
res, type, payload = fa.websocket({mode="recv"})
if payload == "hello" then
  res = fa.websocket({mode="send", type=1, payload="world"}) - 応答
end
```

FlashAir の電源を入れると script が実行され、成功した場合下記のようなログが表示されるはずです。

```
C:¥Users¥tazoe¥ws-hello>node app.js
hello
world
```

公開されている WebSocket サーバに繋いでみる

既にネット上に公開されているサービスに FlashAir を繋げるといった利用方法も可能なはず・・・。例えば下記のような API が WebSocket サーバとして公開されています。

- Slack[5] - Real Time Messaging API
- Coinchecck[6] – WeSocket API

FlashAir を Slack bot にしたり、ビットコインの値段を監視させたりできる･･･はずだったのですが、実はうまく行きませんでした。Slack の方は websocket uri が長すぎる[7]のが原因で失敗したようです 。~~誰だ、実装したのはといいたいところですが、自分でした。すみません。~~・・・・・という感じですが、公開された際は試していただけると幸いです。

4　https://flashair-developers.com/ja/documents/tutorials/lua/2/
5　https://api.slack.com/rtm
6　https://coincheck.com/ja/documents/exchange/api#websocket
7　URL の host が 100 文字、path が 100 文字という制約があります

たぞえ

組み込み向けに WebSocket のライブラリ書いたり、golang でサーバ実装したり、vue.js でフロントエンド書いたり色々なことをしています。

FlashAir IoT Hub 爆誕

南

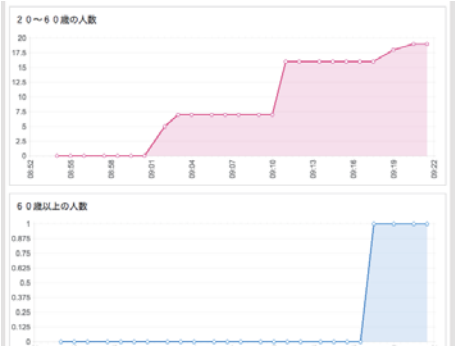
同人誌 3 でご紹介した FlashAir IoT Hub (Beta) が無事に一般公開[1]されました！今回は FlashAir IoT Hub を使ってできることの概要をまとめてみたいと思います。

簡単セットアップ

IoT Hub にアカウントを作って、サンプルスクリプトとアクセストークンを FlashAir に保存し、/SDWLAN/CONFIG をちょちょいっと編集すれば、すぐに IoT Hub にデータが表示されます。うまくいかない時は FlashAir が STA モードで無線 LAN に正しく接続されているか確認してみてください。詳細な手順はチュートリアル[2]を参照してください。

すぐに使える主要機能

計測値



iothub.addMeasurement 関数を呼ぶだけで、最大 5 種類の値をアップロードしてグラフ化することができます。直近の値を CSV ファイルとしてダウンロードすることもできます。

GPIO

iothub.startPioUpload 関数を実行しておけば、iothub.runJob 関数が実行されたタイミングで GPIO 入力値 (High/Low) を IoT Hub にアップロードすることができます。また、GPIO 出力値を IoT Hub から遠隔制御することもできます。L チカ！

スクリプト実行

FlashAir に保存した任意の Lua スクリプトを、IoT Hub から遠隔実行することができます。JSON 形式で任意の引数を渡すこともできます。SPI や I^2C（W-04 から対応）を制御してみると面白いかもしれません。

1 https://iot-hub.flashair-developers.com/
2 https://flashair-developers.com/ja/documents/tutorials/iot-hub/

画像アップロード (New)

画像アップロード機能が追加されました！サンプルスクリプトの upload_image.lua を LUA_SD_EVENT に設定すれば、撮影した画像がすぐにアップロードされます。W-04 を使えばアップロードも高速に！？

CSV アップロード (New)

CSV アップロード機能も追加されました！FlashAir に書き込まれた CSV ファイルをアップロードして、IoT Hub でグラフ化することができます。なんと 100 列まで対応！

汎用ファイルアップロード (Coming soon)

画像や CSV ファイルだけでなく、動画ファイルなど任意形式のファイルをアップロードできるようになります！アップロードされたファイルは IoT Hub の画面から簡単ダウンロードできる予定です。

IFTTT 連携

IFTTT Maker Webhooks を使って（非公式に）IFTTT と連携できます。アップロードされた画像ファイルを Dropbox などのオンラインストレージに保存したり、SNS に投稿したりすることができます。計測値が閾値を上回ったり下回ったりした時に通知やメールを飛ばすこともできます。いつかは IFTTT の公式サービスになりたいなあ。

おわりに

FlashAir IoT Hub は API[3] を公開しています。API を使って独自アプリやサービスを開発すれば、IoT Hub のポテンシャルをフルに引き出すことができます。

FlashAir IoT Hub は私たちにとっても本当に未知の試みです。皆さんに喜んでいただけるものにするために、ご意見・ご要望があればどしどしお寄せください！

3　https://iot-hub.flashair-developers.com/ja/api.html

南（@Nang_JP）

Web 系エンジニア。なにやら今時の若いもんは Vue.js らしい。研究所に勤めながら FlashAir IoT Hub の企画的な仕事をしていますと言ったら、見学に来ていた学生さんが変な顔をしていた気がしました。

FlashAir IoT Hub を始めよう！！

寺田 賢司

突然ですが、FlashAir IoT Hub をご存知でしょうか。私の周りでも「知ってはいるけど利便性がよくわからない」という方が大勢います。IoT Hub というと他のシステムやアプリとの連携を行うだけと思いがちですが、実は FlashAir IoT Hub には連携機能以外にもそれ自身に様々な機能を持っており、それらを無料で使用することができます。そこで今回は「FlashAir IoT Hub ってこんなに便利なんだ！！」というのをわかって頂くため、皆様と一緒に動体検知カメラを作成してみたいと思います。

構築環境について

- Raspberry Pi 2 Model B
- FlashAir W-04
- C270m(カメラ)
- GH-CR45UHK(カードリーダー)

FlashAir IoT Hub を使ってみよう！！

まず始めに FlashAir IoT Hub への登録を行います。詳しい説明は FlashAir Developers[1] に記載しているため省きますが、作成するにあたっての変更点などを説明していきます。

FlashAir IoT Hub へのアカウント登録後、FlashAir の登録を行うため、" アクセストークン " や " スクリプト " を FlashAir IoT Hub からダウンロードして FlashAir へコピーし、"upload_image.lua" を下記のように変更します。強調された箇所は変更、又は追加された箇所となります。

```
…
last_dirname = "/DATA"
fpath = "/DATA"
…
```

次に FlashAir に "DATA" フォルダを作成し、CONFIG ファイルを下記のように設定します。<> の箇所は環境に合わせて変更してください。

1 https://flashair-developers.com/ja/documents/tutorials/iot-hub/1/

```
[WLANSD]

DHCP_Enabled=NO
IP_Address=< 例 )192.168.0.100>
Subnet_Mask=< 例 )255.255.255.0>
Default_Gateway=< 例 )192.168.0.1>
Preferred_DNS_Server=< 例 )192.168.0.1>
Alternate_DNS_Server=< 例 )192.168.0.2>

[Vendor]

CIPATH=/DCIM/100__TSB/FA000001.JPG
APPMODE=5
APPSSID=< 例 )ssid>
APPNETWORKKEY=< 例 )password>
VERSION=F15DBW3BW4.00.00
CID=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PRODUCT=FlashAir
VENDOR=TOSHIBA
APPAUTOTIME=0
LUA_SD_EVENT=/upload_image.lua
```

[WLANSD] セグメントを追加することにより FlashAir の IP アドレスを固定しています。"LUA_SD_EVENT=/upload_image.lua" は FlashAir に対し書込みイベントがあった場合、実行する lua スクリプトを指定しています。

ここまで完了したら FlashAir を再起動 (抜き差し) してください。同じ LAN 環境に接続されている PC などのブラウザを起動し、[WLANSD] セグメントで設定した IP アドレス (192.168.0.10) を入力、FlashAir の画面が表示されれば接続確認は完了です。

次に FlashAir IoT Hub との接続確認を行います。あまりサイズの大きくない画像ファイルを作成した "DATA" フォルダにコピーしてみてください。少し経つと FlashAir IoT Hub 画面が切り替わり " イメージ " タブを押下するとコピーした画像が表示 (図 1) されれば成功です。

図 1: IoT Hub への転送確認

Raspberry Pi を使って動体検知カメラを作成しよう！！

まず始めに Raspberry Pi に Raspbian をインストールします。インストール方法は解説しているサイトがたくさんありますので、検索してみてください。Raspbian をインストール後、Raspberry Pi Configuration ダイアログを表示し、Interface タブを押下後、ssh と camera の使用を許可するように設定し、設定した FlashAir と Web カメラを Raspberry Pi に接続し、認識されていることを確認します。

次に Raspberry Pi の IP アドレス固定するため、/etc/dhcpcd.conf に以下を追記します。

