電子工作愛好者向けセキュリティゲートウェイの構築

第四報：Raspberry Guardian の実証実験へ向けて

大野 浩之[1]　鈴木 裕信[2]

概要：電子回路を活用したものづくり（いわゆる電子工作）の愛好者が増加傾向にある中，2012 年に登場した手の平サイズでありながら Linux を搭載するボードコンピュータ “Raspberry Pi” は，販売価格が 3000 円台であったため販売開始から 1 年半で 200 万台以上を販売するベストセラーとなり，電子工作の世界とインターネットが一気に近づいた．その結果，単に電子工作を楽しみたいだけであっても，作品がネットワークを介した通信を行うのであれば，情報通信分野の専門技術者と同等の情報セキュリティに関する専門知識と運用技術の習得が必須となった．しかし，純粋に電子工作を楽しみたいだけの者にとってこの要求は大きな負担となろう．そこで著者らは，Raspberry Pi のために Raspberry Pi で作ったセキュリティゲートウェイ装置である “Raspberry Gate”[1] と，情報セキュリティ上の脅威を簡単かつ持続的に低減する手法である “Raspberry Guardian”[2] の二つを提案している．本報では，電子工作愛好者向けの脅威の低減を維持し続けるこれらの機構についてその現状と今後について述べる．

Security Gateway for Electronics Hobbyists

(Part 4. Experimental Implementation for Raspberry Guardian)

Hiroyuki Ohno[1]　Hironobu Suzuki[2]

1. はじめに

近年，電子回路を活用したものづくり（いわゆる電子工作）を楽しむ人々が増えている．特に 2012 年初頭に登場した英国製のボードコンピュータ “Raspberry Pi” は，手のひらサイズでありながら UNIX 系 OS をインストールでき，3000 円台で入手できたため，発売開始から 1 年半で 200 万台以上出荷するベストセラーとなった [3].

これまでも，マイクロコントローラにセンサやアクチュエータを接続してものづくりを楽しむ電子工作の世界と，インターネットの普及が作り出した情報通信ネットワークの世界を取り持つ方法は存在したが，複雑な操作を要したり十分な性能がでなかったり高価だったりした．安価ではあるが電子工作用途であれば十分高性能な Raspberry Pi の登場と，IoT(The Internet of Things) 分野の研究成果が一般に普及し始める時期とが一致したこともあり，Raspberry Pi を使ったインターネットと連携する電子工作が愛好者の間に急速に普及し始めた．このことは電子工作という個人の趣味の世界が，セキュリティ上の深刻な脅威をインターネットから直接受ける事態になったことも意味し，個人的に電子工作を楽しみたいだけであっても，彼らの作品がネットワークへの接続性を持つのであれば，業務として情報通信機器を日夜管理するインターネット分野のスペシャリストと同等の情報セキュリティに関する専門知識と運用技術が必要となった．単に電子工作を楽しみたいだけの者にとってこの要求は大きな負担となろう．

一例をあげるなら，Raspberry Pi で最も普及している Raspbian という Debian UNIX 系のディストリビューションをインストールした場合，インストールした時点で設定されている pi というユーザの初期パスワードは広く知られたある文字列に決められている．初期設定のままの Raspbian は，起動とともに sshd が特に制限を受けない状態で起動して遠隔ログインを提供し，ユーザ pi は sudo

[1] 金沢大学 総合メディア基盤センター
Kanazawa University, Kanazawa, Ishikawa 920–1192, Japan
[2] 専修大学 ネットワーク情報学部
Senshu University, Kawasaki, Kanagawa, 214-8580, Japan

コマンドで管理者権限を全面的に取得するのに自分のパスワードを入力するだけでよい．このような初期設定のままの Linux 機が特段の配慮なくインターネットに接続され始めたら何が起こりうるかは情報セキュリティに一定の知見のある者であれば容易に想像できるはずである．しかし，電子工作を楽しみたいだけの者は「動いているのだからそれでよいのではないか」と考えるかもしれない．加えて Raspberry Pi には一般ユーザ権限でコマンドラインから自由に操作できる汎用の I/O ポートがある．インターネットからアクセス可能な Raspberry Pi の GPIO ポートにモーターやヒーターなどの機材が接続されていた場合，これらの不適切な遠隔操作は大規模な物理的破壊を伴うかもしれない．これは「システムを悪用されてメールの不正中継に利用された」といった事案とは次元の異なる深刻な事態であり，なんとしても未然に防ぐ必要がある．

