

Armadillo Base OS 基礎セミナー (VScode版)

株式会社アットマークテクノ

www.atmark-techno.com



準備編

- 第1部 Armadillo Base OSについて
- 第2部 Armadilloの準備

開発編

- 第3部 Armadilloの設定
- 第4部 アプリケーション開発
- 第5部 ソフトウェアアップデート設定
- 第6部 量産準備

運用編

- 第7部 Armadillo Twinについて

第1部：Armadillo Base OSについて



- ・ IoT機器を運用する為に考える事
- ・ Armadillo Base OSの概要

IoT機器向けのOSとは？

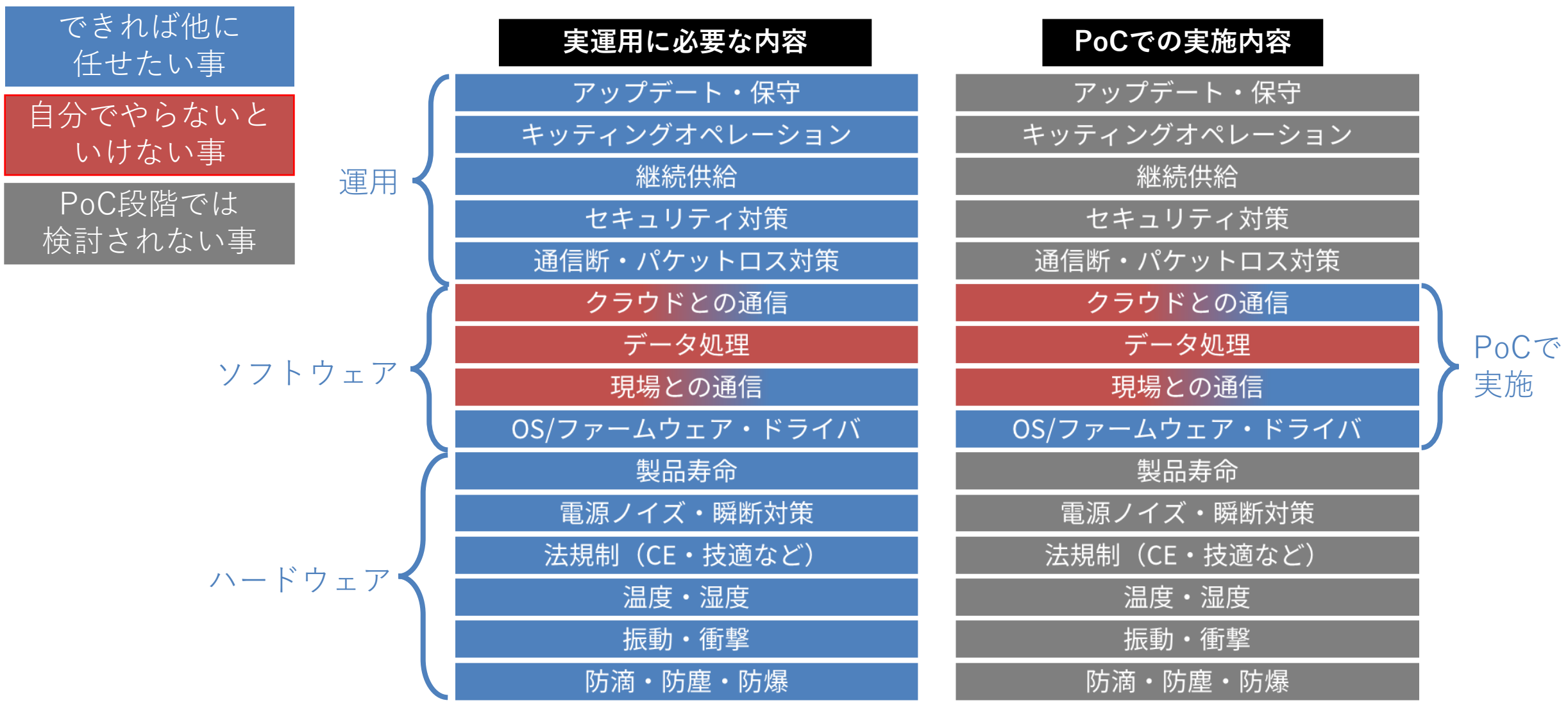
IoT機器にはDebian/Ubuntu由来の汎用ディストリビューションが多い

IoT機器から見た汎用ディストリビューション

- **デスクトップ用途**や**サーバ用途**に便利に作られている
⇒IoT機器としては不要なものが多い（脆弱性の懸念）
- **OSのサイズ**が大きすぎる
⇒安さが求められるIoT機器はストレージ容量が限られる
- **アップデート機能**や**リカバリ機能**が無い
⇒IoT機器が脆弱性に対応していく為には必須
- OSの**サポート期間**が決まっている(5年程度)
⇒IoT機器は5年では運用期間が短すぎる

長年に渡って運用するIoT機器向けのOSには向いていない？

IoT機器を安定運用する為には？



長期安定運用とセキュリティ対策が可能な
IoT機器向けのOSとして



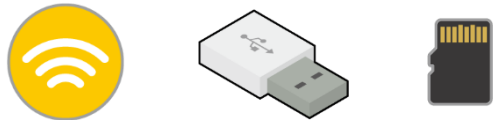
Armadillo
Base OS

を新たに開発しました

IoT機器特化の「Armadillo Base OS」

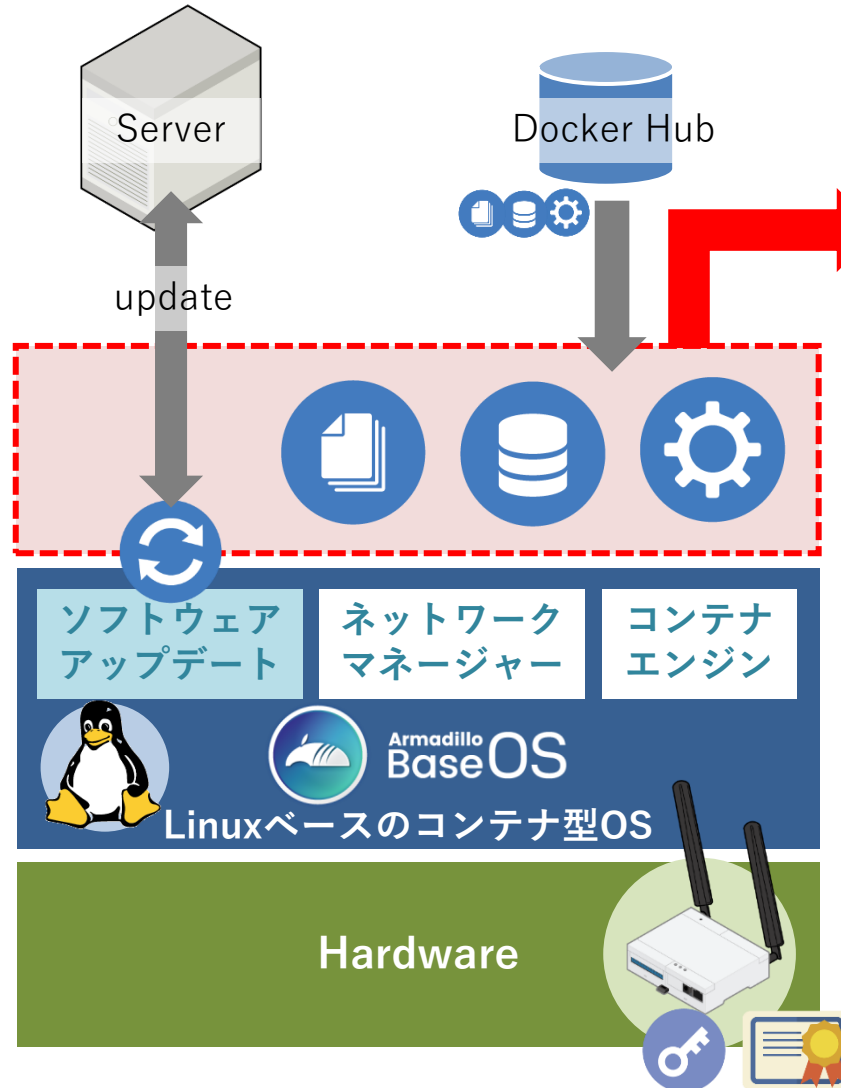
アップデート機能を標準搭載

- イメージ二重化でロールバック可能 (アップデート失敗でもリカバリー)
- 差分更新機能で最小データ転送
- 多様なアップデート手段 (ネットワーク/USB/SD)



長期で安定運用するために

- OSの長期メンテナンス
- 突然の電源断でも壊れにくい堅牢なファイルシステム
- プログラム領域のリードオンリー化
- 運用ログの記録



モダンなソフトウェア環境

- アプリ開発に集中できる (OS部分は最適化済み)
- Docker Hubのコンテナを利用可能
- (A6E)ゲートウェイコンテナを使ってIoTアプリも簡単に実現
- (G4/X2)GUIアプリはFlutterで開発

多面的なセキュリティ機能

- OSコンパクト化で脆弱性を限定
- コンテナ構造でサンドボックス化
- セキュアブートで改ざん防止
- セキュアエレメントで個体認証
- OP-TEEのセキュアな実行環境

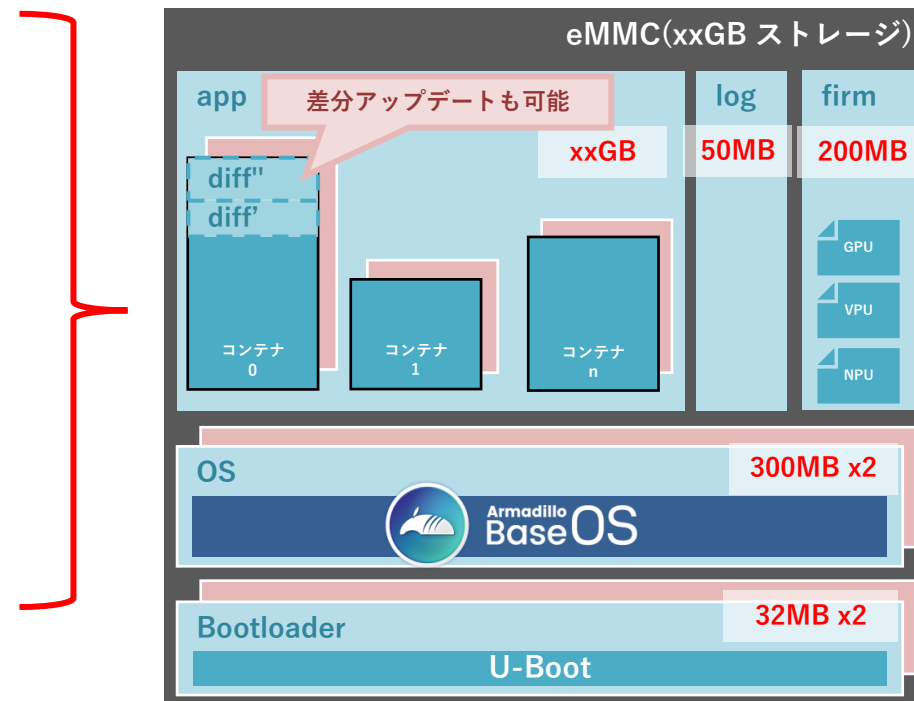
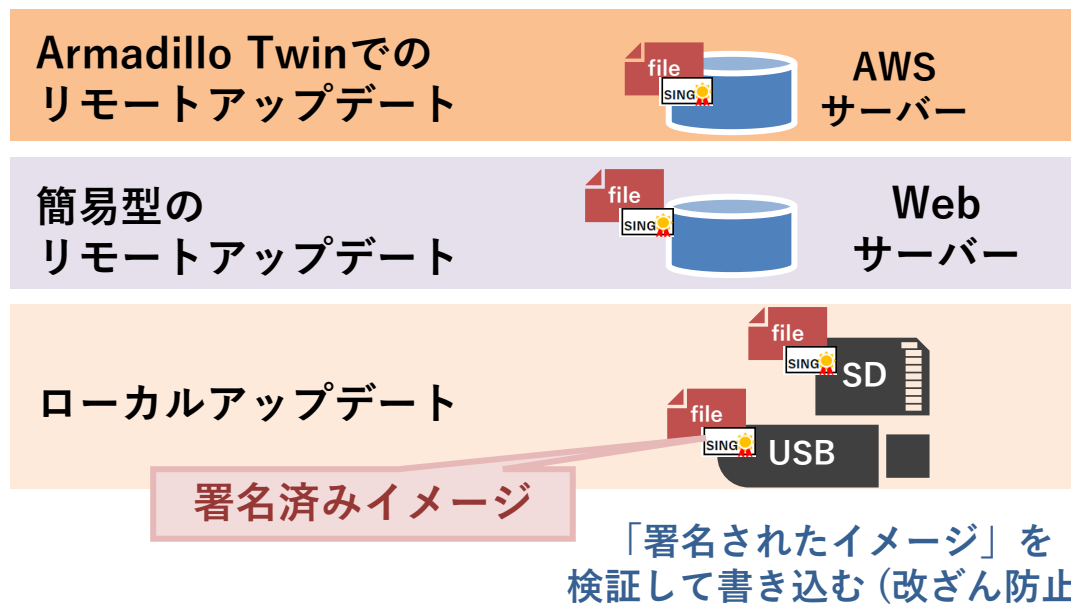
ソフトウェアアップデート機能

ソフトウェアアップデートはリモート/ローカルそれぞれでアップデートが可能

■ アップデート対象

ファイル単体からOSのみ、コンテナ単位やコンテナの差分アップデートも可能

■ アップデート方法



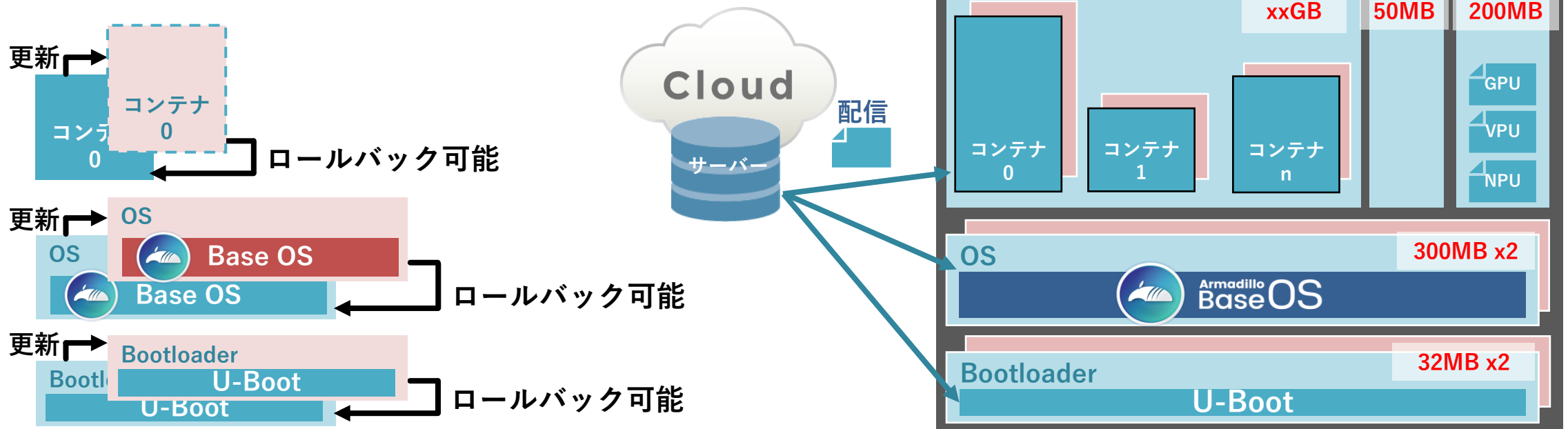
ロールバック機能

ロールバック機能は自動(または手動)でアップデート前のバージョンに戻って起動する仕組み

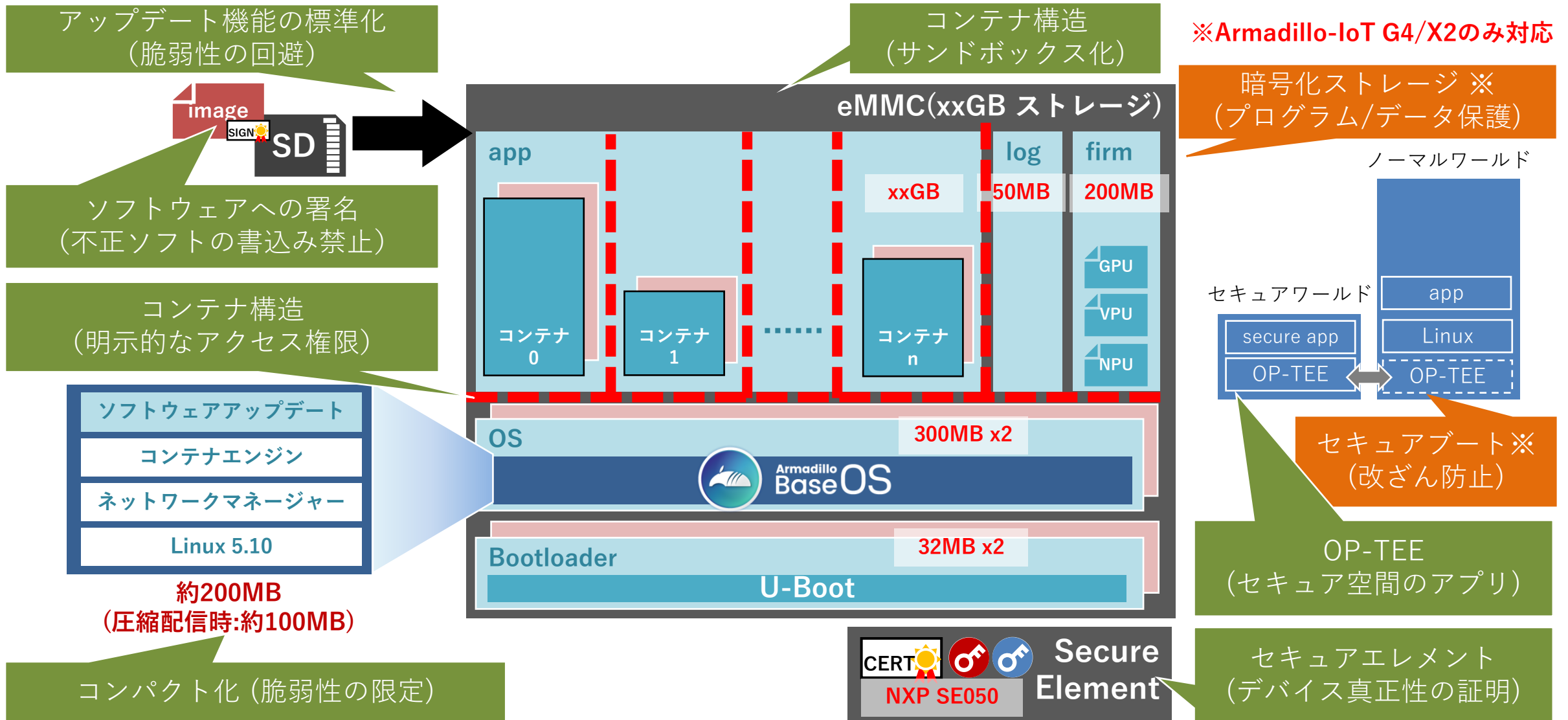
ロールバック条件

- ①起動時にカーネルとdtbファイルが存在しない場合
- ②アップデート直後に3回起動を試してLinuxが正常に起動しなかった場合
- ③abos-ctrl rollback コマンドを実行した場合

Armadillo Base OSはコンテナ / OS領域 / Bootloader領域は二面化により1回分のバックアップが存在し、**ロールバック機能によりリカバリが可能**



様々なセキュリティ対策



- IoT機器を運用する為に考える事
 - ・ 長期運用とセキュリティに対応
- Armadillo Base OSの特徴
 - ・ アップデート機能
 - ・ ロールバック機能
 - ・ 長期安定運用
 - ・ セキュリティ対策

第2部：Armadilloの準備



- ATDE環境の構築
- Armadilloとの接続
- Armadillo Base OSアップデート

- **Oracle VM VirtualBox** (※VMware Workstation Pro でも可)
用途：仮想化ソフトウェア ※既にライセンスをお持ちの場合
- **Atmark Techno Development Environment(ATDE)**
用途：VMwareで使用するLinux用の仮想開発環境
- **Visual Studio Code(VScode)**
用途：Armadilloの初期設定、アプリケーション作成
- **ABOS Web**
用途：ネットワーク設定、SWUインストール
- **at-dtweb**
用途：拡張インターフェース設定
- **Armadillo Twin**
用途：死活監視、リモートアップデートなどのサービス

コンソールの表記について

■ コンソール表記の使い分け

以後、本資料で記載するコマンドがどの環境で実施しているかを分かりやすくする為、下記の様にコンソール表記を使い分けて記載する

```
[armadillo]# //Armadillo Base OS(ホスト)でのコマンド  
[ATDE]$ //ATDE環境(PC)でのコマンド
```

- ・ コマンドは"水色"、コメントは"黄色"、その他は"白"とする
- ・ 水色のコマンド等はコピー&ペーストで実行可能
(例：資料のコマンドをctrl+Cでコピー、Tera Termでは右クリックでペースト)

```
[armadillo]# command //コメント
```


- **Armadillo Base OS搭載製品の開発セット**

- ・ Armadillo本体 + ケーブル、アダプタ類
- ・ 対象製品：Armadillo-IoT G4 , Armadillo-IoT A6E ,
Armadillo-X2, Armadillo-640, Armadillo-610

- **パソコン**

- **LAN接続可能なインターネット環境**

- (任意)USBメモリ(※)

- (任意)SDカード(※)

※ファイル保存用途、ローカルアップデート、インストールディスク作成で使用します。

任意と記載があるものは無くてもセミナー進行には影響はありません。

本セミナーでは仮想マシン用ソフトウェア VMware を使用し、当社で配布している ATDE を起動します。ATDE の環境構築方法は各製品マニュアルに詳細を記載しておりますので、セミナーでは割愛します。

また、以下の様に **VMware のネットワーク設定をブリッジ** にしておくことを推奨します。

- ・ **ATDE 起動前の場合**

[仮想マシン設定の編集] を選択し、[ハードウェア] > [ネットワークアダプタ] を選択、[ネットワーク接続] のラジオボタンを **ブリッジ** に設定します。

- ・ **ATDE 起動後の場合**

左上の [Player] > [管理] > [仮想マシン設定の編集] を選択し、あとは上記と同じです。

NAT でも問題無く開発は可能ですが、VScode 上で Armadillo が表示されません。
また、ネットワーク変更後は ping などでインターネットに接続出来ている事をご確認下さい

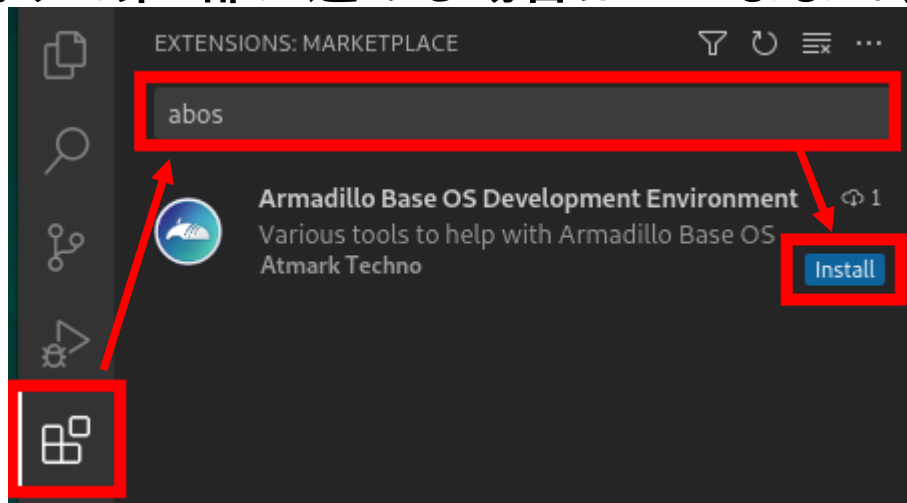
以降、ATDE で minicom の設定が完了した前提で進めます。

ATDE環境の構築

VScodeのエクステンションABOSDE(Armadillo Base OS Development Environment)をインストールします。まず、コンソールを起動して下記コマンドを実行します。

```
[ATDE]$ sudo apt update  
[ATDE]$ sudo apt upgrade  
[ATDE]$ code
```

VScodeが起動したら下記順に進めてABOSDEをインストールします。続けて第3部に進める場合はこのままで次に進みます。

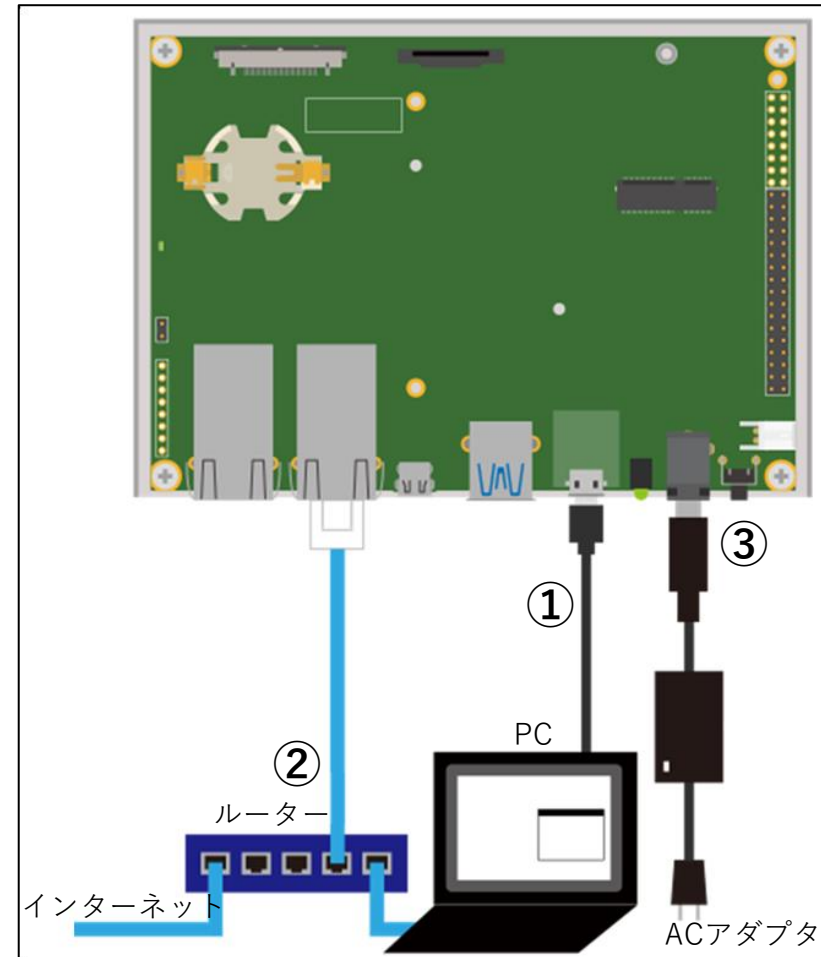


Armadilloとの接続

Armadillo-IoT G4 / Armadillo-X2 の場合

右図の様に下記を接続

- ①パソコンとArmadilloを**USB-USBmicroB**ケーブルで接続
- ②**LANケーブル**をルーター経由(※)でPCとArmadilloに接続
※インターネットに接続可能であること
- ③**ACアダプタ**のプラグ側をArmadilloを接続
※**ACアダプタのAC電源側はまだコンセントに接続しません**



