UIテストの所要時間を10分の1に短縮する取り組み

～ ラズベリーパイのクラスターで並列実行 ～

SQiP2018 Award受賞記念講演会

（2019年3月11日）

レバテック株式会社
ITソリューション事業部
専属ビジネスパートナー
折田 武己
takemi.orita@levtech.jp

目次

自己紹介

折田 武己 <takemi.orita@levtech.jp>

フリーランスのエンジニアとして、様々な開発プロジェクトに携わってきた。また、エンジニア向け技術研修の講師として、後進の指導に当たることも多い。

どこの開発現場でも直面する喫緊の課題であるテストの自動化については、ライフワークのひとつとして長年に渡り取り組んできた。

本日の講演内容も、これまでの開発経験から得られた知見をベースにしているが、独自の視点からアイデアを練り直し、日々カイゼンを施している。

UIテストの時間短縮

Selenium や Appium 等のミドルウェアが整備されたこともあり、UIテストを自動化する事例も増えてきた。

しかし、**自動テストの開発資産が拡充されるのに比例して、その所要時間は増加の一途を辿る。**日々改良を加え、利用者に提供するサービスでは、UIテストがボトルネックになってスピード感が損なわれることも少なくない。

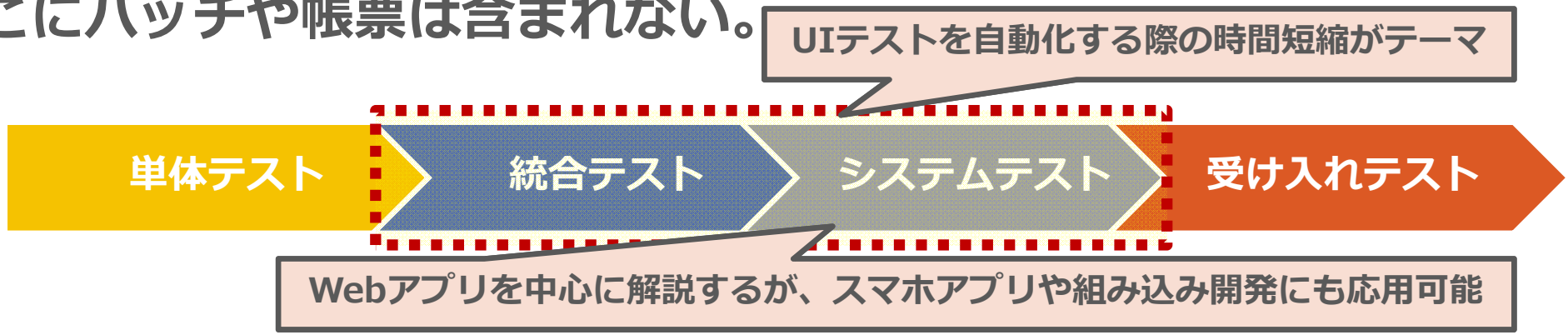
リリース判定に使うテストケースを絞り込み、UIテストの所要時間を短縮することは可能である。しかし、そうすることで（割愛したテストケースに起因する）障害発生のリスクが付きまとう。

UIテストの所要時間を（要求される品質を担保した上で）短縮するためには、テストケースを分割し、複数台のPCで並列実行するしかない。

本発表では、ラズベリーパイを使った時間短縮のソリューションについて紹介する。

テストクライアント

本発表で用いる「UIテスト」や「自動テスト」とは、統合テストやシステムテストで実施する画面を絡めた検証作業のことを指す。そこにバッチや帳票は含まれない。



UIテストを実施する端末のことを、ここでは「テストクライアント」と定義し、まずはその調達方法について検討する。

- 開発者PCの流用
- テスト専用PCの調達
- クラウドサービスの利用
- ラズパイの利用
- ソリューションの比較

【テストクライアント】開発者PCの流用

UIテストを自動化している案件において、開発作業に使っているPCをテストクライアントとして流用すると、次のようなトラブルに見舞われることがある。

- WebブラウザーやExcelが自動的に起動し、マウスカーソルやキーボードのフォーカスが奪われてしまう
- CPUの負荷が一時的に増大し、PCの反応が極端に鈍くなる
- 意図せずに自動テストの実行を阻害してしまい、作業成果をCIサイクルに反映できない
- 自動テストが実行中であるため、PCをシャットダウンすることができない

UIテストの実施時期は、開発者がバグの調査や修正に追われるタイミングとピッタリ重なる。開発作業と自動テストが相互に干渉することで、混乱に拍車がかかる。

【テストクライアント】テスト専用PCの調達

テスト専用PCを用意し、開発環境と分離することで、開発者の負担はかなり軽減される。しかし、問題がすべて解決するわけではない。次のような事例をよく見聞きする。

- 貴重なオフィス空間の一部をテスト専用PCが占拠する
- テストケースが増大するのに伴い、端末の数が不足する
- OSやアプリのアップデート等で管理者の負担が増大する
- まだ必要なのに、他のプロジェクトにリソースを奪われる

プロジェクトの予算が潤沢であれば苦労しないが、実際には限られたリソースでやり繰りする必要がある。スケジュールに遅延が発生した場合、リソース計画が土台から崩れてしまうこともある。