そこで，著者らは Raspberry Pi のために Raspberry Pi で作ったセキュリティゲートウェイを構築することにし，これを Raspberry Gate と名付けた．Raspberry Gate は，情報通信分野の専門家ではない電子工作愛好者が簡単に設置して運用できる Raspberry Pi を用いた小型の装置であり，彼らの作品がネットワーク接続する際の上流側（WAN 側）との境界点に設置する [1]．Raspberry Pi で Raspberry Gate を構成するのは，コストを安く押さえられることに加え，「得体の知れない見知らぬ機材」を新たに使わせるという流れではなく，使い慣れた親しみのある機材である Raspberry Pi を活用したという流れにすることで，望まずにシステム管理者になった電子工作愛好者が抱きかねない導入に際しての抵抗感を少しでも下げようという配慮の結果でもある．

Raspberry Gate は既存の UNIX ベースのセキュリティゲートウェイと同等の構成をとるため，一定の水準の安全確保が期待できるが，セキュリティデバイスの常として，その機能や性能をどのようにして維持し続けるかが当該デバイスの初期性能の確保以上に重要な問題となる．この問題に対処し，情報セキュリティ上の脅威を簡単かつ持続的に低減する組織と手法として，著者らは “Raspberry Guardian” と名付けたソーシャルネットワークを活用した合意形成と機能維持のための手法を提案している [2]．

なお，現時点での議論や開発は，Raspberry Pi を対象として展開しているが，ここでの議論は Raspberry Pi と同様の構成の小型ボードコンピュータにも適用できるため，本研究の成果は広く応用できる．

2. Raspberry Gate と Raspberry Guardian

インターネットへの接続性を有する Raspberry Pi とその周辺環境のセキュリティ確保を検討するにあたり，「ラズベリー」がいちごの一種であることから，いちご畑，いちご農場，いちご農家などに見立て，以下の用語を定義した (括弧内は比喩)．

Raspberry Field　多数の Raspberry Pi が散在している環境（畑・野原）

Raspberry Garden　ひとりまたは複数の所有者の何らかの方針の下でまとめて管理運用されている Raspberry Pi の集合体（庭園）

Raspberry House　Raspberry Garden を管理運用する者の拠点（農家）

Raspberry Farm　Raspberry House と，ひとつまたはいくつかの　Raspberry Garden からなる，一まとまりの実体（農園）

Raspberry Gate　Raspberry Farm を外部と区切る門．この門以外は塀で囲まれていて出入りできない（農園の門）

Raspberry Guardian　Raspberry House を拠点に行動し Raspberry Garden に安全安心をもたらす者．またそのための組織（農園の保護者）

実際の Raspberry Gate は，図 1 に示すような位置づけにあり，ネットワーク的にはフィルタリング機能を持つルータあるいはブリッジである．

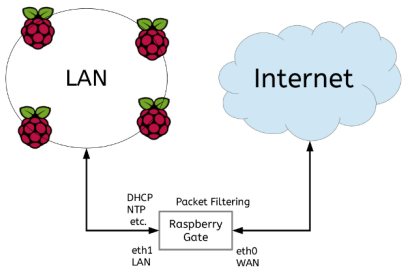

図 1　Raspberry Gate の機能

Raspberry Pi の基本性能は，一般的な PC より劣る（あるいは 10 年前の PC よりは勝る程度である）が Linux を OS とできること，ネットワークインタフェースを複数搭載できること，Linux PC にはファイアウォールのようなセキュリティ機器としての運用実績があることなどから，100Mbps 程度の速度でよいなら Raspberry Pi をルータあるいはブリッジとして利用できることは既報で指摘した．ただし，Raspberry Gate の想定利用者である電子工作愛好家が必ずしも情報セキュリティ分野の専門家ではないことから，起動，機能選択，終了といった操作が簡単に行え，いきなり電源を切断しても次回利用時に影響がでないようにするといった配慮が必要である．

Raspberry Guardian は，図 2 のような異なる役割をもったユーザから構成される．彼らの多くは一つあるいはそれ以上の Raspberry Farm の所有者である．所有者が違う Raspberry Farm は設置場所も運用規模も異なる

し，運用方針も厳密には同一ではない．しかし，Raspberry Guardian においては，ソーシャルネットワークのしくみを活用して意見交換を行い，多少の意見の違いは受け入れつつ Raspberry Gate の運用方針やそれに基づく諸機能の諸設定を共有し，ひとりではなし得ない Raspberry Farm の安全と安心を継続的に享受することを目指す．

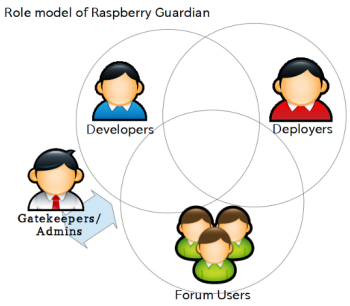

図 2　Raspberry Guardian の概念

3.　設計と実装

3.1　Raspberry Gate のハードウェア

本項では，Raspberry Gate のハードウェアについて，Raspberry Gate の基本部分と拡張部分に分けて述べる．基本部分は Raspberry Gate に必須の部分であるのに対し，拡張部分は可用性を高める部分である．初回設定を終えて安定して動作している Raspberry Gate は，設定を変更しない限り基本部分だけでも動作する．