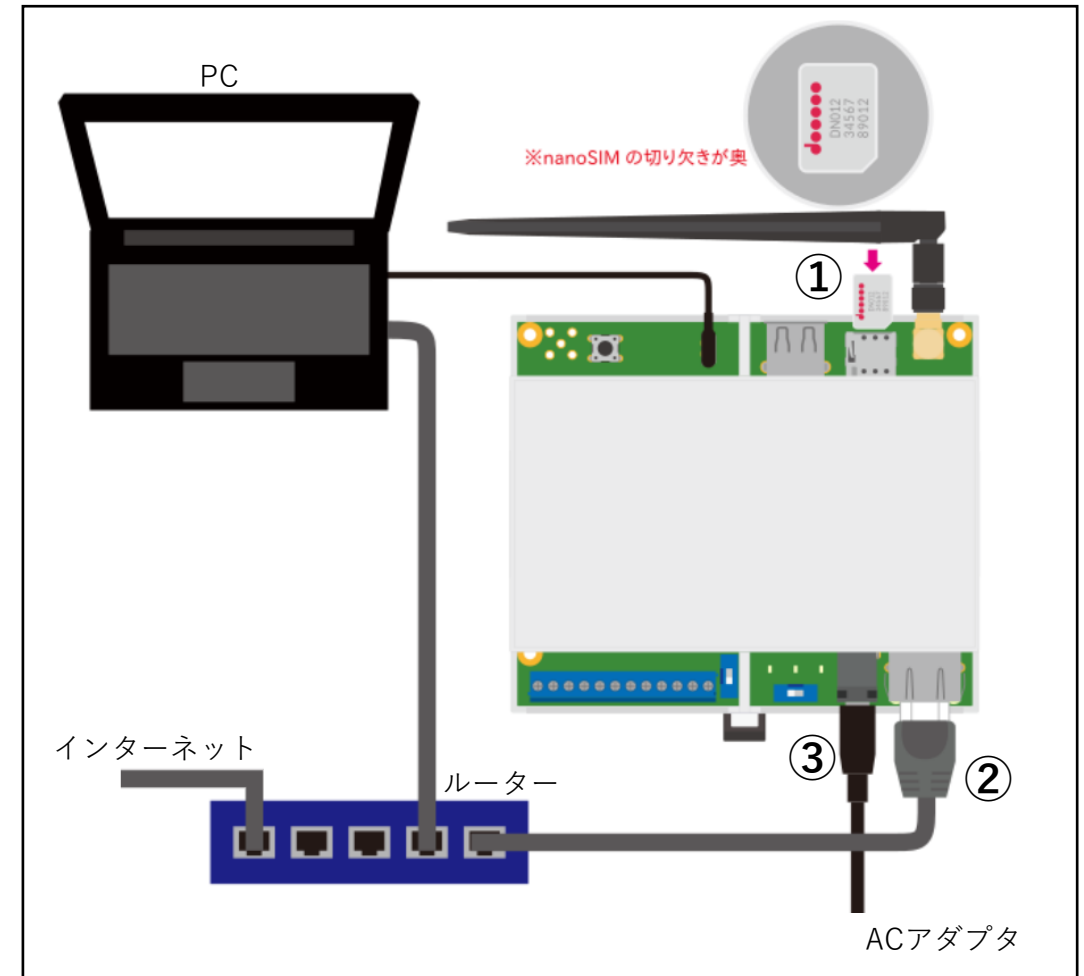
Armadillo-IoT G4接続図

Armadilloとの接続

Armadillo-IoT A6E の場合

右図の様に下記を接続

- ①パソコンとArmadilloを**USB-USBmicroB**ケーブルで接続
- ②**LANケーブル**をルーター経由(※)でPCとArmadilloに接続
※インターネットに接続可能であること
- ③**ACアダプタ**のプラグ側をArmadilloを接続
※**ACアダプタのAC電源側はまだコンセントに接続しません**



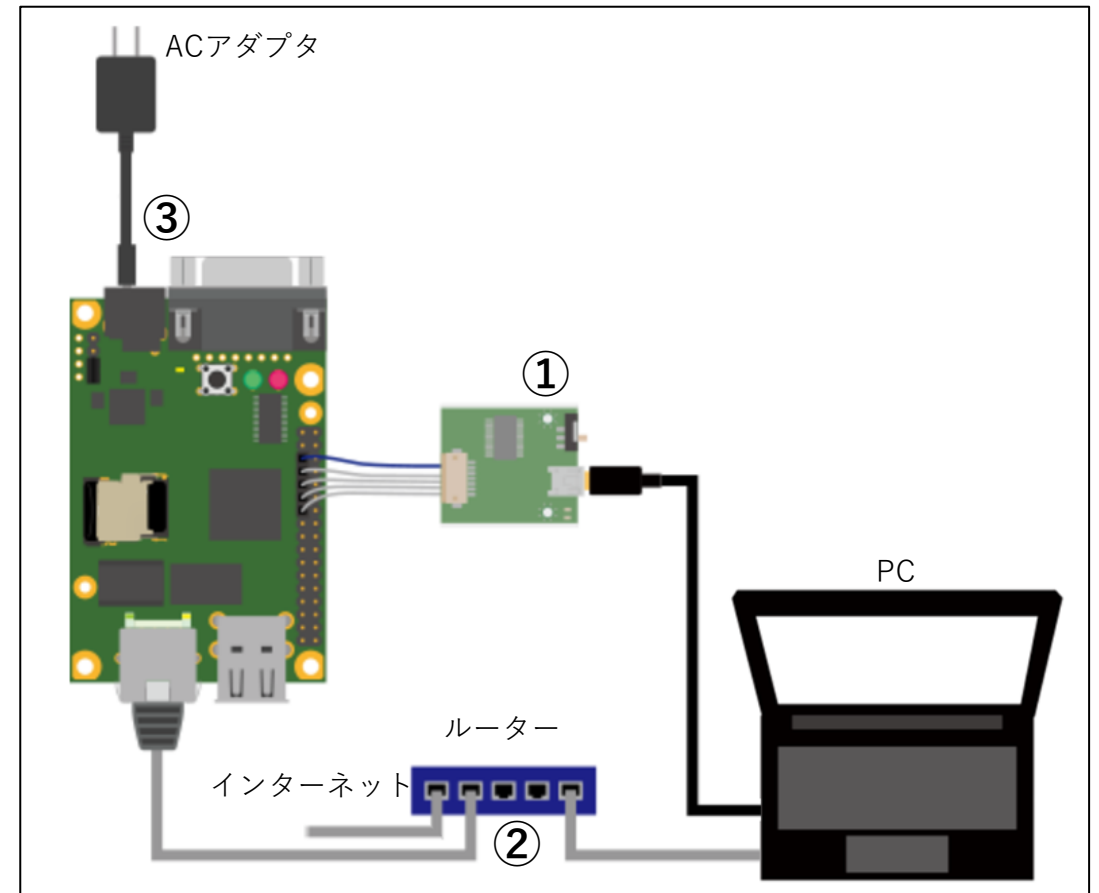
Armadillo-IoT A6E接続図

Armadilloとの接続

Armadillo-640 の場合

右図の様に下記を接続

- ①パソコンとArmadilloを**USB変換ケーブル**で接続
- ②**LANケーブル**をルーター経由(※)でPCとArmadilloに接続
※インターネットに接続可能であること
- ③**ACアダプタ**のプラグ側をArmadilloを接続
※**ACアダプタのAC電源側はまだコンセントに接続しません**



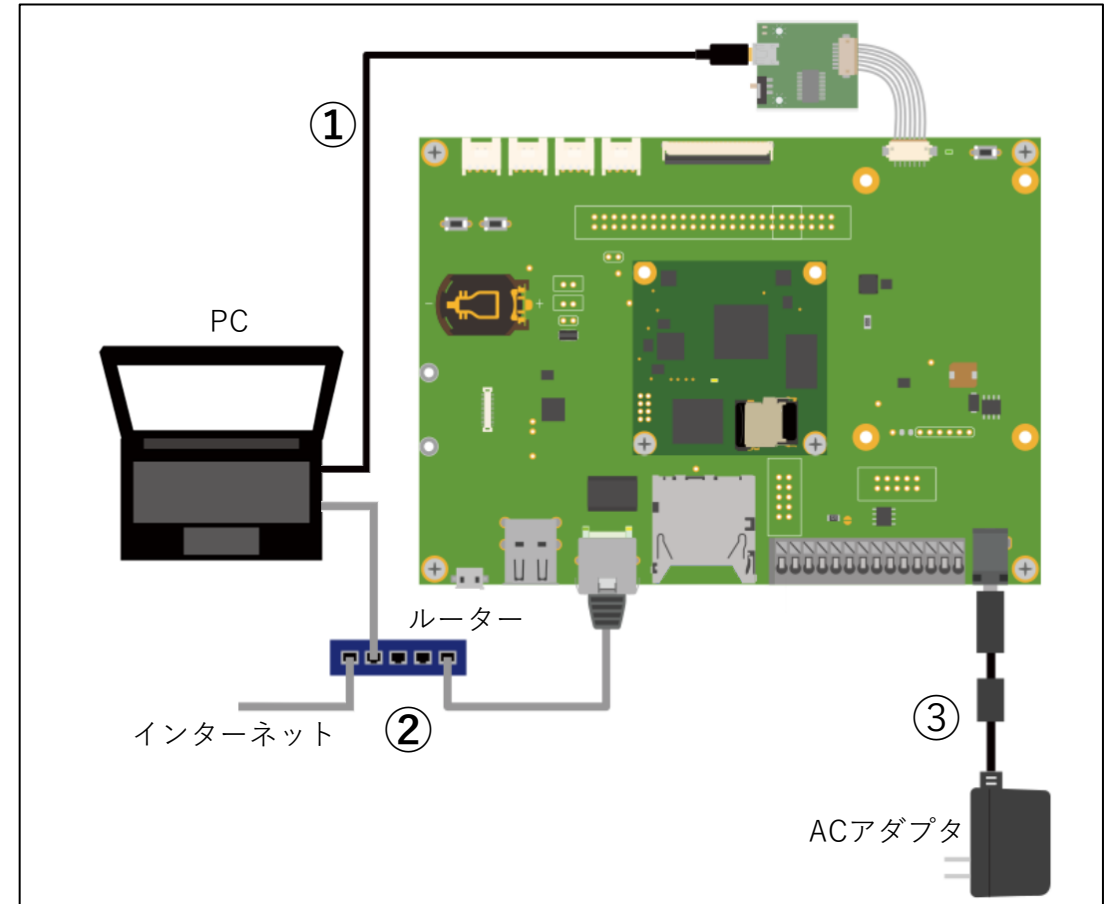
Armadillo-640接続図

Armadilloとの接続

Armadillo-610 の場合

右図の様に下記を接続

- ①パソコンとArmadilloを**USB変換ケーブル**で接続
- ②**LANケーブル**をルーター経由(※)でPCとArmadilloに接続
※インターネットに接続可能であること
- ③**ACアダプタ**のプラグ側をArmadilloを接続
※**ACアダプタのAC電源側はまだコンセントに接続しません**



Armadillo-610接続図

■ コンソールの表示について

開発時はminicomでコンソールを表示しておくことをお勧めします。(Tera Termでも可)
minicomを起動する場合は下記コマンドを実行します。
以後、コンソール有りのケースとしてご説明します。

```
[ATDE]$ sudo LANG=C minicom --wrap --device /dev/ttyUSB0 //minicom起動
```

環境によってはデバイスが **ttyUSB0** ではない場合がありますので、その時は
USBケーブルをPCに挿した後に下記コマンドで表示されるデバイスを指定します。
※下記は ttyUSB0 だった場合の例

```
[ATDE]$ sudo dmesg  
<中略>  
[10189.924460] cp210x 3-2:1.0: cp210x converter detected  
[10189.939602] usb 3-2: cp210x converter now attached to ttyUSB0
```


■ コンソールの表示について

AC側をコンセントに差し込むと電源が入り、Armadilloが起動します。
コンソールを起動している場合は起動ログが流れ、数秒後に下記のログインの
プロンプトが表示されます。

```
Welcome to Alpine Linux 3.18
Kernel 5.10.197-0-at on an aarch64 (/dev/ttymx1)

armadillo login:
```

ここでは**暫定**として、下記でログインします。
※パスワードは後でinitial_setupで変更します。

```
armadillo login : root
New password : root
Retype new password : root
```

■ Armadillo Base OSアップデート手順

初回、開発する前にOSを最新版にアップデートしておく事を推奨します。
アップデート方法は下記どれでも可能です。

■ アップデート方法

- Armadilloのコンソールからインターネット経由でアップデートする方法
- ABOS Webからインターネット経由でアップデートする方法
- USB/SD経由でアップデートする方法

Armadillo Base OSのアップデート

■ Armadilloのコンソールからインターネット経由でアップデートする方法

Armadilloにログイン後、下記コマンドを実行
完了すると自動でリブートし、アップデートしたバージョンで起動します。

```
[armadillo]# swupdate -d '-u [URL]' //アップデートコマンド
```

URLは下記バージョン(latest)を指定する事で最新版にアップデートできます。

Armadillo-IoT G4

<https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-g4/image/baseos-x2-latest.swu>

Armadillo-IoT A6E

<https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-a6e/image/baseos-6e-latest.swu>

Armadillo-X2

<https://armadillo.atmark-techno.com/files/downloads/armadillo-x2/image/baseos-x2-latest.swu>

Armadillo-640

<https://armadillo.atmark-techno.com/files/downloads/armadillo-640/image/baseos-600-latest.swu>

Armadillo-610

<https://armadillo.atmark-techno.com/files/downloads/armadillo-610/image/baseos-600-latest.swu>

Armadillo Base OSのアップデート

■ ABOS Webからインターネット経由でアップデートする方法

下記コマンドでブラウザを立ち上げてABOS Webにログインします。
ABOS Web起動時はセキュリティについての警告が出ますが無視して進めます。

```
[ATDE]$ firefox --url https://armadillo.local:58080 //ABOS Webを開く
```

※同じIPアドレスのセグメントに複数台Armadilloが接続されている場合は、armadillo.local を
当該のArmadilloのIPアドレスに置き換えます。

仮パスワードを設定してログインします。
※パスワードは後でinitial_setupで変更します。

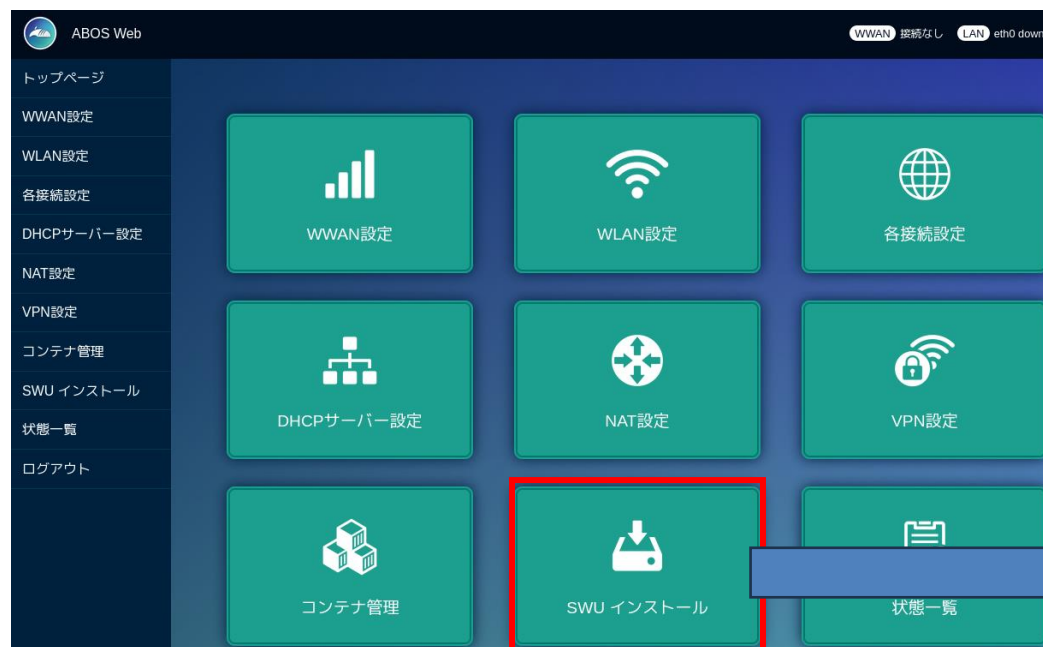


ログイン画面

Armadillo Base OSのアップデート

■ ABOS Webからインターネット経由でアップデートする方法

下記のSWUインストールを選択すると右図の画面が表示されます。
アップデートファイルのURLを入力⇒インストールボタン押下でアップデートが開始され、
アップデートが完了すると自動でリブートします。(URLはP.27と同じ)



ABOS Web トップページ



SWUインストール画面

■ USB/SD経由でアップデートする方法

- ①SWUファイルをダウンロード
- ②USBメモリやSDカードのTOP階層に配置
- ③起動中のArmadilloに挿入する事で、自動でアップデート実行～リブート

**注意：TOP階層に保存するswuファイルは1つにしてください
複数ある場合は順次実行されてしまいます。**

自動リブートが始まらない場合は/var/log/messagesにエラー情報が出ている可能性がありますので確認してください。

TOP階層以外にswuファイルを保存した場合は下記コマンドでも実行可能
※デバイスが/dev/sda1と認識している場合の例です。

```
[armadillo]# mount /dev/sda1 /mnt //USBメモリをマウント  
[armadillo]# swupdate -i /mnt/swu/baseos-x2-[version].swu //G4の例
```

■ アップデート時のエラー

アップデートに失敗すると、Armadilloの/var/log/messages にログが残ります。
エラーの詳細は下記を参照下さい。

swupdateのエラー詳細

<https://armadillo.atmark-techno.com/faq/swupdate-troubleshooting-abos>

- **ATDE環境の構築**
 - ・ VMwareのネットワーク設定
- **Armadilloとの接続**
 - ・ minicomの起動
 - ・ Armadilloへのログイン
- **Armadillo Base OSアップデート**
 - ・ 3つのアップデート方法

第3部：Armadilloの設定



- Armadilloの初期設定
- ネットワーク設定
- ハードウェア設定
- 既存コンテナの削除

■初期設定で行う事

Armadilloのroot/atmarkのパスワード設定、ABOS Webのパスワード設定
アップデート用の署名書き込みを最初に実施します。

1. VScodeでinitial_setup.swuを作成する

- ATDEの1ユーザーで1回だけ実行(以後、署名を再作成する場合以外は実施不要)
- 下記の情報を書き込むアップデートファイルを作成
 - Armadilloのパスワード
 - ABOS Webのパスワード
 - 固有の署名情報 (アップデートで使用)

2. initial_setup.swuをインストールする

- 開発時は1台のArmadilloに1回実行(量産では別手段有り)
- 同じArmadilloに2回は実行できない
- **インストールせずに実運用でArmadilloを使用するのは非推奨**

1. VScodeでinitial_setup.swuを作成する

VScodeを起動します。第2部で起動済みの場合は次頁に進みます。
UIから起動する方法とコマンドから起動どちらでも可能です。

◆UIからVScodeを起動する場合

ATDE左上のアプリケーション>プログラミング>Visual Studio Code(VScode)を実行

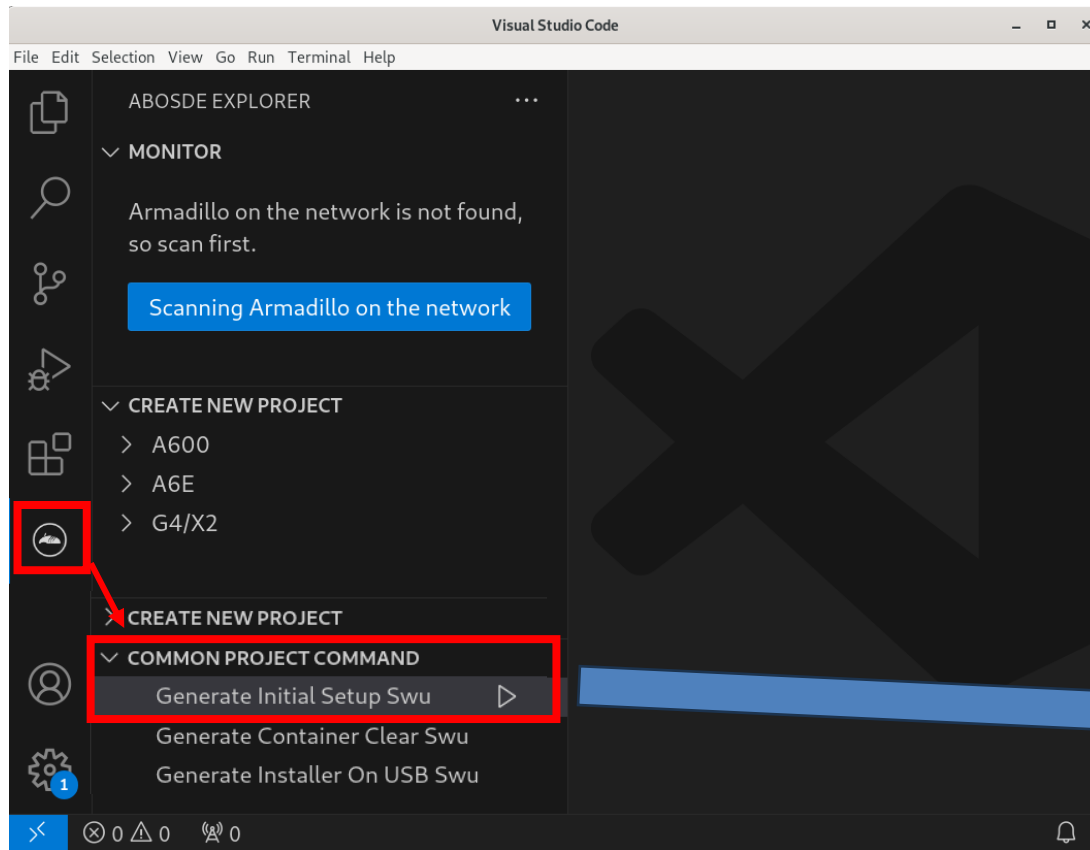
◆コンソールからVScodeを起動する場合

Armadilloのコンソールとは別にコンソール開き、下記コマンドを実行します。

```
[ATDE]$ code //VScode起動
```

Armadilloの初期設定

1. VScodeでinitial_setup.swuを作成する
ABOSDE Explorer を選択し、COMMON PROJECT COMMAND にある
Generate Initial Setup Swu の右の△をクリック



現在のユーザーで一度もGenerate Initial Setup Swu を作成していない場合は、コンソールに下記の様な設問が幾つか出てきます。
例としてセミナーでは次頁の様に入力します。

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... bash - Task + - [ ] [ ] ... ^ x
* Executing task: /home/atmark/.vscode/extensions/atmark-techno.armadillo-base-os-developme
nt-environment-1.5.0/shell/generate_initial_setup_swu.sh

mkdir: ディレクトリ '/home/atmark/mkswu' を作成しました
設定ファイルを更新しました: /home/atmark/mkswu/mkswu.conf
証明書の共通ネーム(一般名)を入力してください:
```

1. VScodeでinitial_setup.swuを作成する

◆initial_setupの入力内容

※以下の赤字はセミナーでの入力内容

セミナーでは簡易的なパスワードにしていますが、実際には強固なパスワードを設定してください。

証明書の共通ネーム(一般名)を入力してください: **test**

⇒証明鍵の「common name」として会社や製品が分かるような任意の名称を入力します。

証明書の鍵のパスワード: **testpw**

⇒SWUファイルを作る際に聞かれるパスワードです。確認でもう一度入力します。

アップデートイメージを暗号化しますか? (N/y) **n**

⇒暗号化する場合のみy、セミナーではnが良いです。

アットマークテクノが作成したイメージをインストール可能にしますか? (Y/n) **y**

⇒Armadilloの/etc/swupdate.pemにアットマークテクノの署名を残します。

それによりアットマークテクノが作成したArmadillo Base OSやコンテナ等のアップデート(SWU)をそのまま使えるようになります。

”y”に設定しても自動ではインストールされません。

1. VScodeでinitial_setup.swuを作成する

◆initial_setupの入力内容

※以下の赤字はセミナーでの入力内容

セミナーでは簡易的なパスワードにしていますが、実際には強固なパスワードを設定してください。

rootパスワード: **at-semi1**

⇒8文字以上でArmadilloのrootのパスワードを設定します。確認でもう一度入力します。

atmarkユーザのパスワード（空の場合はアカウントをロックします）: **（空でEnter）**

⇒ 8文字以上でArmadilloのatmarkユーザのパスワードを設定します。確認でもう一度Enter

BaseOSイメージのarmadillo.atmark-techno.comサーバーからの

自動アップデートを行いますか？ (N/y) **n**

⇒OSのアップデートがリリースされたら自動的にアップデートを行う設定です。

特に希望が無い限りは”n”を推奨します。

abos-webユーザのパスワード（空の場合はアカウントをロックします）: **abos-web**

⇒Armadilloのabos-webユーザのパスワードを設定します。（8文字以上）

確認でもう一度入力します。

1. VScodeでinitial_setup.swuを作成する

Generate Initial Setup Swu の実行が完了すると~/mkswuというディレクトリが作成されます。ディレクトリ内の詳細は下記の通りです。

注意！！

これらのファイルを無くすと以後同じ署名でのアップデートが出来なくなりますのでディレクトリごとバックアップを取っておく事をお勧めします。

mkswuディレクトリ下のファイル

- initial_setup.swu 初期設定をArmadilloに書き込むSWUファイル
- initial_setup.desc 上記SWUファイルを作成する為の書式ファイル
- mkswu.conf mkswuのコンフィグファイル
- swupdate.key 鍵情報
- swupdate.pem 署名情報（Armadilloにはこの署名が書き込まれる）

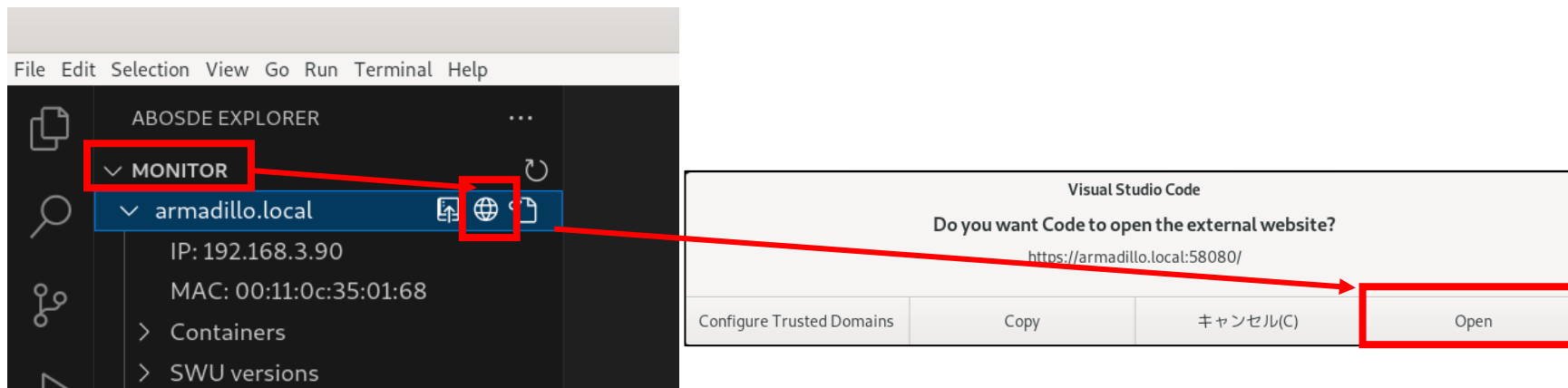
Armadilloの初期設定

2. initial_setup.swuをインストールする

※USBメモリ・SDカードでも実施可能です

◆VMwareのネットワーク設定をBridgeに設定している場合

ABOSDE Explorer の MONITORの **Scanning Armadillo~** をクリックします。表示された当該のArmadilloの armadillo(-x).local を選択し、丸いアイコンをクリックしてABOS Webを立ち上げます。(複数台ある場合はIPで判断)



◆VMwareのネットワーク設定をNATに設定している場合

MONITORには表示されない為、下記コマンドをコンソールで実行

