Armadillo-IoT ゲートウェイ A6E
製品マニュアル

AG6221-C01D0
AG6221-C01Z

Version 1.3.0
2023/01/27

株式会社アットマークテクノ [https://www.atmark-techno.com]

Armadillo サイト [https://armadillo.atmark-techno.com]

Armadillo-IoT ゲートウェイ A6E 製品マニュアル

株式会社アットマークテクノ

Version 1.3.0
2023/01/27

目次

1. はじめに 16
1.1. 本書で扱うこと扱わないこと 16
1.1.1. 扱うこと 16
1.1.2. 扱わないこと 17
1.2. 本書で必要となる知識と想定する読者 17
1.3. ユーザー限定コンテンツ 17
1.4. 本書および関連ファイルのバージョンについて 17
1.5. 本書の構成 18
1.6. 表記について 18
1.6.1. フォント 18
1.6.2. コマンド入力例 18
1.6.3. アイコン 19
1.7. 謝辞 19
2. 注意事項 20
2.1. 安全に関する注意事項 20
2.2. 取扱い上の注意事項 21
2.3. 製品の保管について 22
2.4. ソフトウェア使用に関しての注意事項 23
2.5. 電波障害について 23
2.6. 無線モジュールの安全規制について 24
2.7. 保証について 25
2.8. 輸出について 25
2.9. 商標について 25
3. 製品概要 26
3.1. 製品の特長 26
3.1.1. Armadillo とは 26
3.1.2. Armadillo-IoT ゲートウェイ A6E とは 26
3.1.3. Armadillo Base OS とは 29
3.2. 製品ラインアップ 31
3.2.1. Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 開発セット 31
3.2.2. Armadillo-IoT ゲートウェイ A6E 量産用 32
3.3. 仕様 32
3.4. ブロック図 33
3.5. ストレージデバイスのパーティション構成 34
4. Armadillo の電源を入れる前に 35
4.1. 準備するもの 35
4.2. 開発/動作確認環境の構築 35
4.2.1. ATDE のセットアップ 36
4.2.2. 取り外し可能デバイスの使用 41
4.2.3. コマンドライン端末(GNOME 端末)の起動 42
4.2.4. シリアル通信ソフトウェア(minicom)の使用 43
4.3. インターフェースレイアウト 46
4.4. 接続方法 48
4.5. 起動デバイス設定スイッチについて 50
4.6. vi エディタの使用方法 50
4.6.1. vi の起動 51
4.6.2. 文字の入力 51
4.6.3. カーソルの移動 52
4.6.4. 文字の削除 52
4.6.5. 保存と終了 52

5. 起動と終了 53
5.1. 起動 53
5.2. ログイン 55
5.3. 終了方法 57
6. ユーザー登録 59
6.1. 購入製品登録 59
7. 動作確認方法 60
7.1. ネットワーク 60
7.1.1. 接続可能なネットワーク 60
7.1.2. ネットワークの設定方法 60
7.1.3. nmcli の基本的な使い方 60
7.1.4. 有線 LAN の接続を確認する 64
7.1.5. LTE (Cat.M1 モデルのみ) 65
7.1.6. 無線 LAN 73
7.1.7. BT 74
7.2. ストレージ 75
7.2.1. ストレージの使用方法 76
7.2.2. ストレージのパーティション変更とフォーマット 77
7.3. LED 78
7.3.1. LED を点灯/消灯する 79
7.3.2. トリガを使用する 79
7.4. ユーザースイッチ 80
7.4.1. イベントを確認する 80
7.5. RS485 81
7.6. 接点入力 81
7.6.1. 入力レベルの確認 82
7.7. 接点出力 82
7.7.1. 出力レベルの設定 82
7.8. RTC 82
7.8.1. RTC に時刻を設定する 82
8. 省電力・間欠動作機能 84
8.1. 動作モード・起床条件と状態遷移図 84
8.1.1. 動作モード・起床条件 84
8.2. シャットダウンモードへの遷移と起床 85
8.2.1. poweroff コマンド 85
8.2.2. aiot-alarm-poweroff コマンド 85
8.3. スリープモードへの遷移と起床 86
8.3.1. RTC アラーム割り込み以外での起床 86
8.3.2. RTC アラーム割り込みでの起床 87
8.3.3. 起床要因のクリア 87
8.4. スリープ(SMS 起床可能)モードへの遷移と起床 88
8.5. 状態遷移トリガにコンテナ終了通知を利用する 88
9. 開発の基本的な流れ 91
9.1. Armadillo への接続 91
9.1.1. シリアルコンソール 91
9.1.2. ssh 91
9.2. overlayfs の扱い 91
9.3. アプリケーション開発パターン 91
9.3.1. ゲートウェイコンテナとは 92
9.4. ゲートウェイアプリケーション開発の流れ 92
9.4.1. ゲートウェイアプリケーション開発 92
9.4.2. ゲートウェイコンテナを利用せず、一からコンテナを作成する 94
9.5. アプリケーションコンテナの運用 95

9.5.1. アプリケーションの自動起動 95
9.5.2. アプリケーションの送信 95
9.5.3. インストール確認：初期化 96
9.5.4. アプリケーションのアップデート 96
10. Howto 97
10.1. ゲートウェイアプリケーションを開発する 97
10.1.1. VSCode を用いた開発の流れ 97
10.1.2. ATDE 上でのセットアップ 97
10.1.3. Armadillo 上でのセットアップ 100
10.1.4. ゲートウェイアプリケーション開発 100
10.1.5. ゲートウェイアプリケーションの拡張例 100
10.1.6. リリース版のビルド 101
10.1.7. 製品への書き込み 101
10.2. ゲートウェイコンテナを動かす 102
10.2.1. ゲートウェイコンテナ利用の流れ 102
10.2.2. ゲートウェイコンテナ起動確認 102
10.2.3. 接続先の クラウド 環境を構築 (AWS) 102
10.2.4. 接続先の クラウド 環境を構築 (Azure) 111
10.2.5. 設定ファイルの編集 115
10.2.6. コンテナ起動・実行 124
10.2.7. クラウドからの操作 138
10.2.8. コンテナの終了 152
10.2.9. ログ内容確認 152
10.2.10. ゲートウェイコンテナの構成 153
10.3. ゲートウェイコンテナを拡張する 153
10.4. アプリケーションコンテナを作成、実行する 154
10.4.1. Podman - コンテナ仮想化ソフトウェア 154
10.4.2. コンテナを操作する 154
10.4.3. 近距離通信を行う 172
10.4.4. ネットワークを扱う 174
10.4.5. サーバを構築する 175
10.4.6. 異常検知 179
10.5. コンテナの運用 180
10.5.1. コンテナの自動起動 180
10.5.2. pod の作成 184
10.5.3. network の作成 185
10.5.4. コンテナからのコンテナ管理 185
10.5.5. コンテナの配布 185
10.6. Armadillo のソフトウェアをビルドする 189
10.6.1. ブートローダーをビルドする 189
10.6.2. Linux カーネルをビルドする 190
10.6.3. Alpine Linux ルートファイルシステムをビルドする 193
10.7. SD ブートの活用 196
10.7.1. ブートディスクの作成 196
10.7.2. SD ブートの実行 198
10.7.3. ゲートウェイコンテナのインストール 198
10.8. Armadillo のソフトウェアの初期化 199
10.8.1. インストールディスクの作成 199
10.8.2. インストールディスクを使用する 202
10.9. Armadillo のソフトウェアをアップデートする 202
10.9.1. SWU イメージとは 202
10.9.2. SWU イメージの作成 202
10.9.3. イメージのインストール 205

10.9.4. hawkBit サーバーから複数の Armadillo に配信する 207
10.9.5. mkswu の desc ファイル 221
10.9.6. swupdate_preserve_files について 226
10.9.7. SWU イメージの内容の確認 227
10.9.8. SWUpdate と暗号化について 227
10.10. Armadillo Base OS の操作 227
10.10.1. アップデート 227
10.10.2. overlayfs と persist_file について 227
10.10.3. ロールバック状態の確認 230
10.10.4. ボタンやキーを扱う 230
10.10.5. Armadillo Base OS 側の起動スクリプト 232
10.10.6. u-boot の環境変数の設定 232
10.10.7. Network Time Protocol (NTP, ネットワーク・タイム・プロトコル) 234
10.11. Device Tree をカスタマイズする 234
10.11.1. at-dtweb の起動 235
10.11.2. Device Tree をカスタマイズ 236
10.11.3. DTS overlays によるカスタマイズ 242
10.12. eMMC のデータリテンション 243
10.13. SMS を利用する (Cat.M1 モデルのみ) 243
10.13.1. 初期設定 243
10.13.2. SMS を送信する 244
10.13.3. SMS を受信する 244
10.13.4. SMS 一覧を表示する 244
10.13.5. SMS の内容を表示する 245
10.13.6. SMS を削除する 245
10.13.7. SMS を他のストレージに移動する 245
10.14. 動作中の Armadillo の温度を測定する 246
10.14.1. 温度測定の重要性 246
10.14.2. atmark-thermal-profiler をインストールする 246
10.14.3. atmark-thermal-profiler を実行・停止する 246
10.14.4. atmark-thermal-profiler が出力するログファイルを確認する 247
10.14.5. 温度測定結果の分析 248
11. 動作ログ 250
11.1. 動作ログについて 250
11.2. 動作ログを取り出す 250
11.3. ログファイルのフォーマット 250
11.4. ログ用パーティションについて 250
12. Linux カーネル仕様 251
12.1. デフォルトコンフィギュレーション 251
12.2. ブートパラメータ 251
12.3. Linux ドライバ一覧 252
12.3.1. Armadillo-IoT ゲートウェイ A6E 252
12.3.2. UART 252
12.3.3. LTE 253
12.3.4. WLAN 253
12.3.5. BT 254
12.3.6. Ethernet 255
12.3.7. SD ホスト 255
12.3.8. USB ホスト 256
12.3.9. リアルタイムクロック 257
12.3.10. LED 257
12.3.11. ユーザースイッチとイベント信号 258
12.3.12. I2C 258