```
…
interface eth0
static ip_address=<例）192.168.0.11/24>
static routers=<例）192.168.0.1>
static domain_name_servers=<例）192.168.0.1>
```

dhcpcd を再起動し、Raspberry Pi のパッケージ更新を行います。

```
> service dhcpcd reload
> apt-get update
> apt-get upgrade
```

撮影アプリのインストールを行います。撮影アプリは色々あるのですが今回は多機能な撮影アプリ、Motion(http://lavrsen.dk/foswiki/bin/view/Motion/WebHome) を使用します。

```
> apt-get install motion
```

次に Motion アプリの /etc/motion/motion.conf の以下の項目の値を変更します。

```
logfile /tmp/motion.log
width 480
height 320
output_pictures best
ffmpeg_output_movies off
target_dir /media/pi/XXXX-XXXX/DATA
stream_localhost off
```

設定変更後、Motion アプリを起動します。

```
> motion -n
```

同じ LAN 環境に接続されている PC などのブラウザを起動し、"dhcpcd.conf" で設定した IP アドレスにポート番号を追加し (192.168.0.11:8081) 入力、カメラの撮影画像が表示 (図 2) されれば Motion カメラの起動確認は完了です。

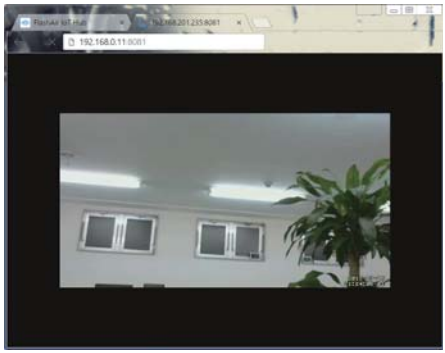


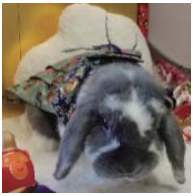
図 2: Motion アプリ起動確認

次にカメラの前で手を振ってみてください。少し経つと FlashAir の "DATA" フォルダに JPEG 画像が作成されているでしょうか。作成されている場合、ブラウザで FlashAir IoT Hub を開き、" イメージ " に撮影された写真が表示 (図 3) されれば完成です。

図 3: 撮影写真の転送

ずいぶん駆け足の説明となってしまいましたが、いかがだったでしょうか。今回は動体検知カメラとして作成しましたが、Motion アプリを使用し、タイムラプスで撮影した画像などもアップロードできますし、Raspberry Pi に温度センサーや照度センサーなどをつけて FlashAir IoT Hub でグラフを確認するなども面白いかと思います。

皆様のアイデア次第で色々できる FlashAir IoT Hub を今後とも宜しくお願い致します。



寺田 賢司

家でウサギを飼っているのですが、私自身はウサギアレルギーのためあまり近寄れません。せめて画像だけでもと思い、このシステムを思いつきました…。

TOY Board で簡単 IoT をはじめよう！

株式会社マクニカ　矢野 均　大松 良司

みなさん、こんにちは！マクニカの矢野です。ここでは FlashAir を使った小型な BLE-無線 LAN ゲートウェイ基板（TOY Board）をご紹介させて頂きます。

TOY Board のコンセプトは FlashAir を刺すだけで BLE から取得したセンサー情報を「データ蓄積」し、「クラウド」へデータをアップロード出来る基板です。

今回、TOY Board 誕生の背景と接続事例についてご説明させて頂きます。

マクニカ (Mpression) について

マクニカは、エレクトロニクス、情報通信業界をリードする国内外の大手電機・電子機器メーカーをはじめとするお客さまに、半導体、電子デバイス、ネットワーク関連機器、ソフトウェアなどの高付加価値商品とサービスを提供しています。

1972 年の創業以来、マクニカは技術サポートに力を注ぎ、技術の深化を推し進めてきました。業界に先駆け、技術支援重視の事業スタイルを確立。「技術商社」という新しい商社像を打ち立てました。

マクニカでは、半導体デバイス・ネットワーク製品などを技術サポートと供に提供することに加え、そこで得たノウハウを詰め込んだ、より高度で、使いやすく、手堅い、サービスおよびシステムレベルソリューションを開発し提案しています。

「Mpression（エムプレッション）」はお客様のイノベーション、課題解決を加速するオリジナル技術ブランドです。今回の TOY Board も Mpression の中の一つのブランドとなります。

図 1: Mpression ロゴ

TOY Board 誕生の背景

2016 年 11 月初旬、高田さん[1]、余熱さんとの打ち合わせ時に BLE で取得したセンサーデータを FlashAir からクラウドにアップロードするゲートウェイ基板を作りたいとお話を頂きました。マクニカではマイコン、BLE モジュール、周辺部品を提案と基板の受託開発も行うことが可能である為、Mpression として進めさせて頂きました。

開発着手までに時間が限られる中、展示会等で多忙な高田さん、余熱さんに時間確保頂く為、ET2016 の会場での仕様決めや緊急ミーティングなども実施頂きました。

そのような状況の中、何とか仕様確定、ハード設計、ソフト設計の開発着手を行う

1　高田さんについては FlashAir 同人誌 1 ～ 3 をご参照ください。
https://flashair-developers.com/ja/documents/books/

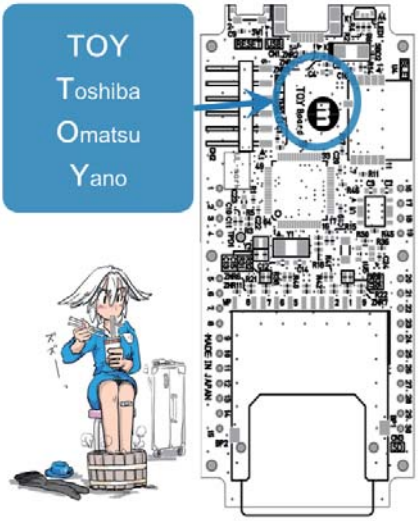


図 2: TOY Board アートワーク

事が出来ました。基板開発を行う最中、Mpressionロゴとともに基板ネームも印字しようという話になりました。

当初 FlashAir ゲートウェイと名付けようと考えていましたが、それだけでは面白味がないと思い、（当時）東芝さんとマクニカで作る基板にちなんだ名称にしたいと考えておりました。

そこで、技術担当の大松と案出しをする中、東芝さんの “T”、大松の “O”、矢野の “Y” から TOY と名付けました（図 2）。TOY Board という名称に決定して以降、2017 年 2 月の FlashAir Developers Summit や IoT/M2M 展 2017、Interop Tokyo 2017 と各種イベントでご紹介を続けている次第です。

後で気づいたことですが、TOY Boad の “T” と ”Y” は高田さんの “T”、余熱さんの “Y” でもあって無理やりではありますが、遊び心のある名称に出来たと思います。

下記より TOY Board 接続事例は TOY の名前にちなんで簡単に楽しく IoT をはじめられる事を意識して技術担当大松よりご説明させて頂きます。

大松さん、バトンタッチしますね！ TOY っと！笑

TOY Board の基本動作

“ つぉっ ” とバトンを受け取りました技術兼ゴーストライターの大松です。

ここからは、TOY Board の基本動作を説明します。まず、TOY Board は、IoT 向けのシンプルな BLE- 無線 LAN ゲートウェイ基板です。世の中の BLE 通信機能を備えた IoT センサーデバイスからデータを受信して、FlashAir にデータを保存する事ができます。

FlashAir の無線 LAN 機能と「FlashAir IoT Hub」を使う事で、データの可視化もできます。ま、ま、まっさかの IoT ソリューションとして活用できるのです。

図 3 は、“ 世の中にありそうでなかった ”TOY Board です。意外と小さく手のひらサイズ！

図 4 の TOY Board のデモ構成を少し説明します。IoT センサーは、オムロン社の環境センサーを使わせてもらっています。環境センサーは、温度、湿度、照度、気圧、騒音、UV の周辺環境をまとめてセンシングできます。

様々な用途に使えるので、TOY Board と相性ピッタリなのです。素晴らしい！

オムロンさんありがとう！色々な人と連携して面白い事を考え中です。そしてアイデアも絶賛募集中ですので、矢野をフォローしてみてください !!

図 3: これが TOY Board だ！

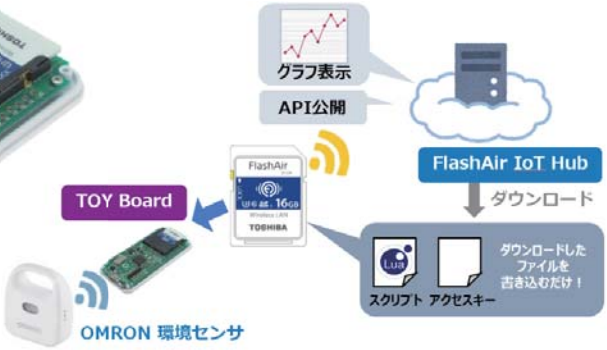


図 4: TOY Board デモ構成

TOY Board の応用例　～奮闘記～

TOY Board の反響も良く、様々な機器につなげて、新しい用途を奮闘中です！！

現在、 3つの活用事例を進めています。FlashAir を使って、世の中をもっと便利に！

もっと快適に！を目指して企画、開発、販売をしています。