```
[ATDE]$ firefox -url https://[IPアドレス]:58080 //firefoxでABOS Webにアクセス
```

2. initial_setup.swuをインストールする

ABOS Web起動時はセキュリティについての警告が出ますが、無視して進めます。

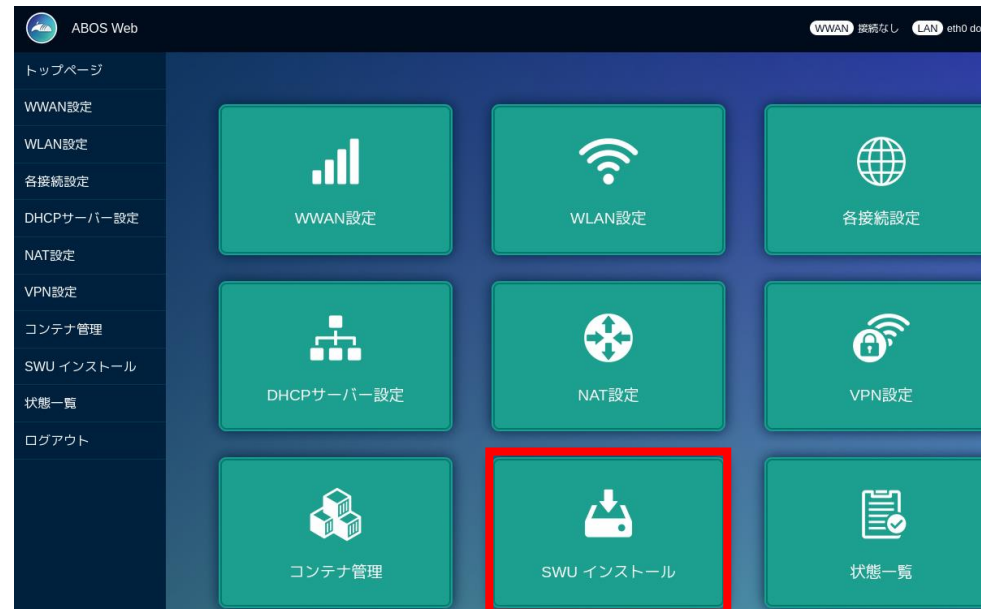
仮パスワードでログインし、SWUインストールを選択します。

仮パスワード未設定の場合は任意のパスワードを入力します。

※この後にinitial_setupで設定したパスワードに上書きされます。



ログイン画面



ABOS Web トップページ

2. initial_setup.swuをインストールする

①SWUファイルの参照ボタンを押してinitial_setup.swuを選択する
ファイルパス：/home/atmark/mkswu/initial_setup.swu

②インストールボタンを押す



① SWU ファイル入力

SWU ファイル
参照... ファイルが選択されていません。

② インストール

SWU URL 入力

SWU URL
https://download.atmark-techno.com/armadillo-iot-g4/image/baseos-x2-lat

インストール

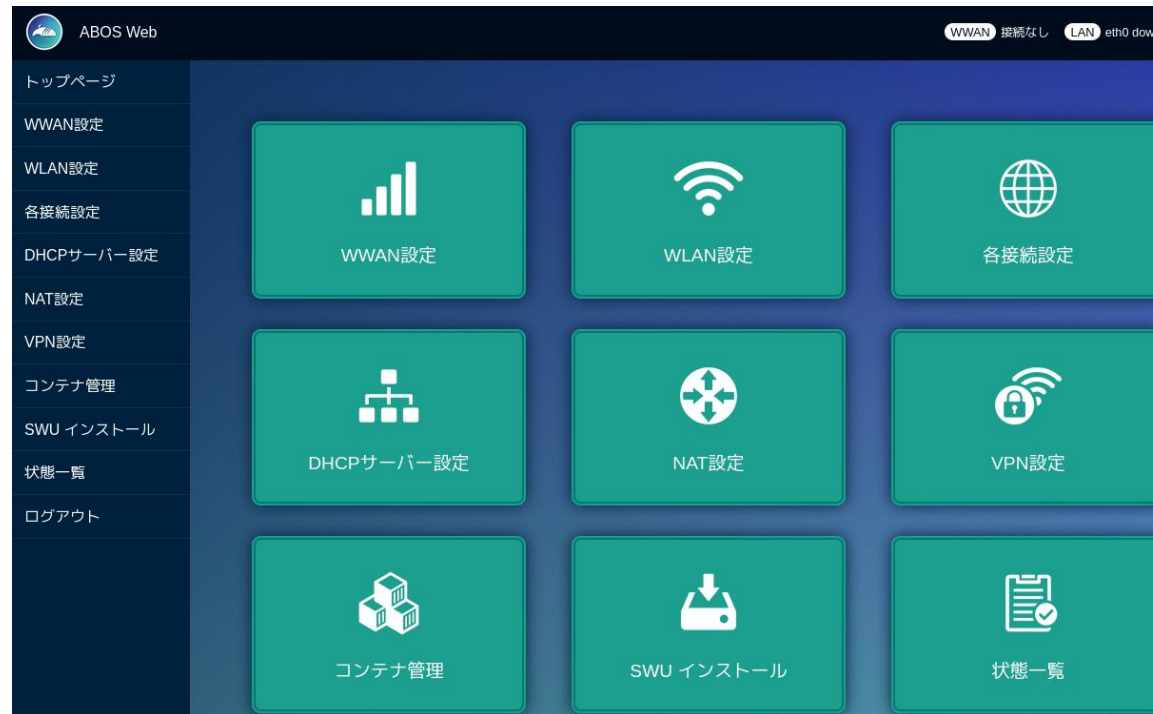
《補足》

SWUのインストールが完了すると、自動でArmadilloが再起動します。コンソール(minicomやTeraTerm)を立ち上げておくと再起動しているか、立ち上がったかどうかを確認出来ます。

以上で、Armadilloの初期設定は完了です。

■ ABOS Webの役割

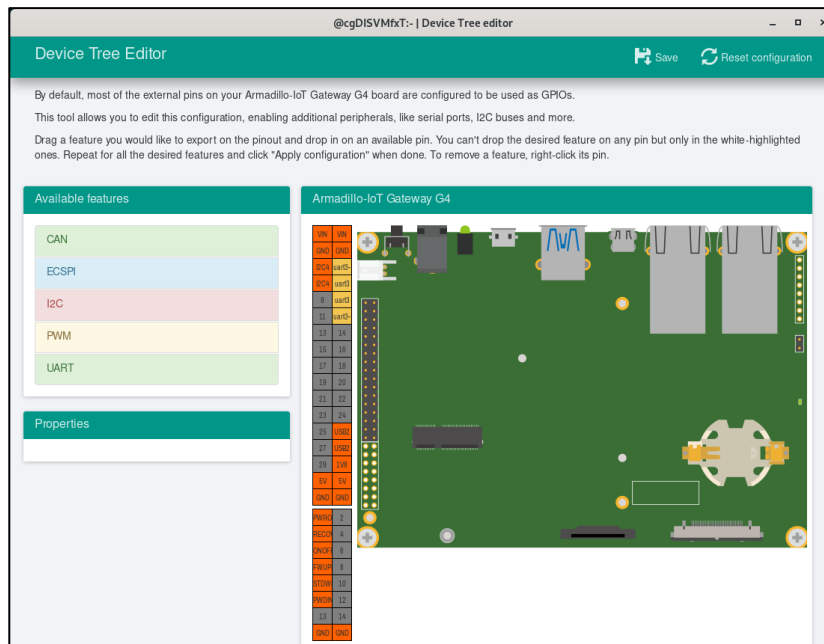
ABOS WebではArmadilloで使用するLTEやWLAN、NATなどの設定を行います。
EthernetはデフォルトでDHCPとなっており、そのまま使う場合は設定は不要です。
各設定方法の詳細は製品マニュアルを参照下さい。（本セミナーでは実施しません）



ABOS Web トップページ

ハードウェア設定

各製品には拡張インターフェースを搭載しており、UARTやI2C,SPI,CANなど様々な機能の割り当てを行う事が可能です。これらを使用しない場合は本内容は設定不要です。手順の詳細やコマンドは製品マニュアルを参照下さい。



at-dtweb設定画面

《概略手順》

- ①カーネルソースファイルをArmadilloサイトからダウンロード&展開
- ②デフォルトコンフィグを適用
- ③at-dtweb を起動
- ④製品を選択
- ⑤カーネルディレクトリを選択
- ⑥端子機能をドラッグ&ドロップで変更
- ⑦上部の Save ボタンを押してセーブ&ビルド
- ⑧ATDE上でアップデートファイル作成(次頁)

参考ブログ

<https://armadillo.atmark-techno.com/blog/15349/19122>

ハードウェア設定

at-dtwebで作成されたファイルを使用し、下記コマンドを実行する事で割り当てた機能を有効にすることができるSWUファイルが生成されます。SWUをArmadilloにインストールする事でハードウェアの設定は完了です。(ファイルの編集は不要)