12.3.13. GPIO 259
12.3.14. パワーマネジメント 260
13. ソフトウェア仕様 262
13.1. SWUpdate 262
13.1.1. SWUpdate とは 262
13.1.2. swu パッケージ 262
13.1.3. A/B アップデート(アップデートの 2 面化) 262
13.1.4. ロールバック（リカバリー） 263
13.2. hawkBit 263
13.2.1. hawkBit とは 263
13.2.2. データ構造 264
14. ハードウェア仕様 265
14.1. 電気的仕様 265
14.1.1. 絶対最大定格 265
14.1.2. 推奨動作条件 265
14.1.3. 入出力仕様 265
14.1.4. 電源回路の構成 266
14.2. インターフェース仕様 267
14.3. CON1(SD インターフェース) 269
14.3.1. microSD カードの挿抜方法 269
14.4. CON3(nanoSIM インターフェース) 272
14.5. CON4(LAN インターフェース) 272
14.6. CON5(電源入力インターフェース) 273
14.7. CON6(入出力インターフェース) 273
14.7.1. 接点入力 274
14.7.2. 接点出力 274
14.7.3. RS485 275
14.8. CON7(USB コンソールインターフェース) 275
14.9. CON8(拡張インターフェース) 276
14.10. CON9(USB インターフェース) 277
14.11. CON10(RTC バックアップインターフェース) 277
14.12. ANT1(LTE アンテナインターフェース 1) 278
14.13. SYS、APP、WWAN(LED) 278
14.14. SW1(ユーザースイッチ) 279
14.15. SW2(起動デバイス設定スイッチ) 279
14.16. SW3(RS485 終端抵抗設定スイッチ) 279
14.17. 形状図 281
14.17.1. 筐体形状図 281
14.17.2. 基板形状図 282
14.17.3. アンテナ形状図 285
14.18. 組み立てと分解 285
14.18.1. ケースの組み立て手順 288
14.18.2. ケースの分解 288
14.19. 設計情報 290
14.19.1. 信頼性試験データについて 290
14.19.2. 放射ノイズ 290
14.19.3. ESD/雷サージ 290
14.19.4. 拡張ボードの設計 291
14.20. オプション品 291

図目次

2.1. LTE モジュール:EMS31-J 認証マーク 24
2.2. WLAN+BT コンボモジュール:Sterling LWB5+ 認証マーク 24
3.1. Armadillo-IoT ゲートウェイ A6E とは 27
3.2. 間欠動作の例 28
3.3. 様々なデバイスとの接続例 28
3.4. Armadillo Base OS とは 29
3.5. コンテナによるアプリケーションの運用 30
3.6. ロールバックの仕組み 30
3.7. Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 31
3.8. Armadillo-IoT ゲートウェイ A6E ブロック図 33
4.1. GNOME 端末の起動 42
4.2. GNOME 端末のウィンドウ 43
4.3. minicom の設定の起動 43
4.4. minicom の設定 43
4.5. minicom のシリアルポートの設定 44
4.6. 例. シリアル通信用 USB ケーブル(A-microB)接続時のログ 44
4.7. minicom のシリアルポートのパラメータの設定 45
4.8. minicom シリアルポートの設定値 45
4.9. minicom 起動方法 46
4.10. minicom 終了確認 46
4.11. インターフェースレイアウト 47
4.12. Armadillo-IoT ゲートウェイ A6E の接続例 49
4.13. 起動デバイス設定スイッチの操作 50
4.14. vi の起動 51
4.15. 入力モードに移行するコマンドの説明 51
4.16. 文字を削除するコマンドの説明 52
7.1. nmcli のコマンド書式 60
7.2. コネクションの一覧表示 61
7.3. コネクションの有効化 61
7.4. コネクションの無効化 61
7.5. コネクションの作成 61
7.6. コネクションファイルの永続化 61
7.7. コネクションの削除 62
7.8. コネクションファイル削除時の永続化 62
7.9. 固定 IP アドレス設定 63
7.10. DNS サーバーの指定 63
7.11. DNS サーバーの指定 63
7.12. コネクションの修正の反映 63
7.13. デバイスの一覧表示 63
7.14. デバイスの接続 64
7.15. デバイスの切断 64
7.16. 有線 LAN の PING 確認 64
7.17. LTE のコネクションの作成 67
7.18. LTE のコネクションの設定の永続化 67
7.19. MCC/MNC を指定した LTE コネクションの作成 67
7.20. PAP 認証を有効にした LTE コネクションの作成 68
7.21. LTE のコネクション確立 68
7.22. LTE の PING 導通確認 68
7.23. LTE コネクションを切断する 69
7.24. nmcli connection modify コマンドで LTE のパスワードを設定する 69

7.25. LTE 再接続サービスの設定値を永続化する 70
7.26. LTE 再接続サービスの状態を確認する 70
7.27. LTE 再接続サービスを停止する 70
7.28. LTE 再接続サービスを開始する 71
7.29. LTE 再接続サービスを無効にする 71
7.30. LTE 再接続サービスを有効にする 71
7.31. 認識されているモデムの一覧を取得する 72
7.32. モデムの情報を取得する 72
7.33. SIM の情報を取得する 72
7.34. 回線情報を取得する 73
7.35. 無線 LAN アクセスポイントに接続する 73
7.36. 無線 LAN のコネクションが作成された状態 74
7.37. 無線 LAN の PING 確認 74
7.38. bluez のインストール 74
7.39. bluetooth.service の有効化 75
7.40. bluetoothctl スキャン開始 75
7.41. bluetoothctl スキャン停止 75
7.42. bluetoothctl 終了 75
7.43. mount コマンド書式 76
7.44. ストレージのマウント 77
7.45. ストレージのアンマウント 77
7.46. fdisk コマンドによるパーティション変更 77
7.47. EXT4 ファイルシステムの構築 78
7.48. LED を点灯させる 79
7.49. LED を消灯させる 79
7.50. LED の状態を表示する 79
7.51. 対応している LED トリガを表示 80
7.52. LED のトリガに timer を指定する 80
7.53. ユーザースイッチ: イベントの確認 81
7.54. 入力レベルの確認 82
7.55. 出力レベルを "0" に設定する場合 82
7.56. システムクロックを設定 83
7.57. ハードウェアクロックを設定 83
8.1. 状態遷移図 84
8.2. aiot-alarm-poweroff コマンド書式 85
8.3. aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込み以外での起床のとき） 86
8.4. aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込みでの起床の場合） 87
10.1. ゲートウェイアプリケーション開発の流れ 97
10.2. ソフトウェアをアップデートする 98
10.3. ゲートウェイアプリケーションを展開する 98
10.4. 初期設定を行う 98
10.5. VSCode で初期設定用の swu を作成する 98
10.6. VSCode のターミナル 99
10.7. SSH 用の鍵を生成する 99
10.8. ssh_config を編集する 99
10.9. ゲートウェイアプリケーションを実行する 100
10.10. アプリケーションを停止する 100
10.11. ログファイルの Count_value の出力例 101
10.12. リリース版をビルドする 101
10.13. Armadillo-IoT ゲートウェイ A6E クラウド設定データをダウンロードする 112
10.14. コンフィグファイルを編集する 113
10.15. コンフィグファイル設定例 113
10.16. Azure IoT Hub と DPS の設定を実行する 114

10.17. /var/app/rollback/volumes/gw_container/config/sensing_mgr.conf のフォーマット 115
10.18. /var/app/rollback/volumes/gw_container/config/cloud_agent.conf のフォーマット 120
10.19. 接点入力制御シャドウ設定例 142
10.20. 接点入力制御デバイスツイン設定例 142
10.21. 接点出力制御シャドウ設定例 143
10.22. 接点出力制御デバイスツイン設定例 144
10.23. RS485 レジスタ読み出しシャドウ設定例 145
10.24. RS485 レジスタ読み出しデバイスツイン設定例 146
10.25. 接点出力制御シャドウ設定例 148
10.26. RS485 レジスタ書き込みシャドウ設定例 150
10.27. LED 点灯制御シャドウ設定例 152
10.28. ログファイルのフォーマット 153
10.29. コンテナを作成する実行例 154
10.30. コンテナ一覧の表示実行例 155
10.31. podman ps --help の実行例 156
10.32. コンテナを起動する実行例 156
10.33. コンテナを起動する実行例(a オプション付与) 156
10.34. podman start --help 実行例 157
10.35. コンテナを停止する実行例 157
10.36. podman stop --help 実行例 157
10.37. my_container を保存する例 157
10.38. chattr によって copy-on-write を無効化する例 158
10.39. podman build の実行例 159
10.40. podman build でのアップデートの実行例 159
10.41. コンテナを削除する実行例 160
10.42. $ podman rm --help 実行例 160
10.43. イメージを削除する実行例 160
10.44. podman rmi --help 実行例 161
10.45. Read-Only のイメージを削除する実行例 161
10.46. コンテナ内部のシェルを起動する実行例 161
10.47. コンテナ内部のシェルから抜ける実行例 162
10.48. podman exec --help 実行例 162
10.49. コンテナを作成する実行例 162
10.50. コンテナの IP アドレスを確認する実行例 163
10.51. ping コマンドによるコンテナ間の疎通確認実行例 163
10.52. GPIO を扱うためのコンテナ作成例 164
10.53. コンテナ内からコマンドで GPIO を操作する例 164
10.54. gpiodetect コマンドの実行 164
10.55. gpioinfo コマンドの実行 164
10.56. I2C を扱うためのコンテナ作成例 165
10.57. i2cdetect コマンドによる確認例 165
10.58. シリアルインターフェースを扱うためのコンテナ作成例 166
10.59. setserial コマンドによるシリアルインターフェイス設定の確認例 166
10.60. USB シリアルデバイスを扱うためのコンテナ作成例 166
10.61. setserial コマンドによる USB シリアルデバイス設定の確認例 167
10.62. USB カメラを扱うためのコンテナ作成例 167
10.63. USB メモリをホスト OS 側でマウントする例 167
10.64. ホスト OS 側でマウント済みの USB メモリを扱うためのコンテナ作成例 168
10.65. USB メモリに保存されているデータの確認例 168
10.66. USB メモリをマウントするためのコンテナ作成例 168
10.67. コンテナ内から USB メモリをマウントする例 168
10.68. USB デバイスのホットプラグに対応する例 169
10.69. RTC を扱うためのコンテナ作成例 169