3.1.1　基本部分

Raspberry Gate の基本部分は，Raspberry Pi, SD カード, USB メモリ, USB LAN アダプタおよび電源である．その主な仕様を表 1 に，外観を図 3 に示す．

表 1　ハードウェア仕様

型式	Raspberry Pi Type B
CPU	Broadcom BCM2835 700 MHz ARM1176JZFS
メモリ	512 MB
ストレージ	SD カード (OS 起動用) 及び USB メモリ (動作記録用)
USB I/F	USB 2.0 × 2
ネットワーク I/F (本体)	10 / 100 Base-T
ネットワーク I/F (追加)	100 / 1000BASE-T USB LAN アダプタ
電源	DC 5V, 700mA

後述のように OS のブートは SD カードから行う．SD カードは 4GB 以上の容量が必要である．将来的にはこの SD カードは起動直後に読み出し専用で再マウントし，予期せぬ電源断などにも耐えられるように改良する．USB インターフェースに接続した USB メモリは，ストレージ領域として各種ログの記録に利用しているが，SD カードが読み出し専用でマウントされるようになった場合は，一時的作業領域などはこの USB メモリ上に確保することになる．Raspberry Pi ボード本体にある Ethernet インタフェースは OS により eth0 に割り当てられる．1000BASE-T (あるいは 100BASE-T) USB LAN アダプタは USB インタフェースに接続され，OS により eth1 に割り当てられる．

電源は，DC+5V で 最低でも 700mA 程度必要である．電源コネクタは micro USB コネクタなので，形状的も容量的にもスマートホン用のものが流用できる．なお，USB LAN アダプタによってはこれ以上の電力が必要になり電源供給型の USB ハブを別途用意する場合もある．

図 3　Raspberry Gate の実装例

3.1.2　拡張部分

Raspberry Gate は，USB キーボードや HDMI ディスプレイの存在を前提としない．必要な設定を終えて安定稼働している状態では上述の基本部分だけで動作するが，起動操作，機能選択，機能確認，終了操作などをキーボードやディスプレイに頼らずに容易に行えるしくみがあると可用性が向上する．この可用性向上に寄与する部分を拡張部分と呼ぶ．幸い Raspberry Pi にはユーザがコマンドラインからも利用できる GPIO ポートが用意されているので何らかのハードウェアを接続して利用するのは比較的容易である．しかし，ハードウェアの追加に伴って設置コストが上昇することは望ましくない．

今回は，Raspberry Gate の基本部分と拡張部分の購入費用の合計金額を Raspberry Pi 2 台分強の 10,000 円に設定して検討した結果，以下の構成になった．これらのうち，電源アダプタについては既存品の流用できる場合があり，ケースについては別途自作となる場合があるので「別途」と表記した．現時点の概算では，基本部分だけで 6,500 円，拡張部分を含めると総額 8,000 円．その他に電源アダプタとケースを購入すると 10,000 円となる．

- 基本：Raspberry Pi 本体 - 4,000 円 [*1]
- 基本：SD カード - 1,000 円
- 基本：USB イーサネットアダプタ - 1,500 円
- 拡張：ディップスイッチ，LED, ブザー等で構成する

[*1] 発売当時は 3000 円台であったが，為替レートの変動と消費税の変更で 2014 年 5 月現在では 4000 円台

拡張モジュールの部品類 - 1,500 円
- 別途：電源アダプタと簡易なケース - 2,000 円

Raspberry Pi の GPIO 機能をユーザが使う場合には，本体上にある 26 ピンの「P1 ヘッダ」に引き出されている GPIO ポートを使うのが一般的である．P1 ヘッダには電源線も配置されているため実際に I/O に使えるのは 17 ピンであるが，これらのうち 7 ピンは標準起動状態では UART(2 ピン), I2C(2 ピン), SPI(3 ピン) に割り当てられている．これらのピンもデジタル I/O に利用することは問題なくできるが，これらの機能を温存するとデジタル I/O に使えるピンは 10 ピン (10bit) となる．今回はこの 10 ピンを以下のように割り当てて試作した．

- デジタル入力 (2bit) (起動モード選択)
- デジタル入力 (4bit) (ネットワークアドレス選択)
- デジタル入力 (1bit) (シャットダウン指示)
- デジタル出力 (1bit) (ウォッチドッグタイマリセット)
- デジタル出力 (1bit) (LED)
- デジタル出力 (1bit) (ブザー)