```
[ATDE]$ ls //Armadillo-IoT G4の例
armadillo_iotg_g4-at-dtweb.dtbo at-dtweb.desc
update_overlays.sh update_preserve_files.sh
[ATDE]$ mkswu at-dtweb.desc //SWUファイル生成コマンド
```

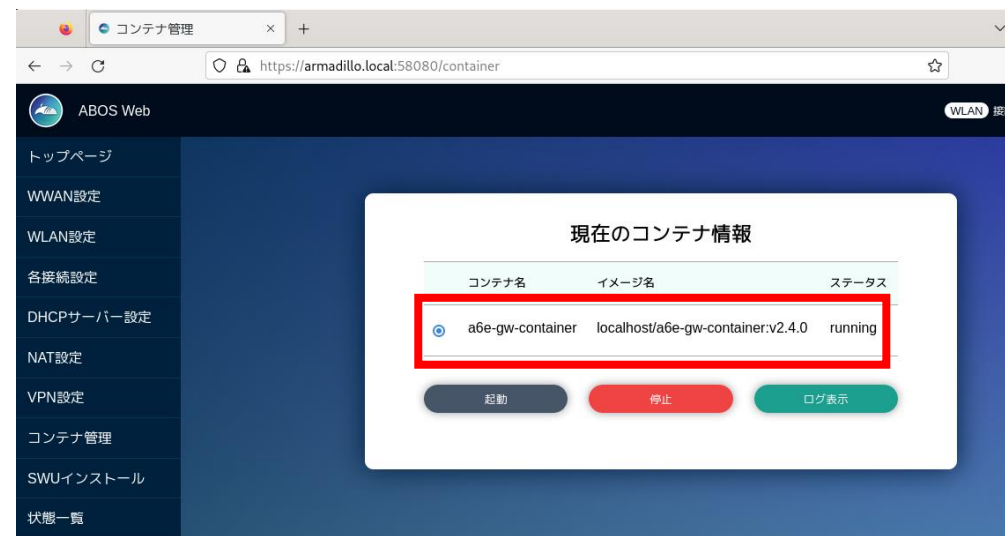
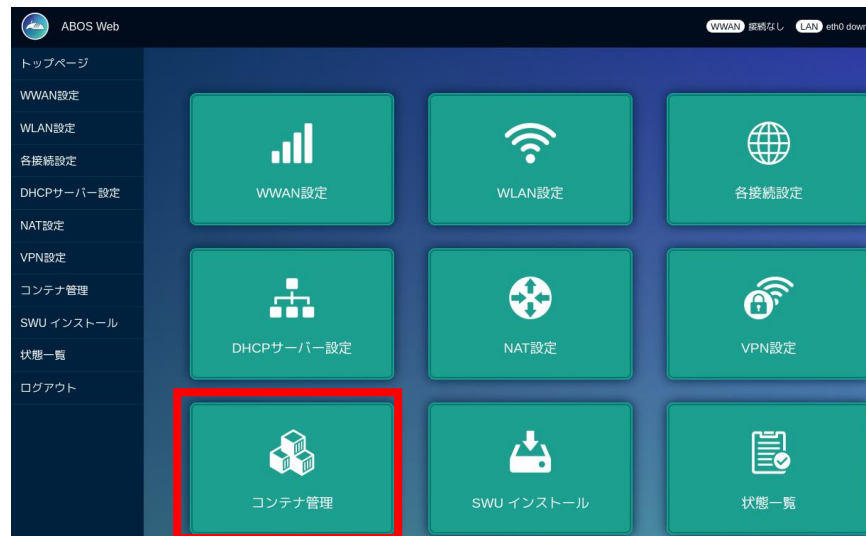
本SWUの実行内容は下記の通りです。

- [製品名]-at-dtweb.dtbo を/bootにコピー
- [製品名]-at-dtweb.dtbo を /etc/swupdate_preserve_files に追記(※スクリプトで実行)
⇒swupdate_preserve_filesに記載する事でArmadillo Base OSをアップデートしてもファイルは引き継がれます。
- [製品名]-at-dtweb.dtbo を overlays.txt に追記(※スクリプトで実行)
⇒overlays.txtに記載されたdtboファイルの設定が適用されます。

既存コンテナの削除

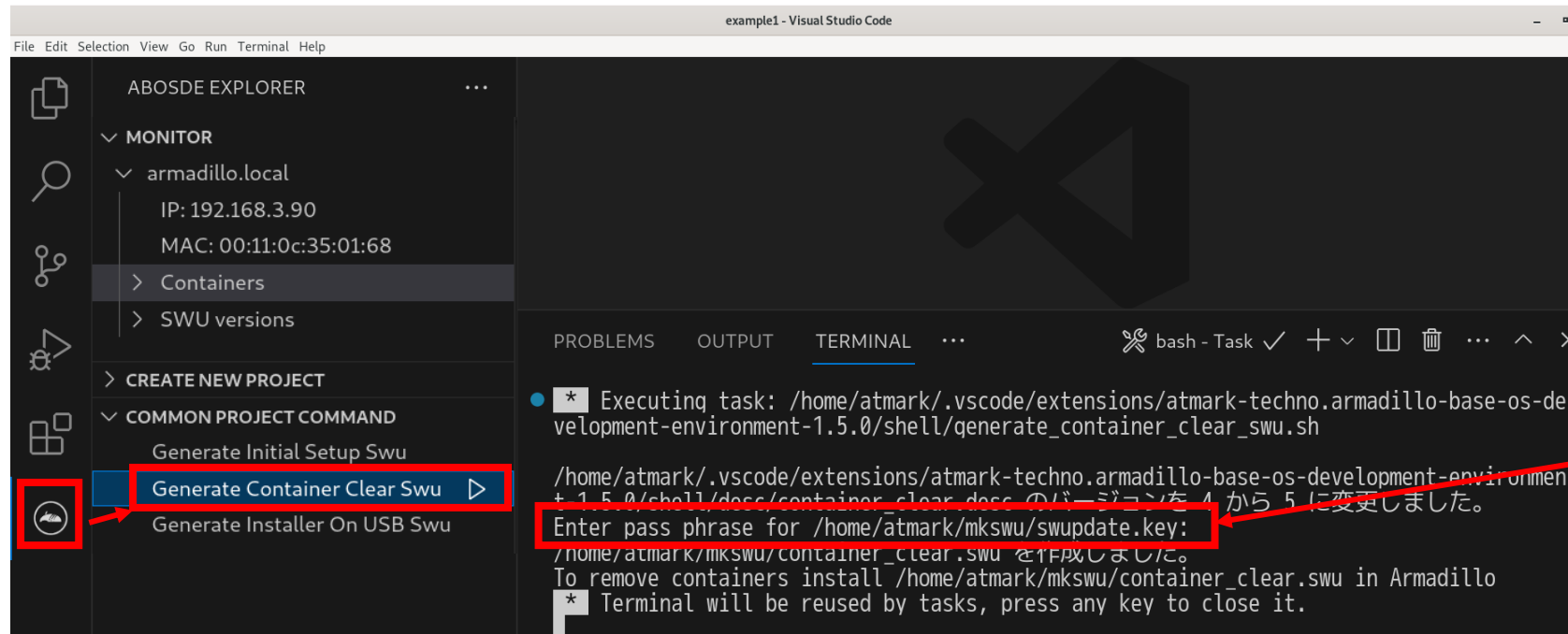
Armadilloで既にコンテナが作成されている状態で、これからの開発に使用しない場合には、**デバイスの競合を防ぐ事やeMMCの使用領域削減**を目的として、事前にコンテナを削除しておきます。コンテナが作成されているかどうかはABOS Webで下記の様に確認出来ます。(セミナーでは独自にコンテナを作成する為削除します)

※Armadillo-IoT A6Eでは出荷状態または当社インストールディスクイメージでの初期化後はゲートウェイコンテナ、またはNode-REDコンテナがプリインストールされています。



既存コンテナの削除

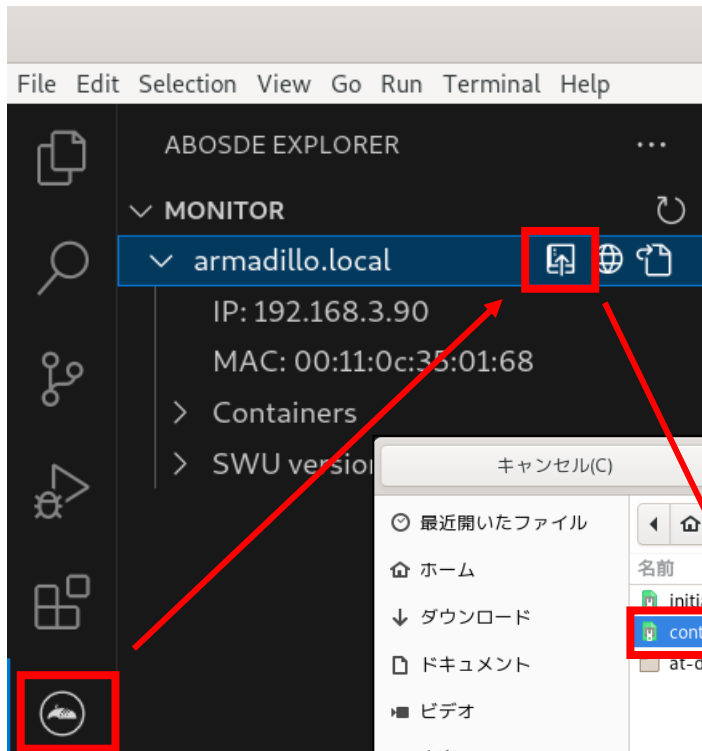
既存コンテナを削除する方法は、VScodeでGenerate Container Clear Swuの右の△をクリックすると、コンテナを削除するSWUファイルが作成されます。本SWUをインストールすると**全てのコンテナを削除**する事が出来ます。
作成場所：`/home/atmark/mkswu/container_clear.swu`



パスワード `testpw` を入力

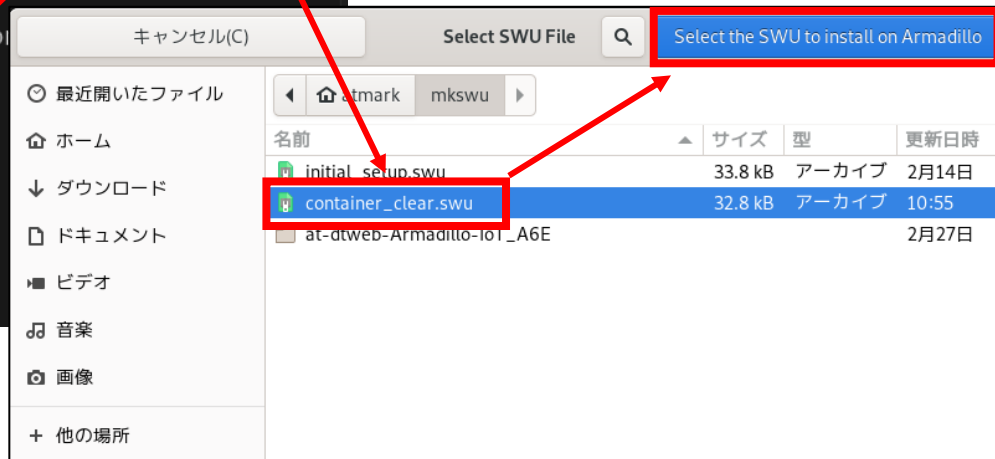
既存コンテナの削除

一度initial_setup.swuをインストール(ABOS Webのパスワードを登録)した後は、下記手順で簡単にSWUファイルをインストール出来ます。



《手順》

- ①左図の↑マークのInstall SWU on Armadilloを選択
- ②ABOS Webのパスワードを入力(初回のみ)
- ③ファイル選択ウインドウが開いたら、SWUファイルを指定(VScodeのウインドウ裏に表示される場合有り)
- ④右上のSelect the SWU to Install on Armadilloを押す



《補足》

Armadilloを初期化したり別のArmadilloで実行する場合、下記エラーが出る事があります。その場合はDeleteを選択する事で実行できます。

Certificate associated with token does not match.
Have you changed Armadillo's certificate?
Delete tokens stored in ABOSDE that have failed access?

- **Armadilloの初期設定**
 - ・ Initial_setupで設定書き込み
- **ネットワーク設定**
 - ・ ABOS WebでのLTE,WLAN等の設定
- **ハードウェア設定**
 - ・ at-dtwebでの拡張インターフェース設定
- **既存コンテナの削除**
 - ・ container_clear.swuの作成&インストール

第4部：アプリケーション開発



- **アプリケーションの開発**
 - ① 新規プロジェクト作成
 - ② SSH接続設定
 - ③ アプリケーションの作成
 - ④ ビルド～SWUインストール
 - ⑤ デバッグ～リリース版SWUインストール
 - ⑥ sshd無効化(任意)

■作成するアプリケーション

Armadillo Base OS搭載製品共通

①LED点滅のシェルスクリプトを動かす . . . P.53～

Armadillo-IoT A6Eの場合

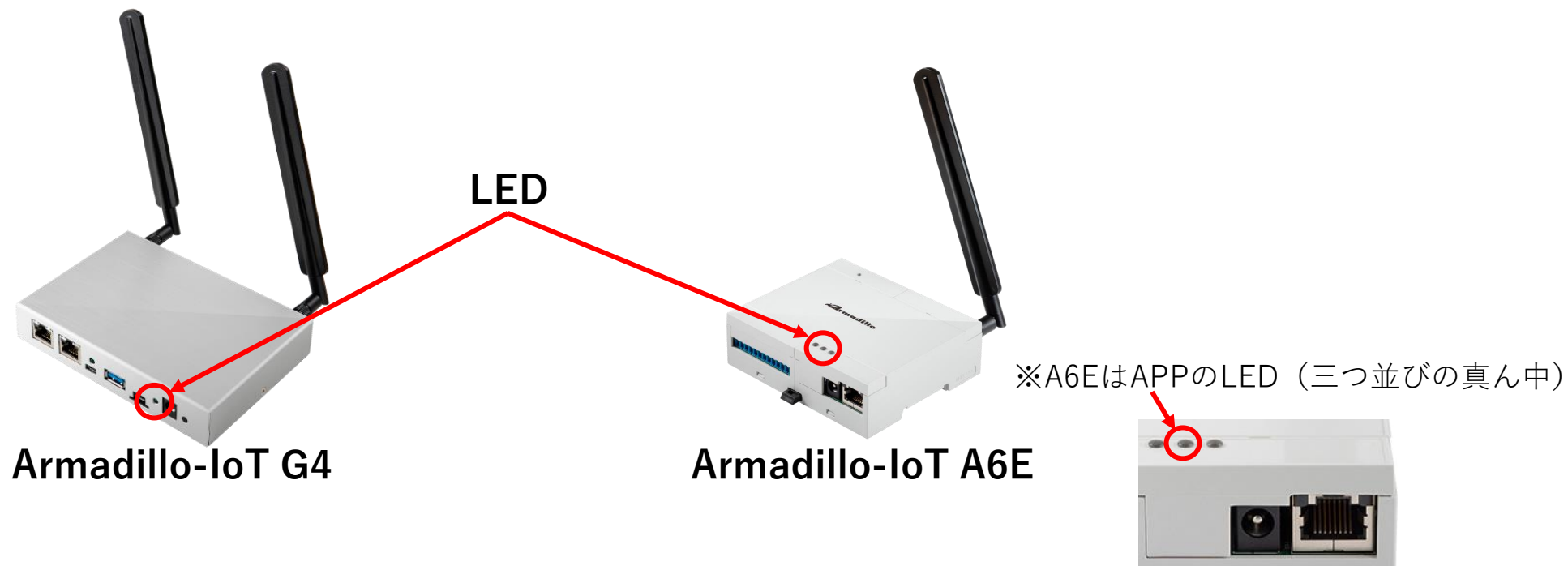
②DI(Digital Input)操作し、LEDを点灯/消灯させる . . . P.88～

Armadillo-IoT G4の場合

③USBカメラ画像をHDMI出力しつつVPUでエンコードする . . . P.108～

■作成するアプリケーション

①LED点滅のシェルスクリプトを動かす（製品共通）



■ アプリケーション開発手順

アプリケーションは主に下記手順を実施頂く事で開発出来ます。

- ① 新規プロジェクト作成
- ② SSH接続設定
- ③ アプリケーションの作成
- ④ ビルド～SWUインストール
- ⑤ デバッグ～リリース版SWUインストール
- ⑥ sshd無効化(任意)

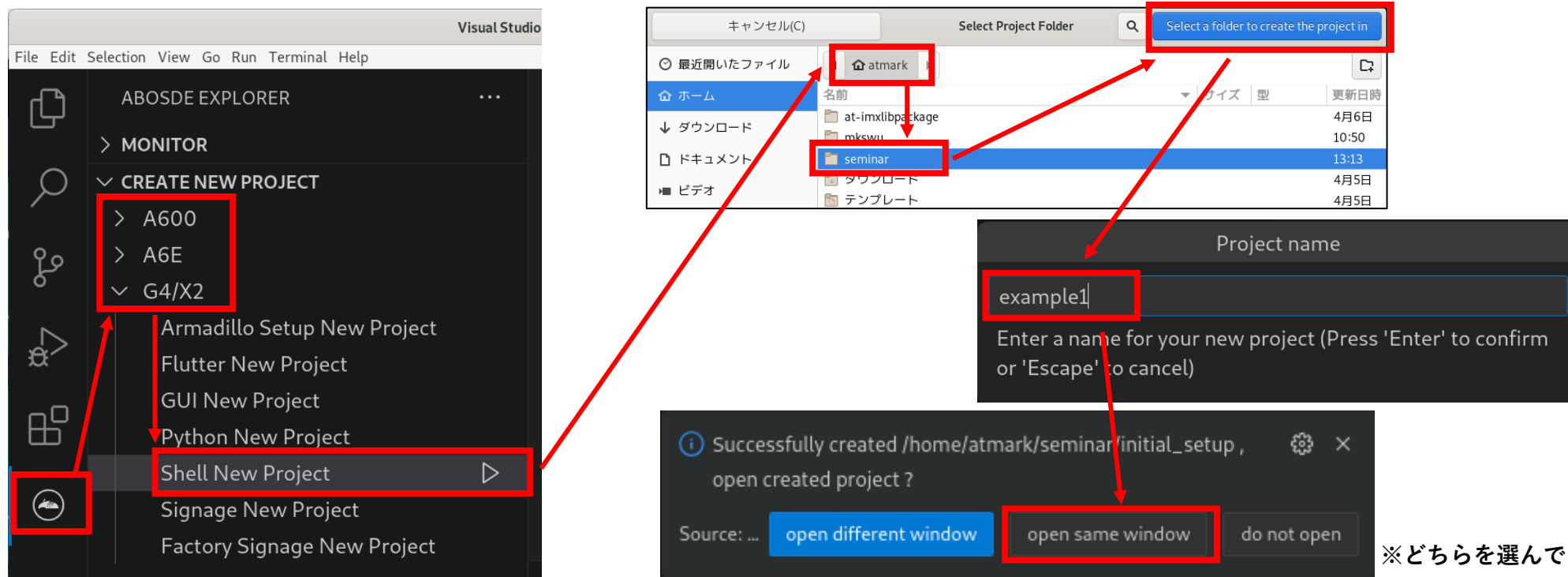
■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

アプリケーション開発①

①新規プロジェクト作成

アプリケーション作成の為のプロジェクト **example1** を作成します。
下記手順に沿って、ABOSDE Explorerから **製品** ⇒ **Shell New Project** の右の△をクリックし、**seminar**ディレクトリの中に **example1** ディレクトリを作成します。
画面右下で別ウインドウにするか聞かれますが、ここでは同じウインドウとします。

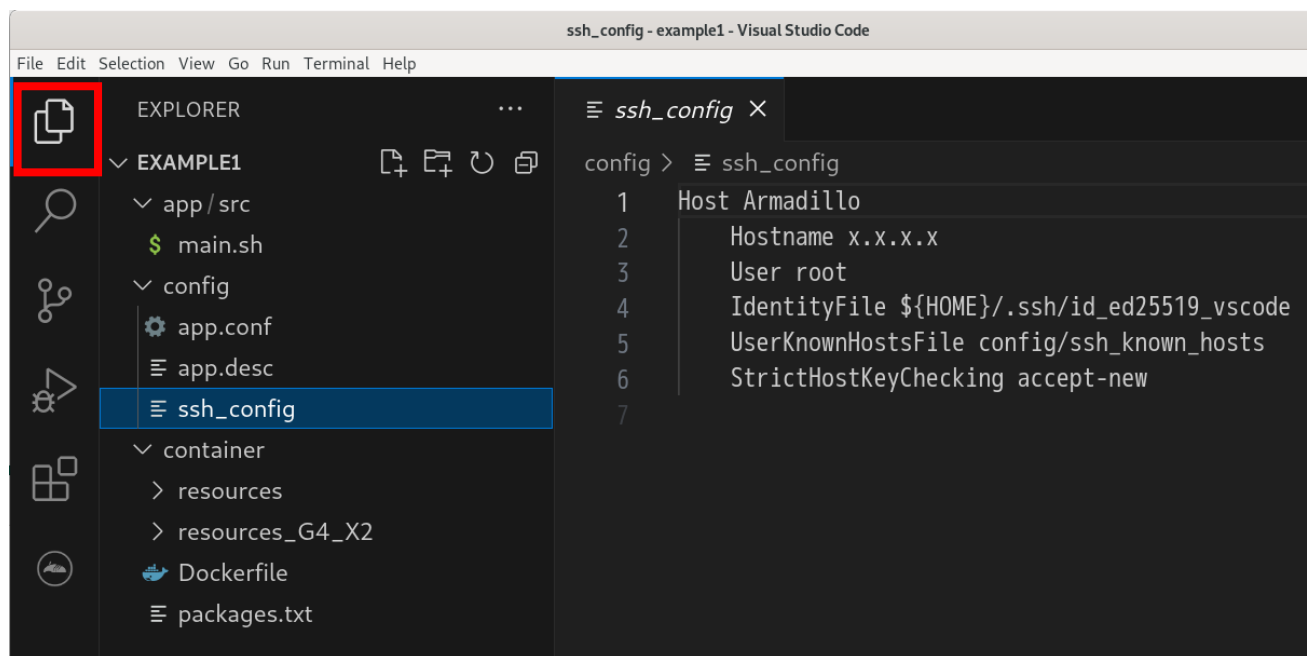


※どちらを選んでも開発可能です

①新規プロジェクト作成

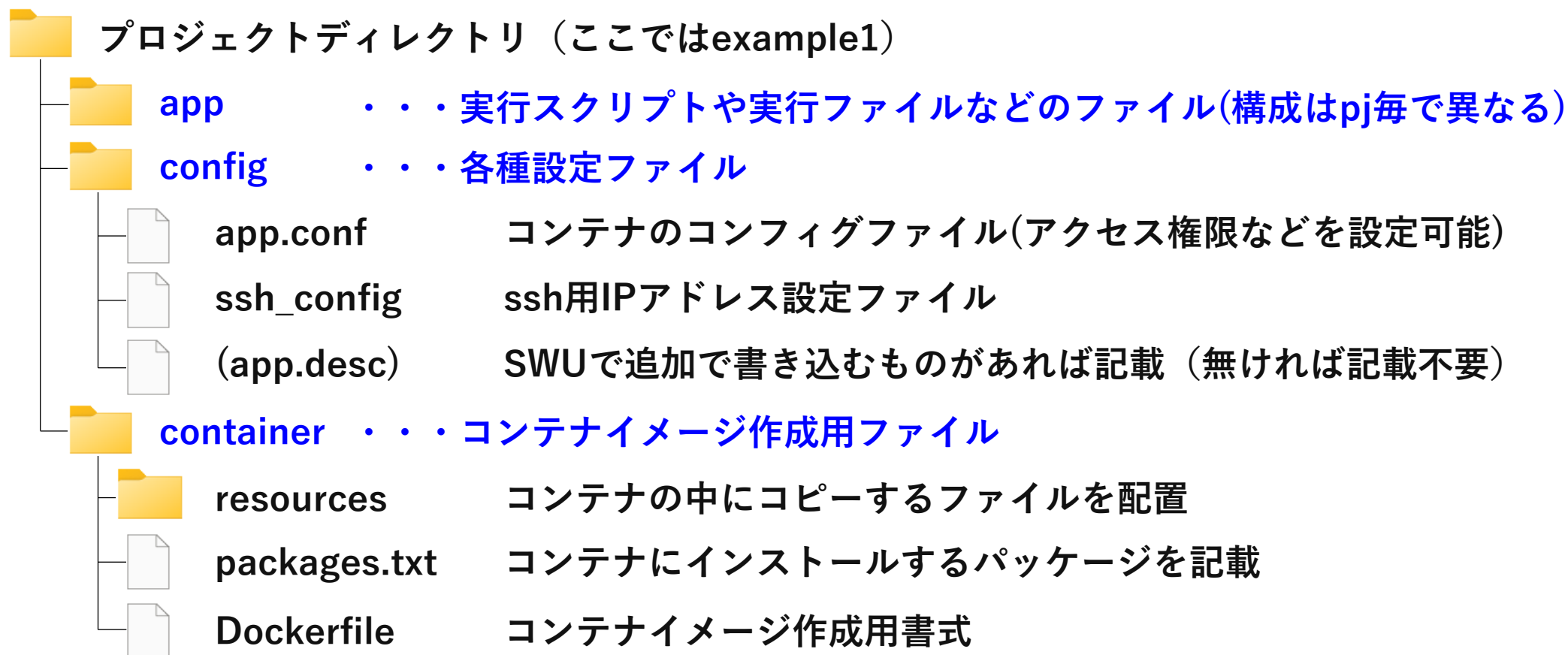
VScodeの下記赤枠のExplorerを選択すると、プロジェクトのディレクトリ及びファイルが表示されます。これらのファイルをそれぞれ修正してアプリケーションを作成します。

※下記以外にもディレクトリ/ファイルは存在しますが、VScode上では使用しない為、表示されません。



①新規プロジェクト作成

Projectの基本ディレクトリ構成は下記の様になっています。 ※編集不要のファイルは省略



《補足》各プロジェクトの説明

■各プロジェクトの補足

※選択可能なプロジェクトは製品によって異なります

以下のプロジェクトはゲートウェイコンテナを除き、デフォルトでサンプルアプリが動作する様になっています。どのプロジェクトを流用して開発しても問題ありません。

全製品共通プロジェクト

- ・ Python New Project : Pythonアプリの標準プロジェクト
- ・ Shell New Project : Shellアプリの標準プロジェクト
- ・ C New Project : C言語アプリの標準プロジェクト

G4/X2用プロジェクト

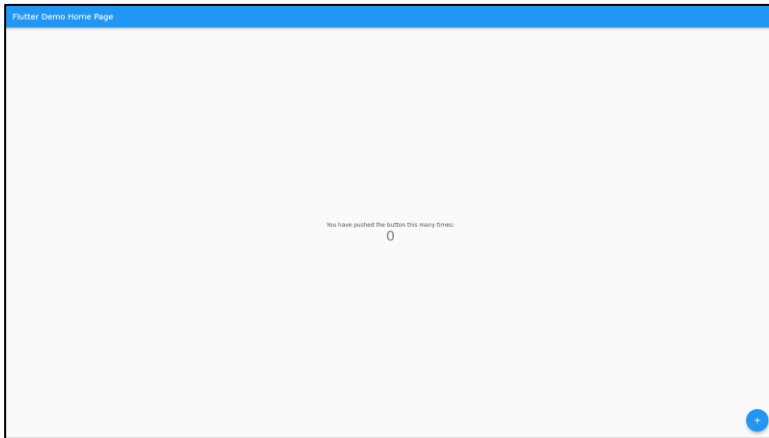
- ・ Flutter New Project : Flutterアプリの標準のプロジェクト
- ・ GUI New Project : サンプルデモ用プロジェクト
- ・ Signage New Project : サンプルデモ用プロジェクト
- ・ Factory Sinage New Project : サンプルデモ用プロジェクト

A6E用プロジェクト

- ・ GW New Project : ゲートウェイコンテナ標準プロジェクト

《補足》各プロジェクトの説明

■各Flutterプロジェクトのサンプルデモ



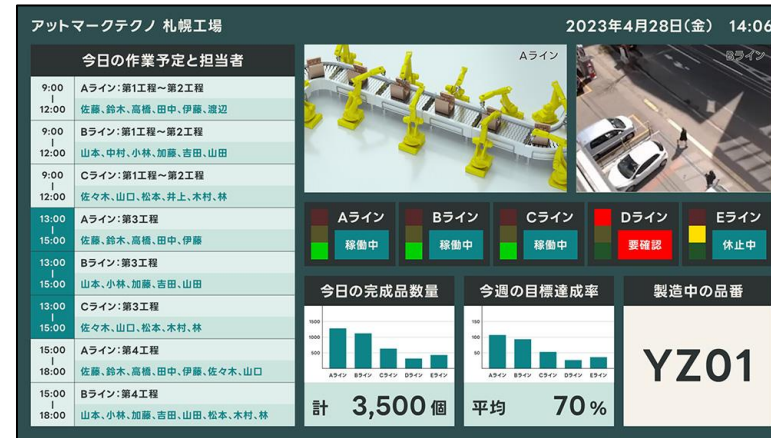
Flutter New Project



GUI New Project



Signage New Project



Factory Signage New Project

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

②SSH接続設定

Armadilloでデバッグを行う場合はSSH接続が必要になります。
SSH接続を行うには下記手順で行います。デバッグが不要な場合は本手順は不要です。

- **Setup environment**を行う
ATDEでSSHの鍵を生成します。
- **対象のArmadilloのIPを設定する**
example1/config/ssh_configを書き換えます。
- **(後の作業で)アプリ作成後に Generate development swu を実行**
本SWU内にArmadilloのsshd有効化の処理が入っています。
Generate release swuにはsshdの有効化/無効化処理は入っていません。

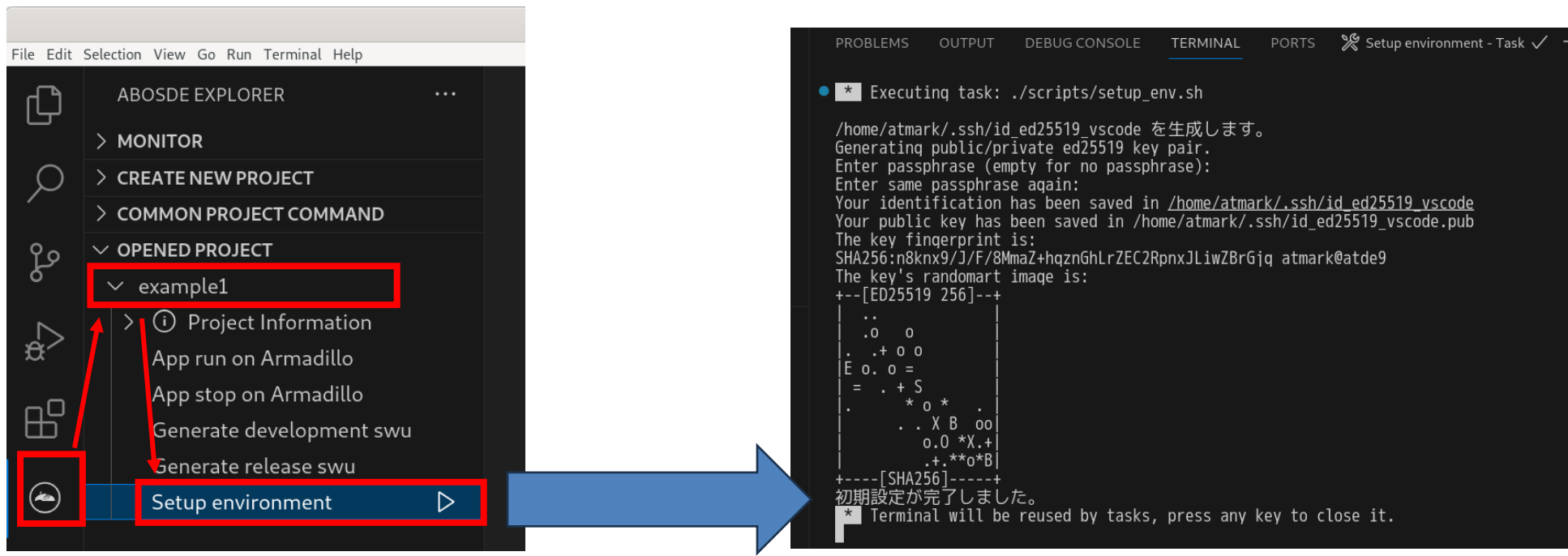
《注意》

Generate development swu を一度インストールすると、その後に Generate release swuをインストールしてもsshdは解除されません。sshd解除にはP.84を参照下さい。

②SSH接続設定

■ Setup environmentを行う

Setup environmentの右の△を押してSSHの鍵を生成します。
SSHのパスワードを聞かれるので、セミナーでは **ssh** とします。(2度入力)
鍵(~/.ssh)を消さない限りは別プロジェクト作成時も実行する必要はありません。

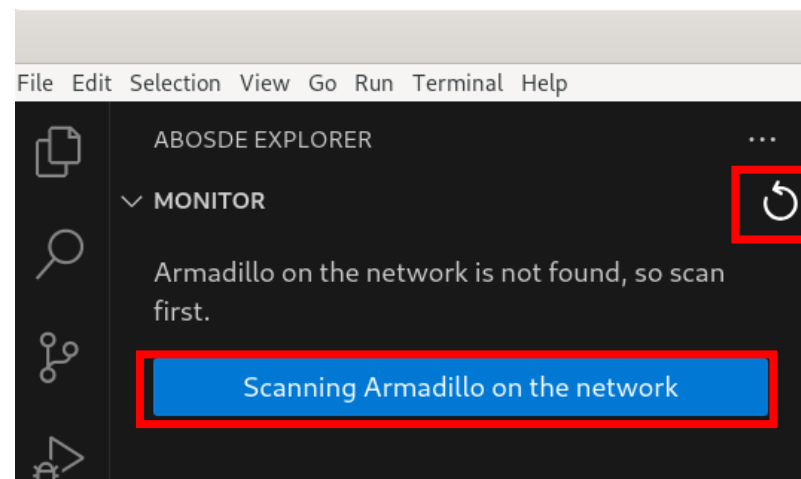
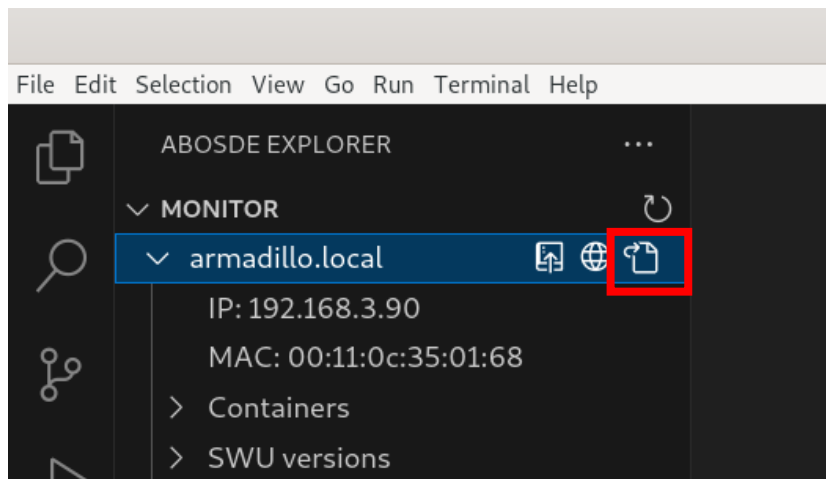


②SSH接続設定

- 対象のArmadilloのIPをssh_configに設定する
ATDEのネットワーク設定がBridgeの場合とNAT または手入力 の場合で設定方法が異なります。

◆Bridge設定の場合

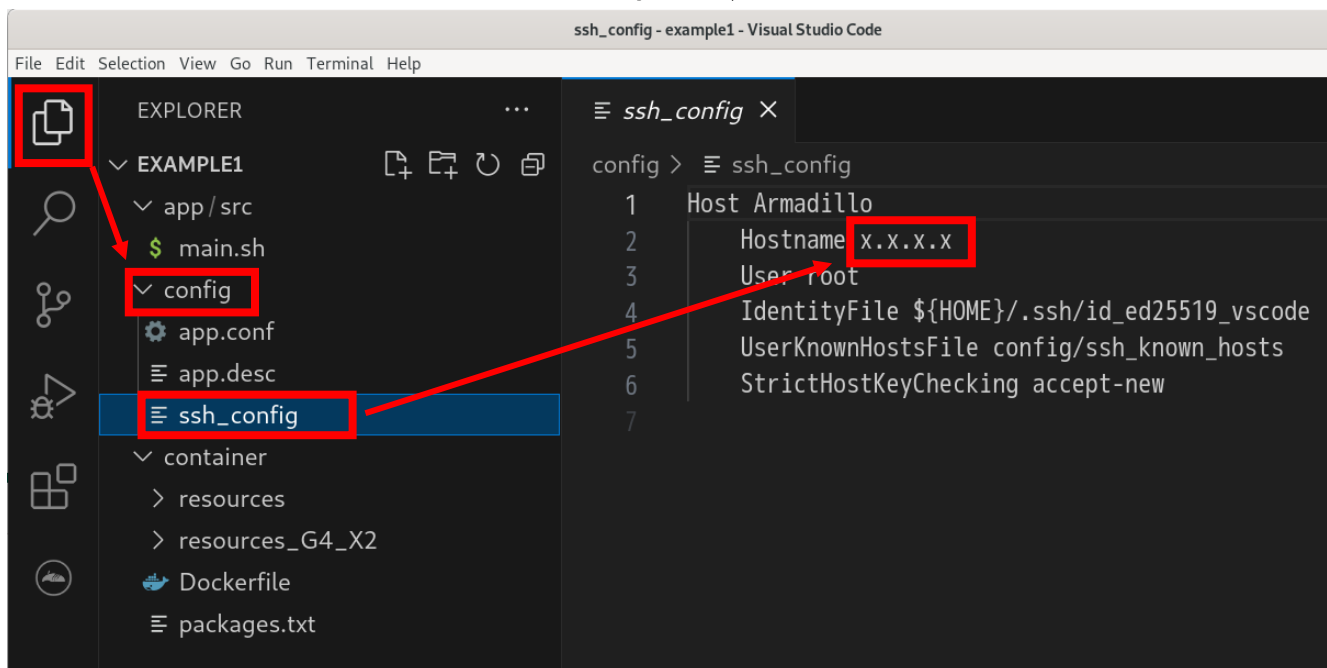
ABOSDE ExplorerのMONITORに同じセグメント上にあるArmadilloが表示されます。左図の赤枠を選択すると [example1/config/ssh_config](#) が当該IPで設定されます。表示されない場合は右図の赤枠どちらかを押して更新してください。



②SSH接続設定

◆NAT設定の場合（または手入力する場合）

Explorer ⇒ EXAMPLE1/config/ssh_config を選択すると内容が表示されます。
Hostname x.x.x.x の部分をArmadilloのIPアドレスに書き換えます。
Ctrl+S で保存して閉じます。（上部タブのFile⇒Saveでも可）

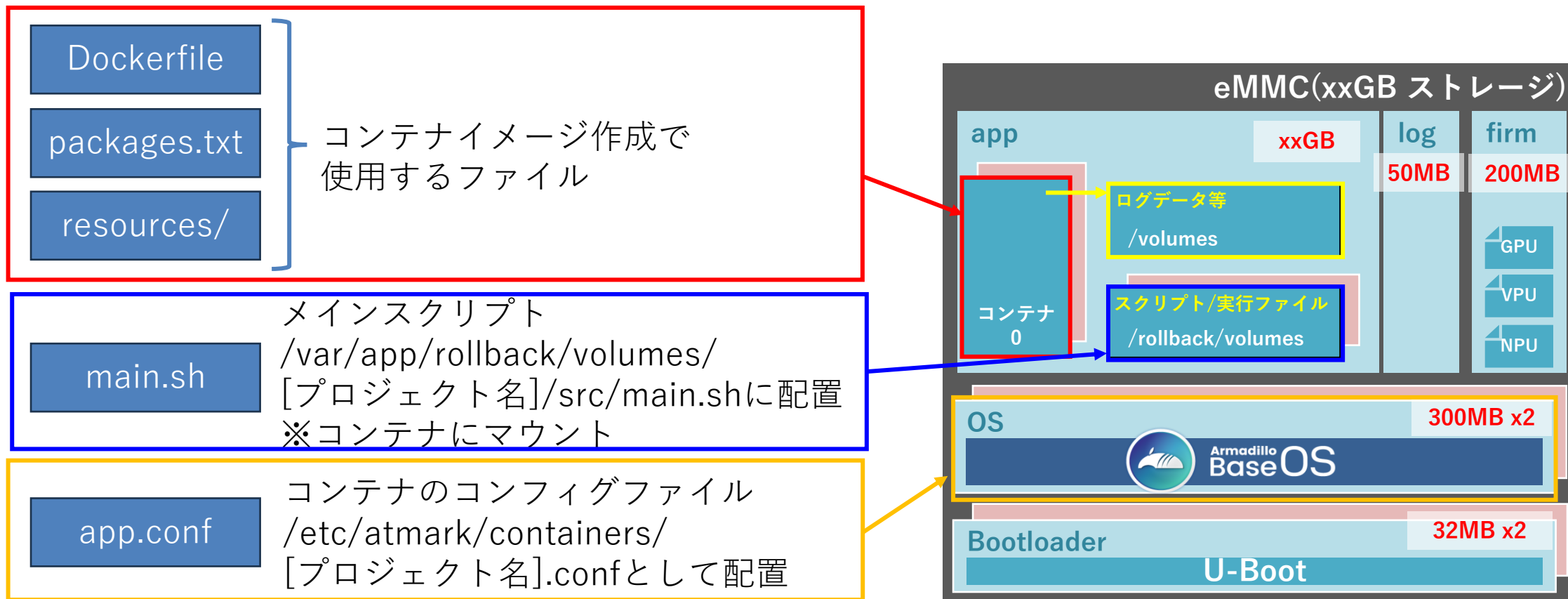


■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

③アプリケーションの作成

アプリケーション作成では下記ファイルをそれぞれ編集して、コンテナを作成します。
この後の手順でArmadilloに書き込んだ後はArmadilloに下記の様に配置されます。



■ volumesディレクトリについて

volumesディレクトリはapp領域にあるeMMC保存可能な領域(常時OverlayFS無効)

- `/var/app/volumes`

ロールバックの有無に関わらずファイルを保持

- `/var/app/rollback/volumes`

ロールバック発生時、ロールバック先の状態(前の状態)に戻る

コンテナはRAM上で動作する為、**コンテナ内のデータは電源OFFでクリア**される
電源OFFでもデータを保持したい場合は本ディレクトリを使用する

③アプリケーションの作成

■ コンテナの設定 (app.conf)

コンテナを起動する為の設定を app.conf に記載します。(書式は製品マニュアルを参照)
Shell Projectではデフォルトで下記設定になっています。本アプリ開発では変更しません。

Dockerfileでビルドするイメージを指定(変更不要)

```
set_image localhost/{{PROJECT}}:latest
```

コンテナ外とのファイル・ディレクトリ共有(変更可能)

```
add_volumes /var/app/rollback/volumes/{{PROJECT}}:/vol_app
```

```
add_volumes /var/app/volumes/{{PROJECT}}:/vol_data
```

```
add_volumes /sys:/sys //LEDを制御する為に必要
```

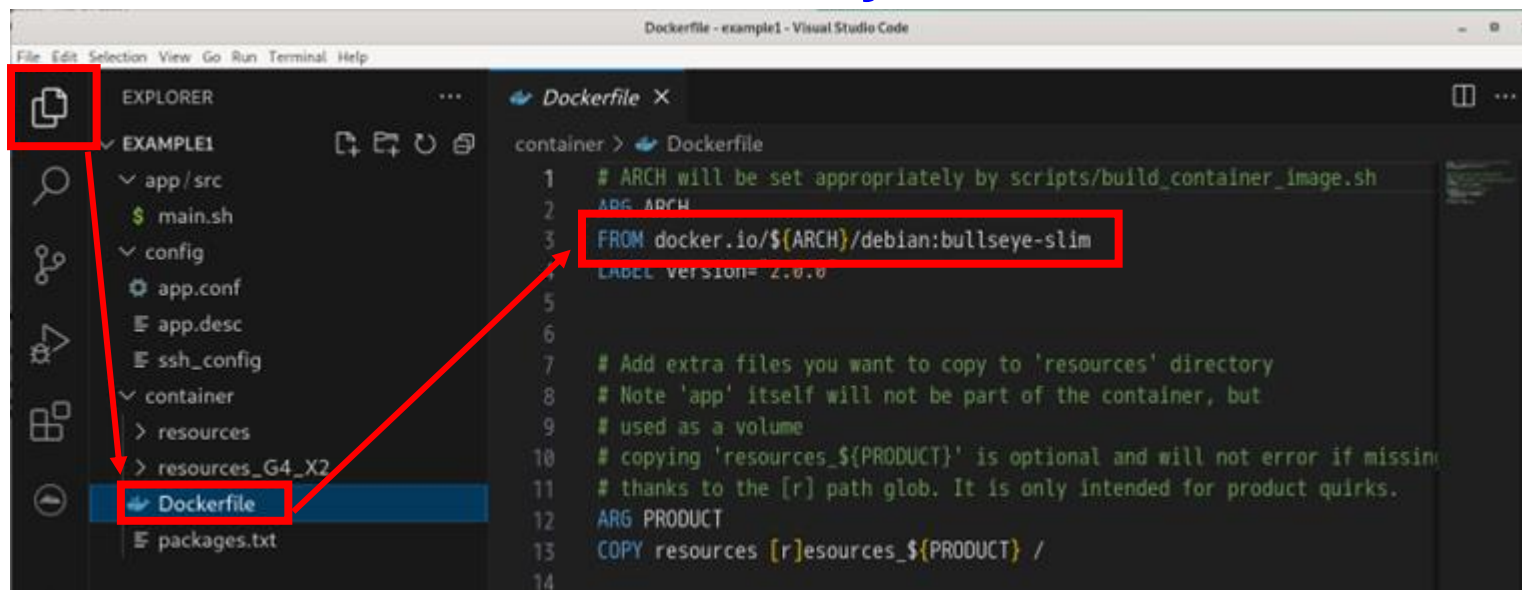
コンテナで実行するコマンド(変更不要)

```
set_command bash /vol_app/src/main.sh
```

コンテナは上記set_commandが終了すると自動的に終了します。
終了させない場合は無限ループなどで上記コマンドが終了しないスクリプトにします。

③ アプリケーションの作成

- 使用するコンテナイメージを決定(Dockerfile)
使用するコンテナイメージはDockerfileで指定できます。
セミナーではそのまま `debian:bullseye-slim` を使用します。