テスト専用PCは、繁忙期には台数が不足するのに、それ以外の時期には役立たずな存在で、むしろ持て余してしまう。

【テストクライアント】クラウドサービスの利用

クラウドサービスには次のような特徴がある。

- デバイスやOSの種類を任意に*1選ぶことができる
- セキュリティパッチ等の管理の手間が不要である
- Webで完結し、ハードウェアを意識しなくて済む
- 必要な数の端末をオンデマンドで利用できる

自前でテストクライアントを調達するのに比べると、多くのメリットがある。しかし、ほとんどのサービスは従量課金制を採用*2しているため、計画遂行には精緻さが求められる。

全てが計画通りに進めば、コストを大幅に抑制できる。ただし、機密保持契約に抵触したり、特殊なデバイスやソフトウェアに依存する場合、利用できないこともある。

*1 すべてのデバイスやOSのバージョンが利用できるわけではない。
*2 継続的デリバリーが必要な開発案件ではサービスの運用コストにも影響する。

【テストクライアント】ラズパイの利用

もうひとつの選択肢として今回紹介するラズパイには次のような特徴がある。

- 本体だけなら 5,000円 以下で買える
- 使わない時は引き出しの中にしまっておける
- ソフトウェアだけでなく、ハードウェアも制御できる
- 技術情報が世界中のユーザーから発信され続けている

”エンジニアのオモチャではないか “と揶揄する声もあるが、単なる思い込みに過ぎない。初期のハードウェアに比べ、最新モデルは格段に進化している。

テストクライアントとして十分に役立つどころか、パソコンには難しい芸当を難なくこなせるユニークな能力が備わっている。

【テストクライアント】ソリューションの比較

それぞれのソリューションを比較すると下表のようになる。

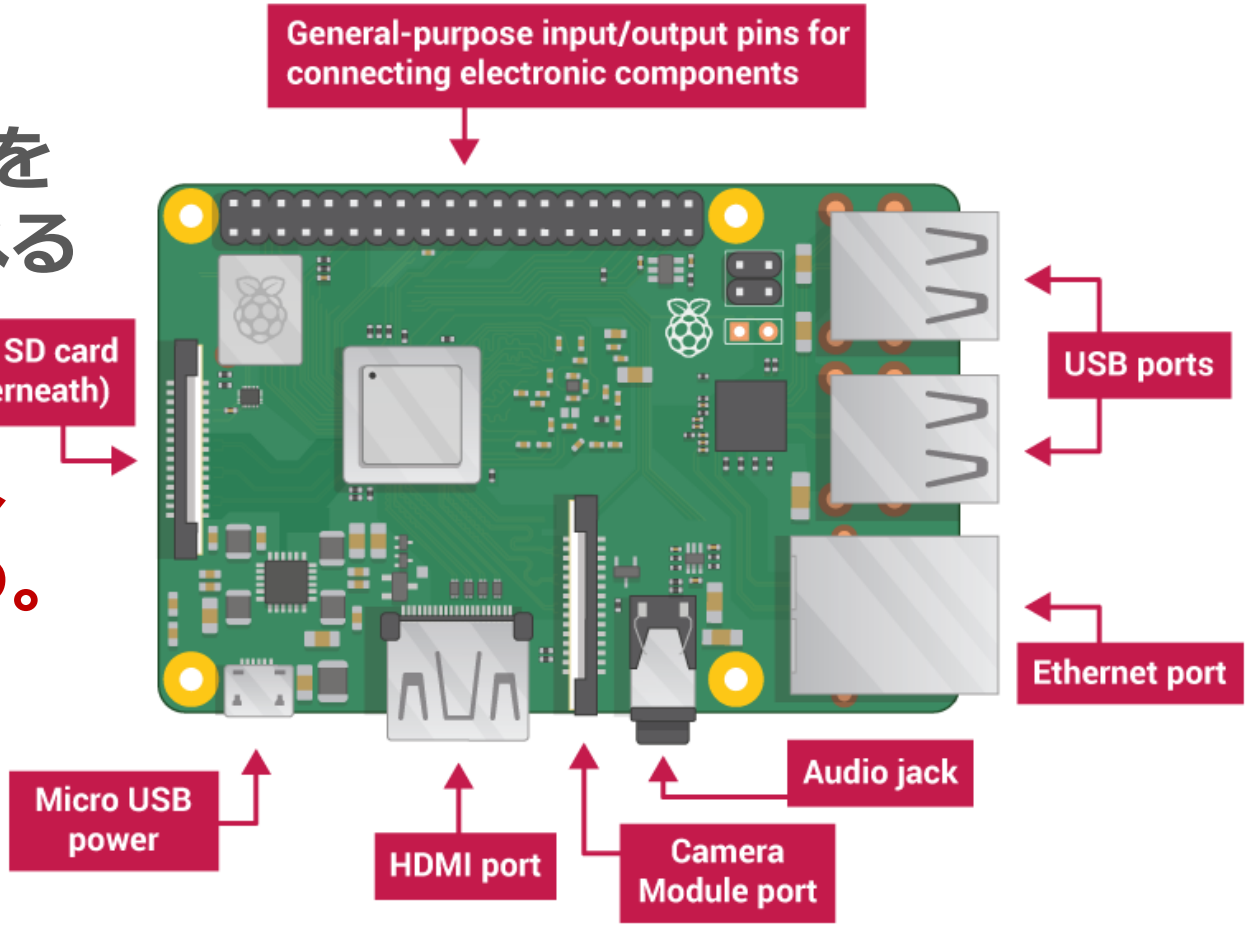
	開発者PCの流用	テスト専用PCの調達	クラウドサービスの利用	ラズパイの利用
通常業務への影響	×*1	○	○	○
CIサイクルへの影響	×*1	○*2	○*3	○*5
再帰テストの実施	×*1	△*2	△*3	○*5
テスト計画の柔軟性	×*1	△*2	△*3	○*5
デバイス選択の自由度	△	○*2	△*4	△
オフィス空間の有効活用	○	×	○	○
導入コスト	○	×	△	△*6
運用コスト	○	×	×*3	○*6