10 ピンのうち 7 ピンはデジタル入力モードに割り当てた．後述のように Raspberry Gate には 4 つの動作モードが存在するのでモード選択用に 2 ピン分を割り当てた．さらに DHCP サーバが割り当てるサブネットの選択用に 4 ピンを，シャットダウン指示用に 1 ピンを割り当てた．これらのピンは内部的に 10K Ωの抵抗でプルアップし，シャットダウン指示用のピン以外の 6 ピンはディップスイッチに接続した．

一方，出力モードにした 3 ピンは，ウォッチドッグタイマのリセット用に 1 ピンを，動作確認のための LED とブザーに 1 ピンずつ割り当てた．LED とブザーは 1 ビットずつしかないためエラーメッセージ類の表現には制限があるが，LED の点滅とブザー音のオンオフで表現する方針である．LED やブザーのオンオフでのエラー表示は，追加部品点数を減らし製作コストは押さえられるがあまりに不便という判断であれば，I2C インタフェースを利用してデジタル I/O を拡張する方法，さらに液晶表示装置を接続する方法がある．製作コストを押さえるなら 16 文字× 2 行のモノクロキャラクタ液晶表示装置を使うことになるが，製作コストをの制約がないなら QVGA 相当のフルカラー液晶表示器を SPI インタフェースで接続し，X11 のディスプレイにする方法もある [4] .

3.1.3 RTC(リアルタイムクロック)

Raspberry Pi は，RTC を持っていないため，Raspberry Pi 用の Linux の一つであり今回 Raspberry Gate に採用した Raspbian では，起動時にネットワーク接続が有効であればインターネット上の NTP サーバを用いて時刻を合わせ，以後はタイマー割り込みで内部時計を更新しつつ適宜 NTP で時刻同期を行っている．もし，起動時に現在時刻が得られなければ前回停止した際の時刻の直後から計時を再開する．このため，時刻が過去に戻ることはないが，再起動時に NTP サーバへアクセスができないと再起動のたびに実際の時刻との乖離が大きくなる．そこで Raspberry Gate には，バッテリーバックアップ機能付の外付 RTC（リアルタイムクロック）を追加することを推奨している．このような用途に適している Maxim 社の DS1307 を使えば 1000 円未満の追加支出で目的を達せられる．DS1307 は I2C インタフェースで Raspberry Pi と接続できるので [5], 今回の I2C ポートを温存した GPIO ポートの割当であれば DS1307 の追加は容易である．

3.1.4 ウォッチドッグタイマ

Raspberry Gate の可用性向上のためにはウォッチドッグタイマによる異常検出と再起動，これらと連動した自電源の制御が必要であるが Raspberry Pi にはいずれも装備されいない．そこで，かつて Raspberry Pi を車載運用した際に製作した回路 [6] を流用して動作を確認した．

通常，自動車の電源系には常時通電している系とエンジン作動時にのみ通電する系がある．今回流用した回路は，本来は前者の電源系から給電して回路を動かしつつ後者の電源系を監視し，一定時間以上のエンジン停止を検出するとシャットダウン信号を外部機器（この場合は Raspberry Pi）に送出する機能を持つ．また，外部機器側が定期的にパルス信号を出すようプログラムしておき，一定時間以上パルスがこなければハングアップと判断して強制リセットするウォッチドッグタイマ機能も持つ．Raspberry Gate では，Raspberry Gate 用の電源を常時通電系と見なし，Raspberry Gate の下流に位置する他の Raspberry Pi への給電を監視し，下流の Raspberry Pi への給電が全て止まったら Raspberry Gate も一定時間後に停止する体制を用意した．タイマ IC を始めとする 10 個程度の部品で構成されるこのウォッチドッグタイマの詳細は著者らが関わる ε-ARK プロジェクトの Web ページ [*2] から取得できるので，ここでは部品配置を図 4 に示すに留める．

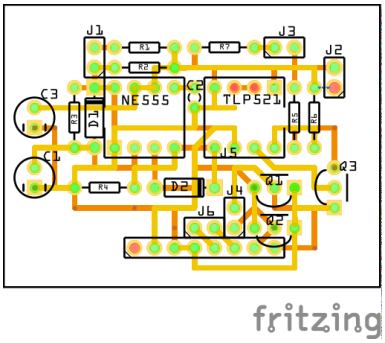

図 4 ウォッチドッグタイマの部品配置

[*2] http://e-ark.jp/

3.2 Raspberry Gate のソフトウェア

Raspberry Gate で採用した OS の主な情報 を表 2 に示す．Raspberry Gate が提供するさまざまなサービスやアプリケーションソフトウェアはこの OS の上で動作する．