```
1 # ARCH will be set appropriately by scripts/build_container_image.sh
2 ARG ARCH
3 FROM docker.io/${ARCH}/debian:bullseye-slim
4 LABEL version=2.0.0
5
6
7 # Add extra files you want to copy to 'resources' directory
8 # Note 'app' itself will not be part of the container, but
9 # used as a volume
10 # copying 'resources_${PRODUCT}' is optional and will not error if missing
11 # thanks to the [r] path glob. It is only intended for product quirks.
12 ARG PRODUCT
13 COPY resources [r]resources_${PRODUCT} /
14
```

その他のコンテナイメージを探す場合は下記を参考にしてください。

参考ブログ：<https://armadillo.atmark-techno.com/blog/15349/12089>

③アプリケーションの作成

■ コンテナにインストールするパッケージを設定(Dockerfile)

Dockerfile内では下記のように記載していますが、独自に変更も可能です。
セミナーでは編集しません。

```
15 # Add extra packages to containers/packages.txt
16 ARG PACKAGES
17 RUN apt-get update && apt-get upgrade -y \
18     && apt-get install -y ${PACKAGES} \
19     && apt-get clean
```

※デフォルトのコンテナイメージから変更する場合は
左記のコマンドは適宜変更下さい。

デフォルト設定でDebianのコンテナイメージを使用する場合は、
[example1/container/packages.txt](#) にインストールしたいパッケージを
記載すると apt-get install で一括でインストールできます。

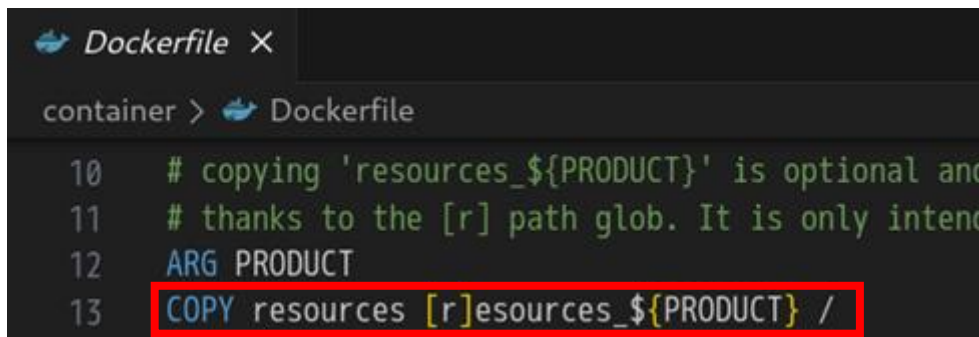
その他、上記ではインストールできない様なものも、個別に追記する事で
インストールする事が出来ます。

例) `RUN python3 -m pip install pyserial`

③アプリケーションの作成

- コンテナ内で使用するファイルを配置(Dockerfile)

example1/container/resources ディレクトリにコンテナ内にコピーしたいファイルがある場合、ビルド時に一括コピーする事が出来ます。



```
Dockerfile X
container > Dockerfile
10 # copying 'resources_${PRODUCT}' is optional and
11 # thanks to the [r] path glob. It is only intend
12 ARG PRODUCT
13 COPY resources [r]resources_${PRODUCT} /
```

例として、コンテナ内に `/root/app/app.txt` というファイルを配置したい場合は `resources/root/app/app.txt` に配置する事で当該ディレクトリにコピーされます。

製品名の付いた `resources_[製品名]` がある場合がありますが、こちらは製品固有のファイルの為、編集はしません。

③アプリケーションの作成

- その他特殊な設定 (Dockerfile)
以下は一例です。
 - コンテナ内のユーザーの追加
例) RUN useradd -m -u 1000 atmark
 - コンテナ内でビルド
例) RUN gcc test.c -o test.exe

その他、Dockerfileでボリューム指定(VOLUME)、実行コマンド指定(CMD)、環境変数設定(ENV)の指定なども行う事ができますが、それらに関しては [example1/config/app.conf](#) で設定できますのでそちらで説明します。

③アプリケーションの作成

■ アプリを実行するスクリプトを作成(main.sh)

シェルスクリプトを作成する場合は、下記の main.sh を編集します。

デフォルトではLEDが1秒周期で点滅し、CPU/SoCの温度をログ保存するサンプルスクリプトが記述されています。セミナーではまずはこのまま動作させてみます。

```
main.sh - example1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
EXAMPLE1
  app/src
    $ main.sh
  config
    app.conf
    app.desc
    ssh_config
  container
    resources
    resources_G4_X2
    Dockerfile
    packages.txt
main.sh
1  #!/bin/bash -e
2
3  # For debug. If you need it, please comment in.
4  #set -x
5
6  LOG_DIR="/vol_data/log"
7  LOG_FILE="$LOG_DIR/temp.txt"
8  LEDS=( app led1 green )
9
10 log_temp() {
11     local now cpu_temp soc_temp
12
13     now=$(TZ=JST-9 date -Iseconds)
```

《補足》

デバッグ用にecho等で標準出力すると、App run on Armadillo実行時にVScodのコンソールに表示されます。

ログ内容

ログ： /var/app/volumes/example1/log/temp.txt

```
atmark@atde9: ~/install_disk
armadillo:~# cat /var/app/volumes/example1/log/temp.txt
time,cpu temperature,soc temperature
2023-10-11T17:39:56+09:00,44,46
```

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

④ビルド～SWUインストール

ここからはArmadilloでデバッグを行う手順となっています。
デバッグを行わない場合は、⑤デバッグ～リリース版SWUインストールのP.83へ
進んでください。

Generate development swu で開発用のSWUファイルを作成します。

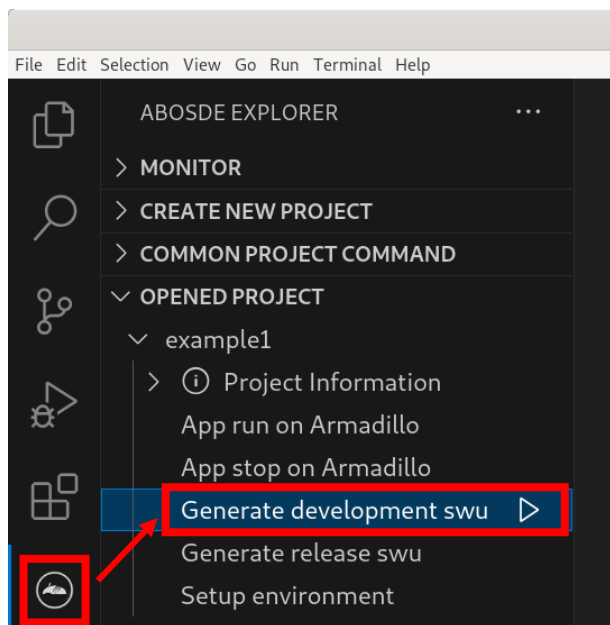
Generate development swuの処理内容

- Dockerfileで設定したコンテナイメージを作成
- スクリプトを/var/app/rollback/volumes/example1/src/main.shに配置
⇒デバッグしやすい様にコンテナ外に配置（コンテナ内の修正には再ビルドが必要な為）
- Armadilloのsshdを有効化する処理を実施
⇒sshdを有効化し、**App run on Armadillo** でソースファイルを書き換え可能とする

④ビルド～SWUインストール

実施手順は ABOSDE Explorer ⇒ [プロジェクト] ⇒ **Generate development swu** の右側の△をクリックで作成します。

ビルド後、SWUファイルを作成する際に署名用のパスワードを聞かれますので `initial_setup` で設定した **testpw** を入力します。



出力ログ

```
bash is already the newest version (5.1-2+deb11u1).
bash set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
--> 34b467f995c
STEP 7: RUN useradd -m -u 1000 atmark
STEP 8: COMMIT localhost/arm64v8/example1:latest
--> 9ed08362447
9ed08362447eff0292ee7a7cb2ee26a5711924a136177b54ae82c4dbc83a3994
コンテナイメージを ./swu/example1.tar に保存しました。
./swu/app.desc のバージョンを 0 から 1 に変更しました。
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
./development.swu を作成しました。
次は Armadillo に ./development.swu をインストールしてください。
* Terminal will be reused by tasks, press any key to close it.
```

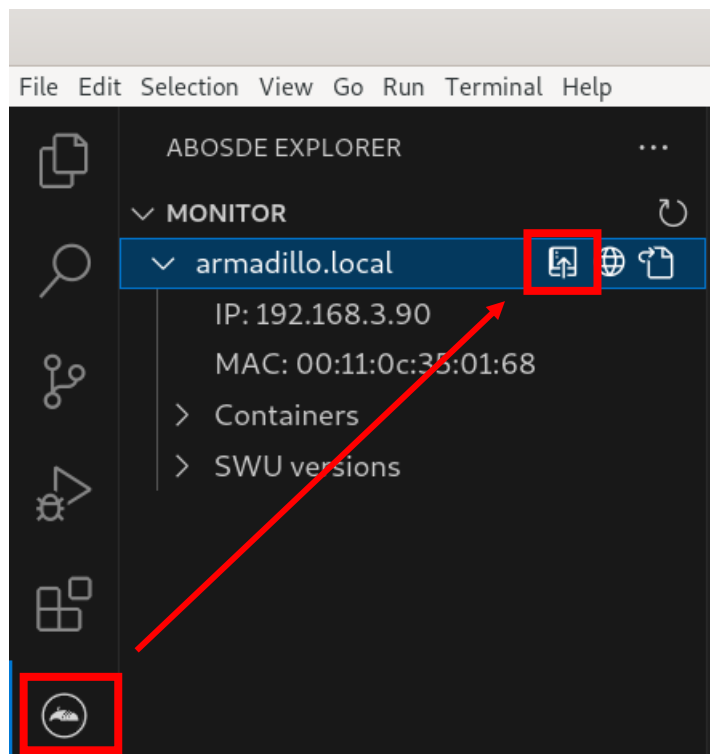
testpw と入力としてEnter

④ビルド～SWUインストール

下記ボタンからABOS Web経由でSWUファイルをインストールします。

※ABOS Webのブラウザ経由でインストールしても問題ありません

ファイルパス：[~/seminar/example1/development.swu](#)



インストールが完了するとArmadilloが自動で再起動し、本アプリケーションのコンテナが立ち上がると対応するLEDが点滅するようになります。

※点滅するLEDは製品毎で異なります

《補足》

Armadilloを初期化したり別のArmadilloで実行する場合、下記エラーが出る事があります。その場合はDeleteを選択する事で実行できます。

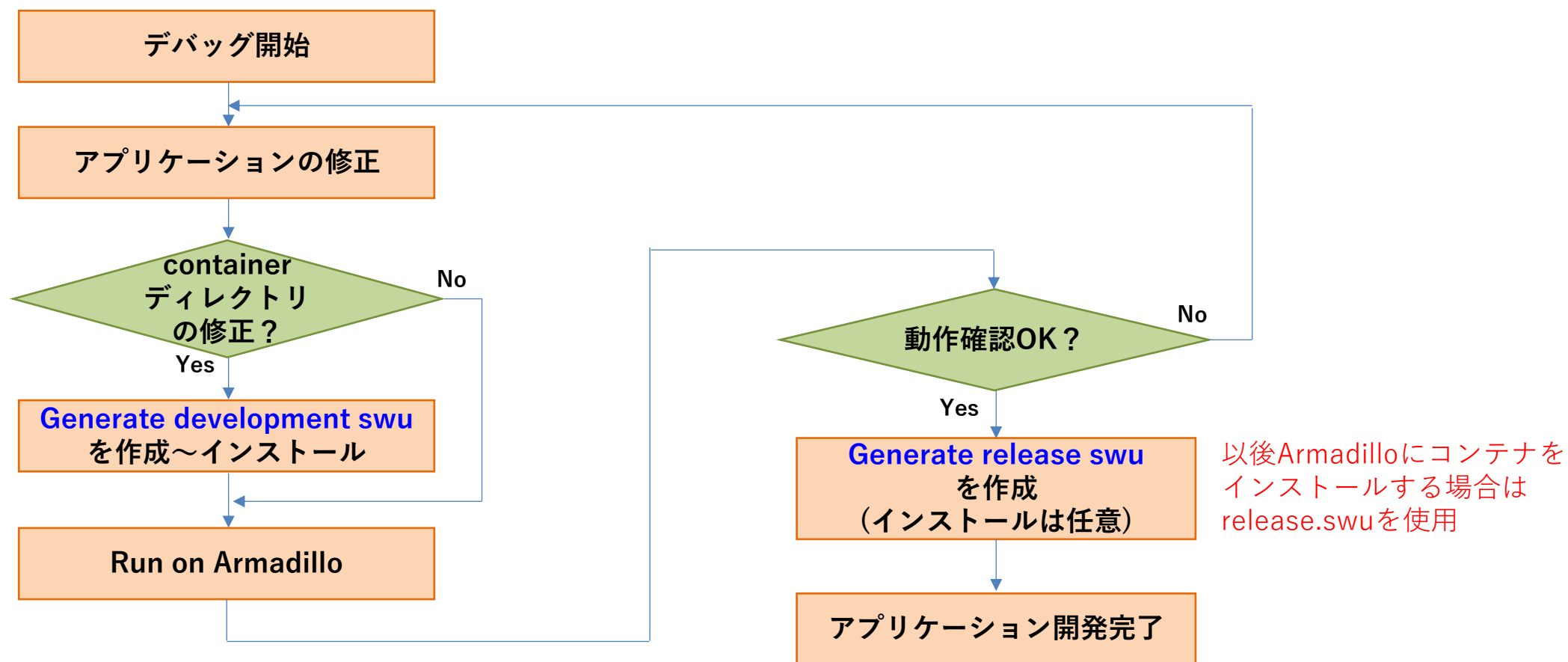
Certificate associated with token does not match. Have you changed Armadillo's certificate?
Delete tokens stored in ABOSDE that have failed access?

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

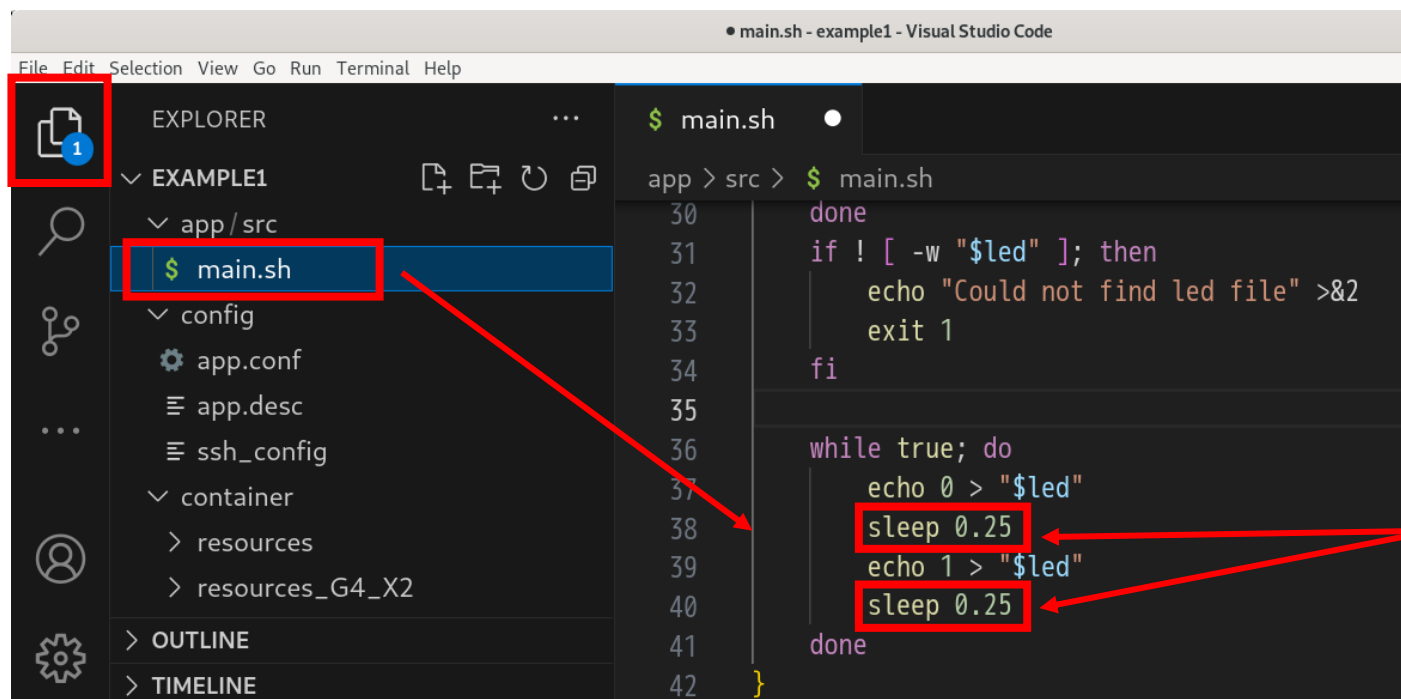
⑤ デバッグ～リリース版SWUインストール

以下のフローでArmadilloでの動作確認を行いつつ、ソースやコンテナの修正を行います。



⑤ デバッグ～リリース版SWUインストール

ここではデバッグの例として、LEDの点滅間隔を変えてみます。
現在の点滅周期1秒(0.5秒でON/OFF)を、0.5秒(0.25秒でON/OFF)とします。
修正が完了したら Ctrl+S で保存します。



```
File Edit Selection View Go Run Terminal Help
EXPLORER
EXAMPLE1
  app/src
    $ main.sh
  config
    app.conf
    app.desc
    ssh_config
  container
    resources
    resources_G4_X2
  OUTLINE
  TIMELINE

$ main.sh
app > src > $ main.sh
30 done
31 if ! [ -w "$led" ]; then
32     echo "Could not find led file" >&2
33     exit 1
34 fi
35
36 while true; do
37     echo 0 > "$led"
38     sleep 0.25
39     echo 1 > "$led"
40     sleep 0.25
41 done
42 }
```

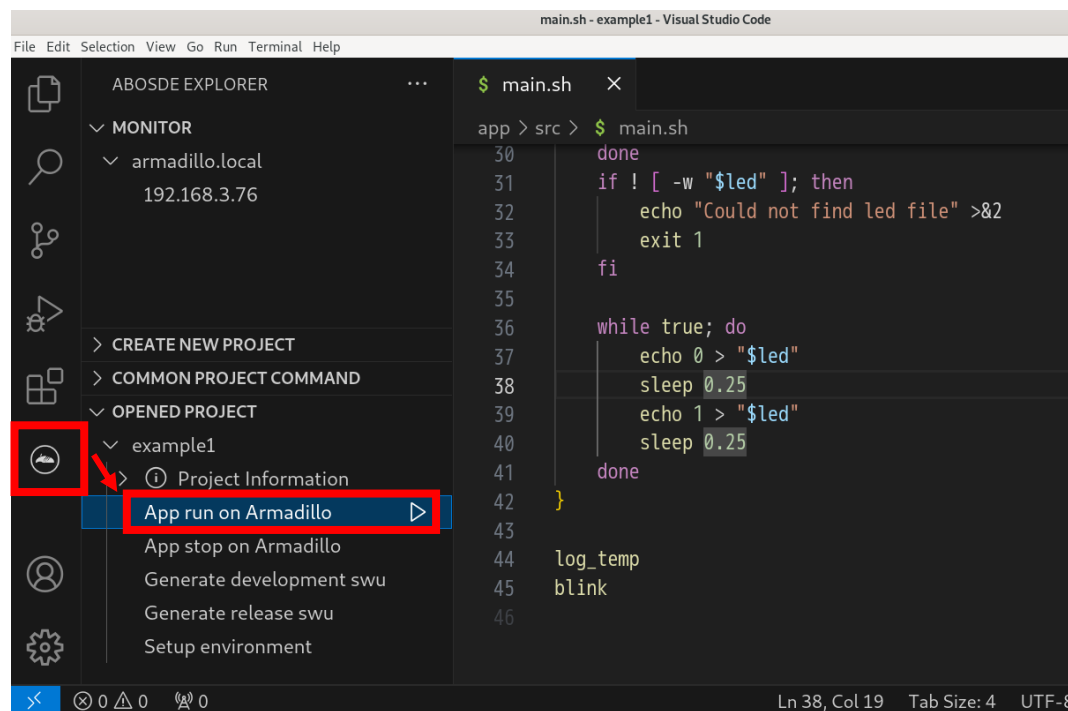
《補足》

main.sh以外にapp.confの
修正もApp run on Armadillo
で適用する事が可能です。

sleepを0.25(秒)に変更

⑤ デバッグ～リリース版SWUインストール

ABOSDE Explorerから、**App run on Armadillo** を実行します。初回実行時にsshのパスワードを聞かれたら **ssh** と入力します。LEDの点滅が0.5秒周期で早く表示されていれば、変更部分がソースコードに反映されています。セミナーではここでデバッグ完了とします。



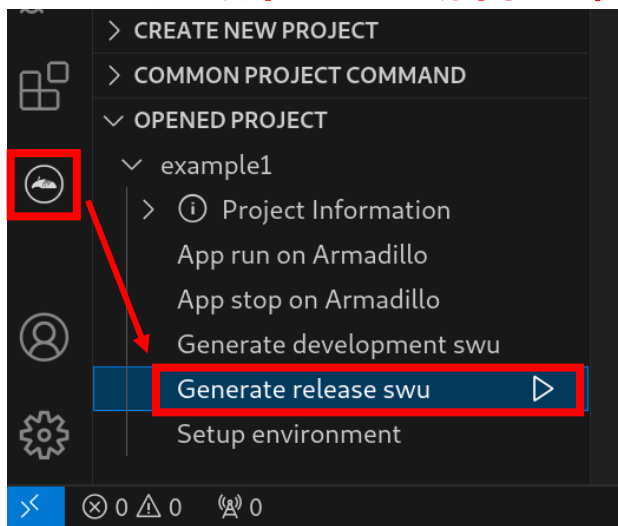
⑤デバッグ～リリース版SWUインストール

デバッグが完了したら、**Generate release swu** の右の△をクリックするとビルドが始まります。途中署名のパスワードを聞かれる為、initial_setupで設定した **testpw** と入力します。

ビルドが成功すると **/seminar/example1/release.swu** が作成されます。

これまでと同様の手順でSWUをインストールし、最終動作確認を行います。

※sshを無効化する場合は最終動作確認前に次頁の内容を実施します。



[インストール手順]

1. Generate release swuを作成
2. ABOS Webを開く
3. ABOS WebでSWUをインストール
4. 最終動作確認

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化

⑥sshd無効化

sshdを無効化する方法は下記方法がありますが、ここではArmadillo上でコマンド実行でsshdを無効化する方法をご案内します。

1. Armadillo上でコマンド実行
2. swuファイルで実行

Armadillo上でsshdを無効化する手順
下記順にコマンドを実行します。

```
[Armadillo]# rc-service sshd stop //sshdサービスを停止
[Armadillo]# rc-update delete sshd //sshdをサービスから消去
[Armadillo]# persist_file -d /etc/runlevels/default/sshd //実行スクリプトを消去
[Armadillo]# persist_file -d /etc/ssh/*key* //鍵の消去
```

⑥sshd無効化(任意)

sshdの有効/無効の状態は下記コマンドで確認できます。

sshd有効化時

```
[Armadillo]# rc-status
Runlevel: default
podman-atmark [ started ]
zramswap [ started ]
sshd [ started ]
buttd [ started ]
abos-web [ started ]
connection-recover [ started ]
avahi-daemon [ started ]
chronyd [ started ]
reset_bootcount [ started ]
local [ started ]
Dynamic Runlevel: hotplugged
Dynamic Runlevel: needed/wanted
fscck [ started ]
root [ started ]
dbus [ started ]
Dynamic Runlevel: manual
modemmanager [ started 00:02:59 (0) ]
```

sshdが消えている

sshd無効化時

```
[Armadillo]# rc-status
Runlevel: default
podman-atmark [ started ]
zramswap [ started ]
buttd [ started ]
abos-web [ started ]
connection-recover [ started ]
avahi-daemon [ started ]
chronyd [ started ]
reset_bootcount [ started ]
local [ started ]
Dynamic Runlevel: hotplugged
Dynamic Runlevel: needed/wanted
fscck [ started ]
root [ started ]
dbus [ started ]
Dynamic Runlevel: manual
modemmanager [ started 00:07:01 (0) ]
```

《参考》 コンテナイメージの削除

ATDEのコンテナイメージ削除

ATDEで開発をしていく過程でATDEのディスク容量が増えていきますので、適宜削除していくことを推奨します。削除コマンドは下記で実行出来ます。

```
[ATDE]$ podman images //全コンテナイメージの表示  
[ATDE]$ podman rmi debian:bullseye-slim //当該イメージの削除  
[ATDE]$ podman image prune -a //全イメージの削除
```

ビルドしたコンテナイメージの最新版は各プロジェクトのswuディレクトリの下に **プロジェクト名.tar** として保存されています。
このコンテナイメージを使用する場合は下記コマンドを実行します。

```
[ATDE]$ podman load -i example1.tar //コンテナイメージのロード
```


※P.89～P.108までArmadillo-IoT A6E向けの内容となります。

■作成するアプリケーション

Armadillo-IoT A6Eの場合

②DI(Digital Input)操作し、LEDを点灯/消灯させる

コンテナ起動後、下記内容のスクリプトを実行

- ・ DI1がHighを検出している間はAPPのLEDを消灯
- ・ DI1がLowを検出するとAPPのLEDを点灯
- ・ DI1がLowを検出した日時をログ出力
- ・ 5秒間隔で検出

本内容のポイント

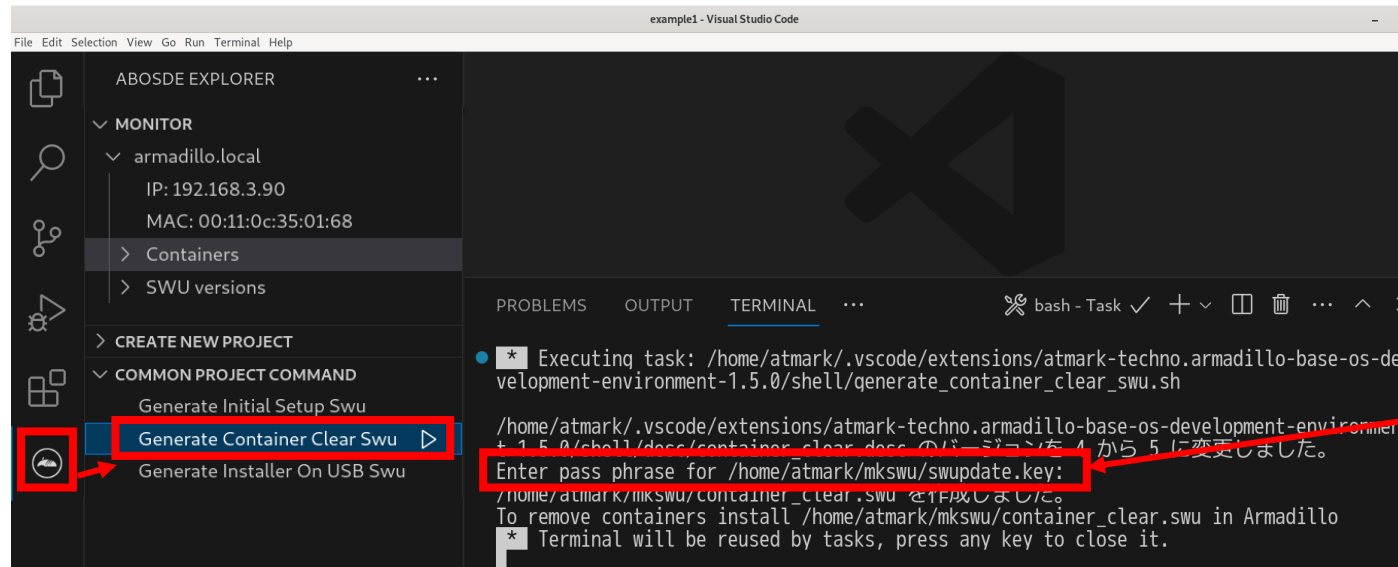
- ・ 別のコンテナイメージを使用する (Debian⇒Alpine)
- ・ デバイス(GPIO)をコンテナで操作する
- ・ 動作ログの保存

■作成するアプリケーション

まず最初に、アプリケーション開発①で作成したコンテナを消去します。
再度 **Generate Container Clear Swu** の右の△をクリックし、container_clear.swuを再作成し、本SWUをインストールしてコンテナを削除します。

作成場所： `/home/atmark/mkswu/container_clear.swu`

※SWUにはバージョンがあり、同じバージョンは使えない為再作成します。



《インストール手順》

- ①SWUを作成
- ②ABOS Webを開く
- ③SWUをインストール

パスワード **testpw** を入力

■作成するアプリケーション

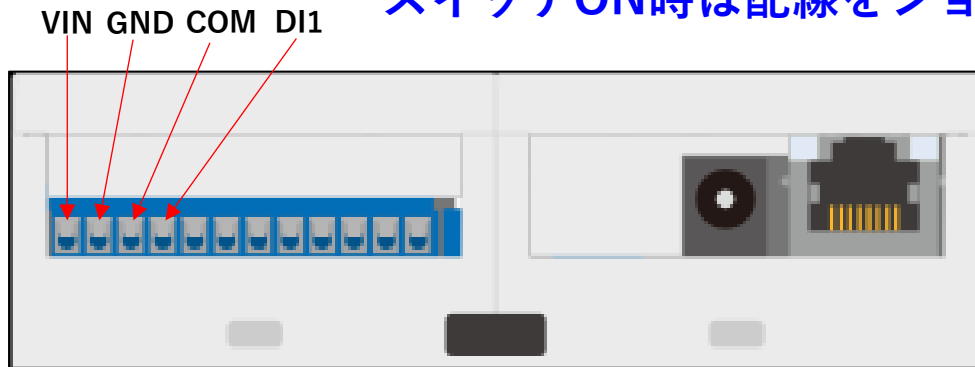
コンテナの削除が終わったら、下記の様に接続を変更します。
配線作業は安全を考慮し、**Armadilloをpoweroffしてシャットダウンした後、ACアダプターを抜いた状態**で実施下さい。配線が完了したら電源投入します。