10.70. hwclock コマンドによる RTC の時刻表示と設定例 169
10.71. 音声出力を行うためのコンテナ作成例 170
10.72. alsa-utils による音声出力を行う例 170
10.73. ユーザースイッチのイベントを取得するためのコンテナ作成例 170
10.74. evtest コマンドによる確認例 171
10.75. LED を扱うためのコンテナ作成例 171
10.76. LED の点灯/消灯の実行例 172
10.77. Bluetooth デバイスを扱うためのコンテナ作成例 172
10.78. Bluetooth を起動する実行例 172
10.79. bluetoothctl コマンドによるスキャンとペアリングの例 172
10.80. Wi-SUN デバイスを扱うためのコンテナ作成例 173
10.81. EnOcean デバイスを扱うためのコンテナ作成例 174
10.82. コンテナの IP アドレス確認例 174
10.83. ip コマンドを用いたコンテナの IP アドレス確認例 174
10.84. ユーザ定義のネットワーク作成例 175
10.85. IP アドレス固定のコンテナ作成例 175
10.86. コンテナの IP アドレス確認例 175
10.87. コンテナに Apache をインストールする例 176
10.88. コンテナに lighttpd をインストールする例 176
10.89. コンテナに vsftpd をインストールする例 176
10.90. ユーザを追加する例 177
10.91. 設定ファイルの編集例 177
10.92. vsftpd の起動例 177
10.93. コンテナに samba をインストールする例 177
10.94. ユーザを追加する例 178
10.95. samba の起動例 178
10.96. コンテナに sqlite をインストールする例 178
10.97. sqlite の実行例 179
10.98. ソフトフェアウォッチドッグタイマーを使うためのコンテナ作成例 179
10.99. コンテナ内からソフトウェアウォッチドッグタイマーを起動する実行例 179
10.100. ソフトウェアウォッチドッグタイマーをリセットする実行例 179
10.101. ソフトウェアウォッチドッグタイマーを停止する実行例 180
10.102. コンテナを自動起動するための設定例 180
10.103. ボリュームを shared でサブマウントを共有する例 182
10.104. pod を使うコンテナを自動起動するための設定例 184
10.105. network を使うコンテナを自動起動するための設定例 185
10.106. abos-ctrl podman-rw の実行例 187
10.107. abos-ctrl podman-storage のイメージコピー例 187
10.108. ブートローダーのソースコードをダウンロードする 189
10.109. デフォルトコンフィギュレーションの適用 189
10.110. ブートローダーのビルド 190
10.111. ブートローダーの結果確認 190
10.112. Linux カーネルソースコードの展開 191
10.113. Linux カーネルデフォルトコンフィギュレーションの適用 191
10.114. Linux カーネルコンフィギュレーションの変更 191
10.115. Linux カーネルコンフィギュレーション設定画面 191
10.116. Linux カーネルコンフィギュレーションの変更 192
10.117. Linux カーネルビルドしたファイルの確認 193
10.118. 自動マウントされた microSD カードのアンマウント 196
10.119. ゲートウェイコンテナ SWU イメージアーカイブをダウンロードし、 SWU イメージを作成する 199
10.120. hawkBit コンテナの TLS なしの場合（テスト用）の実行例 208
10.121. hawkBit コンテナの TLS ありの場合の実行例 209

10.122. persist_file のヘルプ 228
10.123. persist_file 保存・削除手順例 228
10.124. persist_file ソフトウェアアップデート後も変更を維持する手順例 229
10.125. persist_file 変更ファイルの一覧表示例 229
10.126. persist_file でのパッケージインストール手順例 229
10.127. /var/at-log/atlog の内容の例 230
10.128. buttond で SW1 を扱う 231
10.129. local サービスの実行例 232
10.130. uboot_env.d のコンフィグファイルの例 233
10.131. chronyd のコンフィグの変更例 234
10.132. at-dtweb の起動開始 235
10.133. ボード選択画面 235
10.134. Linux カーネルディレクトリ選択画面 236
10.135. at-dtweb 起動画面 236
10.136. UART1(RXD/TXD) のドラッグ 237
10.137. CON8 8/9 ピンへのドロップ 237
10.138. 信号名の確認 238
10.139. プロパティの設定 239
10.140. プロパティの保存 239
10.141. 全ての機能の削除 240
10.142. I2C4(SCL/SDA) の削除 240
10.143. dtbo/desc ファイルの生成 241
10.144. dtbo/desc の生成完了 241
10.145. /boot/overlays.txt の変更例 242
10.146. 言語設定 243
10.147. SMS の作成 244
10.148. SMS 番号の確認 244
10.149. SMS の送信 244
10.150. SMS の一覧表示 244
10.151. SMS の内容を表示 245
10.152. SMS の削除 245
10.153. SIM カードのストレージに SMS を移動 245
10.154. LTE モジュールの内蔵ストレージに SMS を移動 246
10.155. atmark-thermal-profiler をインストールする 246
10.156. atmark-thermal-profiler を実行する 247
10.157. atmark-thermal-profiler を停止する 247
10.158. ログファイルの内容例 247
10.159. サーマルシャットダウン温度の確認(Armadillo-IoT ゲートウェイ G4 を例に) 248
10.160. Armadillo-IoT ゲートウェイ G4 で取得した温度のグラフ 249
11.1. 動作ログのフォーマット 250
13.1. hawkBit が扱うソフトウェアのデータ構造 264
14.1. 電源回路の構成 267
14.2. Armadillo-IoT ゲートウェイ A6E のインターフェース 表面 268
14.3. Armadillo-IoT ゲートウェイ A6E のインターフェース 裏面 268
14.4. カバーのロックを解除する 270
14.5. カバーを開ける 270
14.6. microSD カードの挿抜 270
14.7. カードマークの確認 271
14.8. カバーを閉める 271
14.9. カバーをロックする 271
14.10. CON4 LAN LED 272
14.11. AC アダプタの極性マーク 273
14.12. CON6 接点入力周辺回路 274

14.13. CON6 接点出力周辺回路 275
14.14. CON6 RS485 トランシーバ周辺回路 275
14.15. ANT1 接続可能なアンテナコネクタ形状 278
14.16. スイッチの状態と起動デバイス 279
14.17. スイッチの状態と終端抵抗の ON/OFF 280
14.18. 筐体形状 281
14.19. 基板形状およびび固定穴寸法 282
14.20. コネクタ、スイッチ、LED 位置 283
14.21. 部品高さ 284
14.22. アンテナ形状 285
14.23. ケースモデル展開図 286
14.24. フック取り付け 1 287
14.25. フック取り付け 2 288
14.26. フックのツメ 289
14.27. ケースボトムのツメ 289
14.28. カバーのツメ 290
14.29. 拡張ボード例 291

表目次

1.1. 使用しているフォント 18
1.2. 表示プロンプトと実行環境の関係 19
1.3. コマンド入力例での省略表記 19
2.1. 推奨温湿度環境について 23
2.2. LTE モジュール:EMS31-J 適合証明情報 24
2.3. WLAN+BT コンボモジュール:Sterling LWB5+ 適合証明情報 24
3.1. Armadillo-IoT ゲートウェイ A6E ラインアップ 31
3.2. 仕様 32
3.3. eMMC メモリマップ 34
3.4. eMMC ブートパーティション構成 34
3.5. eMMC GPP 構成 34
4.1. ユーザー名とパスワード 39
4.2. 動作確認に使用する取り外し可能デバイス 41
4.3. シリアル通信設定 43
4.4. インターフェース内容 47
4.5. 入力モードに移行するコマンド 51
4.6. カーソルの移動コマンド 52
4.7. 文字の削除コマンド 52
4.8. 保存・終了コマンド 52
7.1. ネットワークとネットワークデバイス 60
7.2. 固定 IP アドレス設定例 62
7.3. ems31-boot.conf の設定内容 66
7.4. psm の tau と act-time に設定可能な値 66
7.5. edrx の pcl と ptw に設定可能な値 66
7.6. APN 情報設定例 67
7.7. 通信モジュールのネットワークデバイス 67
7.8. 再接続サービス設定パラメーター 70
7.9. ストレージデバイス 75
7.10. eMMC の GPP の用途 76
7.11. LED クラスディレクトリと LED の対応 79
7.12. LED トリガの種類 79
7.13. インプットデバイスファイルとイベントコード 80
7.14. RS485 に対応する CON6 ピン番号 81
7.15. 接点入力に対応する CON6 ピン番号 81
7.16. 接点出力に対応する CON6 ピン番号 82
7.17. 時刻フォーマットのフィールド 83
8.1. aiot-modem-control TRIGGER 一覧 86
8.2. 設定パラメーター 89
8.3. 遷移先の動作モード 89
8.4. 起床条件 89
9.1. 利用できるインターフェース・機能 92
9.2. 利用できるクラウドベンダー・サービス 92
10.1. [DEFAULT] 設定可能パラメータ 117
10.2. [LOG] 設定可能パラメータ 117
10.3. [CPU_temp] 設定可能パラメータ 118
10.4. [DI1,DI2] 設定可能パラメータ 118
10.5. [DO1,DI2] 設定可能パラメータ 119
10.6. [RS485_Data1, RS485_Data2, RS485_Data3, RS485_Data4] 設定可能パラメータ 120
10.7. [CLOUD] 設定可能パラメータ 121
10.8. [CLOUD] 設定可能パラメータ 121