*1 バグの修正対応に追われている開発者の作業効率を著しく低下させてしまう。
*2 ピーク時の負荷に合わせて必要な機材を調達するのは現実的でない。
*3 慎重に計画しないと、確保していた予算をあっという間に使い果たしてしまう。
*4 特定のデバイスやソフトウェアに依存する場合、調達先の見直しを迫られる。
*5 安価で設置場所にも困らないため、必要な台数を調達できる。
*6 PCの調達に比べれば、必要なコストは無視できる範囲内に収まる。

ラズベリーパイ（ラズパイ）

Raspberry Pi は、英国のラズベリーパイ財団が開発した名刺サイズのシングルボードコンピューターである。**安価で拡張性に優れており、Linux の豊富なソフトウェア資産を利用することができる。**

Intel Core プロセッサを搭載したパソコンに比べると非力な感は否めないが、**最新モデルなら、テストクライアントとして十分な性能を持つ。**



- 基本スペック
- パフォーマンス

【出典】https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started/3

【ラズベリーパイ】基本スペック

「Raspberry Pi 3 Model B+」のスペックは下記の通り。

CPU	1.4GHz クアッドコア Cortex-A53
GPU	デュアルコア VideoCore IV OpenGL ES 2.0、OpenVG、24G FLOPSの性能
RAM	1GB DDR2 450MHz 低電圧 SDRAM
HD/SSD	なし（micro SD カード）
Ethernet	10/100/1000 Base-T（最大転送速度は300Mbps）
WiFi	IEEE 802.11 b/g/n/ac 2.4/5GHz
Bluetooth	Bluetooth 4.2, Bluetooth Low Energy
USB	USB 2.0 × 4
Video	HDMI、コンポジット 3.5mm 4極ジャック、DSI
Audio	3.5mm 4極ジャック、HDMI、I2Sピンヘッダ
GPIO	5V、3.3V、GND、UART、I2C、SPI、I2S、PWM
Power	Micro USB 5V 2.5A / 2.54mm ピンヘッダ / PoE
OS	Raspbian、Ubuntu MATE、Windows 10 IoT Core など