表 2 OS 仕様

名称	Raspbian
ディストリビューション	Debian Wheezy
ベース	
リリース日	2014-01-07
Kernel Ver.	3.10

3.2.1 インストール

Raspberry Gate 構築のために Raspberry Pi Type B に Raspbian (Debian Wheezy [7]) を最小限インストールしている．

特定のディストリビューションに依存せず拡張性を担保することを考慮する．そのため Raspberry Gate 自体のインストールはシェルスクリプトをベースとしたオリジナルのインストーラで行う．各ディストリビューションに用意されているサービス (ntpd や dhcpd など) を使う部分のみ，Debian のパッケージ管理システム APT(Advanced Pakaging Tool) に依存する．各種サービスの脆弱性対応などは Debian のディストリビューションのアップデートにしたがう．

3.2.2 Github リポジトリの利用

現在のバージョンでは tar ボールを展開し，スクリプトを稼動させることによってインストールを行う．

```
$ tar xvf raspgate.tgz
$ cd raspgate
$ sudo ./install.sh
```

今後は Github [8] のリポジトリから直接ダウンロード及びアップデートを行えるようにする．既に組織名 RaspberryGuardian として Github サイトには登録している．その時は次のような形でリポジトリを取得し，インストールを行う．

```
$ git clone git://github.com/RaspberryGuardian
  /raspgate.git
```

github 移行後は，アップデート用に本システムで用意されているシェルスクリプト update.sh を実行すると内部で git pull が呼び出され，関連するアップデートファイルがダウンロードされた後，自動的に関連するファイルも含めて自動アップデートされる．

Raspberry Gate のアップデートは，インストール済みのソフトウェアに発生した脆弱性への対処などのメンテナンスだけではなく，Raspberry Gate が新たに必要とする外部パッケージの新規導入や，使用するデータファイルの更新，あるいは新規導入など多岐にわたって行われる．

Debian パッケージ管理システムのみを考えるならば，deb 形式で対応する方が良いと思われるが，将来の拡張性，汎用性を考えた場合，特定のディストリビューションを前提に構築するのは良いとは思われない．かといって，Raspberry Gate のために拡張性と汎用性を持ち合わせた特別なツールを自作するのも開発者の負担になるので，既存のアップデートの枠組みを効率よく利用するのが望ましい．そこで，オープンソース開発環境では広く使われているリポジトリである Github を利用することにした．

3.2.3 起動時スクリプト raspgate

Raspberry Gate には以下で述べるように「初期化モード」「メンテナンスモード」「ブリッジモード」「ルーターモード」の 4 つのモードがある．ブート時にこのいずれか 1 つを外部スイッチで選択し，Raspberry Gate 環境を立ち上げるための起動スクリプトとして /etc/init.d/raspgate を用意した．

raspgate 内でロードされるモジュールは/lib/raspgate/ 以下のディレクトリにインストールされる．以下のような関連モジュールファイルがある．

- /lib/raspgate/bridge-functions : ブリッジモード設定で利用する時に必要なモジュール
- /lib/raspgate/router-functions : ルータモード設定で利用する時に必要なモジュール
- /lib/raspgate/iptables-functions : iptables 設定で利用する時に必要なモジュール

3.2.4 秘匿初期化ベクトル initvec の生成

インストール時に設定される 8K バイトのランダム値をファイル /etc/raspgate/conf.d/initvec に格納する．これをシステム固有の値とし SECKEY(後述) などを作る．そのため，この initvec の内容を変更するとシステム的には別の個体 (システム) として認識する．

3.2.5 その他のインストール関連情報

ブリッジ環境のために “bridge-utils” が，ルータ環境のために “dhcpd” と “ntpd” がインストーラ内部で apt-get を呼び出す形でインストールされる．

さらに Raspberry Gate のブリッジモードで利用される設定ファイル及びルータモードで利用される以下の設定ファイルがインストールされる．

- /etc/raspgate/conf.d/bridge/eth0-01-iptables : ブリッジモード稼働中に eth0 に設定可能な iptables のパラメータ
- /etc/raspgate/conf.d/router/eth0-01-iptables : ルータモード稼働中に eth0 に設定可能な iptables のパラメータ

3.2.6 個々の Raspberry Gate 識別

現在の Raspberry Gate は 1 つ 1 つ独立して管理する必

要があるが，将来的にはグループ管理などの必要性があると考える．そこで最初の設計の段階から個別の Raspberry Gate 認識のための ID を組み込むことにした (表 3)．

表 3 内部で利用されるキーの種類

IDKEY(認識キー)	Raspberry Gate のもつアカウントに相当
SECKEY(秘密キー)	Raspberry Gate のもつパスワードに相当
EXKEY(拡張キー)	厳密にユニークを確認する際に使う値

個々の Raspberry Gate にはインストール時に 8Kbyte のランダム値ファイルを作り，それを秘匿することにより秘密情報生成時の初期化ベクトルとして利用する．