目新しいセンサーを探し歩き、センサーのラインナップを拡大中、今後は、クラウドサービスを強化して、もっと FlashAir を広めていきたいという熱い想いです。

もはや、ゴーストライターの枠を超えたスーパーゴーストライターに変身しつつあります。

センサーのコンフィグレーションツールとして活用 <実装中>

マクニカにて IoT ワンパックモジュールを開発中です。IoT ワンパックモジュールは、モーションセンサー、環境センサー、測距センサーを評価する事ができます。お客様の目的に応じてセンサーの入れ替えができるボードとなっているのです。

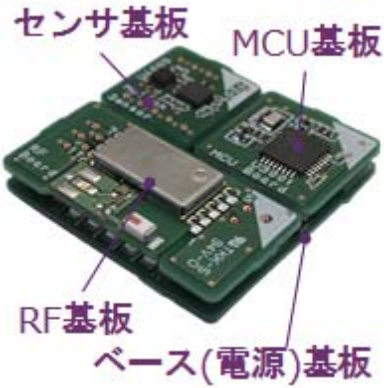


図 5: IoT ワンパックモジュール

TOY Board は、IoT ワンパックモジュールのセンサーをコンフィグレーションできるツールとして実装中です。FlashAir に入っているコンフィグレーションファイルを編集する事で実現できます。

これで、センサーを細かくチューニングして評価をしたいというニーズに応える事ができます。また、大量のセンサーデータを FlashAir に貯める事ができるので、センサーを評価したい方にピッタリかもしれません。

人、物の位置検知・管理に中継器として活用 <実装中>

Beacon センサーと TOY Board とクラウドをつなげて、人、物の位置を検知して、もっと便利に活用できるように実装中です。様々な業界から器材の管理が大変という声があり、位置、管理ができると便利になると考えています。

また世の中の働き方が変わってきていますので、人の正確な稼働状況を把握する事で、生産性向上のきっかけにならないか、労働者と経営者がハッピーになれるツールの一つになると私もハッピーです。

生体センサー (脳波、心電図) の付加価値として活用 <企画検討中>

脳波、心電センサーと TOY Board を接続して、様々なシーンで世の中の役に立てるソリューションを企画しています。これは、人的被害が拡大しないように予期せぬ事故を防ぐ事や地方の方がもっと快適に生活できないかを考えています。

在宅介護、医療の補助として FlashAir を活用できたらステキなことです。

脳波、心電センサーは、人のストレス、集中の度合い、眠気の度合い、人の認知の特定などに活用できるので、生活、働く事に困っている人を助けられる可能性を感じています。

おわりに

またまた、登場しました矢野です。TOY Board 誕生から接続事例についてご説明をさせて頂きました。FlashAir をつかった TOY Board ! いろいろな可能性を感じて日々ブラッシュアップ出来ないか、面白そうなセンサやソリューションをご提案できないか考えています。ご意見やご要望がございましたらご連絡ください !

矢野 均 (@kin0719)

半導体商社で営業を担当。余熱さんとの打合せの中で FlashAir の面白さ、IoT への無限の可能性を感じています。

FlashAir Developers Summit や Interop でセミナーも担当させて頂きました。

大松 良司

半導体商社で技術を担当。表にでるゴーストライターという一面もあります。展示会の説明員、日々のお客様に TOY Board と FlashAir の魅力を伝える事に楽しさを感じています。FlashAir の機能を勉強中です。

冷蔵庫ドアの見守りセンサーを作ろう！

余熱

電気ポットを使うとスマホに通知がいく というシステムがありますが、似たようなノリで冷蔵庫のドアを開けたらスマホに通知がいくシステムが作りたい！そんな時、FlashAir と TOY Board、環境センサ、それから FlashAir IoT Hub を使えば簡単に実現可能なんです！というわけでチュートリアル形式で説明しようと思います。FlashAir IoT Hub と IFTTT の連携のチュートリアル[1]も適宜参照してください。

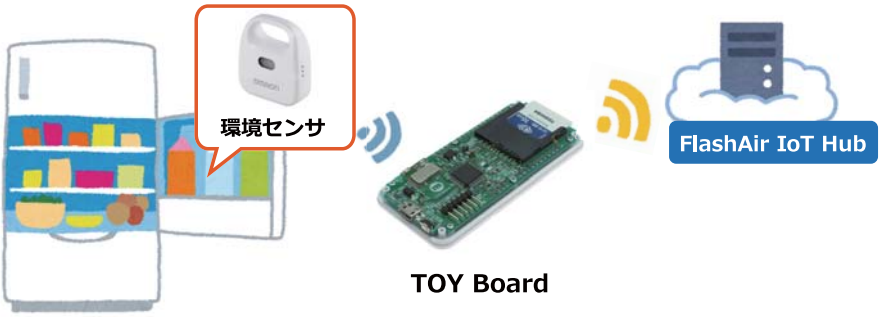


図 1: システム概要

さっそく作ってみよう！

環境センサ

今回はオムロン製の「環境センサ[2]」を使用します。環境センサは、**温度、湿度、気圧、照度、紫外線、音圧、不快指数、熱中症指**数を取得することが可能で、Bluetooth Low Energy を用いて通信を行います。環境センサには電源スイッチが無く、CR2032 ボタン電池を入れることで約 6 か月使用できます。

環境センサには専用のスマホアプリが用意されており、アプリを用いて環境センサの設定を変更することが可能です。今回は測定間隔を 10 秒に設定します。

TOY Board

TOY Board のサンプルファームウェアを用いて環境センサに接続することが可能です。今回は、温度、湿度、照度、電池残量をテキストファイルとして保存することにします。「**25.75,35.64,3,283**」というように値をカンマで区切って4つの値を格納したテキストファ

1 https://flashair-developers.com/ja/documents/tutorials/iot-hub/8/#add_applet_measure
2 2JCIE-BL01 http://www.omron.co.jp/ecb/products/sensor/special/environmentsensor/

イルを書き込むファームウェアを作成しました。TOY Board へのプログラムの書き込み用の治具は **ST-Link V2** を使用し、ソフトウェアは **ST-LINK utility** を使用します。

FlashAir

FlashAir を挿入して TOY Board を起動します。TOY Board が環境センサと接続すると LED の点滅が青色から白に変わります。一度 FlashAir を取り出し PC で読み込むと concentscale フォルダの下に Env_D7FAEE65C354.txt のような名前のテキストファイルが作成されているはずです。このファイル名は接続する環境センサによって異なります。テキストファイルの中身はカンマ区切りで 4 つの数値が保存されています。

続いて FlashAir IoT Hub から**アクセスキー**と **Lua スクリプト**をダウンロードして書き込みます。テキストファイルから温度、湿度、照度、電池残量を抜き出し、FlashAir IoT Hub にアップロードするため、**bootscript.lua** を下記のように変更する必要があります。読み込んだテキストの操作は検索関数 (**string.find**) や切り出し関数 (**string.sub**) を使います。IoT Hub へのアップロードは **iothub.addMeasurement** という関数を使います。

```
local iothub = require("iothub")

-- iothub.startPioUpload(0x03)
-- iothub.stopPioUpload()

local cnt = 0

while(1) do
  iothub.runJob()

  file = io.open("/consentscale/Env_D7FAEE65C354.txt", "r")
  for line in file:lines() do
    comma_1 = string.find(line, ",")
    tmp = tonumber(string.sub(line, 1, comma_1-1))
    line2 = string.sub(line, comma_1+1, string.len(line))

    comma_2 = string.find(line2, ",")
    hum = tonumber(string.sub(line2, 1, comma_2-1))
    line3 = string.sub(line2, comma_2+1, string.len(line2))

    comma_3 = string.find(line3, ",")
    ill = tonumber(string.sub(line3, 1, comma_3-1))
    bat = tonumber(string.sub(line3, comma_3+1, string.len(line3)))

    iothub.addMeasurement({tmp, hum, ill, bat, cnt})
  end
  file:close()

  cnt = cnt + 1
  sleep(5000)
  collectgarbage("collect")
end
```

FlashAir の **WLAN_SD/CONFIG** を下記のように変更します。WLAN_SD フォルダは隠しフォルダなので注意してください。