10.9. [CLOUD] 設定可能パラメータ 122
10.10. [CLOUD] 設定可能パラメータ 123
10.11. デバイス情報データー覧 124
10.12. CPU 温度データー覧 124
10.13. 接点入力データー覧 124
10.14. RS485 データー覧 124
10.15. ユーザースイッチ関連データー覧 125
10.16. Azure Stream Analytics ジョブ設定値 130
10.17. Azure Stream Analytics ジョブ入力設定値 131
10.18. 接点入力設定値 141
10.19. 接点出力設定値 143
10.20. RS485 レジスタ読み出し設定値 144
10.21. 接点出力制御設定値 148
10.22. RS485 レジスタ書き込みシャドウ設定値 149
10.23. LED 点灯制御設定値 151
10.24. microSD カードのパーティション構成 197
10.25. thermal_profile.csv の各列の説明 247
12.1. Linux カーネルの主要デフォルトブートパラメータ 251
12.2. キーコード 258
12.3. I2C デバイス 258
12.4. 対応するパワーマネジメント状態 260
14.1. 絶対最大定格 265
14.2. 推奨動作条件 265
14.3. 電源入力仕様 265
14.4. 電源出力仕様 265
14.5. 入出力インターフェース(CON6)の入出力仕様 266
14.6. 拡張インターフェース(CON8)の入出力仕様(OVDD = VCC_3.3V) 266
14.7. Armadillo-IoT ゲートウェイ A6E インターフェース一覧 268
14.8. CON1 信号配列 269
14.9. CON3 信号配列 272
14.10. CON4 信号配列 272
14.11. CON4 LAN LED の動作 272
14.12. CON6 信号配列 273
14.13. CON6 接続可能な電線 274
14.14. CON7 信号配列 275
14.15. CON8 搭載コネクタと対向コネクタ例 276
14.16. CON8 信号配列 276
14.17. CON9 信号配列 277
14.18. CON10 信号配列 278
14.19. LED 信号配列 278
14.20. LED 状態と製品状態の対応について 279
14.21. SW1 信号配列 279
14.22. ケースモデル展開図パーツ一覧 287

1. はじめに

このたびは Armadillo-IoT ゲートウェイ A6E をご利用いただき、ありがとうございます。

Armadillo-IoT ゲートウェイ A6E は、各種センサーとネットワークとの接続を中継する IoT 向けゲートウェイの開発プラットフォームです。標準インターフェースとして RS485、接点入出力 2ch/2ch、Ethernet、USB を搭載、様々なセンサ・デバイスを接続することができ、ソフトウェアをカスタマイズして、オリジナルのゲートウェイを素早く、簡単に開発することができます。

Armadillo-IoT ゲートウェイ A6E は、Armadillo-IoT ゲートウェイシリーズの中でも、省電力や間欠動作機能に特化した IoT ゲートウェイです。自立型のシステムを構築する際には、ソーラーパネルや蓄電池をより小さなものにでき、システム全体のコストを大幅に低減することができます。

ゲートウェイを間欠動作させることで、さらに細かな節電が可能です。スリープ時はほとんど電力を消費せず、その状態からすぐに高速起動することができます。必要なときだけゲートウェイを起動しクラウドと通信、データ送信後は再スリープといった運用を実現します。

Armadillo-IoT ゲートウェイ A6E は、用途に合わせて複数のモデルを用意しています。超低消費電力でクラウドと通信できるセルラー LPWA LTE-M(Cat.M1)モジュールを搭載した「Cat.M1 モデル」を標準として、より高速な通信を必要とする用途向けに LTE(Cat.1)を採用した「Cat.1 モデル」、既設の LAN/無線 LAN を利用する安価な「LAN モデル」の 3 モデルが選択可能です。

Armadillo-IoT ゲートウェイ A6E には Linux ベースのディストリビューションとして専用設計の Armadillo Base OS を搭載しています。Armadillo Base OS はユーザーアプリケーションをコンテナとして管理する機能、Armadillo Base OS 自体とコンテナの両方を安全にリモートアップデートする機能、ネットワークや HW セキュリティに関する機能を集約したコンパクトな Armadillo 専用 OS です。

Armadillo Base OS とユーザーアプリケーションを含むコンテナはどちらも、Armadillo Base OS のリモートアップデート機能で安全にアップデートすることができます。Armadillo Base OS はアップデートの状態を 2 面化しているので電源やネットワークの遮断でよって中断してもアップデート前の状態に復旧します。

ユーザーアプリケーションをコンテナとして管理できる機能を利用し、各種クラウド IoT サービス(Azure IoT や AWS IoT Core)に対応したゲートウェイコンテナを用意しました。これまでの Armadillo-IoT ゲートウェイシリーズでは、ユーザー自身が開発するアプリケーションソフトウェアで、センサーからのデータ取得、クラウドへのアップロード等のゲートウェイとしての機能の他、通信障害時の対応、セキュリティ対応、間欠動作時の挙動などの難しい課題を自ら解決する必要がありました。あらかじめ用意されたゲートウェイコンテナを活用することで、これらの課題に対処することができ、短期間に IoT システムを構築可能です。

以降、本書では他の Armadillo ブランド製品にも共通する記述については、製品名を Armadillo と表記します。

1.1. 本書で扱うこと扱わないこと

1.1.1. 扱うこと

本書では、Armadillo-IoT ゲートウェイ A6E の使い方、製品仕様(ソフトウェアおよびハードウェア)、オリジナルの製品を開発するために必要となる情報、その他注意事項について記載しています。Linux あるいは組み込み機器に不慣れな方でも読み進められるよう、コマンドの実行例なども記載しています。

また、本書では、アットマークテクノが運営する Armadillo サイトをはじめ、開発に有用な情報を得る方法についても、随時説明しています。

1.1.2. 扱わないこと

本書では、一般的な Linux のプログラミング、デバッグ方法やツールの扱い方、各種モジュールの詳細仕様など、一般的な情報や、他に詳しい情報があるものは扱いません。また、(Armadillo-IoT ゲートウェイ A6E を使用した)最終製品あるいはサービスに固有な情報や知識も含まれていません。

1.2. 本書で必要となる知識と想定する読者

本書は、読者として Armadillo-IoT ゲートウェイ A6E を使ってオリジナルのゲートウェイ機器を開発するエンジニアを想定して書かれています。また、「Armadillo-IoT ゲートウェイ A6E を使うと、どのようなことが実現可能なのか」を知りたいと考えている設計者・企画者も対象としています。Armadillo-IoT ゲートウェイ A6E は組込みプラットフォームとして実績のある Armadillo をベースとしているため、標準で有効になっている機能以外にも様々な機能を実現することができます。

ソフトウェアエンジニア	端末からのコマンドの実行方法など、基本的な Linux の扱い方を知っているエンジニアを対象読者として想定しています。プログラミング言語として C/C++を扱えることは必ずしも必要ではありませんが、基礎的な知識がある方が理解しやすい部分もあります。
ハードウェアエンジニア	電子工学の基礎知識を有したエンジニアを対象読者として想定しています。回路図や部品表を読み、理解できる必要があります。

1.3. ユーザー限定コンテンツ

アットマークテクノ Armadillo サイトで購入製品登録を行うと、製品をご購入いただいたユーザーに限定して公開している限定コンテンツにアクセスできるようになります。主な限定コンテンツには、下記のものがあります。

- 各種信頼性試験データ・納入仕様書等製造関連情報

限定コンテンツを取得するには、「6. ユーザー登録」を参照してください。

1.4. 本書および関連ファイルのバージョンについて

本書を含めた関連マニュアル、ソースファイルやイメージファイルなどの関連ファイルは最新版を使用することをおすすめいたします。本書を読み始める前に、Armadillo サイトで最新版の情報をご確認ください。

Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ドキュメントダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6e/resources/documents

Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6e/resources/software

1.5. 本書の構成

本書には、Armadillo-IoT ゲートウェイ A6E をベースに、オリジナルの製品を開発するために必要となる情報を記載しています。また、取扱いに注意が必要な事柄についても説明しています。

- **はじめにお読みください。**

  「1. はじめに」、「2. 注意事項」

- **Armadillo-IoT ゲートウェイ A6E の仕様を紹介します。**

  「3. 製品概要」

- **工場出荷状態のソフトウェアの使い方を紹介します。**

  「4. Armadillo の電源を入れる前に」、「5. 起動と終了」

- **ご購入ユーザーに限定して公開している情報の紹介やユーザー登録について紹介します。**

  「6. ユーザー登録」

- **ハードウェアの動作を確認する方法を紹介します。**

  「7. 動作確認方法」、「8. 省電力・間欠動作機能」

- **システム開発に必要な情報を紹介します。**

  「9. 開発の基本的な流れ」、「10. Howto」、「12. Linux カーネル仕様」

- **工場出荷状態のソフトウェア仕様について紹介します。**

  「11. 動作ログ」、「13. ソフトウェア仕様」

- **拡張基板の開発や、ハードウェアをカスタマイズする場合に必要な情報を紹介します。**

  「14. ハードウェア仕様」

1.6. 表記について

1.6.1. フォント

本書では以下のような意味でフォントを使いわけています。

表 1.1 使用しているフォント

フォント例	説明
本文中のフォント	本文
`[PC ~]$ ls`	プロンプトとユーザ入力文字列
`text`	編集する文字列や出力される文字列。またはコメント

1.6.2. コマンド入力例

本書に記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「 / 」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「 ~ 」で表します。

表 1.2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
[PC /]#	作業用 PC の root ユーザで実行
[PC /]$	作業用 PC の一般ユーザで実行
[ATDE ~/]#	ATDE 上の root ユーザで実行
[ATDE ~/]$	ATDE 上の一般ユーザで実行
[armadillo /]#	Armadillo 上 Linux の root ユーザで実行
[armadillo /]$	Armadillo 上 Linux の一般ユーザで実行
[container /]#	Podman コンテナ内で実行
⇒	Armadillo 上 U-Boot の保守モードで実行

コマンド中で、変更の可能性のあるものや、環境により異なるものに関しては以下のように表記します。適宜読み替えて入力してください。

表 1.3 コマンド入力例での省略表記

表記	説明
[VERSION]	ファイルのバージョン番号

1.6.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

用語の説明や補足的な説明を記載します。

1.7. 謝辞

Armadillo で使用しているソフトウェアの多くは Free Software / Open Source Software で構成されています。Free Software / Open Source Software は世界中の多くの開発者の成果によってなりたっています。この場を借りて感謝の意を表します。