X-Windowも快適

SDカードの性能が全体のパフォーマンスに大きく影響

高速なネットワーク

USBデバイスの活用

ハードウェアHack

Debianベース

【出典】https://raspberry-pi.ksyic.com/news/page/nwp.id/75

【ラズベリーパイ】パフォーマンス

クワッドコアのCPUを搭載したラズパイ3であれば、GUI環境でのWebブラウジングも快適にこなせる。

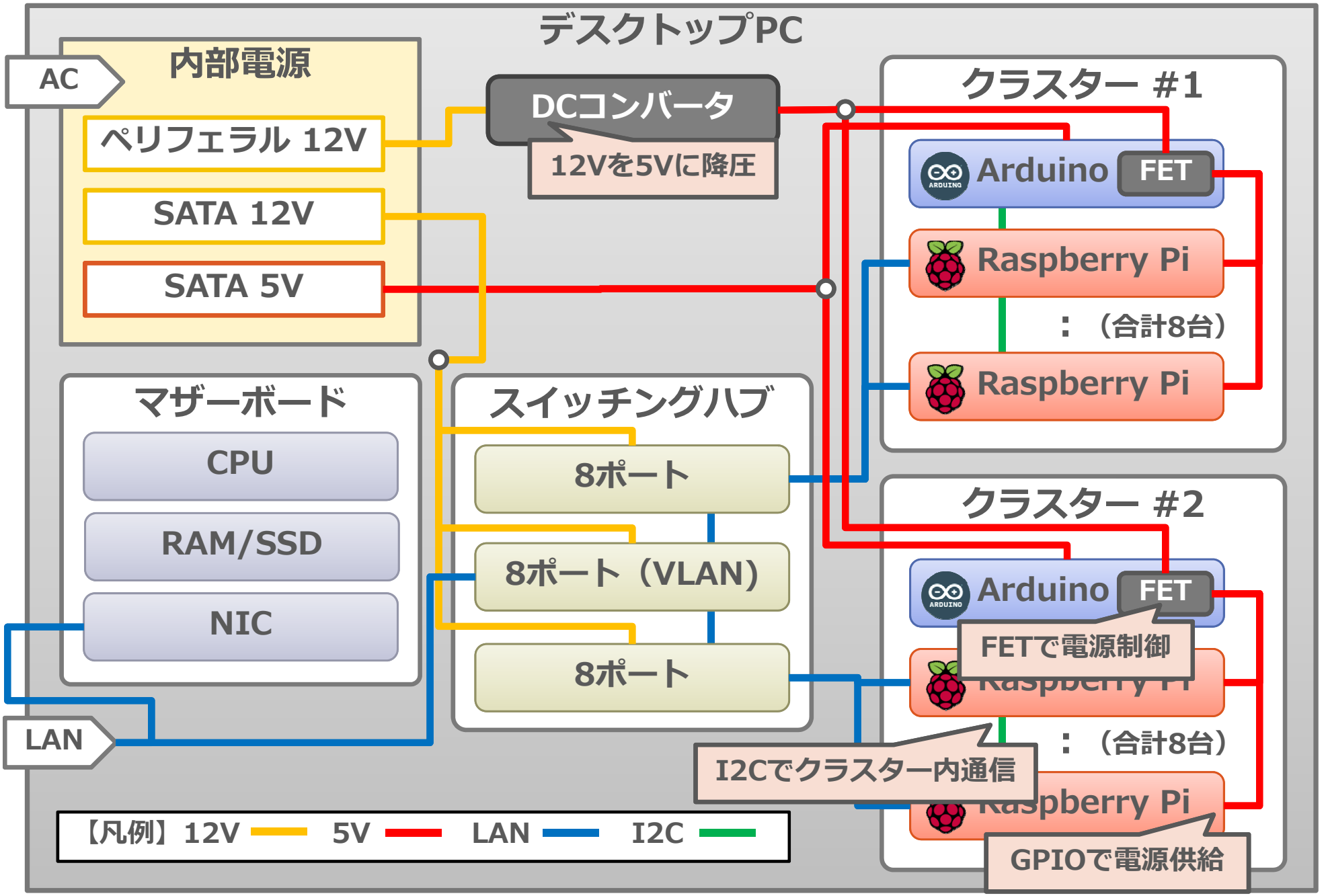
Chrome/Chromiumに WebDriver を導入し、複数の画面にまたがるテストスクリプトの所要時間を計測した。

	サーチエンジン	ECサイト
自作PC（Windows 10 Pro） 【CPU】Core i5 【RAM】16GB	19.187 秒	483.297 秒
MacBook 2017（macOS High Sierra） 【CPU】Core i7 【RAM】16GB	18.587 秒	503.101 秒
Raspberry Pi 3 Model B+（Raspbian） 【CPU】Cortex A-53 【RAM】1GB	22.115 秒	561.414 秒

ハードウェアのスペックには大きな違いがあっても、速度差がほとんど見られない点が興味深い。実際にはサーバー通信に多くの時間が割かれている。**UIテストの実施に足りていないのは（端末の数であって）CPUのパワーではない。**

テストクライアントとして利用する場合、今日の標準的なパソコンは明らかにオーバースペックである。ある一定の基準を越えれば、むしろ「数こそが力」となる。

クラスター構築


デスクトップPC
内部電源
AC
ペリフェラル 12V
SATA 12V
SATA 5V
DCコンバータ
12Vを5Vに降圧
クラスター #1
Arduino
FET
Raspberry Pi
：（合計8台）
Raspberry Pi
マザーボード
CPU
RAM/SSD
NIC
スイッチングハブ
8ポート
8ポート（VLAN）
8ポート
クラスター #2
Arduino
FET
FETで電源制御
：（合計8台）
I2Cでクラスター内通信
Raspberry Pi
GPIOで電源供給
LAN
【凡例】12V 5V LAN I2C

【クラスター構築】電源供給

クラスター用の電源を外部に求めず、**デスクトップPCの内部電源から調達することが可能である。**実際のクラスター構築で使用した550Wの内部電源のスペックは下記の通り。

	供給電流	供給電力	SATA	ペリフェラル
3.3V	0～18A	90W		N/A
5V	0～15A		Max 1.5A	Max 10A
12V	0～45.8A	549.6W		

8台のラズパイで最大100W

【出典】http://www.enermaxjapan.com/Revolution-SFX_Series/ERV550SWT-ERV650SWT_spec.html

4ノードのクラスターであれば、ペリフェラルの5Vをそのまま電源として利用できる。しかし、それ以上の規模する場合、**ペリフェラルの12Vを5Vに降圧する**必要がある。