```
[Vendor]

CIPATH=/DCIM/100__TSB/FA000001.JPG
APPMODE=5
APPNETWORKKEY=********
VERSION=FA9CAW3AW3.00.01
CID=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PRODUCT=FlashAir
VENDOR=TOSHIBA

MASTERCODE=XXXXXXXXXXXX
LOCK=1
APPSSID=flashair
APPNAME=flashair
LUA_RUN_SCRIPT=/bootscript.lua
```

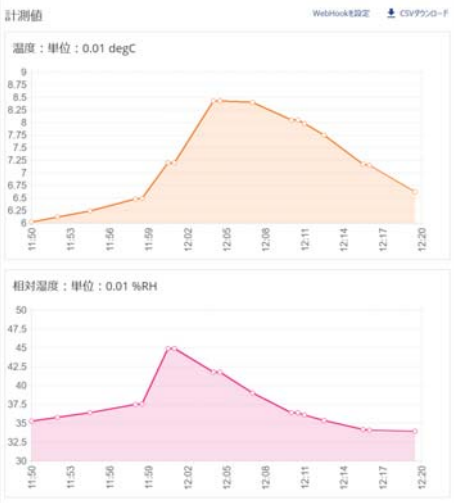


図 2: FlashAir IoT Hub の画面

FlashAir IoT Hub

間もなく FlashAir IoT Hub の画面にグラフが表示されはじめます (図 2)。10 時間ほど動作させ、FlashAir IoT Hub から CSV ダウンロードを行いグラフ化してみた (図 3) のですが、温度はおおよそ 5 ～ 10[℃]、湿度は 30 ～ 40[%] で遷移していました。冷蔵庫のドアを開けると湿度が 40%を超えるようなので、その値を Webhook のトリガにしてみます。

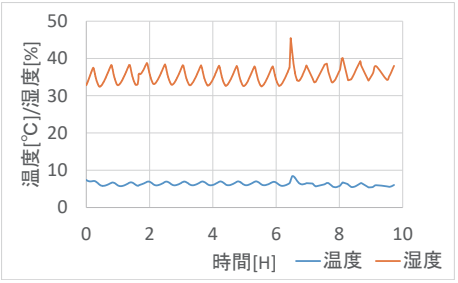


図 3: ダウンロードした CSV のグラフ

グラフが表示されている画面の右上の「**WebHook を設定**」をクリックすると図 4 のようなダイアログが開きます。[**event**] に「**Door_Open**」、[**key**] には後述する key

図 4: WebHook 設定

を入力し、通知の条件も入力したのち「**登録**」をクリックします。すると[**一覧**]にWebhook設定の一覧が表示されます。これでFlashAir IoT Hub側の準備は完了です。

図 5: IFTTT の key

IFTTT

IFTTTにログインし、Maker Webhooksのページを開き、「**Documentation**」をクリックすると、図5のようにkeyが表示されます。このkeyを前述のFlashAir IoT Hubの Webhook設定で入力してください。

つづいてIFTTTのAppletを作成します。My Appletのページから手順に従って設定を行うと、図6のようにアプレットを作成することが出来ます。今回はメールではなくTwitterに投稿としてみました。

図 6: IFTTT の Applet

動作

冷蔵庫のドアを開けると無事にツイートされていました(図7)。遠隔地の誰かを見守るシステムをとても簡単に構築することができました。

冷蔵庫の中以外でも、部屋に置いて不快指数や熱中症危険度をロギングするのも良いですね。通知もTwitter以外にも色々な方法が考えられます。

試作が簡単なのはFlashAirの大きなメリットの1つです。是非ともFlashAirを使って何か作ってみて頂ければと思います。

図 7: 無事にツイートされた模様

余熱 (@yone2_net)

FlashAir同人誌編集担当。FlashAirチームに配属になってから2年経ちました。引き続き頑張りますので応援よろしくです。

今回も社内外のサポーターの皆さまに寄稿頂きました。この場を借りてお礼申し上げます。ありがとうございました。

シンプルなリモートカメラから始める

「Take a picture」

こんにちは！図 1 は趣味の菜園で撮影したリモートカメラの画像です。キュウリとトマトを育てています。

iPhone のブラウザからボタンを押すと写真が表示されるシンプルなリモートカメラです。カメラの撮影は GR-LYCHEE (図 2) を使っています。SD ソケットに FlashAir がそのまま接続できます！ (図 3)

図 1: FlashAir List.htm の表示

GR-LYCHEE のスペック

GR-LYCHEE は ARM® mbed™ に対応しており、スペックは右表の通りです。2017 年 11 月に販売開始予定です。

搭載 CPU の RZ シリーズは容量の大きな RAM を内蔵しているため、Open CV で顔検出等もできます。若干機能は限定されますが、ユニークなカメラ画像にできます。

表 1: GR-LYCHEE のスペック

項目	スペック
CPU	RZ/A1LU
動作周波数	384MHz
ROM	8MB
RAM	3MB
カメラ有効画素数	640x480
カメラフレームレート	最大 60fps

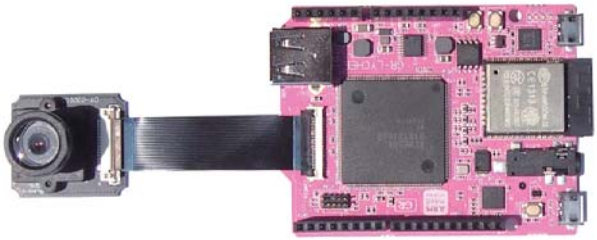


図 2: GR-LYCHEE

図 3: GR-LYCHEE
With FlashAir

リモートカメラのサンプル

今回のリモートカメラは FlashAir の共有メモリ 512 バイトの内の 1 バイトをトリガーとして写真を撮影し、FlashAir に写真を保存するだけという簡単な作りになっています。

Arduino ライクな「IDE for GR」(V1.01 以降) には、FlashAir Developers のチュートリアルやリモートカメラのサンプルが用意されています (図 4)。List.htm も Web に掲載し、サンプルにリンクを貼りましたので是非、お試しください。

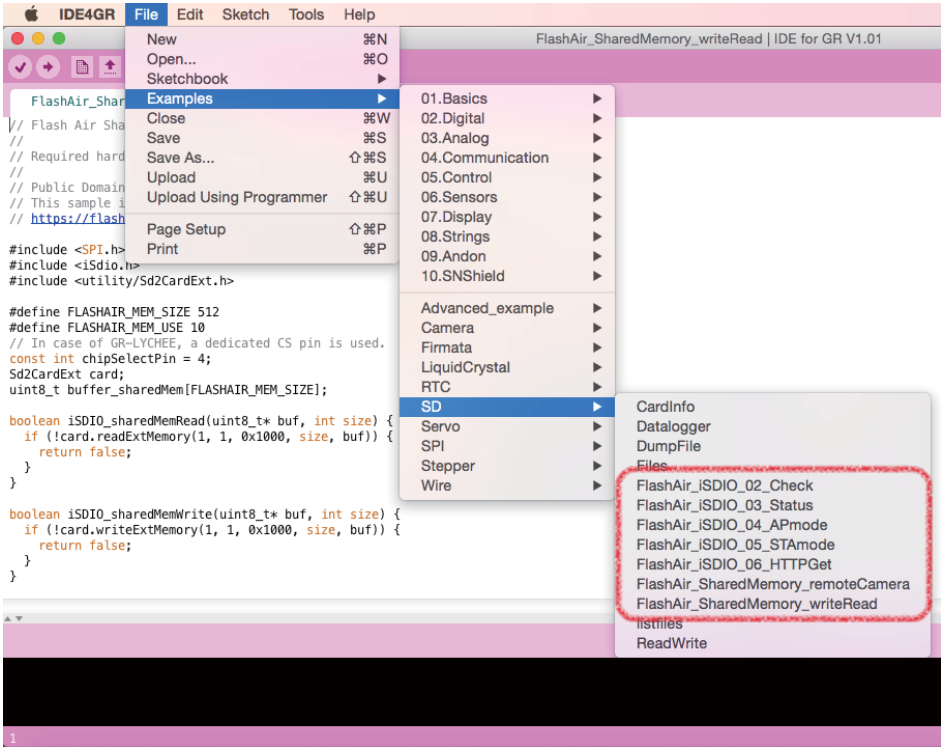


図 4: IDE for GR FlashAir のサンプル

Yuuki Okamiya (@Renesas_FUN)

GADGET RENESAS(略称がじぇるね) で企画、ソフト、Web やイベントなどを担当しています。一緒にものづくりを盛り上げて行きましょう!

がじぇるね : gadget.renesas.com

FlashAir は IoT の牛丼屋

Takehiro Yamaguchi

まずは吉報です。がじぇっとるねさすさんから GR-LYCHEE が登場します。

2017/4/22　GR-LYCHEE プロデューサーミーティングが開催され GR-LYCHEE β版とカメラが参加者に配布されました。β版の現時点では、がじぇるね Web コンパイラでの開発環境となっていますが、ARM® mbed™ 環境に対応予定となっています。発売時には OpenCV への対応にも期待したい所です。なぜ吉報かというと最近のボードは microSD 用ソケットが基板に実装されている場合が多く、あの手この手で FlashAir を使っていたのですが、なんとなんと GR-LYCHEE にはですね FlashAir がささるフルサイズのソケットが基板に実装されているんです。この驚きを表現するなら、けものフレンズのボス:ラッキービーストの耳が赤くなることを初めて知った時と同じ位の衝撃がはしりました。

私が初めて FlashAir を購入したのは W-02 です。GPIO を HIGH/LOW できた時

「かっこええやん」と、すっかり FlashAir の芸達者ぶりに惚れてしまい、今では IoT のひとつの手段として FlashAir を見ています。最近は LPWA や LTE などといった IoT の選択肢も増え、電力会社から届いたスマートメーターへのお取替えのお知らせを見ながら、ドイツが推進するインダストリ 4.0 の到来を身近に感じるようになりました。

では、生活の中の通信インフラを見ると、やはり WiFi 環境が一番身近な通信環境だということは事実です。

ここで言いたい。

まさに FlashAir は IoT の牛丼屋みたいなものなのです。

FlashAir IoT Hub の存在

「はやい・やすい・うまい」を見事に実現しているのが FlashAir IoT Hub です。

まだまだ進化を続ける FlashAir IoT Hub なのですが、IFTTT との連携もできるようになったと知り、早速使ってみました。

冒頭でご紹介した GR-LYCHEE での計測値を AS-289R2 プリンタシールドで印字しながら、FlashAir IoT Hub に計測値をアップロード。更にサーマルプリンタの DTR を監視し、もし感熱紙が無くなったら IFTTT を使って Twitter に「紙が無くなりました」とお知らせするものを作ってみました。

まず GR-LYCHEE で計測値を測定しながらサーマルプリンタで計測値を印字、DATA.TXT ファイルを作成します。DATA.TXT の 1 行目には計測値を、2 行目には紙切れ情報（ある場合は 1、無い場合は 0）を DATA.TXT ファイルに書き込みます。

例えば、計測値が 25 で、紙がある場合の DATA.TXT ファイルの中身は以下です。