2. 注意事項

2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。

- ご使用の前に必ず製品マニュアルおよび関連資料をお読みになり、使用上の注意を守って正しく安全にお使いください。
- マニュアルに記載されていない操作・拡張などを行う場合は、弊社 Web サイトに掲載されている資料やその他技術情報を十分に理解した上で、お客様自身の責任で安全にお使いください。
- 水・湿気・ほこり・油煙等の多い場所に設置しないでください。火災、故障、感電などの原因になる場合があります。
- 本製品に搭載されている部品の一部は、発熱により高温になる場合があります。周囲温度や取扱いによってはやけどの原因となる恐れがあります。本体の電源が入っている間、または電源切断後本体の温度が下がるまでの間は、基板上の電子部品、及びその周辺部分には触れないでください。
- 本製品を使用して、お客様の仕様による機器・システムを開発される場合は、製品マニュアルおよび関連資料、弊社 Web サイトで提供している技術情報のほか、関連するデバイスのデータシート等を熟読し、十分に理解した上で設計・開発を行ってください。また、信頼性および安全性を確保・維持するため、事前に十分な試験を実施してください。
- 本製品は、機能・精度において極めて高い信頼性・安全性が必要とされる用途(医療機器、交通関連機器、燃焼制御、安全装置等)での使用を意図しておりません。これらの設備や機器またはシステム等に使用された場合において、人身事故、火災、損害等が発生した場合、当社はいかなる責任も負いかねます。
- 本製品には、一般電子機器用(OA 機器・通信機器・計測機器・工作機械等)に製造された半導体部品を使用しています。外来ノイズやサージ等により誤作動や故障が発生する可能性があります。万一誤作動または故障などが発生した場合に備え、生命・身体・財産等が侵害されることのないよう、装置としての安全設計(リミットスイッチやヒューズ・ブレーカー等の保護回路の設置、装置の多重化等)に万全を期し、信頼性および安全性維持のための十分な措置を講じた上でお使いください。
- 電池をご使用の際は、極性(プラスとマイナス)を逆にして装着しないでください。また、電池の使用推奨期限を過ぎた場合や RTC の時刻を保持できなくなった場合には、直ちに電池を交換してください。そ

のまま使用すると、電池が漏液、発熱、破裂したり、ケガや製品の故障の原因となります。万一、漏れた液が身体に付着した場合は多量の水で洗い流してください。

- 無線 LAN 機能を搭載した製品は、心臓ペースメーカーや補聴器などの医療機器、火災報知器や自動ドアなどの自動制御器、電子レンジ、高度な電子機器やテレビ・ラジオに近接する場所、移動体識別用の構内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

2.2. 取扱い上の注意事項

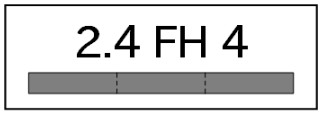
本製品に恒久的なダメージをあたえないよう、取扱い時には以下のような点にご注意ください。

破損しやすい箇所	microSD コネクタおよびそのカバーやフラットケーブルコネクタは、破損しやすい部品になっています。無理に力を加えて破損することのないよう十分注意してください。
本製品の改造	本製品に改造[1]を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設[2]を行う場合は、作業前に必ず動作確認を行ってください。
電源投入時のコネクタ着脱	本製品や周辺回路に電源が入っている状態で、活線挿抜対応インターフェース(LAN, USB) [3]以外へのコネクタ着脱は、絶対に行わないでください。
静電気	本製品には CMOS デバイスを使用しており、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。
ラッチアップ	電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等により、使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには、保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。
衝撃	落下や衝撃などの強い振動を与えないでください。
使用場所の制限	無線機能を搭載した製品は、テレビ・ラジオに近接する場所で使用すると、受信障害を招く恐れがあります。
振動	振動が発生する環境では、Armadillo が動かないよう固定して使用してください。

[1]本書を含めた関連マニュアルで改造方法を記載している箇所および、コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2]改造やコネクタを増設する際にはマスキングを行い、周囲の部品に半田くず、半田ボール等付着しないよう十分にご注意ください。
[3]別途、活線挿抜を禁止している場合を除く

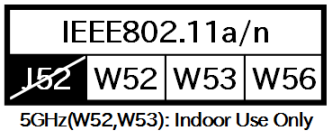
電池の取り扱い	電池の使用推奨期限を過ぎる前に電池の交換をしてください。使用推奨期限を超えて使用すると、電池の性能が充分に発揮できない場合や、電池を漏液させたり、製品を破損させるおそれがあります。
電波に関する注意事項(2.4GHz 帯無線)	2.4GHz 帯の電波を使用する機能(無線 LAN 等)は、自動ドアなどの自動制御電子機器に影響が出る場合、すぐに使用を中止してください。

この無線機(Sterling LWB5+)は 2.4GHz 帯を使用します。全帯域を使用し、かつ移動体識別装置の帯域が回避可能です。変調方式として DS-SS および OFDM 方式を採用し、想定される与干渉距離は 40m 以下です。



この無線機(Sterling LWB5+)は 2.4GHz 帯を使用します。全帯域を使用し、かつ移動体識別装置の帯域が回避不可です。変調方式として FH-SS 方式を採用し、想定される与干渉距離は 40m 以下です。

電波に関する注意事項(5GHz 帯無線)	この無線機(Sterling LWB5+)は 5GHz 帯を使用します。W52、W53 の屋外での利用は電波法により禁じられています。W53、W56 での AP モードは、現在工事設計認証を受けていないため使用しないでください。



電波に関する注意事項 (LTE)	この無線機(EMS31-J)は LTE 通信を行います。LTE 通信機能は、 心臓ペースメーカーや除細動器等の植込み型医療機器の近く(15cm 程度以内)で使用しないでください。
電気通信事業法に関する注意事項について	本製品の有線 LAN を、 電気通信事業者の通信回線(インターネットサービスプロバイダーが提供している通信網サービス等)に直接接続することはできません。 接続する場合は、 必ず電気通信事業法の認定を受けた端末設備(ルーター等)を経由して接続してください。

2.3. 製品の保管について

- 製品を在庫として保管するときは、高温・多湿、埃の多い環境、水濡れの可能性のある場所、直射日光のあたる場所、有毒ガス (特に腐食性ガス)の発生する場所を避け、精密機器の保管に適した状態で保管してください。
- 保管環境として推奨する温度・湿度条件は以下のとおりです。

表 2.1 推奨温湿度環境について

推奨温湿度環境	5～35℃/70%RH 以下[a] [b]

[a]半田付け作業を考慮した保管温度範囲となっております。半田付けを行わない、または、すべての半田付けが完了している場合の推奨温度・湿度条件は、製品の動作温度・湿度範囲となります。
[b]温度変化の少ない場所に保管してください。保管時の急激な温度変化は結露が生じ、金属部の酸化、腐食などが発生し、はんだ濡れ性に影響が出る場合があります。

- 製品を包装から取り出した後に再び保管する場合は、帯電防止処理された収納容器を使用してください。

2.4. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて

本製品の標準出荷状態でプリインストールされている Linux 対応ソフトウェアは、個別に明示されている（書面、電子データでの通知、口頭での通知を含む）場合を除き、オープンソースとしてソースコードが提供されています。再配布等の権利については、各ソースコードに記載のライセンス形態にしたがって、お客様の責任において行使してください。また、本製品に含まれるソフトウェア（付属のドキュメント等も含む）は、現状有姿（AS IS）にて提供します。お客様ご自身の責任において、使用用途・目的の適合について事前に十分な検討と試験を実施した上でお使いください。アットマークテクノは、当該ソフトウェアが特定の目的に適合すること、ソフトウェアの信頼性および正確性、ソフトウェアを含む本製品の使用による結果について、お客様に対し何らの保証も行いません。

パートナー等の協力により Armadillo ブランド製品向けに提供されているミドルウェア、その他各種ソフトウェアソリューションは、ソフトウェア毎にライセンスが規定されています。再頒布権等については、各ソフトウェアに付属する readme ファイル等をご参照ください。その他のバンドルソフトウェアについては、各提供元にお問い合わせください。

以下のソフトウェアは、オープンソースソフトウェアではありません。

ボード情報取得ツール(get-board-info)

2.5. 電波障害について

この装置は、クラス B 情報技術装置です。 この装置は、 住宅環境で使用することを目的としていますが、 この装置がラジオやテレビジョン受信機に近接して使用されると 、 受信障害を引き起こすことがあります。取扱説明書に従って正しい取り扱いをして下さい。VCCI-B

2.6. 無線モジュールの安全規制について

本製品に搭載されている LTE モジュール EMS31-J は、 電気通信事業法に基づく設計認証を受けています。

また、本製品に搭載されている LTE モジュール EMS31-J 、WLAN+BT コンボモジュール Sterling LWB5+ は、 電波法に基づく工事設計認証を受けています。

これらの無線モジュールを国内で使用するときに無線局の免許は必要ありません。

以下の事項を行うと法律により罰せられることがあります。

- 無線モジュールやアンテナを分解/改造すること。
- 無線モジュールや筐体、基板等に直接印刷されている証明マーク・証明番号、または貼られている証明ラベルをはがす、消す、上からラベルを貼るなどし、見えない状態にすること。

認証番号は次のとおりです。

表 2.2 LTE モジュール:EMS31-J 適合証明情報

項目	内容
型式又は名称	EMS31-J
電波法に基づく工事設計認証における認証番号	003-180278
電気通信事業法に基づく設計認証における認証番号	D180162003

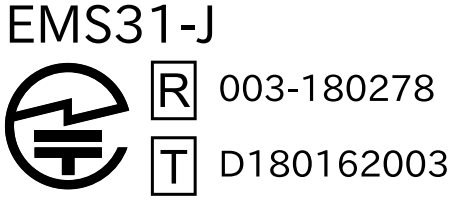


図 2.1 LTE モジュール:EMS31-J 認証マーク

表 2.3 WLAN+BT コンボモジュール:Sterling LWB5+ 適合証明情報

項目	内容
型式又は名称	Sterling LWB5+
電波法に基づく工事設計認証における認証番号	201-200402

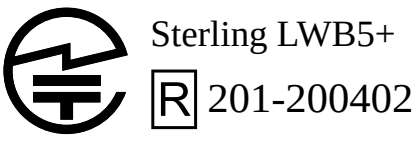


図 2.2 WLAN+BT コンボモジュール:Sterling LWB5+ 認証マーク

2.7. 保証について

本製品の本体基板は、製品に添付もしくは弊社 Web サイトに記載している「製品保証規定」に従い、ご購入から 1 年間の交換保証を行っています。添付品およびソフトウェアは保証対象外となりますのでご注意ください。

製品保証規定 https://armadillo.atmark-techno.com/support/warranty/policy

2.8. 輸出について

- 当社製品は、原則として日本国内での使用を想定して開発・製造されています。
- 海外の法令および規則への適合については当社はなんらの保証を行うものではありません。
- 当社製品を輸出するときは、輸出者の責任において、日本国および関係する諸外国の輸出関連法令に従い、必要な手続を行っていただきますようお願いいたします。
- 日本国およびその他関係諸国による制裁または通商停止を受けている国家、組織、法人または個人に対し、当社製品を輸出、販売等することはできません。
- 当社製品および関連技術は、大量破壊兵器の開発等の軍事目的、その他国内外の法令により製造・使用・販売・調達が禁止されている機器には使用することができません。