```
[armadillo]# poweroff
```

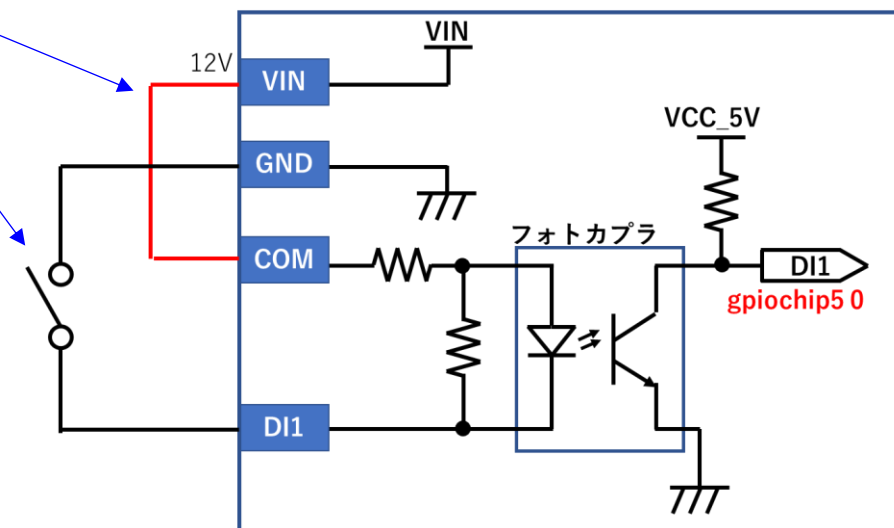
//パワーオフ

この様に配線する
※VINとGNDショートには注意

スイッチON時は配線をショート



Armadillo-IoT A6E(端子台側)



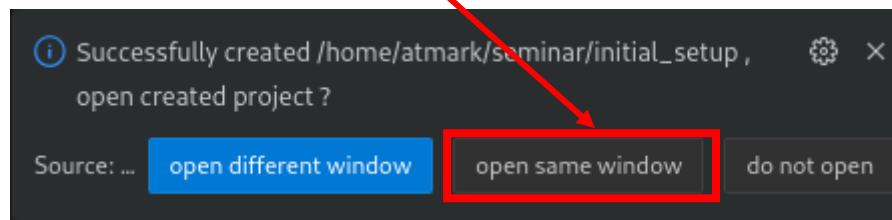
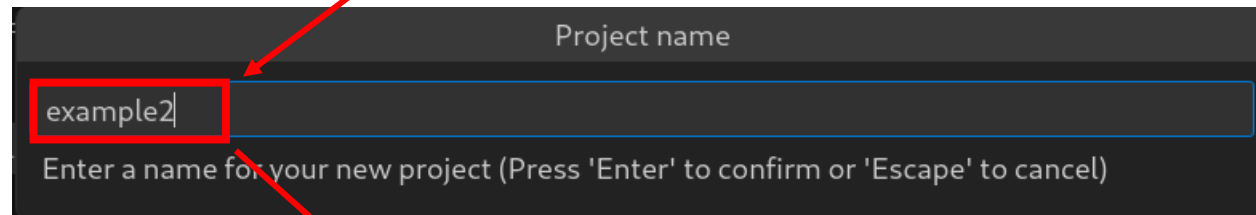
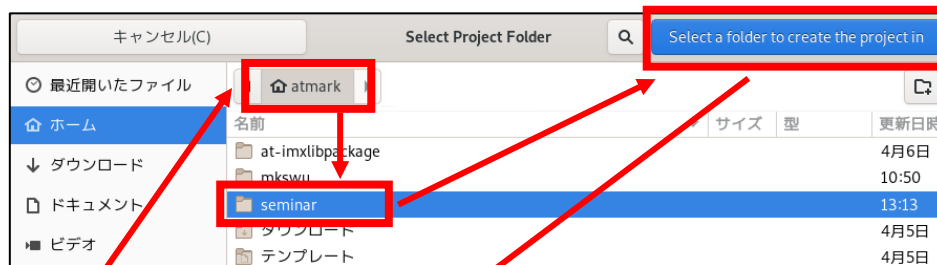
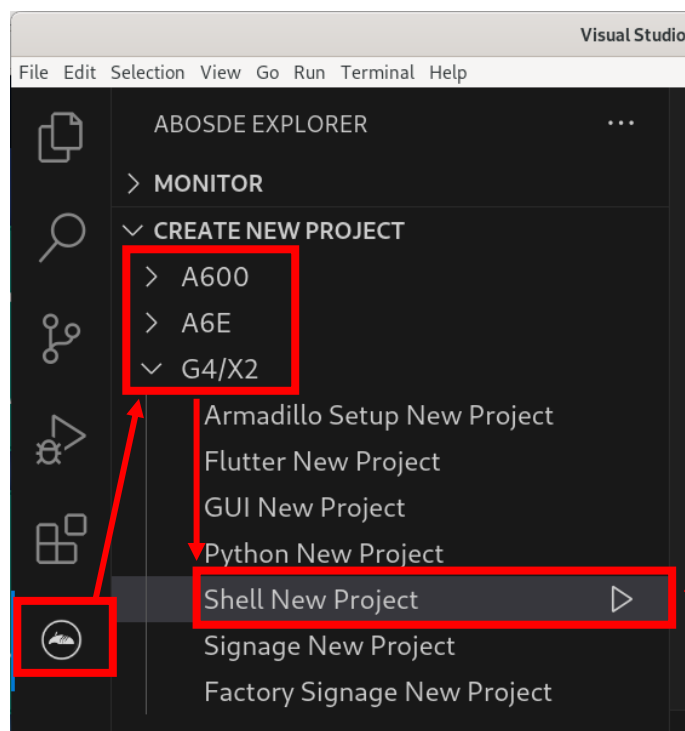
Armadillo-IoT A6E DI部(簡略図)

■ アプリケーション開発手順

- ① 新規プロジェクト作成
- ② SSH接続設定
- ③ アプリケーションの作成
- ④ ビルド～SWUインストール
- ⑤ デバッグ～リリース版SWUインストール
- ⑥ sshd無効化(任意)

①新規プロジェクト作成

アプリケーション作成の為のプロジェクト **example2** を作成します。
手順はアプリケーション開発①と同じです。



※どちらを選んでも開発可能です

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

②SSH接続設定

- **Setup environment**を行う
ATDEで鍵作成済みの為、実施不要です。
- **対象のArmadilloのIPをssh_configに設定する**
アプリケーション開発①と同様に設定します。
 - ◆Bridge設定の場合
ABOSDE ExplorerのMONITORの四角のボタンをクリックし、
example2/config/ssh_config のIPを書き換えます。
 - ◆NAT設定の場合
example2/config/ssh_config を直接編集します。
- **(後の作業で)アプリ作成後に Generate development swu を実行**
後で実施します。

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

③ アプリケーションの作成

■ コンテナの設定 (app.conf)

app.confを下記の様書き換えます。

```
set_image localhost/{{PROJECT}}:latest //変更不要

add_volumes /var/app/rollback/volumes/{{PROJECT}}:/vol_app //変更不要
add_volumes /var/app/volumes/{{PROJECT}}:/vol_data //変更不要
add_volumes /sys:/sys //LEDのアクセス権限付与(デフォルト)

add_devices /dev/gpiochip5 //DI1(gpiochip5 0)のアクセス権限付与

set_command bash /vol_app/src/main.sh //変更不要
```

③ アプリケーションの作成

- 作成するコンテナイメージを設定(Dockerfile)
使用するコンテナイメージを **alpine:3.20** とし、各コマンドを Debian用のコマンドから、alpine用のコマンドに変更します。
以下をコピー&ペーストでDockerfileに貼り付けます。

Dockerfile

```
ARG ARCH
FROM docker.io/${ARCH}/alpine:3.20 //イメージを alpine:3.20に変更
LABEL version="2.0.0"

ARG PRODUCT
COPY resources [r]esources_${PRODUCT} /

ARG PACKAGES
RUN apk update && apk upgrade ¥ //コマンドをalpine用に変更
  && apk add ${PACKAGES} ¥
  && apk cache clean
```

③アプリケーションの作成

- コンテナにインストールするパッケージを設定(Dockerfile)

本アプリケーションでは **libgpod** をインストールする必要があります。
インストールするパッケージが少なければDockerfileに直接書いても
手間ではありませんが、今回はpackages.txt に追記します。

packages.txt

```
bash
```

```
libgpod
```

```
//GPIO操作用にインストール
```

③アプリケーションの作成

■ アプリを実行するスクリプトを作成(main.sh)

スクリプト main.sh に記載されているコードを全削除し、下記の様書き換えます。

※実運用でログを残す場合はログローテーションを行うなどでログのサイズに注意下さい。

main.sh

```
#!/bin/sh

while :
do
  result=$(gpioget gpiochip5 0) //D1(gpiochip5 0)の値を取得
  if [ ${result} = 0 ]; then //D1がLowの分岐
    echo 1 > /sys/class/leds/app/brightness //LEDを点灯させる
    echo `date` >> /vol_data/log.txt //eMMCにログを保存
    echo -e "DI1=${result}\nDI1 Low" //デバッグログ出力例
  else //D1がHighの分岐
    echo 0 > /sys/class/leds/app/brightness //LEDを消灯させる
    echo -e "DI1=${result}\nDI1 High" //デバッグログ出力例
  fi
  sleep 5 //5秒SLEEP
done
```

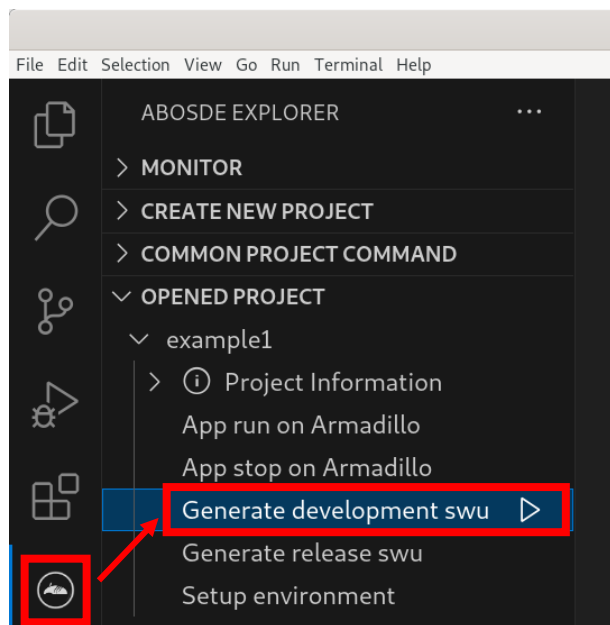
■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

④ビルド～SWUインストール

Generate development swu で開発用のSWUファイルを作成し、Armadilloへインストールします。(sshdが有効化されます)

再起動後、**DI1をショートすると5秒毎でLEDの点灯/消灯の判定**が行われます。その後、**App run on Armadillo** を実行するとVScodeコンソールに先ほど記載したデバッグログが表示されるようになります。



出力ログ

```
bash is already the newest version (5.1-2+deb11u1).
bash set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
--> 34b467f995c
STEP 7: RUN useradd -m -u 1000 atmark
STEP 8: COMMIT localhost/arm64v8/example1:latest
--> 9ed08362447
9ed08362447eff0292ee7a7cb2ee26a5711924a136177b54ae82c4dbc83a3994
コンテナイメージを ./swu/example1.tar に保存しました。
./swu/app.desc のバージョンを 0 から 1 に変更しました。
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
./development.swu を作成しました。
次は Armadillo に ./development.swu をインストールしてください。
* Terminal will be reused by tasks, press any key to close it.
```

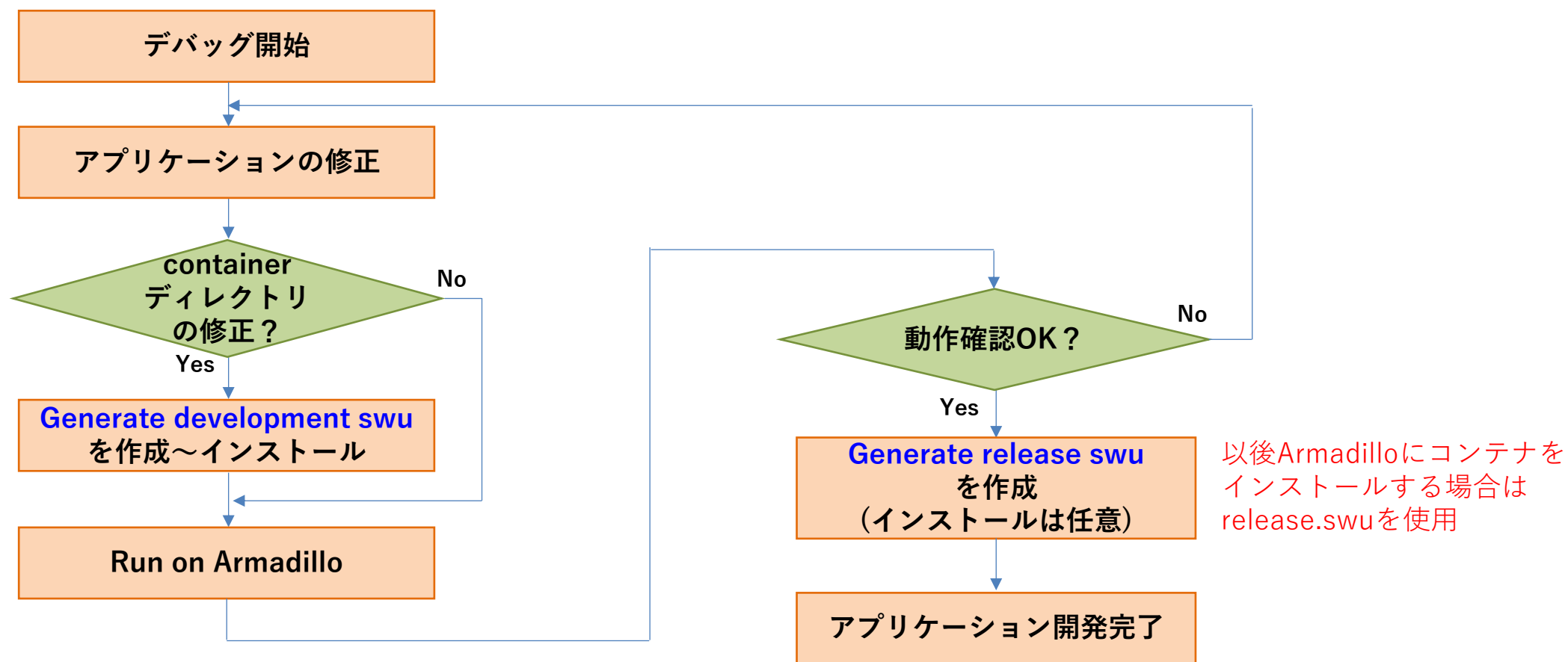
testpw と入力してEnter

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

⑤ デバッグ～リリース版SWUインストール

以下のフローでArmadilloでの動作確認を行いつつ、ソースやコンテナの修正を行います。



⑤ デバッグ～リリース版SWUインストール

分岐は正常に動作していたが検出にディレイが生じる為、 **main.sh** を下記の様に変更し、**App run on Armadillo** で問題無く動作している事を確認します。

- ・ **DI1がLowを検出したら5秒SLEEPとする**
- ・ **5秒毎のSLEEPを削除**
- ・ **デバッグログ削除**

main.sh

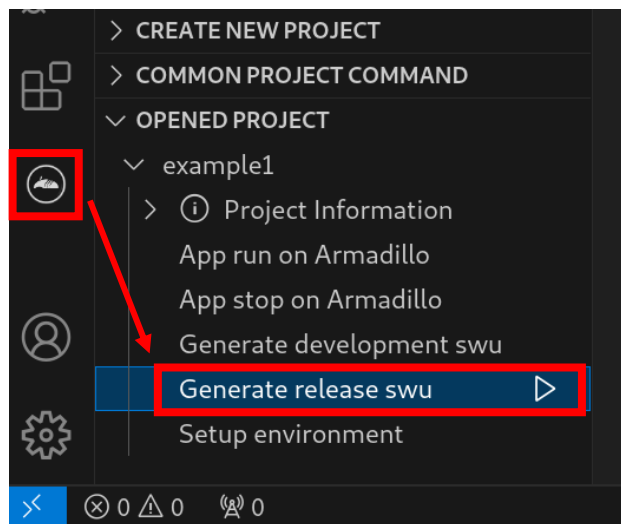
```
#!/bin/sh

while :
do
  result=$(gpioget gpiochip5 0)
  if [ ${result} = 0 ]; then
    echo 1 > /sys/class/leds/app/brightness
    echo `date` >> /vol_data/log.txt
    sleep 5 //5秒SLEEPを追加
  else
    echo 0 > /sys/class/leds/app/brightness
  fi
done
```

⑤ デバッグ～リリース版SWUインストール

デバッグが完了したら、**Generate release swu** を実行します。
作成された **release.swu** をABOS Webでインストールし、再起動後に最終動作確認を行います。

※**sshdを無効化**する場合は最終動作確認前にsshdを無効化しておきます。



[インストール手順]

1. Generate release swuを作成
2. ABOS WebでSWUをインストール
3. (sshd無効化)
4. 最終動作確認

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化

⑥sshd無効化

sshdを無効化する場合は下記を実行します。

Armadillo上でsshdを無効化する手順

```
[Armadillo]# rc-service sshd stop //sshdサービスを停止
[Armadillo]# rc-update delete sshd //sshdをサービスから消去
[Armadillo]# persist_file -d /etc/runlevels/default/sshd //実行スクリプトを消去
[Armadillo]# persist_file -d /etc/ssh/*key* //鍵の消去
```

最終動作確認が問題無ければこれでアプリケーション②の開発は完了です。

■作成するアプリケーション

※P.109～P.130までArmadillo-IoT G4向けの内容となります。
Armadillo-X2でも実施可能です。

Armadillo-IoT G4の場合

③USBカメラ画像をHDMI出力しつつVPUでエンコードする

コンテナ起動後、下記内容のスクリプトを実行

- ・ westonを起動
- ・ gstreamerを使用してUSBカメラからの映像をHDMI表示
- ・ 上記と同時にH.264で録画(音声無し)
- ・ 録画ファイルは/var/app/volumes/example2/に保存

本内容のポイント

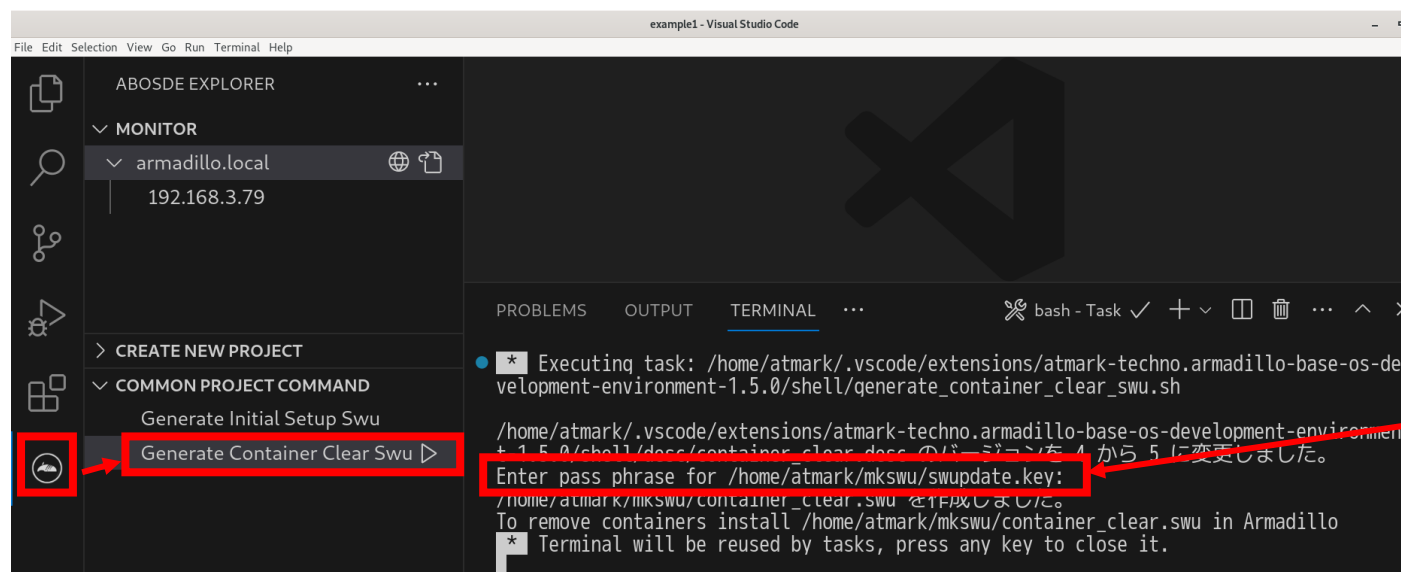
- ・ アットマークテクノ提供のDebianコンテナイメージを使用する
- ・ HDMIで画面出力する
- ・ デバイス(USBカメラ,VPU,HDMI等)をコンテナで使用する
- ・ 録画ファイルを保存する

■作成するアプリケーション

まず最初に、アプリケーション開発①で作成したコンテナを消去します。
再度 **Generate Container Clear Swu** の右の△をクリックし、container_clear.swuを再作成し、本SWUをインストールしてコンテナを削除します。

作成場所： `/home/atmark/mkswu/container_clear.swu`

※SWUにはバージョンがあり、同じバージョンは使えない為再作成します。



《インストール手順》

- ①SWUを作成
- ②ABOS Webを開く
- ③SWUをインストール

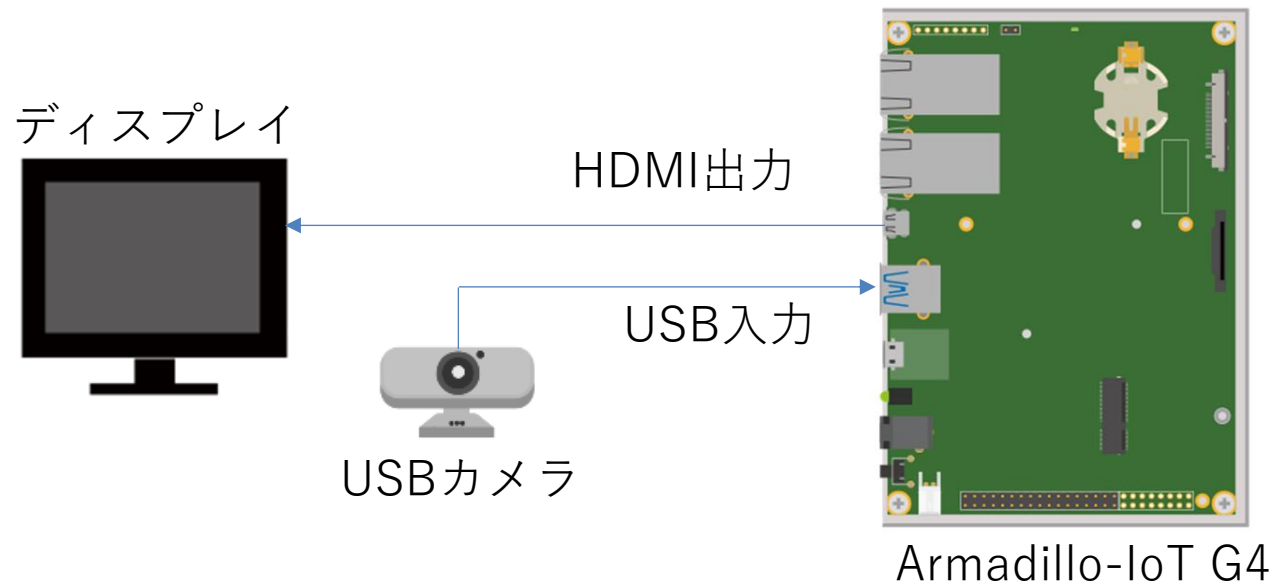
パスワード **testpw** を入力

■作成するアプリケーション

コンテナの削除が終わったら、下記の様に接続を変更します。

■追加のデバイスを接続（右図）

- ・ USBカメラ
- ・ ディスプレイ

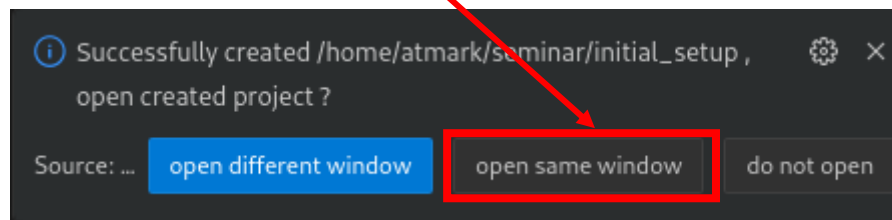
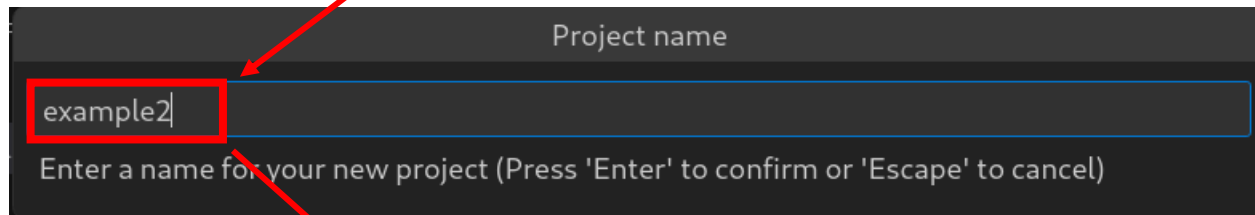
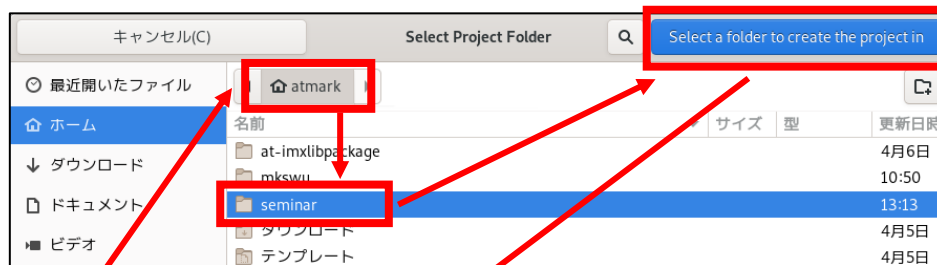
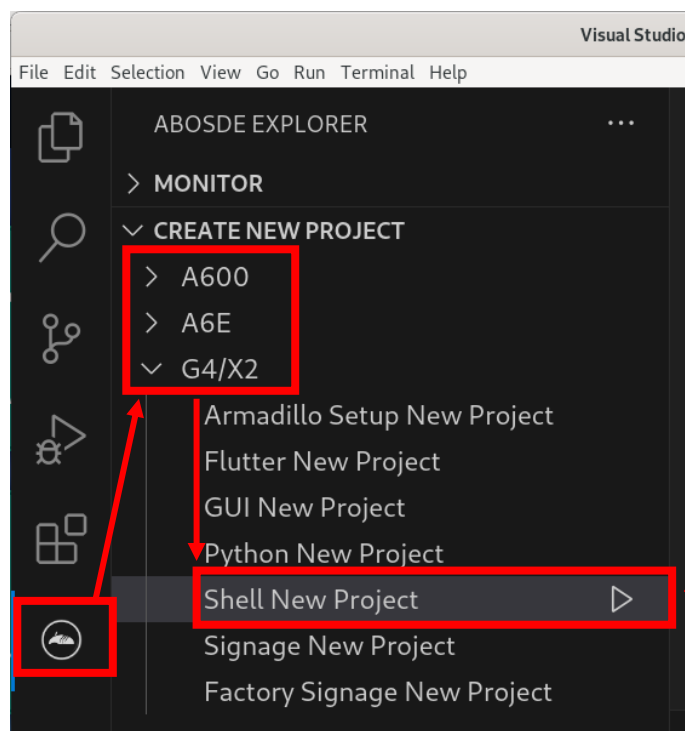


■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

①新規プロジェクト作成

アプリケーション作成の為のプロジェクト **example2** を作成します。
手順はアプリケーション開発①と同じです。



※どちらを選んでも開発可能です

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

②SSH接続設定

- **Setup environment**を行う
ATDEで鍵作成済みの為、実施不要です。
- **対象のArmadilloのIPをssh_configに設定する**
アプリケーション開発①と同様に設定します。
 - ◆Bridge設定の場合
ABOSDE ExplorerのMONITORの四角のボタンをクリックし、
example2/config/ssh_config のIPを書き換えます。
 - ◆NAT設定の場合
example2/config/ssh_config を直接編集します。
- **(後の作業で)アプリ作成後に Generate development swu を実行**
後で実施します。

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

③アプリケーションの作成

■ コンテナの設定 (app.conf)

app.confを下記の様に書き換えます。