```
25
1[EOF]
```

次に lua スクリプト bootscript.lua を作成します。DATA.TXT を読み込み、1 行目の計測値を value1 に、紙切れ情報を value2 としてアップロードするスクリプトです。

```
-- iothub.startPioUpload(0x03)
-- iothub.stopPioUpload()

local linecnt = 0
local v1 = 0
local v2 = 0
while(1) do
  iothub.runJob()
  file = io.open("/DATA.TXT", "r")
  for line in file:lines() do
    if linecnt == 0 then
      v1 = tonumber(line)
    end
    if linecnt == 1 then
      v2 = tonumber(line)
    end
    linecnt = linecnt + 1
  end
  file:close()
  iothub.addMeasurement({v1, v2})
linecnt = 0
  sleep(30000)
  collectgarbage("collect")
end
```

FlashAir IoT Hub 計測値 value1 に計測値が value2 に紙切れ情報が表示されます。

IFTTT との連携は、めっちゃ便利

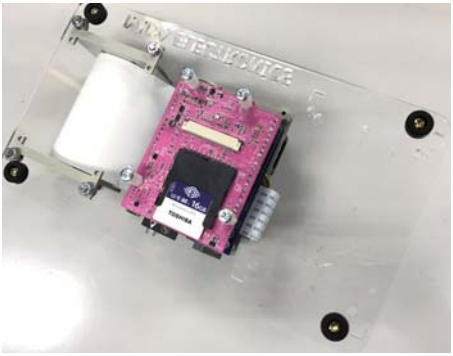
感熱紙の紙切れが発生した場合、IFTTT を使って Twitter に送信します。

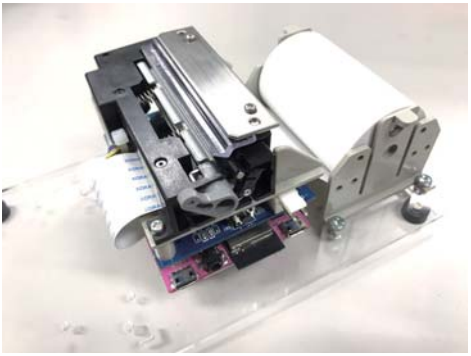
FlashAir IoT Hub の「WebHook を設定」で、IFTTT への送信条件を入力します。感熱紙の紙切れ情報を送信したいので、条件部に「value2」の値が、「<しきい値よりも小さい」とし、しきい値を「1」とします。FlashAir IoT Hub の設定は以上です。

次に IFTTT を設定します。IF [Maker Webhooks] THEN [Twitter] となるように設定します。[Maker Webhooks] の設定については、案内通りに入力すれば OK です。[Twitter] の設定は「Post a tweet」を選択し、Tweet text 部枠内に「紙が無くなりました」と入力すれば設定は完了です。

紙切れが発生すると Twitter にメッセージがでます。FlashAir IoT Hub が IFTTT に対応したことで計測値により、SNS や様々な WEB サービスとの連携が可能になりました。更に W-04 からは、共有メモリが増強され、WebSocket クライアントにも対応予定とのアナウンスがされています。

今後の FlashAir と FlashAir IoT Hub を利用した、IoT の活用事例に注目しています。





FlashAir で未来を拓く

そんな素敵な FlashAir を展示会やイベントなどで熱く語る方がいます。余熱氏です。

余熱氏から弊社製 PM-232 についてお問い合わせを頂きました。PM-232 という機器は RS-232C の通信ログを SD カードに保存する産業用途向けの機器で、主に PLC とプリンタ間の通信を記録する装置として利用されています。

SD カードが動作するので当然 FlashAir も動作するものと思っていたのですが、データ書き込み時のみ SD カードに電源を供給するという回路構成になっていて、FlashAir にデータを書き込めるのですが、肝心の無線 LAN が動作しないという、FlashAir から無線 LAN の機能を奪った、閃ソラさんの額に何本もの線が入りそうな仕様でした。

そこで、基板にパターンカットとジャンパーを施し FlashAir の動作を確認、基板のパターン変更もおこない、FlashAir 対応機器として準備を行っている所です。産業機器にも IoT が求められる今、FlashAir を利用することで幅がグッと広がります。このような通信ログを SD カードに保存する機器でも、機器間の通信内容をを監視しながら、FlashAir IoT Hub に情報をアップロードするといった事も可能になります。

私は IoT は暮らしを豊かにするものだと信じています。IoT の道路というべき通信環境が低価格で入手利用できるようになってきました。当然、道路ができればお店が並び、街ができます。FlashAir と FlashAir IoT Hub は、その街にある一軒の牛丼屋さんのような存在ではないかと思っています。

NADA ELECTRONICS,LTD.

Takehiro Yamaguchi（@nada_tokki）
AS-289R2 Thermal Printer Shield の開発担当者です。
ハードやソフトなどの商品開発業務を担当しています。
各種イベント等にもよく顔を出します。声をかけてください。
同人誌への投稿は初めてなので、うれしはずかしですね。

IoT 完全自立 定点撮影「AirLapse」

（株）タイプ・アール 高瀬 秀樹

AirLapse とは

AirLapse[1] は、定点撮影～動画（タイプラプス）作成までを、機材レンタル、設置工事を含めて、ワンストップで実現できるサービスとして提供している。

主な特徴

- **約 1800 万画素（5184x3456）の高解像度、高画質で広角撮影しているので、長期撮影後の編集でズームやパン等の映像演出が可能**
- **撮影状況を Web からほぼリアルタイムで確認可能**
- **Wi-Fi ルータやバッテリ、ソーラパネルを追加することで、電力供給もいらない、完全自立システムとして稼働が可能（オプション）**
- **最大 3 台の Web カメラ（SD カード内蔵）を実装可能で、高解像度静止画と、ドライブレコーダ的に使用できる動画録画のハイブリッド運用が可能（オプション）**
- **1 日 1000 円からサービス利用が可能（※最低契約期間 3 か月、工事費別途）**

開発の背景

2011 年 3 月 11 日に発生した東日本大震災で多くの街が壊滅的に破壊されその光景から復旧・復興には、10 年位の年月がかかると想像できた。

TBS テレビの知人からこの大規模な復興の様子を 4K 以上の高解像度で定点撮影して、遠隔で見られないかと相談された。

しかも、「設置予定場所の状況を考えると電力や有線回線が確保できない。そのような環境下で 10 年以上安定稼働させたい。」との要求である。この時点では、定点撮影は昔からある技術なので、世の中には何らかのソリューションがあるはずだと考えて、各方面にリサーチを始めたが、要求仕様に合致するものを見つける事ができなかった。

そこで、独自開発するしかないのだが、TBS の研究開発予算は、限られていたので弊社と共同開発という形で進める事となった。

1　http://airlapse.jp/

2013 年 12 月から陸前高田、2014 年 1 月から南三陸志津川町の撮影を開始して現在も日々撮影している。

その後、マサル工業様に独自ケースを、FlashAir 開発陣に技術協力を、お願いし共同開発を進め、2016 年 11 月に「AirLapse」サービスを開始することが出来た。

システム構成と基本動作

被災地に設置しているカメラの現在バージョンのシステム構成を説明する。

カメラ側

定時撮影は、独自開発のコントローラがカメラのレリーズ端子よりシャッターを切る。

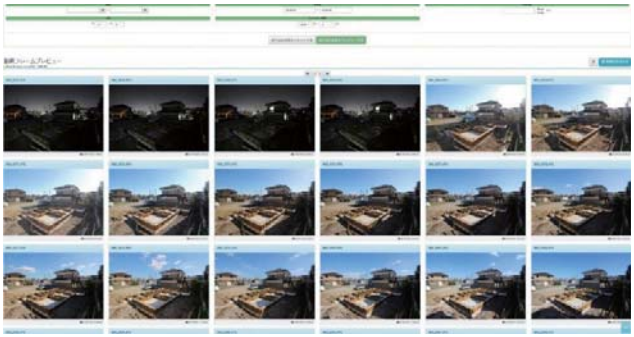
極力消費電力を抑えるために撮影画像データは、 1日1回まとめて伝送するため Wi-Fi ルータの電源制御も同じコントローラにより制御を行う。

クラウドへの転送は、FlashAir に実装した Lua スクリプトが SD-EVENT により実行されて、Wi-Fi ルータ経由で Microsoft Azure を使用したクラウドに転送される。

サーバー側

送られた画像データは、Azure Functions を使用して各種データをデータベースに格納すると同時に、サムネールの作成も行い Web 上で閲覧可能となる。