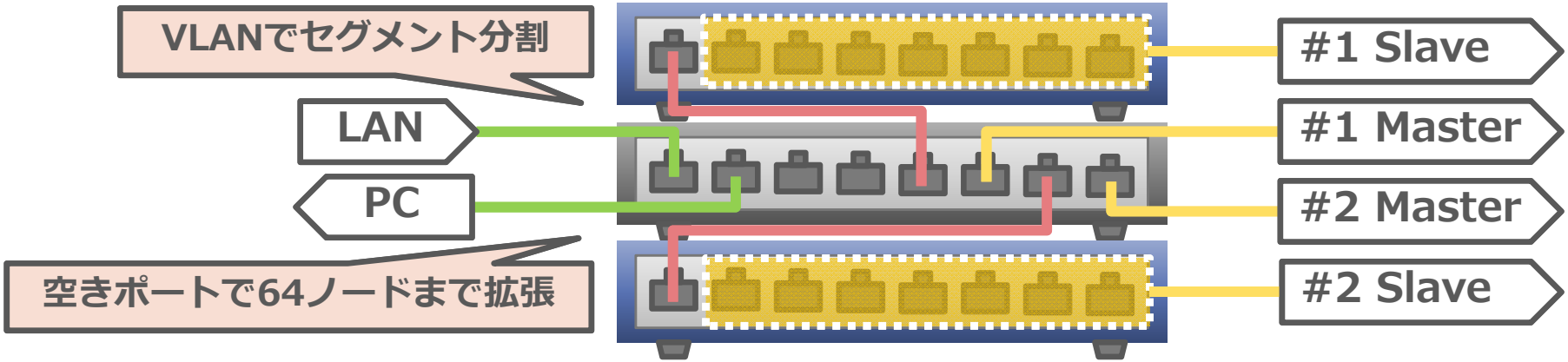
USBハブ経由でラズパイに電源を供給すれば、クラスターの構築は極めて容易になる。しかし、ACアダプターも別途必要になるため、**デスクトップPCの筐体内にクラスターを格納し、両者を同時に稼働させることが難しくなる。**

【クラスター構築】ネットワーク

最新の「Raspberry Pi 3 Model B+」では、高速な5GHz帯の通信規格「IEEE802.11 ac」をサポートしている。しかし、クラスターを構成するノードの数（ラズパイの台数）が多いことが災いして、次のような状況に陥る。

- 無線LANでは、アクセスポイントがパンクする
- 無線LANをOFFにして、有線LANのみを使う
- 有線LANでは、スイッチングハブのポートが不足する

クラスターの各ノード（ラズパイ）が利用するための**スイッチングハブもデスクトップPCの筐体内に格納する。**



【クラスター構築】RASPBERRY PI

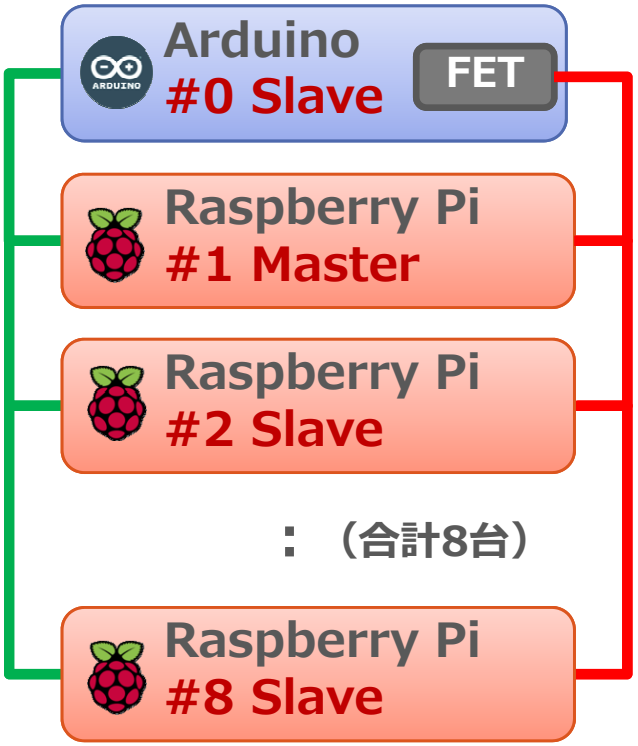
クラスターの最小ユニットは、1台の Arduino と8台の Raspberry Pi から構成される。共通タスクは次の通り。