```
set_image localhost/{{PROJECT}}:latest //変更不要
add_volumes /var/app/rollback/volumes/{{PROJECT}}:/vol_app //変更不要
add_volumes /var/app/volumes/{{PROJECT}}:/vol_data //変更不要 (録画データの保存先)
#add_volumes /sys:/sys //コメントアウト(もしくは削除)
set_command bash /vol_app/src/main.sh //変更不要

add_devices /dev/dri /dev/galcore //以下追加分
add_devices /dev/mxc_hantro /dev/mxc_hantro_vc8000e //画面描画用
add_devices /dev/ion //VPU用
add_devices /dev/input /dev/tty7 //VPU用
add_hotplugs video4linux //入出力用
add_volumes /run/udev:/run/udev:ro //USBカメラ(hotplug対応)
add_volumes /opt/firmware:/opt/firmware:ro //デバイス管理
add_args --env=XDG_RUNTIME_DIR=/run/xdg_home //ファームウェア
add_args --cap-add=SYS_TTY_CONFIG //westonの設定
add_args --cap-add=SYS_TTY_CONFIG //tty操作権限
```

③アプリケーションの作成

■ 作成するコンテナイメージを設定(Dockerfile)

ベースとするコンテナイメージはアットマークテクノが提供するdebianコンテナを使用します。コンテナイメージ名は **at-debian-image:latest** となります。その他はデフォルトから変更はありません。

Dockerfile

```
ARG ARCH
FROM localhost/at-debian-image:latest           //イメージを at-debian-imageに変更
LABEL version="2.0.0"                          (アットマークテクノ提供のdebianコンテナ)

ARG PRODUCT
COPY resources [r]esources_${PRODUCT} /

ARG PACKAGES
RUN apt-get update && apt-get upgrade -y ¥
  && apt-get install -y ${PACKAGES} ¥
  && apt-get clean
```

《参考》 Debianコンテナイメージ

- **アットマークテクノ提供のコンテナイメージとは**

アットマークテクノではArmadillo-IoT G4,Armadillo-X2向けにNPUやVPUを使用可能なDebianコンテナを提供しております。

URL : <https://armadillo.atmark-techno.com/resources/software/armadillo-iot-g4/container>

ATDEで使用可能にする手順 ※下記[version]は適宜修正下さい

```
[ATDE]$ wget https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-g4/container/at-debian-image-[version].tar //コンテナイメージのダウンロード
[ATDE]$ podman load -i at-debian-image-[version].tar //コンテナイメージのロード
[ATDE]$ podman images //コンテナイメージの確認
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/at-debian-image	1.0.11	b199e6407c47	5 weeks ago	262 MB
localhost/at-debian-image	latest	b199e6407c47	5 weeks ago	262 MB

《補足》

at-debian-image-[version].tar はdebian:bullseyeのコンテナイメージにwestonのみインストール済み。その他リポジトリの追加、環境変数などの設定を行っています。詳細は同URLにあるDockerfileの内容を参照下さい。

③アプリケーションの作成

- コンテナにインストールするパッケージを設定(Dockerfile)
本アプリケーションでは gstreamer を使用する為、下記パッケージをインストールします。

packages.txt

```
bash
gstreamer1.0-imx
gstreamer1.0-imx-tools
gstreamer1.0-tools
gstreamer1.0-plugins-good
gstreamer1.0-plugins-bad
```


③アプリケーションの作成

- アプリを実行するスクリプトを作成(main.sh)

スクリプト main.sh に記載されているコードを全削除し、下記の様書き換えます。

※実運用で録画ファイルを残す場合はeMMCの容量にご注意ください。

main.sh

```
#!/bin/bash

weston --tty=7 &                                     //westonを起動
sleep 5                                              //起動待機時間

gst-launch-1.0 -e v4l2src device=/dev/video2 ¥       //gstreamer実行
! video/x-raw,width=640,height=480,framerate=30/1 ¥ //カメラ設定
! tee name=t1 ! queue ! vpuenc_h264 ! h264parse ! queue ¥ //エンコード設定
! qtmux ! filesink location=/vol_data/output.mp4 t1. ! queue ¥ //録画ファイル出力
! waylandsink window-width=1920 window-height=1080 //表示設定
```

■アプリケーション開発手順

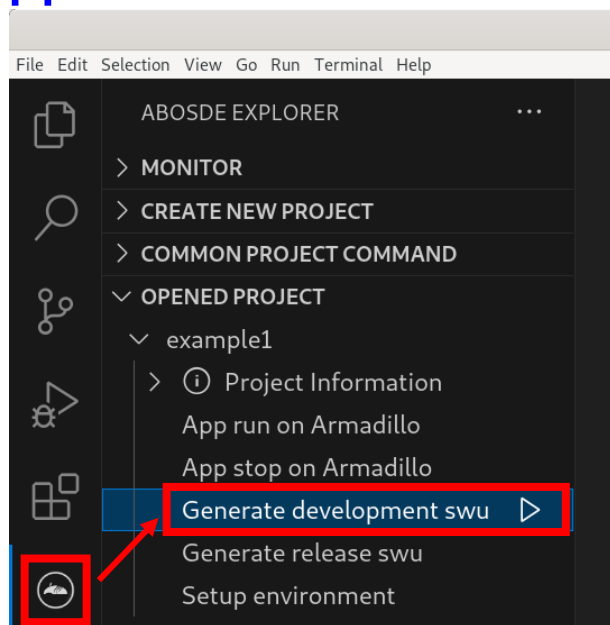
- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

④ビルド～SWUインストール

Generate development swu で開発用のSWUファイルを作成し、Armadilloへインストールします。(sshdが有効化されます)

ファイルパス:~/seminar/example2/development.swu

再起動後、HDMIでwestonが起動(灰色の画面)し、その後カメラ画像が表示されます。**App run on Armadillo** を実行する事でコンソールログが表示されるようになります。



《参考》 Westonの解像度設定

④ビルド～SWUインストール

westonの設定(at-debian-imageで設定)は1920×1080に設定しています。
westonの解像度設定の変更を行う場合は、コンテナ内の/etc/xdg/weston/weston.ini
を書き換えます。下記ファイルを修正してresources/etc/xdg/weston/weston.ini
に配置する事でも書き換えが可能です。

weston.ini

```
[core]
idle-time=0
use-g2d=1
xwayland=true
repaint-window=16

[shell]
panel-position=none

[output]
name=HDMI-A-1
mode=1920x1080

[output]
name=LVDS-1
mode=off
```

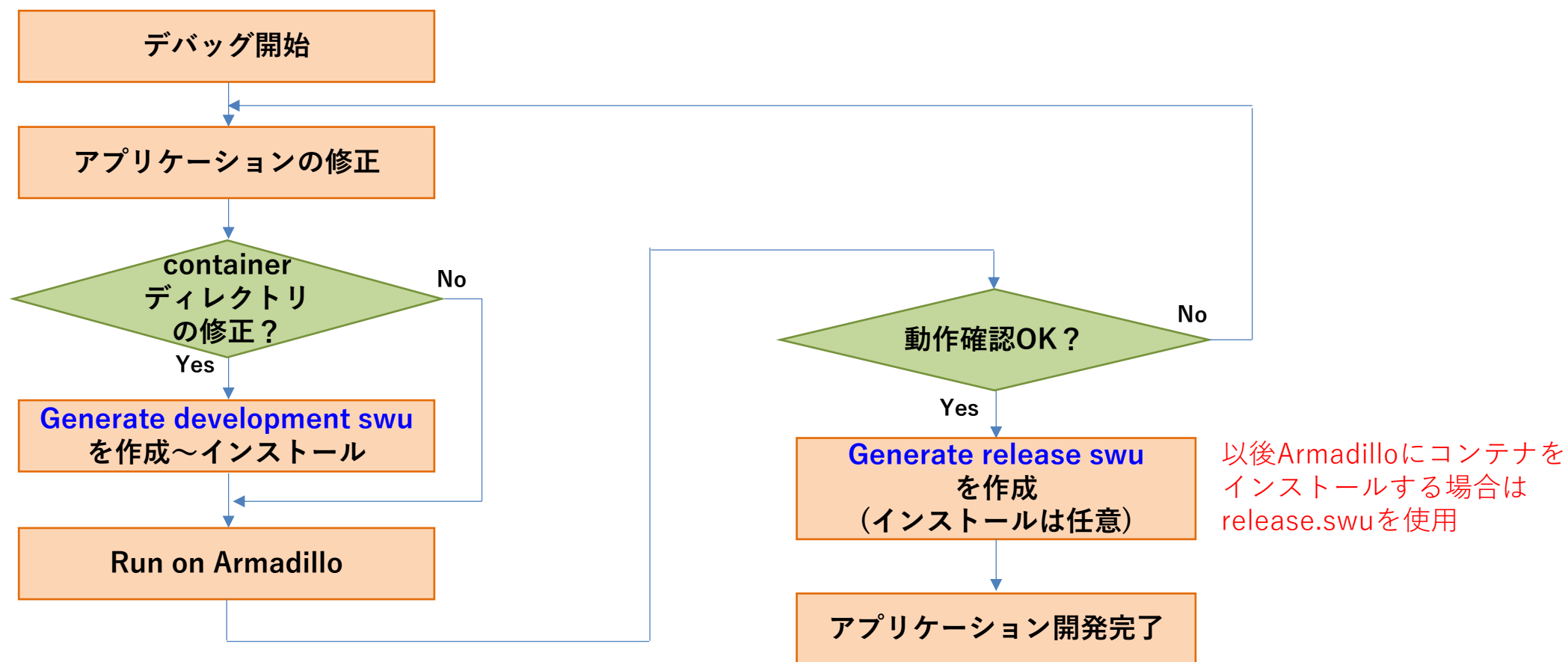
← 解像度設定

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化(任意)

⑤ デバッグ～リリース版SWUインストール

以下のフローでArmadilloでの動作確認を行いつつ、ソースやコンテナの修正を行います。



⑤ デバッグ～リリース版SWUインストール

セミナーでは特にデバッグは行いませんので、ここでアプリケーションのデバッグは完了とします。

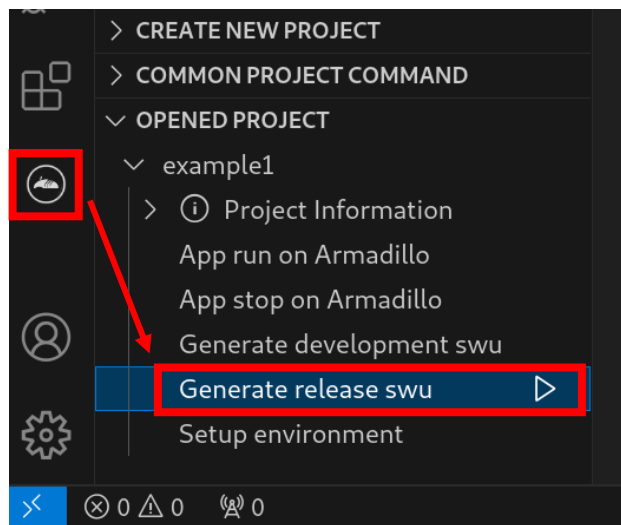
また、アプリケーションを実行したときに、エンコードしてファイル保存する設定にしていたのでファイルの確認だけ行います。（セミナーでは再生はしません）
実行時にエラーが出ていなければ下記ディレクトリに保存されています。

```
[Armadillo]# ls /var/app/volumes/example2  
output.mp4
```

⑤デバッグ～リリース版SWUインストール

デバッグが完了したら、**Generate release swu** を実行します。
作成された **release.swu** をABOS Webでインストールし、再起動後に最終動作確認を行います。

※**sshdを無効化**する場合は最終動作確認前にsshdを無効化しておきます。



[インストール手順]

1. Generate release swuを作成
2. ABOS WebでSWUをインストール
3. (sshd無効化)
4. 最終動作確認

■アプリケーション開発手順

- ①新規プロジェクト作成
- ②SSH接続設定
- ③アプリケーションの作成
- ④ビルド～SWUインストール
- ⑤デバッグ～リリース版SWUインストール
- ⑥sshd無効化

⑥sshd無効化

sshdを無効化する場合は下記を実行します。

Armadillo上でsshdを無効化する手順

```
[Armadillo]# rc-service sshd stop //sshdサービスを停止
[Armadillo]# rc-update delete sshd //sshdをサービスから消去
[Armadillo]# persist_file -d /etc/runlevels/default/sshd //実行スクリプトを消去
[Armadillo]# persist_file -d /etc/ssh/*key* //鍵の消去
```

最終動作確認が問題無ければこれでアプリケーション③の開発は完了です。

- **アプリケーションの開発**
 - ① 新規プロジェクト作成
 - ② SSH接続設定
 - ③ アプリケーションの作成
 - ④ ビルド～SWUインストール
 - ⑤ デバッグ～リリース版SWUインストール
 - ⑥ sshd無効化(任意)

第5部：ソフトウェアアップデート設定



- ソフトウェアアップデート方法
- ソフトウェアアップデート条件
- ソフトウェアアップデートの選択と設定

ソフトウェアアップデート設定

ソフトウェアアップデート方法

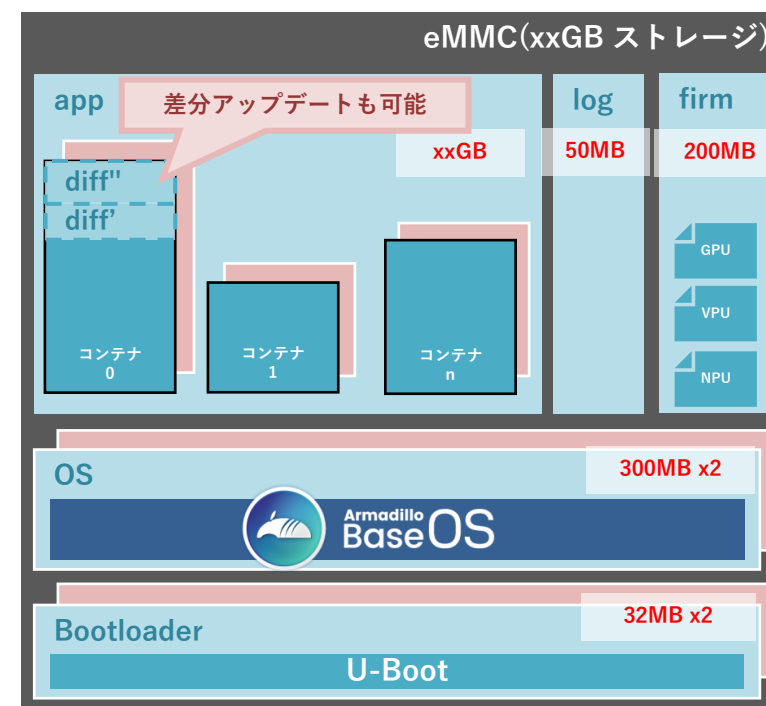
ソフトウェアのアップデートは下記方法でリモート/ローカルで行う事が出来ます。
アップデート対象：ファイル単体からOS、コンテナにも対応（差分アップデートも可）

■アップデート方法



署名済みイメージ

「署名されたイメージ」を検証して書き込む（改ざん防止）



ソフトウェアアップデート手順

アップデートは下記の手順で行います。 ※セミナーでは詳細の手順は省略

アップデート手順

1. **Generate Initial Setup Swu を実行～インストール**
開発時にArmadilloにinitial_setup.swuをインストールしておく
※第3部P.35～43で実施済み
2. **ATDEでアップデート内容を記載したdescファイルを作成**
descファイル書式はマニュアルや、ATDEの/usr/share/mkswu/examples/を参照
3. **ATDEで mkswu コマンドを実行**
コマンド(mkswu xxxx.desc)を実行するとアップデート用ファイル(xxxx.swu)が作成され、内部に~/mkswu/swupdate.pemの署名情報も書き込まれる
4. **Armadilloでアップデートファイルをインストール**
アップデート方法は、リモート/ローカルで実施可能

ソフトウェアのアップデート条件

作成したアップデートファイル(.swu)をArmadilloへインストールするには下記の2つの条件に合致している必要があります。(initial_setupを除く)

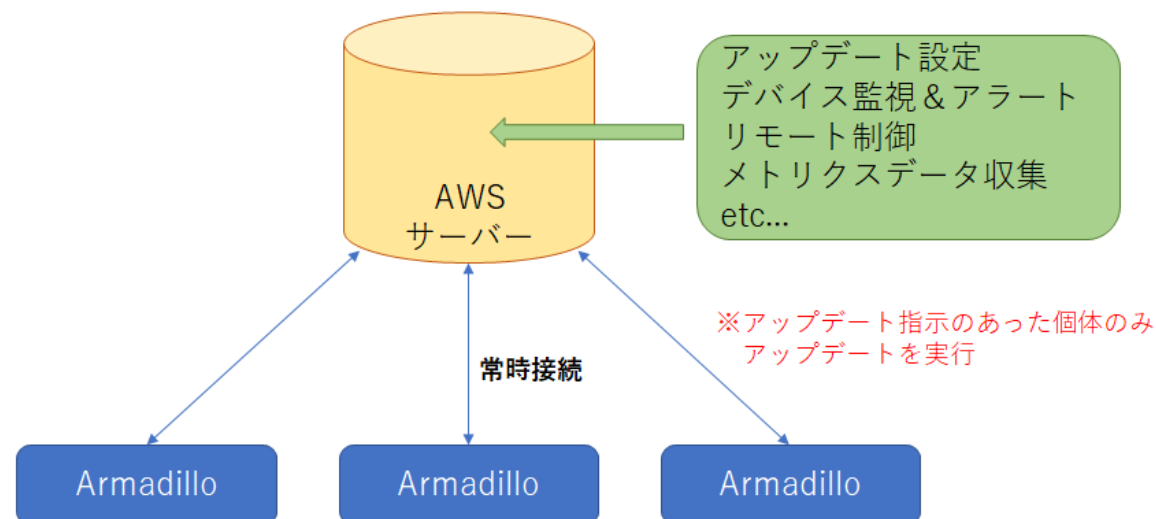
アップデート条件

- アップデートファイル内の署名が対象のArmadillo内の署名のいずれかと一致している事
Armadillo内の署名(/etc/swupdate.pem)はinitial_setup.swuインストール時に同時に書き込まれています。
- アップデートファイル内のバージョン(.descで指定)が対象のArmadilloのバージョン情報(/etc/sw-versions)よりも新しい事(0<9,a<z)、または新規のバージョン名である事
VScodeで release swu で作成する場合は、前回作成のバージョンから自動でカウントアップします。

リモートアップデートの選択と設定

■ Armadillo Twinでのアップデート方法

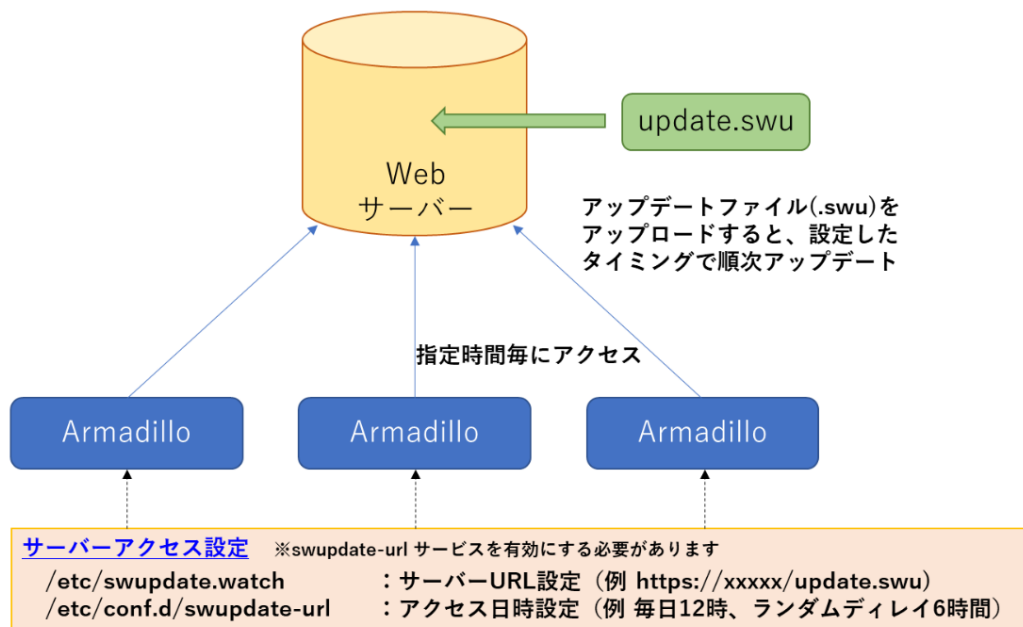
Armadillo Twinは各Armadilloと常時接続している為、AWSサーバーにアップロードしたアップデートファイル(.SWU)を使って各Armadilloにアップデート指示を出す事が出来ます。



リモートアップデートの選択と設定

■ 簡易Webサーバーでのアップデート方法

簡易Webサーバーでアップデートを行う場合、アップデートファイルを指定したサーバーへアップロードします。各々のArmadilloが準備指定時間にポーリングを行い、アップデート可能なファイルがあった場合にアップデートを実行します。



リモートアップデートの選択と設定

アップデート設定及び手順

具体的なアップデート設定や手順につきましては製品マニュアル、または下記ブログを参考に実施頂ければと思います。 ※本セミナーでは省略

Armadillo Base OS : Armadillo Twinでソフトウェアをリモートアップデートする方法
<https://armadillo.atmark-techno.com/blog/15349/19343>

Armadillo Base OS : 簡易Webサーバーでソフトウェアをリモートアップデートする方法
<https://armadillo.atmark-techno.com/blog/15349/12602>

《参考》 ファイルの保護について

- **Armadillo Base OSをアップデート**するとArmadillo Base OS内で作成したファイルは一部を除いて引き継がれません。
- 引き継ぐ場合はArmadilloの `/etc/swupdate_preserve_files` に記載します。
本ファイルに記載する事で、アップデート後に記載のファイルがコピーされます。

■ `persist_file -p`で`/etc/swupdate_preserve_files`に追記（直接編集も可）

```
[armadillo]# touch /root/test.txt //サンプルファイル作成
[armadillo]# persist_file -p /root/test.txt //永続化+上書き保護
[armadillo]# cat /etc/swupdate_preserve_files //追記されている事を確認
<中略>
# persist_file 20221003
/root/test.txt //先ほど作成したファイル
```

- ソフトウェアアップデート方法
 - ・ リモート/ローカルで実施可能
- ソフトウェアアップデート条件
 - ・ 署名とバージョン名が正しい場合のみ実施
- ソフトウェアアップデートの選択と設定
 - ・ Armadilo Twin/簡易Webサーバー

第 6 部：量產準備



- パスワードの変更
- SSH等の固有の鍵情報などの削除
- ROM書き込み方法の選択とファイル作成

■ パスワードの変更

強固なパスワードになっていない場合はパスワードを再設定できます。
不要な場合はスキップして問題ありません。

```
[armadillo]# passwd root //rootのパスワード再設定
[armadillo]# persist_file /etc/shadow //パスワードの永続化

[armadillo]# passwd atmark //atmarkユーザーのパスワード再設定
[armadillo]# persist_file /etc/shadow //パスワード永続化

[armadillo]# adduser test_user //任意のユーザー追加+パスワード設定
[armadillo]# persist_file /etc/shadow ¥ //パスワード、グループの永続化
/etc/passwd /etc/group

[armadillo]# passwd abos-web-admin //ABOS Webのパスワード再設定
[armadillo]# persist_file /etc/shadow //パスワード永続化
```


■ SSH等の固有の鍵情報などの削除

この後に量産用のROM書き込みを行うにあたり、Armadilloの個体毎で固有の鍵を持つ必要が有る場合、鍵をコピーしない様に一旦削除して後で書き込みます。量産時に簡単に固有の鍵を作成する方法は下記ブログを参照下さい。

- インストールディスクでROM書き込みを行う場合
または、ROM書き込みサービスで書き込む場合(製造時に書き込み)
インストール時にスクリプトでsshの鍵を書き込みます。
参考ブログ：<https://armadillo.atmark-techno.com/blog/15349/16876>
- ソフトウェアアップデートでROM書き込みを行う場合
ROM書き込み後、初回起動時にスクリプトで鍵を作成します。
参考ブログ：<https://armadillo.atmark-techno.com/blog/15349/16046>

■ ROM書き込み方法の選択とファイル作成

量産用のROM書き込みを行う方法は下記3種類あります。

- インストールディスクを作成してArmadilloに書き込む
実機をコピーした**インストールディスクを作成**します。
(鍵情報もコピーしてしまう為、前頁で削除しておきます)
- 当社ROM書き込みサービスを利用する
ROM書き込みサービスをご利用頂く事で出荷時に書き込みを行います。
手順は上記**インストールディスク作成**と同じです。
- ソフトウェアアップデートで書き込む
USBメモリやネットワーク経由でROM書き込みを行います。
この場合、ロールバック先も含めて2面(A/B面)とも同じ内容を書き込む必要があります。(2度SWUをインストール)

■ ROM書き込み方法の選択とファイル作成

量産用のROM書き込み用ファイルは下記の様に作成します。

■ インストールディスクを作成する

先にArmadilloに不要なコンテナイメージやコンテナがある場合は、量産機にコピーしない様に事前に削除しておき、microSDカード(※)を入れて下記のコマンドを実行します。

※Armadillo-IoT A6EのWLAN搭載モデルではUSBメモリに作成します。(マニュアル参照)

```
[armadillo]# abos-ctrl make-installer //インストールディスク作成コマンド
```

■ ソフトウェアアップデートで書き込む

下記ブログの様に書き込みが必要なファイルを纏めて、ソフトウェアアップデートで書き込みます。2回書き込む事でA/B面両方に同じものを書き込む事が出来ます。

参考ブログ：<https://armadillo.atmark-techno.com/blog/15349/16549>

- **パスワードの変更**
 - ・ 強固なパスワードに変更
- **SSH等の固有の鍵情報などの削除**
 - ・ 固有鍵の削除と量産時のインストール方法
- **ROM書き込み方法の選択とファイル作成**
 - ・ インストールディスクで書き込み
 - ・ ROM書き込みサービスで書き込み
 - ・ ソフトウェアアップデートで書き込み

第7部：Armadillo Twinについて



- Armadillo Twinとは
- Armadillo Twinの機能

新サービス Armadillo Twin

長期の安定運用を低コストで実現する
デバイス運用管理クラウドサービス



Armadillo
Twin

主な機能

- リモートアップデート
- デバイス死活監視&アラート
- リモート制御(単体/一括/グループ)
- メトリクスデータ収集
etc...