Web サイト上では、期間 / 時刻 / File サイズ / 絞り / シャッタースピードで絞込を行い、目的のサイズとフレームレイトを指定してタイムラプス映像の作成が可能となる。



開発中の様々な問題

Eye-Fi ＋ EOS の問題

開発当初は、FlashAir で Lua が使用できなかった為、Eye-Fi カードを使用していた。

撮影開始から毎日順調に送り続けてきたが、ある日を境に転送されなくなった。調査の結果、ファイル数が増えてくると、Eye-Fi カードが送信前処理として、送信したかどうかの確認をしている間に、EOS 側が送信開始を待ちきれずタイムアウトしてしまう事が判明した為、数か月おきにメンテに行く必要にせまられた。

2015 年 3 月 FlashAir W-03 発表

Lua が使用出来る W-03 の発表でようやく、Eye-Fi からの移行が可能となると考え、早速実験を開始した。ところが、この時点では FlashAir からの制御でカメラの電源を切る方法が無く、消費電力を考えると被災地では使えない事が判明、とは言え長期ランニングテストは必要なので、AC 給電可能な渋谷再開発の模様を撮影開始した。

2016 年 Eye-Fi モード

FlashAir 開発担当から、Eye-Fi モード実装テストが可能になったとの連絡を受けた。Eye-Fi モードを使用すれば、FlashAir からのカメラの電源制御が可能となるので、実験を開始した。当初発生したいくつかの問題をクリアしてもらい無事安定運用、現在全ての Eye-Fi は、FlashAir に取って代わった。

EOS の書き込み禁止問題

カメラは、当初から Canon の EOS シリーズを使用している。正常に転送が終了した素材や、転送の必要が無い時間帯に撮影された素材は、定時撮影後に実行される Lua スクリプトで自動削除を行っている。ところが撮影間隔が短く転送中に次の撮影を行うような場合、File を壊してしまう事が稀に発生する。

対策として、削除前に SD への書き込みを Soft 的に禁止した。PC 上でのテストで無事排他処理が出来ることを確認したので、実機で運用を開始した。ところが一か月くらい運用したところで File 転送をしなくなる問題が何度か発生した。そこで簡単な書き込み禁止プログラムを作成して、EOS の実機で実験したところ、なんと全く無視して書き込みをしてしまう。

どうやら EOS では、起動時に一度書き込み禁止の確認をした後は、書き込み時に再確認しないようで、FlashAir のように自発的に途中で書き込み禁止に切り替わる事を想定していないようだ。

その為、この排他処理に関しては対処療法で対応した中で、いまだ根本的な解決が図れていない。

今後の課題

サービス開始から、大中規模建築、住宅建築、農場、イベント等々様々な場所で撮影をさせていただいていて、お客様から「二度と撮れない貴重な映像をありがとうございます。」と言われるケースが多い。確かに更地から住宅が完成する様子や、畑の四季の移り変わりをただ「ぼーっと」見ているだけでも見飽きない。このような映像をより多くの方々に手軽に撮影するべくサービスの向上と拡大を目指している。

現在まだ、プロトタイプ運用で解決するべき問題も残っているが、量産化に向けてケースとコントローラの設計を進めていて、コントローラにも FlashAir を実装し、各種制御をクラウド経由で実現可能にする予定なので、機会があれば、また本紙面に寄稿させていただきたいと考えている。

（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 でスマートロック！

株式会社 DMM.com ラボ 野秋 拓也

私は学生時代からスマート家電に興味があり、Web から操作できるコーヒーメーカなどを作り続けてきました。去年、就職の為に上京し一人暮らしを始めたのでついに電子錠を開発しました。今回は私が実際に運用しているスマートロック[1]の作り方について解説していこうと思います。

材料

インターネットで DIY の電子錠の作例を探すと、サーボモータを RaspberryPi などで制御し、３Dプリンタで筐体を作る作例が多く存在します。もし３Dプリンタを個人所有していなくても弊社の３Dプリントサービスのように Web からモデリングデータを入稿し、家に完成品が届くサービスがあるので、誰でも初期投資なく始めることができます。とは言え初めての人が一度の設計で完璧なものを作るのは難しいですし、今回は試作ということで材料を買ってきて気軽に始められる工作にしました。材料 (図1) は以下のとおりです。

- FlashAir(W-03 以降)
- MicroUSB ケーブル
- ホットメルト
- Seria ごますり器
- HiLetgo L9110S(モータードライバ)
- リミットスイッチ × 2
- 三端子レギュレータ AZ1086H-3.3
- 秋月電子通商 ＳＤカード配線引出基板
- 抵抗 1k Ω (1/4W)
- TAMIYA 4 速クランクギヤーボックスセット

図 1: 材料たち

制作当時はＳＤカードスロットＤＩＰ化モジュールを使っていたのですが、記事化するにあたってＳＤカード配線引出基板のほうが USB 差込口を搭載しており、パターンを改造せずに利用できるので急遽用意しました。制御機器として FlashAir を選んだ理由としては GPIO と無線ＬＡＮマイコンを含めた値段の安さと、ノンプログラミングでも HTTP の API を標準搭載している点が速度感のある試作にはうってつけであり、気に入っている為採用しました。ごますり器は内鍵のノブを回す回転機構として採用しています。

1 https://youtu.be/5deLVK1l3IE

工作

まずは4速クランクギアーボックスを組み立てます。この製品は4つのギア比を組み替える事ができ、鍵のトルク要件に合ったギアを選択することができます。私の家の鍵だと１５４３：1のギア比がスピードとのバランスが良かったです。また、この製品にはクラッチ機構がついており、物理鍵によってシリンダーを回した場合にも空転するギミックをつけることができます。ただし、クラッチはモータの保護用に設計されているので、ギア比最大まで上げると物理鍵で回した時にモータまで回転が伝わらずに別の部分が空転して壊れます。このクラッチギアは意外と許容トルクが少なく、鍵を回す前に空転してしまうので、ホットメルトでバネの可動を制限してしまいましょう。

次にごますり器のネジを外し分解します。ネジの入っていた穴にシャフトを通したいのでドリルかキリで穴を拡張します。シャフトが通るようになったら穴に合わせてクランクアームを接着します (図2)。ごますり器のフタに内鍵のノブが入る穴を開けて、フタを接着すればノブを回す機構の完成です。

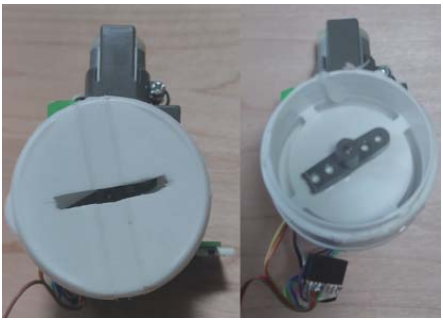


図 2: シャフトの固定

電子工作

ＳＤカード配線引出基板の三端子レギュレータの容量が少なすぎるので AZ1086H-3.3 に載せ替えます。レギュレータの配列が異なるので図3のように取り付けます。このモジュールは定格 1.5A を出力することができますので、モータードライバとモータの電力をここからドライブします。今回は DAT0/DAT1 をモータードライバに、DAT2/DAT3 をマイクロスイッチに接続しました (図4)。スイッチの逆側は GND に落として DAT2/DAT3 はプルアップ抵抗を入れておきます (図5)。ＤＣモータはサーボモータのように決まった角度に止めることが不可能なので、シャフトの鍵とは逆側にクランクアームを貼り付けて、ギアボックスにマイクロスイッチを接着します。

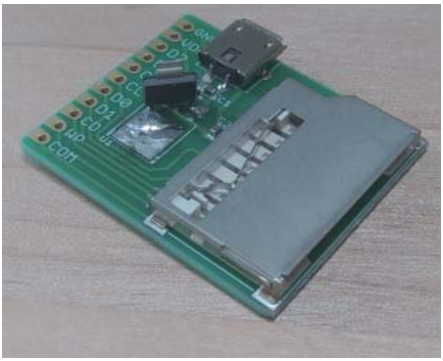


図 3: AZ1086H-3.3 の実装

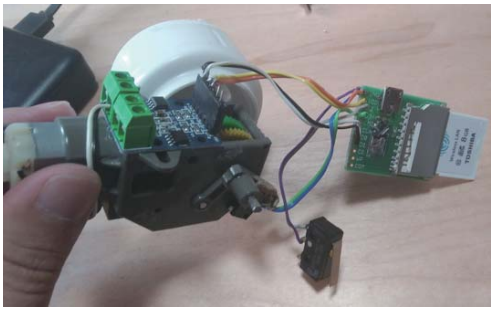


図 4: 実体配線写真

図 5: VCC-D のプルアップ抵抗

プログラミング

FlashAir の SD 端子をデジタル I/O として使用するには CONFIG ファイルを編集し、IFMODE=1 を指定してやる。すると、ON/OFF の I/O 操作が HTTP 越しに行うことができるようになる。FlashAir に接続し、標準の command.cgi に下記の URL でアクセスするとモータのテストができます。