- GPIO経由での電源供給（Arduinoがバルブの役割）
- I2Cによるクラスター内通信

マスターノードだけは常時稼働*1させる。マスターノードの固有タスクは次の通り。

- 電源制御（boot/reboot/halt）
- HTTPサーバー
- DBサーバー
- Dockerレジストリーサーバー
- Flannel

クラスター内部では、独自のテキストコマンドを使ってI2Cで通信

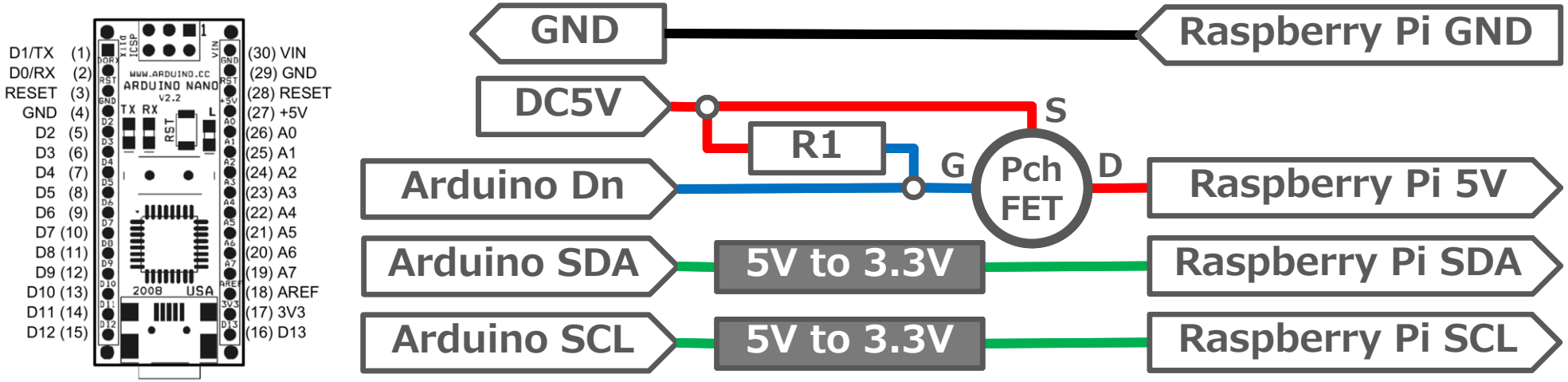


*1 Arduinoは TCP/IP を直接扱えないため。

【クラスター構築】ARDUINO

Raspberry Pi の最大の弱点は、電源管理機構が存在しないことである。OSをシャットダウンしても電力は供給され続け、OSを起動するのには電源ケーブルの抜き差しが必要になる。

この問題を解決するため、Aruduino*1を利用する。
クラスターを構成する各ノードの電源をAruduinoに制御*2させ、必要に応じて稼働する台数をスケール*3する。

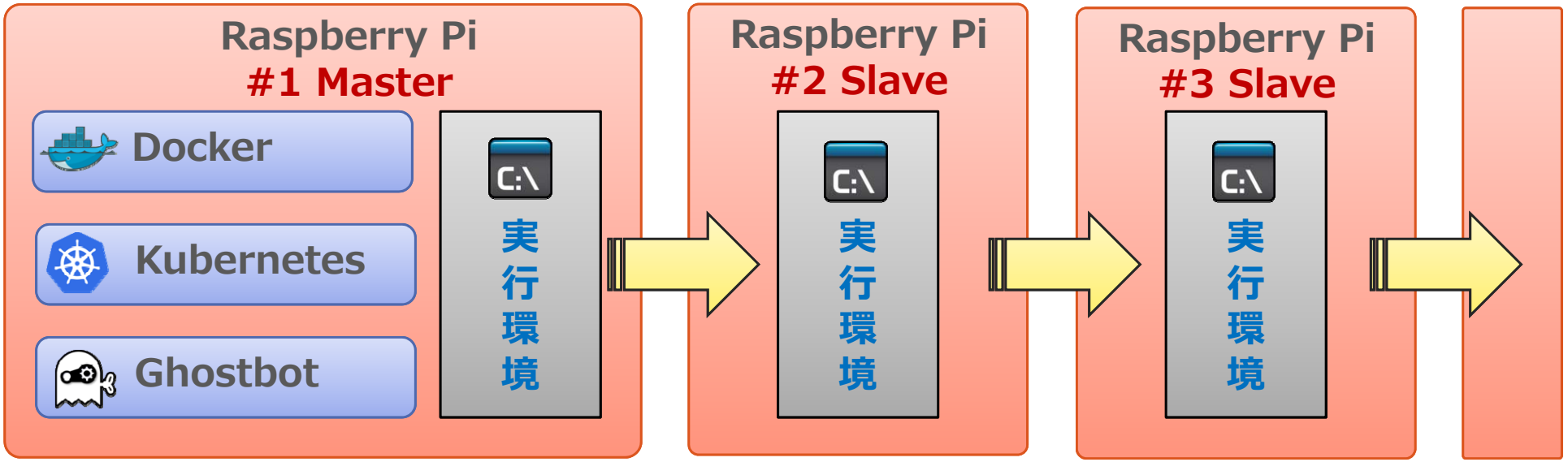


【出典】https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf

*1 省スペースでユニバーサル基板に搭載可能な「Arduino Nano」を利用する。
*2 I2Cによる通信とMOSFETを使ったスイッチングで動的な電源管理を実現する。
*3 HTTPサーバーをマスターノードで稼働させ、Webブラウザーからも制御可能にする。

ミドルウェア

ハードウェアの次はソフトウェアの環境整備である。クラスターは複数のノードから構成されるため、UIテストの実行環境を個別にセットアップするのは得策ではない。ミドルウェアを使って環境整備の手間を軽減する。



- Docker
- Kubernetes
- Ghostbot

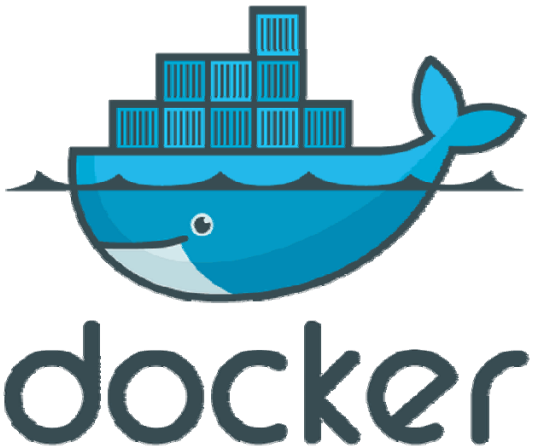
【ミドルウェア】DOCKER

Docker は、Linuxカーネルの機能を利用したコンテナベースの仮想化技術である。VirtualBox などのハイパーバイザー型の仮想化技術に比べて、実行時のオーバーヘッドが少なく、イミュータブルであるという特徴を持つ。