IDKEY は Raspberry Gate の持つアカウントに相当し，MAC アドレスから SHA256 を用いて生成する．MAC アドレスを直接使わないのは，MAC アドレスの上位 24 ビットはベンダー番号なので Raspberry Pi の場合常に同じであり，下位 24 ビットも製品ロット単位で連番的に番号を割り振るため似たような数字がならび，IDKEY の値を取り扱う際に取り違えやすいからである．

一方で，IDKEY と SECKEY の両方を合わせた拡張値 EXKEY (出力は 256 ビット長であるが情報量は実質 128 ビット相当) を使えば実質的なコリジョンを起こす可能性は無視できると考えられる．

各キーの生成ルールを以下に示す．

```
IDKEY :=
  SHA256 (MAC アドレス文字列 || “idkey-1”)
SECKEY :=
  SHA256 (MAC アドレス文字列 || initvec)
EXKEY := SHA256 (IDKEY hex 文字列
  || SECKEY hex 文字列 || "exkey-1")
(いずれも SHA256 の先頭 16 文字を使用)

キーサンプルを 14ca683d5cff35c4 とした場合の
IDKEY，SECKEY の例 (EXKEY は都度生成)：
$ cat /etc/raspgate/raspgate.conf
IDKEY  14ca683d5cff35c4
SECKEY 36a4aba0dcec23e9
```

3.3 Raspberry Gate の動作

Raspberry Gate の動作モードは全部で 4 つある．

1. 初期化モード：フィルタリングパターン（設定ファイル）の選択など
2. アップデートモード: メンテナンス環境
3. ブリッジ動作モード: ブリッジとして動作
4. ルータ動作モード: ルータとして動作

前述したように GPIO ポートには，モード切り換えのためのディップスイッチを接続する．このディップスイッチの ON/OFF でこれらのモードを切り替える．1 及び 2 のモードはディスプレイとキーボードを接続しコンソールからのオペレーションをする．その際は，USB LAN アタプターは取り外した上で USB キーボードを使う (もう 1 つの USB ポートは USB メモリにより既に使われている)．3 及び 4 のモードは組み込みとして動作する．

3.3.1 ブリッジ動作モード

ブリッジは，ネットワーク環境を変えずフィルタリングのみ行うので，内部ネットワークへはフィルタリングのみの影響に留まるが，外部からの遮断はフィルタリングのみである．

```
eth0[WAN 側] ←→ br0 ←→ eth1[LAN 側]
```

図 5 ブリッジモードでの接続関係

フィルタリングは iptables で行う．デフォルトでは eth0 の INPUT/OUTPUT (インバウンド/アウトバウンド) のフィルタリングを行う．図 5 のように接続しているので，eth0 と eth1 双方 (あるいは br0) で INPUT/OUTPUT に対するフィルタリングも可能である．

ブリッジのフィルタリング設定ファイルは /etc/raspgate /conf.d/bridge/に保管されている．どのパターン (設定ファイル) を使うかの選択は初期化モードに入りメニューで選択する．

- フィルタリング設定ファイル: /etc/raspgate/conf.d/bridge/eth0-01-iptables など

3.3.2 ルータ動作モード

ルーターモードはネットワーク切り離し NAT 環境で運用するモードである．WAN 側 eth0 は WAN 側 dhcp で割り当てられる．初期化モードで手動で IP アドレス，ネットワークアドレスなどを設定することも可能である．

LAN 側はデフォルトでは 192.168.32.0/24 が割り当てられる．デフォルトのネットワークアドレスが WAN 側のネットワークアドレスと重複する場合は，前述した GPIO ポートに接続したスイッチの設定を変更すると他のネットワークアドレスに変更できる．

LAN 側に接続される Raspberry Pi のために dhcpd と ntpd が動作している．dhcp 経由で LAN に接続されている Raspberry Pi に必要な DNS サーバ，NTP サーバ，ゲートウェイなどのアドレスが割り振られる．

3.4 Raspberry Guardian

既に述べたように，Raspberry Gate に設定する内容を合議で決めるしくみ (コミュニティ・サイト) と，そのしくみを利用する Raspberry Gate ユーザを Raspberry Guardian と呼ぶ．Raspberry Guardian で合意した結果は更新スク

リプトにまとめられ，github を介して個々の Raspberry Gate にインストールされる．

本項では Raspberry Guardian の構成要素について述べる．

3.4.1 Raspberry Guardian ユーザ認証

Redmine ベースで作成されている Raspberry Guardian サイトは，開発者 (Developer)・利用促進者 (Deployer)・利用者 (User) からなるコミュニティ・サイトとして機能するが，独自にユーザ管理するのではなく，OpenID[9] を使い，Google や Facebook，あるいは twitter などの既存の SNS サイトやコミュニティサイトと連動させる．