```
http://flashair/command.cgi?op=190&CTRL=0x00&DATA=0x02
```

モータのテストができたら下記の Github ページから lock.lua と unlock.lua をダウンロードします。

```
https://github.com/nokkii/flashair_house_key
```

このプログラムを FlashAir に入れてアクセスすると、リミットスイッチを押すまでモータが回り続けます。Command.cgi だと一回のリクエストでは状態遷移待ちのループができないので、Lua の組み込みスクリプト機能を使って鍵開閉検知のループを回しています。このプログラムを使ってリミットスイッチの場所を調節し、接着すれば、晴れてオリジナルの電子錠が完成します。Android/iOS 共に URL のショートカットアイコンをホーム画面に作成することができるので、そこから利用すると良いでしょう。

筐体を作る

今の状態だとメカが丸出しなので、筐体を作ってやる必要があります。私は最初、タッパーで筐体を制作したのですが、貧乏臭くてよくありませんでした。実家からブロックを取り寄せて筐体に利用しました。私が幼少の頃遊んでいたブロックはＬ社の物ではなくＫ社のブロックでした。Ｋ社のブロックは厚みがあり、ギアボックスを覆うのにはちょうど良かったです。逆に接合は緩めに作られている為、うまく強度が出るようにブロックを組む必要

を感じました。玄関扉には吸盤で貼り付けているのですが、吸盤は貼り直しが面倒なので磁石を使える扉であれば磁石のほうが楽だと思います。鍵を回すメカは精度によっては筐体をカッチリ作りすぎると回転軸が合わず、クラッチが空転してしまいます。ダンボールやテイッシュなどでスペースを作りつつ、遊びを作ってやると良いかもしれません。

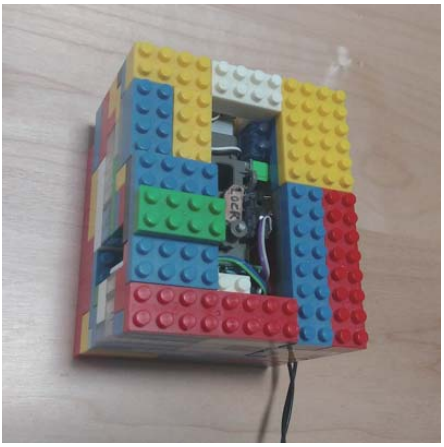


図 6: ブロックの筐体

運用をしてみて

当時は FlashAir の W-03 を使っていたのですが、ブラウザタブのロードなどで 1 秒間に何度も Lua を叩くと処理しきれない場面があり、不安定になる場面がありました。W-04 は性能が向上しているので秒間リクエスト性能も良くなっていると思います。

半年間の運用で一度だけですが、帰ってきた時に FlashAir のアクセスポイントすら見つからず、完全に落ちている日がありました。原因は分からなかったので自作ハード起因のバグか、ソフトウェアの仕様を踏んだのかもしれませんが、丁度その日に全く鍵を持たずに外出していたので完全に締め出され、大家さんに鍵を借りに行きました。流石に家の貸主に「家の鍵を電子錠にしたら締め出されました。」とは言えず、「実家に帰った際に鍵を実家に忘れてきてしまったので、スペアキーを取りに入るために鍵を貸して欲しい。」と言い訳をして鍵を借りました。後学の為に共有致します。

同居人が上京してきた時、「お昼に着くんだけど、仕事は何時に終わるの？」と聞かれ、「電子錠作ったから、無線 LAN に繋いで URL を叩けば入れるよ。」と返したのが、一番未来ポイントの高いエピソードです。今では一応物理鍵を持ち歩いていますが、私は鍵をポケットに入れるのが嫌で、カバンから出すのも面倒くさいので重宝しています。

株式会社 DMM.com ラボ 野秋 拓也 (@pg_nokkii)

R&D部門でIoTエンジニア時々ミドルウェアエンジニアとして活動。ハードウェアもソフトウェアもなんでも作るマン。便利で遊べる機能満載の FlashAir は W-02 時代からのファンです。

初心に帰って (?) L チカしよう！

せいみ まさみ

いつもは MSX の無線 LAN コントローラーを作ったり、PSG を FlashAir 単体で鳴らしたりとか好き放題、斜め上のアイデアの記事を書いてきましたが、今回はちょっと初心に帰って LED をチカチカさせようと思います。決して、夏の模型即売会の準備と重なって時間がないとかそんな逃げのためではないのですよ。きっと。

FlashAir と模型

ここ数年、プラモデルを作る際に LED を使用した電飾を行う人が増えています。戦艦の砲台やエンジンの発光や漏れる艦内の明かり、ロボットの目や武器、その他発光状態などを表現するのに使用されています。熱を出さなく小さい LED はプラモデルに電飾を組み込むのに大変ありがたいものです。

これらの電飾回路は PIC マイコンやタイマ IC などを使用して、ハードウエア的に LED の点滅パターンを設計した上で模型の中に埋め込んでおり、一度模型の中に組み入れてしまうとその後点滅パターンの変更がしづらいものとなっているようです。

そこで、FlashAir の出番です。FlashAir を使用すれば LED の点灯回路のみ作成して模型の中に組み入れておき、後で外部から無線 LAN を使用して点灯・消灯や、プログラムを変更して点滅パターンなどを変更することができます。

まずは基本の LED の点灯・消灯

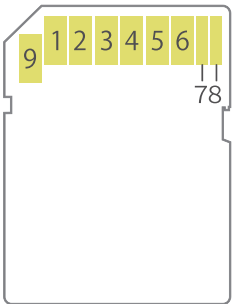
せっかく W-04 という新しい FlashAir が出ましたので、こちらを使用して制作を行ってみました。でも W-04 の新機能は全く使用していないので W-03 でも動作は可能です。関係者の皆さんごめんなさい！

FlashAir では汎用の IO ポートが 5 つあるので、5 つまで LED を同時に制御できますが、今回はサンプルということで IO ポートを 1 つだけ使用してみます。回路図も FlashAir と LED、抵抗のみの超単純回路になります。FlashAir のピンアサインを表 1 に示します。

IO ポートを使用するためには、FlashAir の隠しフォルダ SD_WLAN 内の CONFIG に "IFMODE = 1" を追加する必要がありますので忘れずに追加して下さい。また、リスト 1、

表 1: FlashAir のピンアサイン

ピン	SD I/F	SPI I/F	GPIO	Lua SPI	Lua I²C
8	DAT1	-	0x04	CS	-
7	DAT0	DO	0x02	CLK	SDA
6	Vss2				
5	CLK	SCLK	-	-	-
4	Vcc				
3	Vss1				
2	CMD	DI	0x01	DO	SCL
1	DAT3	CS	0x10	-	-
9	DAT2	-	0x08	DI	-



リスト 2 をそれぞれ led_on.lua 、led_off.lua という名前で FlashAir のルートフォルダに作成して下さい。

この FlashAir と LED を図 1 の回路図通りに接続して電源を投入後、スマホなどを FlashAir の無線 LAN に接続した後に WEB ブラウザから http://flashair/led_on.lua をアクセスすれば LED が点灯し、http://flashair/led_off.lua をアクセスすれば LED が消灯します。

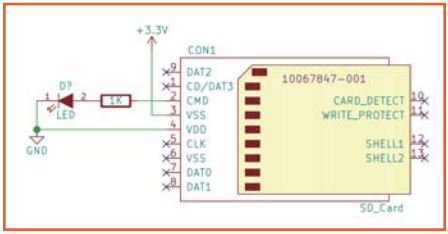


図 1: L チカ回路図