テスト実行によって汚染された環境を（リカバリーすることなく）破棄するというアプローチにはメリットも多い。

- 特定の開発タスクではなく、複数のタスクを切り替えたい
- タスクを終えたら、すぐにクリーンな状態に戻したい
- 構成変更をできるだけ迅速に行いたい

クラスター上でUIテストを実施する際、Docker の持つイミュータブルな特性により、管理コストを大幅に削減できる。



【ミドルウェア】KUBERNETES

Kubernetes は、Docker などのコンテナをクラスター環境で運用するためのオーケストレーションツールである。

もともとは Google Cloud Platform などのクラウドサービス向けに開発されたツールだが、ラズパイ・クラスターでも問題なく利用できる。

- 複数のコンテナを一括デプロイ（スケジューリング機能）
- コンテナのバージョン管理（ロールバック制御）
- コンテナの自動スケーリング（メトリクス指定）
- コンテナのローリングアップデート

Docker にない機能を Kubernetes で補完することで、UIテストの並列実行がオーケストレーション可能になる。

【ミドルウェア】GHOSTBOT

Ghostbot は、Webアプリとスマホアプリの両方に対応*1した自動テスト開発支援ツールである。まだ開発途上ではあるが、開発の現場で重宝する様々な機能を搭載している。

Pythonライブラリ以外に、次の機能を提供予定である。

- 統合コンソール
- スケジューラー
- エビデンス管理
- 画像認識
- 音声認識
- 日本語OCR
- レコーダー

```
# テストスクリプトのサンプル（一部抜粋）
@action(“/”)
def top(self):
    self.open(self.scenario["url"])
    self.validate(":top_page")
    customer = self.container.select(":customer_name")
    if self.resource(":not_login") in customer.text:
        yield "/login"
    yield "/header"
    yield "/footer"
    yield "/carousel"
    yield "/search"
    yield "/logout"
```