2.9. 商標について

- Armadillo は株式会社アットマークテクノの登録商標です。その他の記載の商品名および会社名は、各社・各団体の商標または登録商標です。™、®マークは省略しています。
- SD、SDHC、SDXC、microSD、microSDHC、microSDXC、SDIO ロゴは SD-3C, LLC の商標です。

3. 製品概要

3.1. 製品の特長

3.1.1. Armadillo とは

「Armadillo(アルマジロ)」は、Arm コアプロセッサ搭載・Linux 対応の組み込みプラットフォームのブランドです。Armadillo ブランド製品には以下の特長があります。

- Arm プロセッサ搭載・省電力設計

  Arm コアプロセッサを搭載しています。1〜数ワット程度で動作する省電力設計で、発熱が少なくファンを必要としません。

- 小型・手のひらサイズ

  CPU ボードは名刺サイズ程度の手のひらサイズが主流です。名刺の 1/3 程度の小さな CPU モジュールや無線 LAN モジュール等、超小型のモジュールもラインアップしています。

- 標準 OS として Linux をプリインストール

  標準 OS に Linux を採用しており、豊富なソフトウェア資産と実績のある安定性を提供します。ソースコードをオープンソースとして公開しています。

- 開発環境

  Armadillo の開発環境として、「Atmark Techno Development Environment ATDE)」を無償で提供しています。ATDE は、VMware など仮想マシン向けのデータイメージです。このイメージには、Linux デスクトップ環境をベースに GNU クロス開発ツールやその他の必要なツールが事前にインストールされています。ATDE を使うことで、開発用 PC の用意やツールのインストールなどといった開発環境を整える手間を軽減することができます。

3.1.2. Armadillo-IoT ゲートウェイ A6E とは

Armadillo-IoT ゲートウェイ A6E は、従来モデル以上に省電力で動作する IoT ゲートウェイです。超低消費電力でクラウドと通信できるセルラー LPWA（LTE-M）モジュールを搭載。自立型のシステムを構築する際に、太陽光パネルや蓄電池はより小さな容量を選択可能なため、システム全体のコストを大幅に低減することができます。

高い自由度と、開発のしやすさ、組み込み機器としての堅牢性をバランスよく兼ね備えており、オリジナルの商用 IoT ゲートウェイを市場のニーズに合わせてタイムリーに開発したい方に好適です。

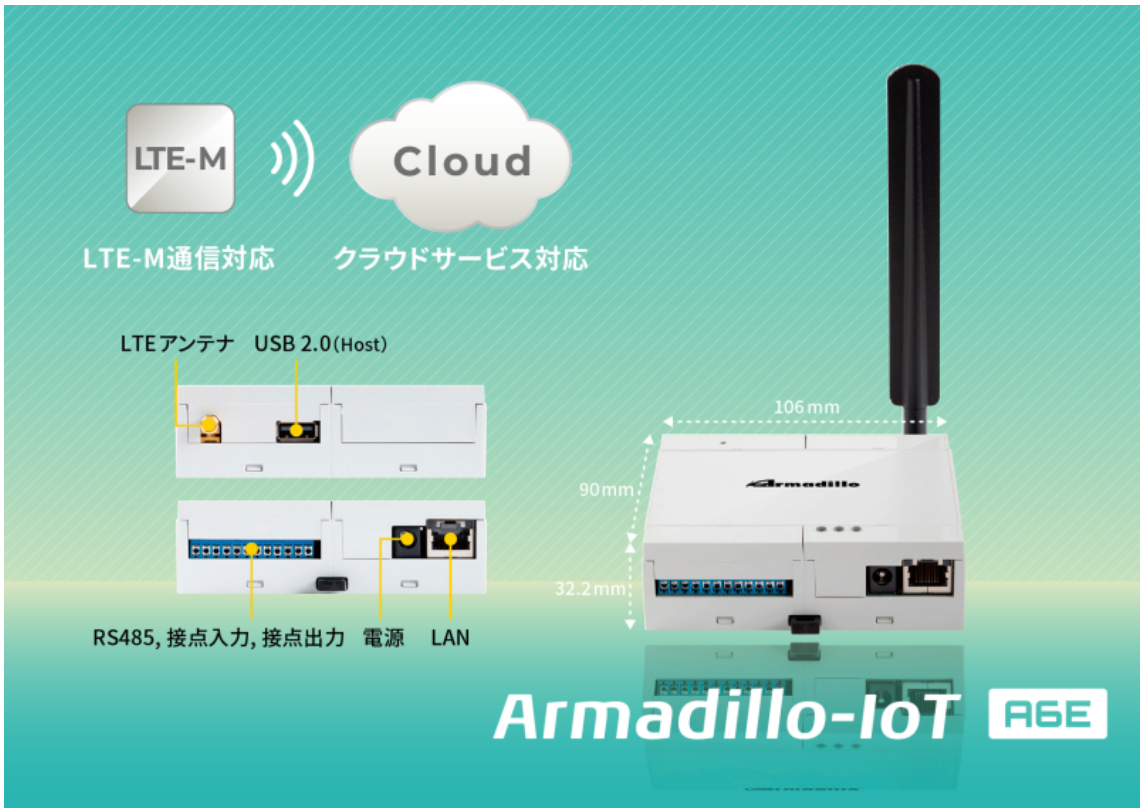


図 3.1 Armadillo-IoT ゲートウェイ A6E とは

・省電力モード搭載・バッテリー駆動の機器に最適

省電力モードを搭載し、「アプリケーションから Armadillo-IoT ゲートウェイ A6E 本体の電源を OFF にする」「RTC(リアルタイムクロック)のアラームで決まった時間に本体の電源を ON にする」「省電力モードで動作させ、SMS の受信で復帰する」といった細かな電源制御、間欠動作が可能です。必要な時だけ本体を起動するといった間欠動作運用が可能なので、バッテリーで稼動させるような機器に適しています。

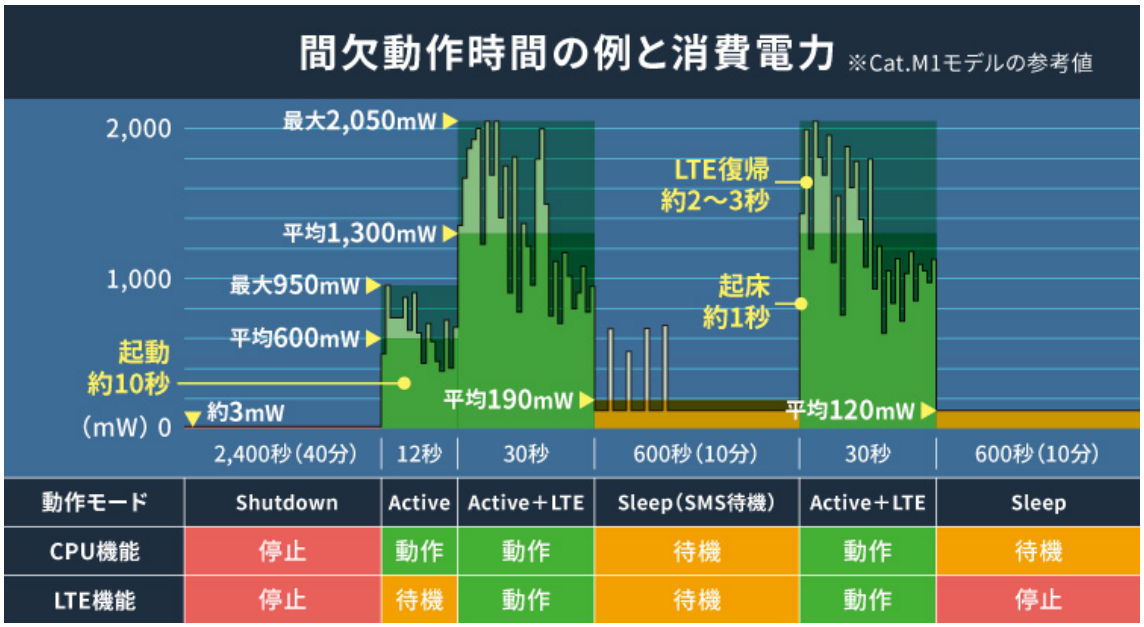


図 3.2 間欠動作の例

・ RS485 や接点入出力を標準搭載

LAN、USB2.0 のインターフェースに加えて、多くの事例で利用されている RS485 シリアル通信(半二重)、接点入力 2ch、接点出力 2ch を標準搭載しました。また、筐体内の基板には更なる拡張インターフェース(UART/GPIO/I2C/SPI/CAN/PWM/他)が用意されており、幅広い目的に沿った拡張をすることができます。

図 3.3 様々なデバイスとの接続例

・ コンテナ型の Armadillo Base OS を搭載し、差分アップデートにも対応

Linux をベースとした Armadillo Base OS はコンパクトでセキュリティリスクが抑えられたコンテナアーキテクチャーの OS で、標準でソフトウェアアップデート機能を有しています。アプリケーションソフトウェアはコンテナ上で動作し、コンテナのアップデートで新機能の追加やセキュリティ

更新をすることができます。また差分アップデート機能にも対応しているため、アップデート時の通信容量を抑えることができ、通信速度が限られている LTE-M 回線でも運用しやすくなっています。

- 各種クラウド IoT サービスに対応したゲートウェイコンテナを提供

  各種クラウド IoT サービス(Azure IoT や AWS IoT Core)に対応したゲートウェイコンテナを用意しました。これまでの従来モデルでは、ユーザー自身が開発するアプリケーションソフトウェアで、ゲートウェイとしての機能の他、通信障害時の対応、セキュリティ対応、間欠動作時の挙動などの難しい課題を自ら解決する必要がありました。今回、あらかじめ用意されたゲートウェイコンテナを活用することで、これらの課題に対処することができ、短期間に IoT システムを構築可能です。