```
-- リスト 1: led_on.lua

fa.pio(0x1,1)
print("HTTP/1.1 200 OK¥r")
print("LED on")
```

```
-- リスト 2: led_off.lua

fa.pio(0x1,0)
print("HTTP/1.1 200 OK¥r")
print("LED on")
```

LED のフェードイン・アウトの実現

プラモデルの電飾の LED は単純にパッと光るだけではちょっと芸が足りない気がします。もわ〜っとアナログ的に段々と明るくなると雰囲気が出るのではないでしょうか。

リスト 3 を led_feed.lua という名前で FlashAir のルートフォルダに作成して WEB ブラウザから http://flashair/led_feed.lua を実行すると LED が段々と明るくなりその後段々と暗くなります。行 13、24 からの処理で人間の目にも止まらぬ速さで LED を ON/OFF

しています。光の残像の効果で LED を高速に ON/OFF していると、LED を光らせている時間が長ければ明るく短ければ暗く見えてしまうように錯覚してしまいます。

この LED を光らせている時間を pattern_pwm で決めていて、今回は 8 段階の明暗を作れるようにしています。ちなみにこのように高速に ON/OFF をすることで、擬似的にアナログ的な制御をすることを一般的に PWM 制御と言います。

リスト 3 の行 19、30 の sleep(1) の数字を変化させて実行してみると、高速の ON/OFF がゆっくりになるので一度試してみると何をやっているのかわかるかもしれません。

```
-- リスト 3 : led_feed.lua

 1: pattern_env ={{1,2,3,4,5,6,7,8,9},                          -- B カーブ ?
 2:               {1,2,3,4,5,6,6,7,7,7,8,8,8,8,9,9,9,9},        -- C カーブ ?
 3:               {1,1,1,1,2,2,2,2,3,3,3,4,4,5,6,7,8,9}}        -- A カーブ ?
 4: pattern_pwm = {{0,0,0,0,0,0,0,0}, {1,0,0,0,0,0,0,0}, --   0%, 12.5%
 5:                {1,1,0,0,0,0,0,0}, {1,1,1,0,0,0,0,0}, --  25%, 37.5%
 6:                {1,1,1,1,0,0,0,0}, {1,1,1,1,1,0,0,0}, --  50%, 62.5%
 7:                {1,1,1,1,1,1,0,0}, {1,1,1,1,1,1,1,0}, --  75%, 87.5%
 8:                {1,1,1,1,1,1,1,1}}                    -- 100%
 9:
10: env=1 --1 ～ 3 で変更できます。光り方が若干変わるかも ??
11: for loop2 = 0, 2 do -- フェードイン・アウトの繰り返し回数。今回は 3 回
12:   for count2 = 1, #pattern_env[env] do -- フェードイン
13:     for count_pwm = 1, 8 do
14:       if (pattern_pwm[pattern_env[env][count2]][count_pwm] == 1) then
15:         fa.pio(0x1,1) --CMD に繋がっている LED ON
16:       else
17:         fa.pio(0x1,0) --CMD に繋がっている LED OFF
18:       end
19:       sleep(1) -- ここの数を増やすと点滅を目視出来るようになります
20:     end
21:   end
22:   sleep(1000)
23:   for count2 = #pattern_env[env], 1, -1 do -- フェードアウト
24:     for count_pwm = 1, 8 do
25:       if (pattern_pwm[pattern_env[env][count2]][count_pwm] == 1) then
26:         fa.pio(0x1,1) --CMD に繋がっている LED ON
27:       else
28:         fa.pio(0x1,0) --CMD に繋がっている LED OFF
29:       end
30:       sleep(1)
31:     end
32:   end
33:   sleep(500)
34: end
35:
36: print("HTTP/1.1 200 OK¥r")
37: print("LED on")
```

最後に

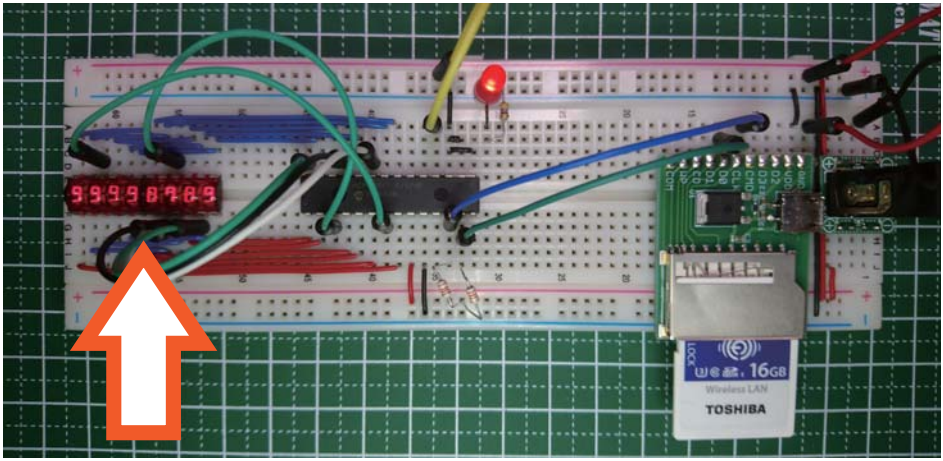
今回の記事は 1 つの LED の制御だけでしたが FlashAir だけでも 5 つの LED を制御できます。3 ポート使用するとフルカラー LED を使用して任意の色を出力するといったことも出来ます。

FlashAir ならではの機能として、WEB サーバ機能を使ってフェードイン・アウトのタイミングを変更することも出来ます。これは HTTP プロトコルを利用して供給メモリにフェードイン・アウトのタイミングを書き込み、Lua のプログラムでその値を読み込むといった方法などで実現できます。

また、FlashAir 同人誌 3 での私の記事[1]で使用した I/O エキスパンダーと FlashAir の SPI、I^2C のアクセス関数を使用するともっと沢山の LED を使用できるようになります。写真の例では矢印の 8 セグメント LED を 8 つ同時に制御して表示させています。

今回の記事では、プログラムの作成など GPS_NMEA さん作の FTLE を使用しました。便利なツール本当に有難うございました。さて、次回は何をして遊びましょうかね〜。

1　FlashAir で音楽を鳴らそう (P.48)
https://flashair-developers.com/ja/documents/books/



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。模型 (おもちゃ) も好きなので今年から模型イベントでも個人販売者側になりました。

音楽、電子工作、MSX に模型と遊びがまた増えていて、知人には「あれ、本職なんだっけ」と言われる最近です。


ブラックソラの正体！？
ブラックソラ！あなたの正体は…
生き別れの双子？
クローン人間？
それとも、
分離した悪の心!?
…それは…
他人のソラ似だッ!!
イラッ
なぁ～んちゃって☆
ニヤッ

■ FlashAir Doujinshi 4 - FlashAir の同人誌 4

2017 年 8 月 5 日　第 1 版第 1 刷発行
2017 年 10 月 11 日　第 1 版第 2 刷発行
著者 : 上岡 裕一 / 伊藤 晋朗 / Pochio / じむ / GPS_NMEA / 寺西 / 綾瀬 ヒロ / 村口 / たぞえ / 南 / 寺田 賢司 / 矢野 均 / 大松 良司 / 余熱 / Okamiya / Takehiro Yamaguchi / 高瀬 秀樹 / 野秋 拓也 / せいみ まさみ
表紙・本文イラスト : じむ / 宮内
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