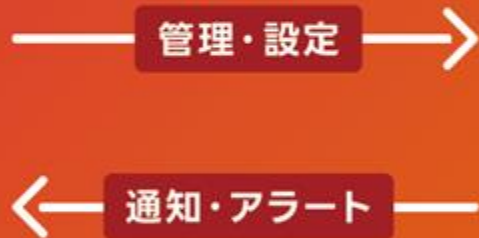
概要イメージ図

長期の安定運用を 低コストで実現



Operator

デバイス状態を可視化して
予防・対処



OTA
(OS・アプリアップデート)

失敗時も
安心!



リカバリー機能
(OSとアプリを二面化)

一括更新

グループ管理



データ収集



状態監視

■ 個体管理

- ▶ 個体コード & 証明書によるデバイス登録
- ▶ ラベル付け
- ▶ デバイスグループ化

■ 遠隔操作

- ▶ 任意コマンド実行
- ▶ ファームウェアアップデート
- ▶ 設定変更
- ▶ グループ一括実行
- ▶ スケジュール実行

■ お知らせ

- ▶ セキュリティアップデート
- ▶ システム障害通知

■ ユーザ管理

- ▶ ユーザーの追加/削除
- ▶ 権限設定

■ 稼働状況監視・データ収集

- ▶ 死活監視 (twinへの接続)
- ▶ アプリケーションコンテナ稼働状況
- ▶ CPU使用率・温度/メモリ使用率
- ▶ ストレージ寿命
- ▶ モバイル回線電波強度
- ▶ モバイル回線基地局 (ざっくり位置情報)
- ▶ アラートメール

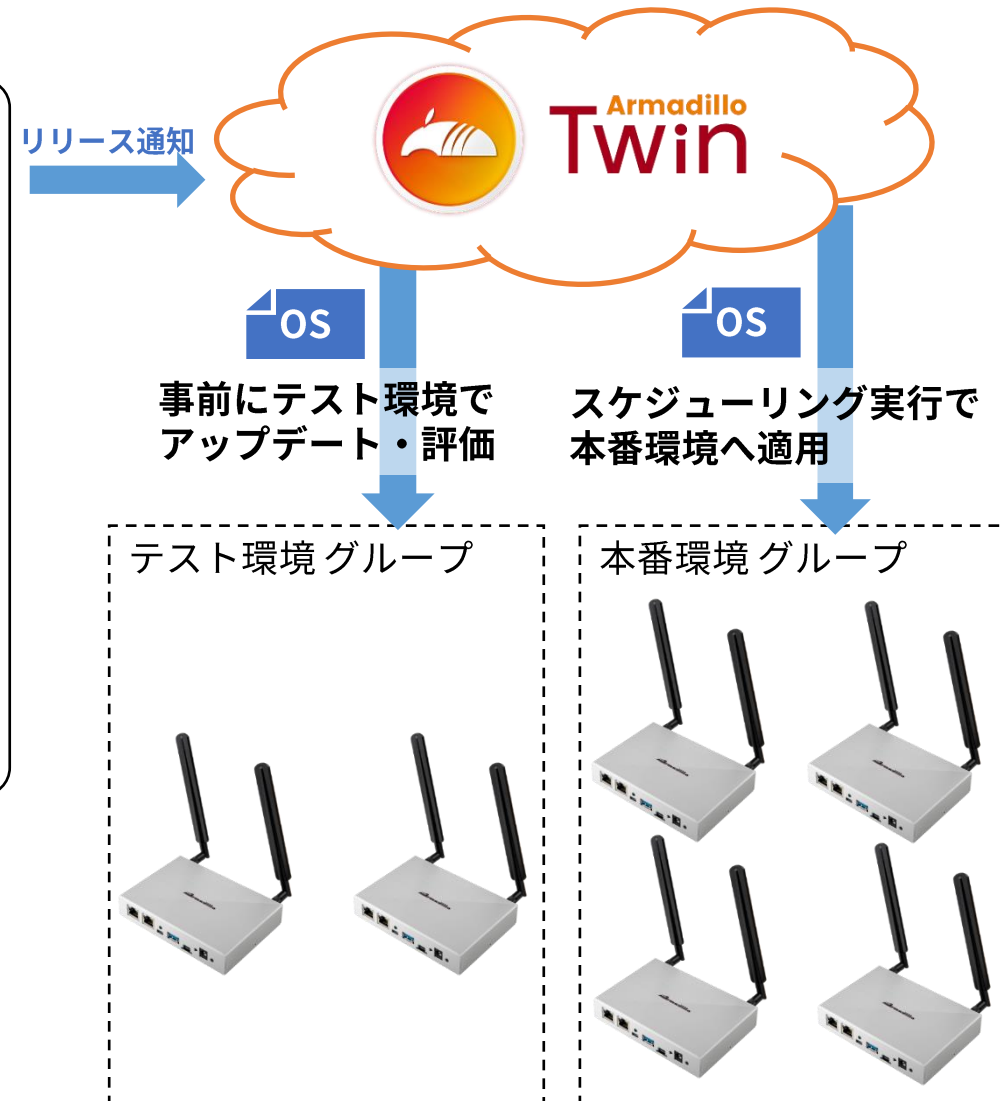
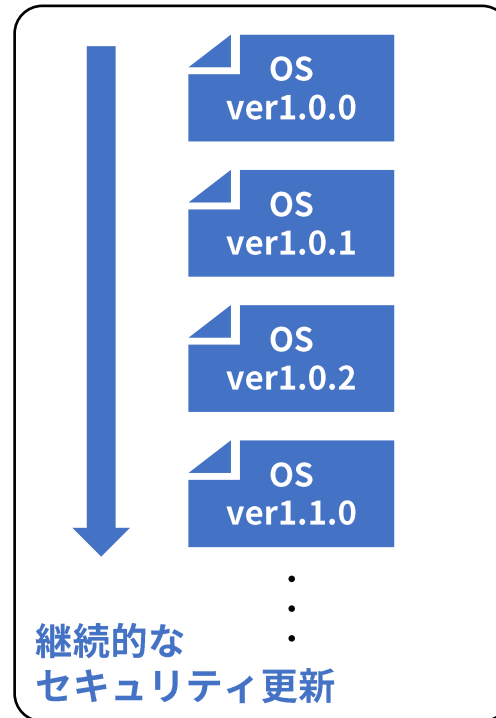
■ 請求書確認

- ▶ 3カ月分保存

機能紹介: OSのアップデート

- アットマークテクノが継続的にOSのセキュリティアップデートや機能追加を実施
- 稼働中の機器に最新のOSを適用し続けることでセキュリティリスクの低減につながる

Armadilloサイト



概要イメージ図

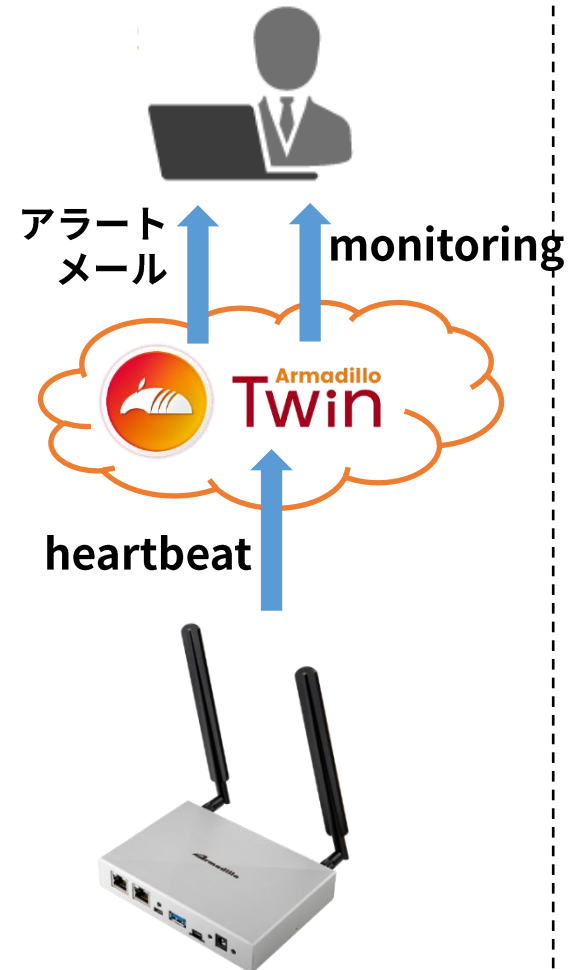
リモートから一括管理・保守業務を効率化

- IoTデバイスに必要と考えられる業務をすべてリモートから実施可能
- 現場に出向くことなく効率的な運用管理を実施

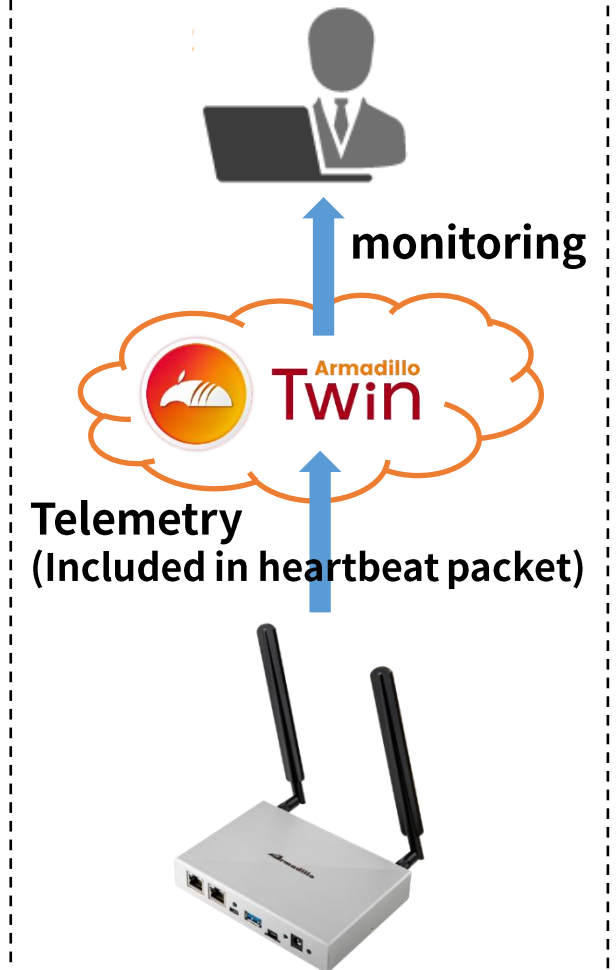
状態監視して異常発生をすぐに検知・予知

CPU使用率	メモリ使用率	ストレージ使用率
LTE電波受信強度	デバイス温度	ストレージ寿命
▼		
異常発生を検知・アラート メール通知 など		

状態監視・アラート



デバイスデータ収集



機能紹介: 任意コマンド実行

任意コマンド



command

即時実行
日時指定実行



command



- 遠隔からリモートでコマンド実行が可能
- SSH(Secure Shell)のための、固定IP契約等が不要
- 問題発生時の対処や予防措置を実施
- デバイスをグループ化した一括実行や日時を指定したスケジュール実行が可能

機能紹介: 個体管理



- SE050のデバイス証明書と個体コードによってデバイス登録を行います
- 個体コードをデバイスのユニークIDとして管理
- ラベル名つけ、グルーピングが可能

Armadillo Twin

テナント A 日本語 お問い合わせ nikko2_test

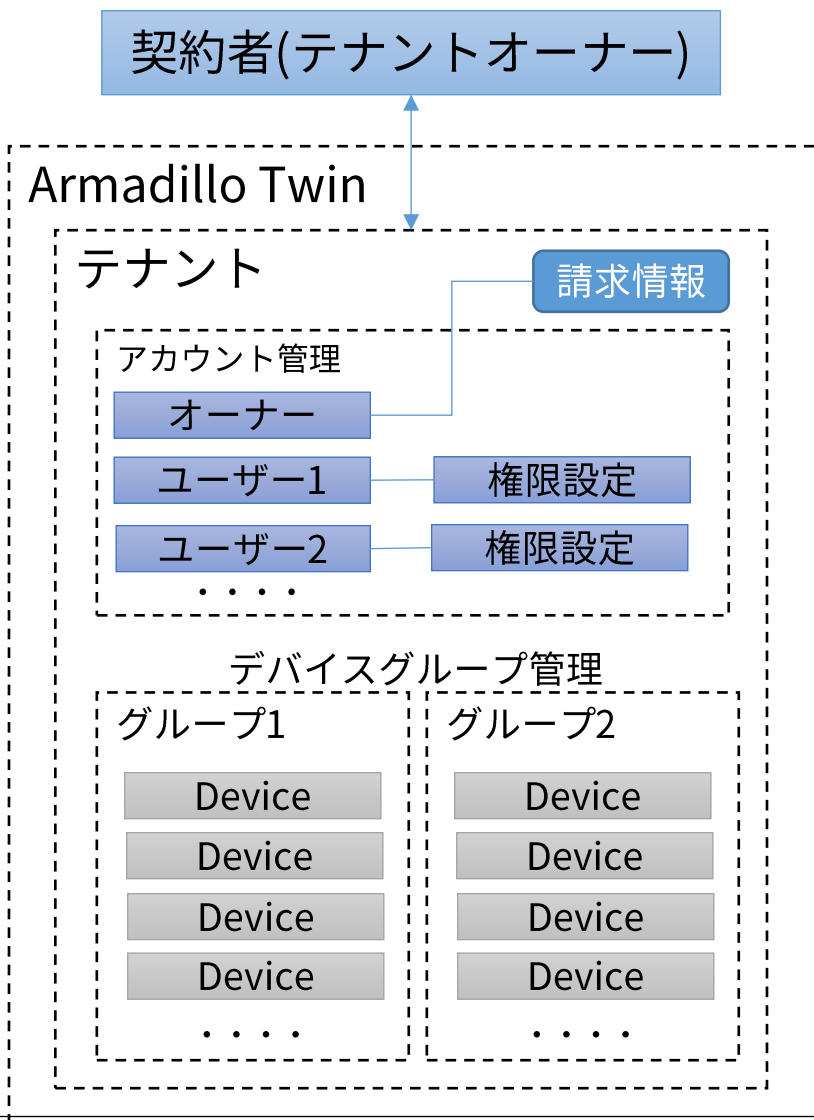
Home > デバイス管理

アクション 登録

個体コード プルダウンメニューで選択された列で検索します

全選択	個体コード	ラベル	ステータス
<input type="checkbox"/>	00C800010005	Demo	異常(オフライン)
<input type="checkbox"/>	00C900010001	TestDevice	異常(オフライン)

機能紹介: ユーザー管理・権限設定

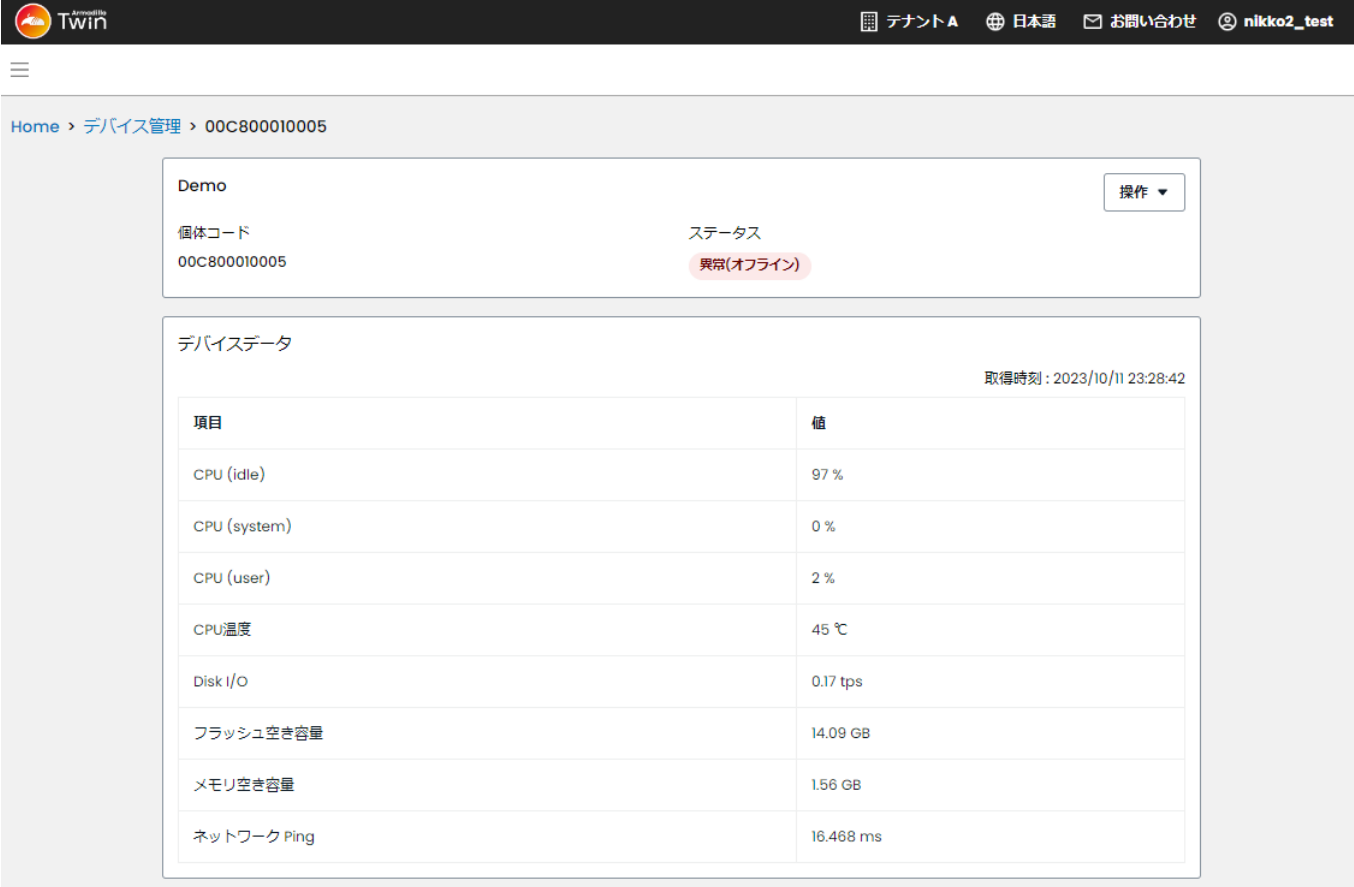


- 本契約、又はトライアルに申し込むとテナントが発行されます
- テナントオーナーはユーザーの追加や権限設定、デバイス追加、デバイスグループ作成などが可能です
- 店舗ごとのグループ作成や、運用ユーザを作成し、柔軟な管理を行なうことができます

収集する主な機器のメトリクスデータ

key	説明
kotai_code	個体コード情報
versions	各種ソフトウェアのバージョン /etc/sw-versionsの内容
timestanp(boot)	デバイス起動時刻(agintの起動時刻)
cpu_temp	CPU温度
cpu_idle	アイドル状態時間の割合
cpu_user	ユーザープロセスの実行時間の割合
cpu_system	カーネルプロセスの実行時間の割合
net_ping	ネットワークの応答速度
app_available_space	フラッシュメモリの使用可能容量 (APPパーティション)
memory_available_space	メモリの使用可能容量
disk_io	秒間 I/O リクエスト数
timestamp (heartbeat)	ハートビート取得時間
cellular_signal_quality	セルラー通信の電波強度
cellular_cellid	セルラー通知のキャリアと基地局情報 (ざっくりとした設置場所がわかる)
emmc_pre_eol_info	eMMCの寿命情報
container_status	Appコンテナの起動状況 podman ps -aの結果

メトリクスデータ画面イメージ



Home > デバイス管理 > 00C800010005

Demo 操作 ▾

個体コード
00C800010005

ステータス
異常(オフライン)

デバイスデータ 取得時刻: 2023/10/11 23:28:42

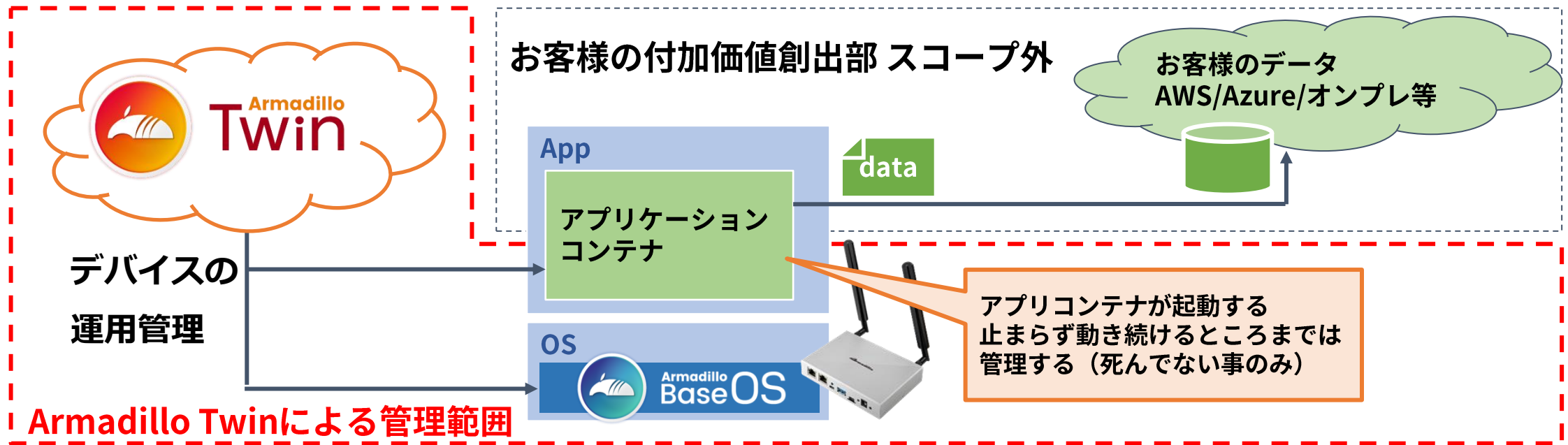
項目	値
CPU (idle)	97 %
CPU (system)	0 %
CPU (user)	2 %
CPU温度	45 °C
Disk I/O	0.17 tps
フラッシュ空き容量	14.09 GB
メモリ空き容量	1.56 GB
ネットワーク Ping	16.468 ms



Armadillo Twinの実施範囲

やらない：アプリ(コンテナ)の中身やデータ（付加価値創出部）の管理

お客様はこちらに注力



プラットフォームとして提供

やる：デバイスとArmadillo Base OS, アプリ(コンテナ)の”箱”の運用管理

無料トライアル(3カ月間)

■ 無料トライアル条件

- ▶ 無料期間: 3カ月
 - 本契約に移行可能、データ引継ぎ可能
 - トライアル期間満了前での移行も可能
 - 無料期間内に本契約をしなければテナント凍結
- ▶ 1テナントに登録可能なデバイス数: 5台

■ 申込方法

- ▶ Twinサービスページのフォームからサインアップ、
即日テナント発行・使用可能

プラン

契約形態 ^{※1 ※2}	月額料金 (税抜価格)		サービス内容		
	基本料金	デバイス利用料金	OTA機能	遠隔稼働監視 遠隔操作	個体管理 ユーザー管理
	1アカウント	1デバイス			
無償トライアル 提供中!					
ベーシックプラン	10,000円	300円 ^{※5}	○	○	○
ライトプラン		100円	—	○	○
スタンバイプラン		0円	—	—	○
リザーブプラン ^{※3}	0円 ^{※4}	0円	—	—	—

デバイス契約料金 (税抜価格)	
発注時(先契約)の場合	納品後(後契約)の場合
1デバイス	
発注時の契約が おすすめ!	
1,000円 ^{※6}	または 3,000円
デバイス登録代行 ^{※7} +100円 ^{※6} (オプション)	デバイス登録代行 申し込み不可
1,000円 ^{※6}	申し込み不可

[※1] 記載以外の料金の詳細についてはお問い合わせください。

[※2] デバイス1台毎に契約中のプランを変更することができます。リザーブプランに他のプランから変更することはできません。変更時は、プラン変更料金として1回につき300円（税抜）/台の費用が発生します。

[※3] 利用を開始するには、アカウントの新規作成とサービスを利用できるいずれかのプランに変更する必要があります。製品出荷後180日以内の初回プラン変更は無料です。180日を過ぎてからの変更は300円（税抜）/台の費用が発生します。




[※4] アカウントを新規作成した時点で、10,000円（税抜）の月額料金が発生します。

[※5] OTA通信量4GB/月の料金を含みます。アカウント内のデバイス全体で、定額部分の容量4GB×登録台数分の容量をシェアすることができます。4GBを超える通信は1GB毎に100円（税抜）が従量課金されます。従量課金は初期状態では無効に設定されているため、有効にする場合は別途設定が必要です。無償トライアル中のOTA通信量の上限は4GB/月で、それを超える通信はできません。

[※6] メーカー希望小売価格です。

[※7] メーカー出荷時に、あらかじめ指定のアカウントとデバイス情報を紐づける「デバイス登録代行オプション」を選択できます。デバイス登録はユーザー自身が行うことも可能です。

製品・OS・サービスの対応関係

製品	OS	サービス
<p><i>Armadillo-610</i></p> <p><i>Armadillo-640</i></p> <p><i>Armadillo-X2</i></p> <p><i>Armadillo-IoT</i> A6E</p> <p><i>Armadillo-IoT</i> G4</p>	 <p>Armadillo Base OS</p>	 <p>Armadillo Twin</p>
<p><i>Armadillo-X1</i></p> <p><i>Armadillo-IoT</i> G3</p> <p><i>Armadillo-IoT</i> G3L</p>	 <p>debian</p>	<p>node-eye</p>

- **Armadillo Twinについて**
 - ・ IoTデバイスに必須な運用管理をリモートで実行
- **Armadillo Twinの機能**
 - ・ ソフトウェアのリモートアップデート
 - ・ 任意コマンド実行
 - ・ アラートメール通知
 - ・ メトリクスデータ取得
 - ・ 個体管理

参考情報



■ Tips集 ※記事は随時追加

Armadillo Base OS開発手順 & Tips集

<https://armadillo.atmark-techno.com/blog/15349/18093>

Armadillo-IoT G4/X2

<https://armadillo.atmark-techno.com/tips/g4>

Armadillo-IoT A6E

<https://armadillo.atmark-techno.com/tips/a6e>

■ Armadilloフォーラム

<https://armadillo.atmark-techno.com/forum/armadillo>

■ How to

<https://armadillo.atmark-techno.com/howto>

■ ブログ

<https://armadillo.atmark-techno.com/blog>