- 用途に合わせて複数のラインアップを用意

  クラウドとの通信回線に LTE-M(Cat.M1)を採用した「Cat.M1 モデル」を標準として、より高速な通信を必要とする用途向けに LTE(Cat.1)を採用した「Cat.1 モデル」、既設の LAN/無線 LAN を利用する安価な「LAN モデル」が用意されています。※ 2022 年 10 月現時点で、「Cat.1 モデル」「LAN モデル」は開発中です。

3.1.3. Armadillo Base OS とは

Armadillo Base OS は、アットマークテクノが提供する専用ディストリビューションです。Linux5.10 をベースに、コンテナ管理機能、ソフトウェアアップデート機能、ネットワークマネージャーなどに対応。機能を限定したコンパクトな OS で、安全性の高い運用を実現します。

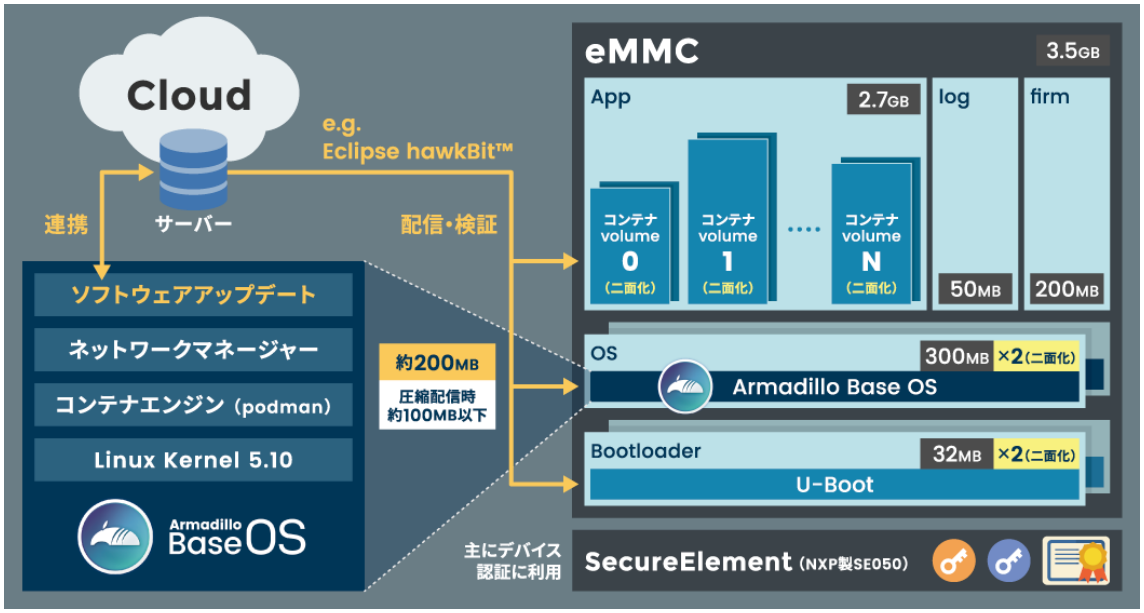


図 3.4 Armadillo Base OS とは

- OS のコンパクト化

  OS 基盤の機能を最小限にしたことで、セキュリティリスクを低減しています。アットマークテクノが継続的にアップデートを提供するため、高セキュリティな IoT 機器として長期間に渡り運用することができます。

- コンテナによるアプリケーション運用

アプリケーションを「コンテナ」単位で OS から分離して管理できるため、コンテナごとのアップデートが可能です。サンドボックス化されることにより、悪意あるソフトウェアからの攻撃に対する機器全体の保護に有効性を発揮します。

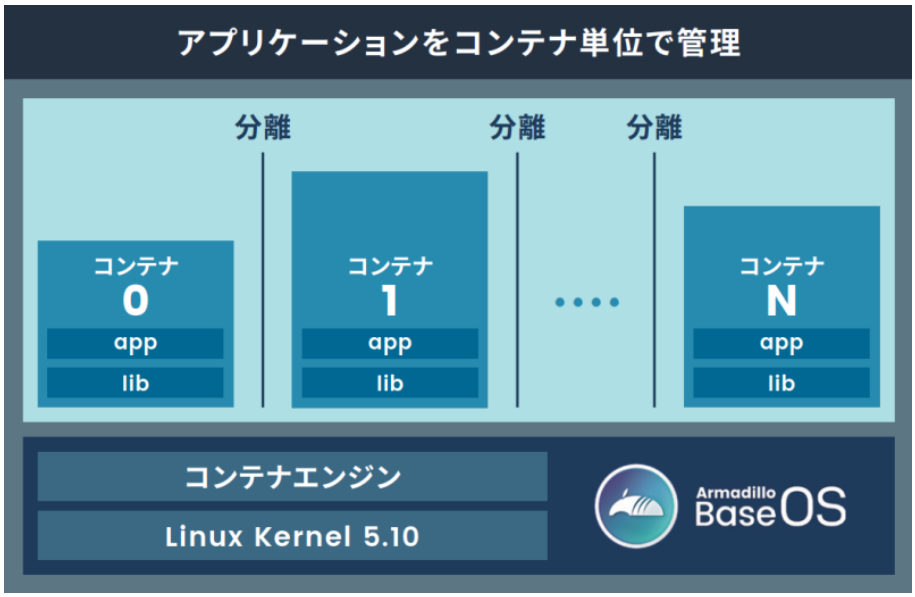


図 3.5 コンテナによるアプリケーションの運用

- アップデート機能を標準搭載

  ネットワークや USB メモリ、microSD カードによるアップデート機能を標準搭載しています。正しく署名されたソフトウェアのみアップデートできる仕組みや、差分アップデート機能も用意されています。OS・ブートローダー・コンテナ部分は、安全性を担保するため二面化し、リカバリー機能を備えています。万が一アップデートに失敗した場合でも、作業前の状態にロールバックすることができます。

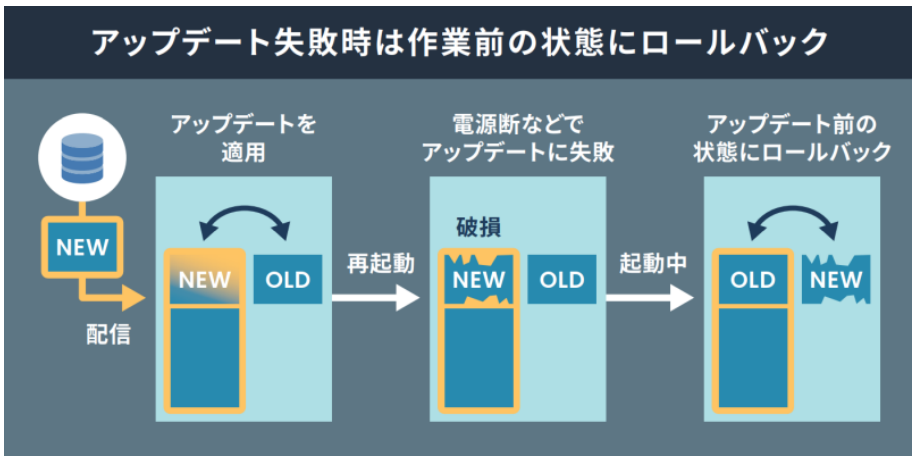


図 3.6 ロールバックの仕組み

- 堅牢性の向上

  安定性の高いファイルシステムで、ストレージへの書込みを減らして消耗を抑制するなど、高い堅牢性を有します。運用ログの記録機能も標準搭載しています。

・セキュリティ機能の向上

コンテナにアクセス権限を設けて管理することができます。デバイス証明に利用できるセキュアエレメントを搭載するほか、セキュア環境「OP-TEE」を利用可能な状態で提供しています。

3.2. 製品ラインアップ

Armadillo-IoT ゲートウェイ A6E の製品ラインアップは次のとおりです。

表 3.1 Armadillo-IoT ゲートウェイ A6E ラインアップ

名称	型番
Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 開発セット (LTE アンテナセット付属)	AG6221-C01D0
Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 量産用 (LTE アンテナセット付属)	AG6221-C01Z

3.2.1. Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 開発セット

Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 開発セット(型番:AG6221-C01D0)は、Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル を使った開発がすぐに開始できるように、開発に必要なものを一式含んだセットです。

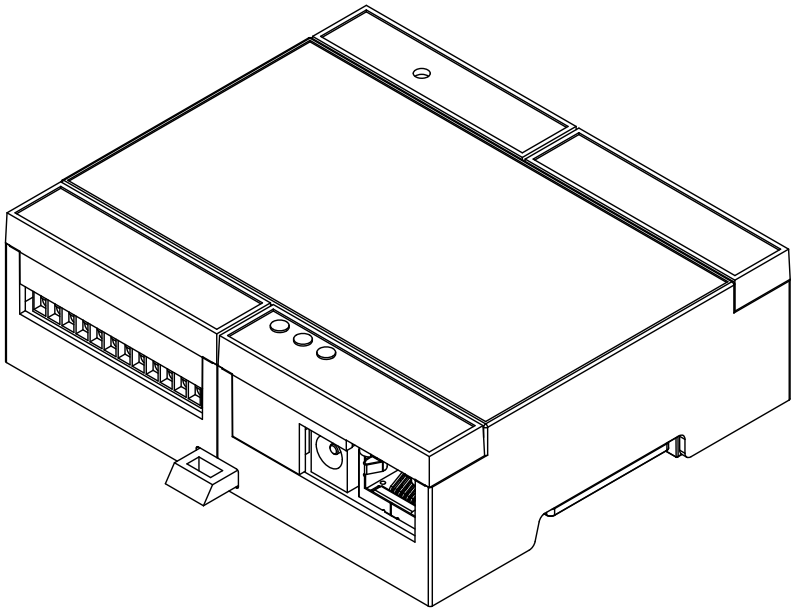


図 3.7 Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル

・Armadillo-IoT ゲートウェイ A6E Cat.M1 モデル 本体

・LTE 用外付けアンテナ

・USB(A オス-microB)ケーブル

・AC アダプタ(12V/2.0A)

3.2.2. Armadillo-IoT ゲートウェイ A6E 量産用

Armadillo-IoT ゲートウェイ A6E 量産用は、Armadillo-IoT ゲートウェイ A6E 開発セットのセット内容を必要最小限に絞った量産向けのラインアップです。Cat.M1 モデル(AG6221-C01Z)の 1 種類となります。