Redmine は OpenID でのログインをサポートしているが，OpenID のみでの運用に絞った方法はサポートしていない．また OAuth でのログインはサポートしていない．OpenID や OAuth を利用することでユーザは Raspberry Guardian のためにパスワードを用意したりする手間が省ける．また，Raspberry Guardian サイトでもパスワードのような秘密情報を管理する手間を軽減できる．

またユーザは Raspberry Guardian のコミュニティには加わりたいが，しかし，利用者情報を Raspberry Guardian の運営者には知られたくない，あるいは利用者情報入力する手間を省きすぐに使いたいといったニーズもあると考える．これらのニーズに対し，OpenID や OAuth を提供することで解決する．

現在，Jorge Barata の手による OpenID Selector[10] を導入すれば Google，Yahoo!，OpenID，Wordpress などといった大手の OpenID を利用できる (図 6)．

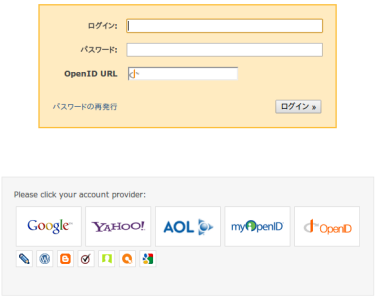

図 6 OpenID Selector 利用画面

一方で Facebook や Twitter は OAuth(oauth2) を採用している．OpenID と OAuth の両方を統合化して使えるプラグインは現在存在していないので，OpenID と OAuth のどちらでも使える枠組みを実装する必要がある．

3.4.2 Ruby on Rails での統合化枠組み

Redmine は Ruby on Rails ベースで実装されている．

OpenID Selector のコードを参考としながらも，OpenID と OAuth を扱うための Redmine プラグイン実装を試みる．これにより OpenID を使う場合と，OAuth を使う場合の両方をユーザに提供できることになる．OpenID ライブラリである janrain.com の OpenID Enabled と oauth2 ライブラリである intridea の oauth2 を使い実装の準備を進めている．

3.4.3 使い捨てアカウント問題

OpenID や OAuth を用いても，その提供する先のサイトにアカウントがあることが保証されるだけで，どのような状況で利用されているか把握できない．悪意をもって利用するために作成するアカウント，いわゆる使い捨てアカウントの問題は OpenID や OAuth では解決できない．

これらを解決するためには，利用先サイトでのユーザのアクティビティをチェックし，有効なアカウントか否かを判断するロジックを入れるなど，踏み込んだ判断を必要とする．技術的には実装は難しくないが，これを導入するのにはサイトポリシーの面から議論が必要である．

4. 現状と今後

今後の試験的な実装をもとにより現実的な用途に安心して使えるように機能を拡張する．なお，既に述べたように Raspberry Gate では将来の機能拡張や変更に備えたバージョン確認機能を要所に埋め込み，将来のバージョンとそれまでのバージョンが混在してもそれが原因でセキュリティ上の問題が生じないよう配慮している．

インターネット上に離れて存在する複数の Raspberry Farm 同士を相互接続するしくみはすぐにでも必要とされている．たとえば，Raspberry Pi を用いた「作品」を運用する展示会場と，その作品の製作者のオフィスの Raspberry Farm を接続してオフィスから遠隔制御したり動作を監視したいという要求はすでにある．この例は二つの Raspberry Farm を一対一接続しているに過ぎないが，一対多，多対多の接続要求もある．

このような場合，安全性を確保しながら，インターネット上に Raspberry Farm 同士を相互接続する「交換点」を設けることになる．このような相互接続を実現する方法は OpenVPN や L2TP/IPSec を用いるにせよ SSH トンネルを用いるにせよ手法としては新しいものではないが，著者らが想定する電子工作愛好者にとっては未経験の技術であるので，安全安心で高い可用性をもつサービスとして提供したい．このサービスを小規模に実施するのであればそれほど難しくないがスケーラビリティまで含めると十分な検討が必要となる．

5. 関連研究

汎用的な PC に Linux をインストールし，情報セキュリティ面の脅威を軽減するゲートウェイを作ることは以前から広く行われている．たとえば，著者の一人（鈴木）は 2000 年に自著 [11] でこの手法について論じており，それ以前から日常的に利用してきた．

今回の試みでは，一般的な PC と比べると処理能力の劣る Raspberry Pi という小型機で Linux を動かしているため，PC を用いた場合と比べると性能面での懸念があるが，その点を除けばすでに確立している技術を選び組み合わせて実現している．したがって，そのこと自体には新規性はないが，小型機で十分な性能を出すための機能の取捨選択や性能調整，さらに GPIO ポートを活用した支援機能を伴うことなどには一定の新規性がある．