エビデンスの自動取得
コンテンツの動的解析

URIベースのテストスクリプト管理

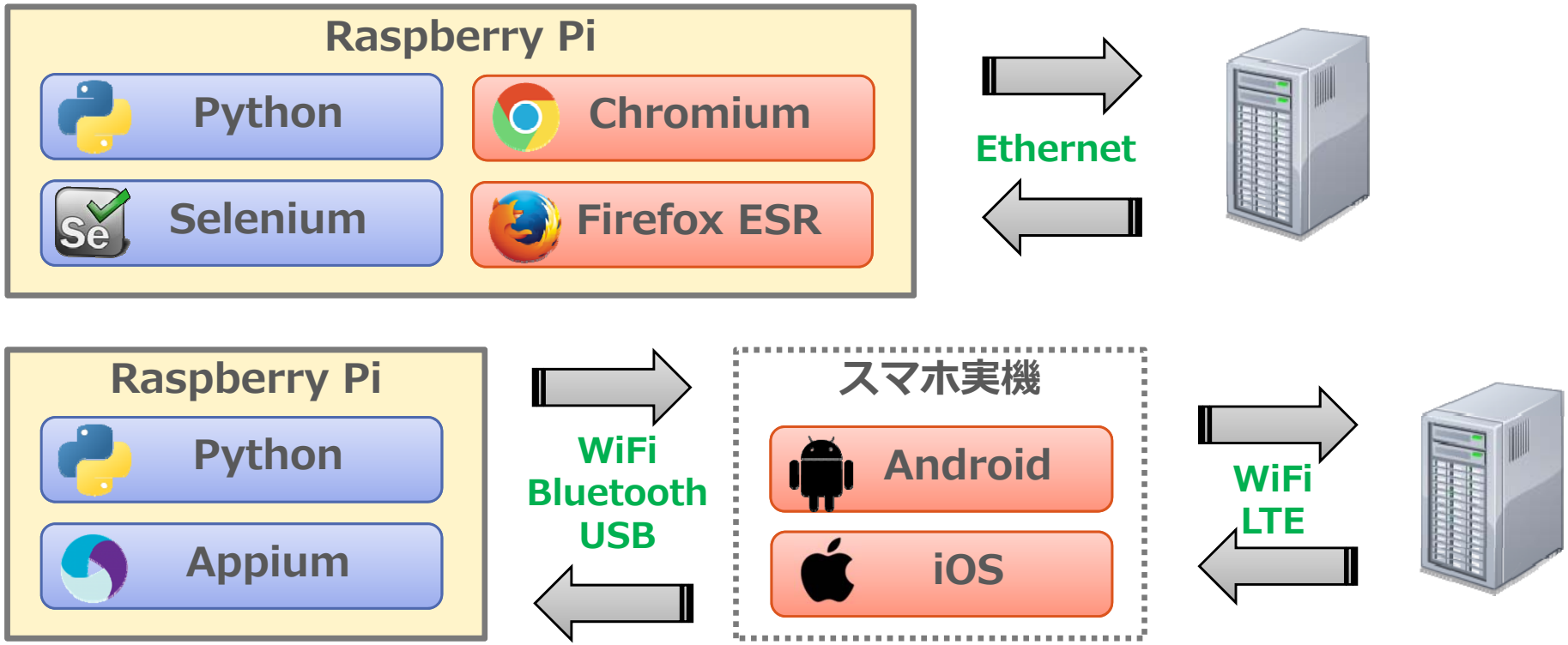
yieldコントロールにより、テストシナリオの実行粒度を動的に制御

ghostbot

*1 デバイス制御のため、内部では Selenium や Appium を利用している。

応用事例

ラズパイを使ってUIテストを実施する場合、どのような点に注意すべきか？ 重要なポイントに絞って解説する。



- Webアプリ
- スマホアプリ

【応用事例】WEBアプリ

WebアプリのUIテストをラズパイで行う場合、Firefox または Chromium が順当な候補となる。

	ARMv7 Binary	WebDriver	Headless
Firefox ESR	○	△*1	×
Chromium	○	△*1	○

実際の開発案件は、パソコンやスマホからの利用を想定している。ラズパイでの動作検証は、あくまでも補助的な位置づけに過ぎない。

- ラズパイでのUIテストは機能検証を目的とする
- 実機上での動作検証*2は必要不可欠である

上記のような制約があっても、膨大な数のテストケースをラズパイ・クラスターで並列実行できるメリットは大きい。

*1 最新版ではなく、使用するWebブラウザーに適合するバージョンを導入する必要がある。
*2 JavaScriptの挙動の変化や搭載フォントの違いによる表示崩れが発生する可能性がある。

【応用事例】スマホアプリ

実機を使ったスマホアプリのUIテストでは、ラズパイの拡張性の高さとフットプリントの小ささを活かして、パソコンでは実現困難なタスクを解決できるかも知れない。

	CSI カメラ	USB	Bluetooth
画像解析	△*1	○*2	×
音声解析	×	○	○*4
リモート制御	×	△*3	○*5

ラズパイにはUSBポートが4基搭載されており、Bluetoothの入出力チャネルを増やすこともできる。**OSSのメリットを最大限に活用し、様々なデバイスに擬態できるのも Linuxエコシステムならではの強みである。**

*1 720Pで30fpsの取り込みが可能だが、固定焦点のため微調整が欠かせない。
*2 USB接続のビデオキャプチャー機器が利用できるが、USB接続のWebカメラの方が便利。
*3 バスパワーだけではバッテリーの充電速度が消費速度に追いつかない。
*4 Bluetoothスピーカーとして認識させることで、音声入力が可能になる。
*5 HIDデバイスやコントローラーとして認識させることで、Bluetooth経由で操作できる。

自動テストの開発資産

UIテストを効率よく並列実行するためには、既存の開発資産についても、以下のような見直しを行う必要がある。

- テストスクリプトを可能な限り疎結合にする
- テストデータをアカウント毎に用意する
- テストシナリオとテストスクリプトを協調させる
- Cookie/LocalStorageを複製する

Single Page Application や AltJS で構築されたWebアプリの並列実行は一筋縄ではいかない。Ajax通信がメインとなるため、コンテンツが更新されてもURLが変化せず、加えてクライアント側でも状態を保持しているからである。

このような場合、クライアントの状態をスキャンするための JavaScript コードを動的に注入し、Webページを複製する。

パーソナルクラウド

UIテストの時間短縮という目的のためにラズパイ・クラスターを製作したが、数世代前のスパコンに相当する計算能力を好きな時に利用できるメリットは、想像した以上に大きい。

- UIテスト以外の用途にも使える
- 外部のラズパイ・クラスターと連携できる
- 休眠状態のGPUをGPGPUとして活用できる

パーソナルコンピュータをインターネットに接続することで全く新しい付加価値が生まれたように、パーソナルクラウドが相互接続されることによって、新たな創造と破壊*1が繰り返される可能性がある。

ラズパイ・クラスターは、ふと思いついた実験的なアイデアを試すのに最適な道具である。

*1 わずか16ノードだと過小評価すべきではない。会社が所有する全てのデスクトップPCの台数に、16または32を掛け合わせてみて欲しい。

乞うご期待

現在も（気の趣くまま）雑多な開発タスクに取り組んでいる。
近々その成果を発表できそうなもの*1は次の通り。

- **スマホアプリのUIテスト**
  ラズパイのUSBポートに接続したWebカメラでスマホ画面を撮影し、リアルタイムに画像解析しながらUIテストを自動実行する。
- **クラウドコンピューティング**
  ラズパイ・クラスターで構築した自前のデータセンターを使って、サーバーの台数を1/3に削減する新たなアーキテクチャー＆ミドルウェアの実証実験を行う。
- **パーソナルアシスタント**
  ラズパイにUSBドングル型のマイクと液晶ディスプレイを接続し、（ベンダーの縛りを受けずに）自由に拡張できる持ち運び可能なスマートスピーカーを作る。

*1 コラボレーション可能なパートナー企業を募集中。

おわりに

平成も終わり、5月からは新たな元号が始まります。

３つもの元号をまたぐことになる自分自身に驚きつつも、これからも現役のエンジニアとして活躍できるように、さらに精進したいと思います。

下記のWebサイトでラズパイ・クラスターに関する技術情報を公開します。興味を持たれた方は、ぜひ参考にしてみてください。

【URL】http://metaworks.io/raspi-cluster

ご静聴ありがとうございました。