また，単なるセキュリティゲートウェイではないので Raspberry Gate との直接の比較はできないが，Rapsberry Pi に実装したゲートウェイである Onion Pi は著名である．Onion Pi は，複数のプロキシノードを暗号化した多段トンネルで接続して匿名通信を行う技術である Tor (The Onion Router) を Raspberry Pi に実装した装置である．Onion Pi と Raspberry Gate は，その目的は異なるが，Raspberry Pi の USB ポートに LAN インタフェースを取り付けてルータとし，対外接続を支援するというしくみは同じである．実装方法の差が性能の差に現れるとしても，Onion Pi がルータとして機能するのと同程度の通信性能は Raspberry Gate でも期待できる．

Raspberry Guardian が目指している，セキュリティ確保のための諸設定をソーシャルなしくみの上で合意形成して定め，参加者が成果を共有して自らの機器に自動的にインストールするというアプローチと全く同じものは見いだせていないが，規模の大きなソフトウェアに一般的に見られるソフトウェアアップデートのしくみを変形させたものだと言える．一般的なソフトウェアアップデートでは，何をアップデートするかの判断と具体的な内容は当該ソフトウェアを供給する側が行い，利用する側はこれを自動的に受け入れる（あるいは個別に判断して受け入れるか否かを決める）．利用する側もソフトウェアアップデートに加われるようにしたのが Raspberry Guardian だと言える．

6. おわりに

本報では，Raspberry Gate と Raspberry Guardian の基本設計と試験的実装について述べた．

これらの開発はまだ緒についたばかりであるが，Raspberry Gate が何のセキュリティ対策も施していない Raspberry Pi を保護できること，それを実現するために必要な Raspberry Gate の諸設定を Raspberry Guardian が提供できることを今後の運用を通じて客観的に示したい．著者らは，2014 年度第 2 四半期からアルファテストを開始するので，本報を口頭発表する 2014 年 7 月下旬にはアルファテストに至る道のりとテスト開始後の状況についての報告が可能である．

本研究の進捗状況は，Raspberry Guardian ホームページ [*3] あるいは著者らが関わる ε-ARK プロジェクトのホームページ [*4] で随時報告する．

謝辞

著者の一人（大野）は，電子工作愛好者が実際に集って技術的な情報交換する「木いちごの会」というハンズオンを定期的に石川県野々市市や同県金沢市で開催している．本研究は，同会における Raspberry Pi を用いた IoT デバイスの安全安心な運用についての議論がきっかけとなって始まった．それぞれ異なる技術的背景と好奇心を持って同会に集う，実行力あふれる電子工作愛好者各位に謝意を表したい．

参考文献

[1] 大野浩之, 鈴木裕信. 6D-4 電子工作愛好者向けセキュリティゲートウェイの構築 (第一報：設計と実装). 全国大会講演論文集, Vol. 2014, No. 3, pp. 47–48, mar 2014.

[2] 大野浩之, 鈴木裕信. 6D-5 電子工作愛好者向けセキュリティゲートウェイの構築 (第二報：運用と管理). 全国大会講演論文集, Vol. 2014, No. 3, pp. 49–50, mar 2014.

[3] kanai. Raspberry Pi が 200 万台を突破. http://makezine.jp/blog/2013/11/two-million-raspberry-pis-in-the-wild.html 2014 年 5 月 16 日閲覧.

[4] Lady Ada. Adafruit PiTFT – 2.8 inch Touchscreen Display for Raspberry Pi. https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen\-display-raspberry-pi/ 2014 年 5 月 16 日閲覧.

[5] Lady Ada. Adding a Real Time Clock to Raspberry Pi. https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi 2014 年 5 月 16 日閲覧.

[6] 大野浩之. 車載 Raspberry Pi (UNIX ネイティブの電子工作塾 夏休み特別編電子工作塾作品発表会). USP マガジン, Vol. 2013, No. 3, p. 45, 6 2013. ISSN 2187-9443.

[7] debian.org. Debian wheezy リリース情報. http://www.debian.org/releases/wheezy/index.ja.html 2014 年 5 月 12 日閲覧.

[8] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pp. 1277–1286, New York, NY, USA, 2012. ACM.

[9] specs@openid.net. OpenID Authentication 2.0 - Final. Technical report, OpenID Foundation, December 2007. http://openid.net/specs/openid-authentication-2_0.txt.

[10] Jorge Barata. OpenID Selectorp. http://www.redmine.org/plugins/openid-selector 2014 年 5 月 12 日閲覧.

[11] すずきひろのぶ. 実践 Linux セキュリティ. インプレス, 9 2000. ISBN4-8443-1401-1.

[*3] http://raspbian-guardian.info/

[*4] http://e-ark.jp/