3.3. 仕様

Armadillo-IoT ゲートウェイ A6E の主な仕様は次のとおりです。

表 3.2 仕様

型番	AG6221-C01D0, AG6221-C01Z
プロセッサ	NXP Semiconductors i.MX6ULL
	ARM Cortex-A7 x 1 ・命令/データキャッシュ 32KByte/32KByte ・L2 キャッシュ 128KByte ・内部 SRAM 128KByte ・メディアプロセッシングエンジン(NEON)搭載 ・Thumb code(16bit 命令セット)サポート
システムクロック	CPU コアクロック(ARM Cortex-A7): 528MHz DDR クロック: 396MHz 源発振クロック: 32.768kHz, 24MHz
RAM	DDR3L: 512MByte バス幅: 16bit
ROM	eMMC: 3.5GB [a]
LAN(Ethernet)	100BASE-TX/10BASE-T x 1 AUTO-MDIX 対応
モバイル通信	LTE CAT-M1（タレス DIS 製 EMS31-J 搭載） [b] [c] SIM スロット: nanoSIM 対応
USB	USB 2.0 Host x 1 (High Speed)
SD	microSD スロット x 1 [d]
入出力インターフェース	接点入力(電流シンク出力タイプに接続可能) x 2 接点出力(無極性) x 2
シリアル(RS485)	2 線式(Data+, Data-, GND) x 1 最大データ転送レート: 5Mbps 終端抵抗 120Ω 内蔵 [e]
拡張インターフェース [f]	GPIO x 24、UART x 2、I2C x 1、SPI x 1、CAN x 2、I2S x 1、PWM x 4、A/D x 4、MQS x 1
カレンダ時計	リアルタイムクロック搭載 外部バックアップ用電源入力対応
スイッチ	ユーザースイッチ x 1 設定用スイッチ x 2
LED	SYS(Green) x 1 APP(Green) x 1 WWAN(Green) x 1
メンテナンスポート	USB micro B シリアルコンソール
セキュアエレメント	NXP Semiconductors SE050
入力電源	DC 8～26.4V
消費電力(参考値)[g]	約 3mW : シャットダウン時 約 120mW : スリープ時 約 200mW : スリープ時 (SMS 起床可能) 約 600mW : アクティブ時 約 2050mW : 最大消費電力
動作温度範囲	-20～+60℃ (結露なきこと)
外形サイズ(基板)	103 x 87 mm (突起部、アンテナを除く)

型番	AG6221-C01D0, AG6221-C01Z
外形サイズ(ケース)	106.2 x 90 x 32.2 mm (突起部、アンテナを除く)

[a]pSLC での数値です。出荷時 pSLC に設定しています。

[b]モバイル通信を利用する時は、外付けアンテナを接続する必要があります。

[c]認証取得済みキャリア: docomo/Softbank/KDDI、対応バンド: (1/8/18/19/26)、下り 300kbit/s、上り 375kbit/s※Softbank をご利用予定の場合はお問い合わせください。※KDDI は料金プランが LPWA (LTE-M) の SIM のみ動作いたします。LTE Cat 1 などの料金プランでは動作しません。

[d]ケースに入れた状態で操作することはできません。

[e]ディップスイッチの操作で抵抗の切り離しが可能です。

[f]i.MX6ULL のピンマルチプクレスの設定で、優先的に機能を割り当てた場合に拡張可能な最大数を記載しています。

[g]LTE の signal quality が 80%かつ周辺機器が未接続の時の参考値となります。電波環境や接続するデバイスにより消費電力は変化します。

3.4. ブロック図

Armadillo-IoT ゲートウェイ A6E のブロック図は次のとおりです。

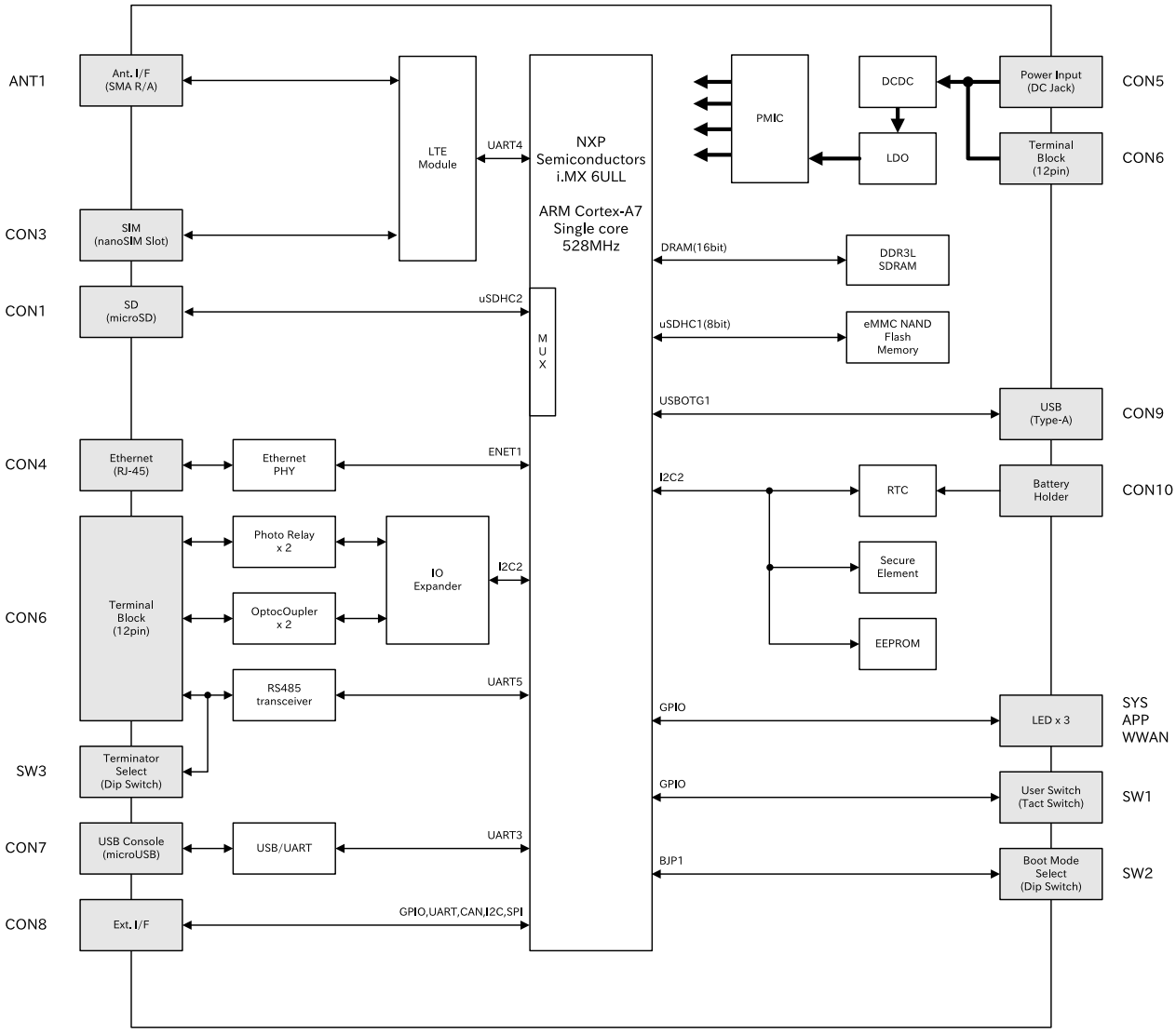


図 3.8 Armadillo-IoT ゲートウェイ A6E ブロック図

3.5. ストレージデバイスのパーティション構成

Armadillo-IoT ゲートウェイ A6E の eMMC のパーティション構成を「表 3.3. eMMC メモリマップ」に示します。

表 3.3 eMMC メモリマップ

パーティション	サイズ	ラベル	説明
1	300MiB	rootfs_0	A/B アップデートの A 面パーティション(Linux カーネルイメージ, Device Tree Blob, Alpine Linux rootfs を含む)
2	300MiB	rootfs_1	A/B アップデートの B 面パーティション(Linux カーネルイメージ, Device Tree Blob, Alpine Linux rootfs を含む)
3	50MiB	logs	ログ書き込み用パーティション
4	200MiB	firm	ファームウェア用パーティション
5	2.5GiB	app	アプリケーション用パーティション

Armadillo-IoT ゲートウェイ A6E の eMMC のブートパーティションの構成を「表 3.4. eMMC ブートパーティション構成」に示します。

表 3.4 eMMC ブートパーティション構成

ディスクデバイス	サイズ	説明
/dev/mmcblk0boot0	4 MiB	A/B アップデートの A 面
/dev/mmcblk0boot1	4 MiB	A/B アップデートの B 面

Armadillo-IoT ゲートウェイ A6E の eMMC の GPP(General Purpose Partition)の構成を「表 3.5. eMMC GPP 構成」に示します。

表 3.5 eMMC GPP 構成

ディスクデバイス	サイズ	説明
/dev/mmcblk0gp0	8 MiB	ライセンス情報等の保存
/dev/mmcblk0gp1	8 MiB	予約領域
/dev/mmcblk0gp2	8 MiB	予約領域
/dev/mmcblk0gp3	8 MiB	ユーザー領域

4. Armadillo の電源を入れる前に

4.1. 準備するもの

Armadillo を使用する前に、次のものを必要に応じて準備してください。

作業用 PC	Linux または Windows が動作し、ネットワークインターフェースと 1 つ以上の USB ポートを持つ PC です。「開発/動作確認環境の構築」を参照して、作業用 PC 上に開発/動作確認環境を構築してください。
ネットワーク環境	Armadillo と作業用 PC をネットワーク通信ができるようにしてください。
tar.xz 形式のファイルを展開するソフトウェア	開発/動作確認環境を構築するために利用します。Linux では、tar で展開できます。Windows では、7-Zip や Lhaz などが対応しています。
nanoSIM(UIM カード)と APN 情報	LTE モデルで LTE の動作を確認する場合に利用します。通信事業者との契約が必要です。SMS の動作を確認する場合は、SMS が利用可能な nanoSIM(UIM カード)が必要です。

4.2. 開発/動作確認環境の構築

アットマークテクノ製品のソフトウェア開発や動作確認を簡単に行うために、VMware 仮想マシンのデータイメージを提供しています。この VMware 仮想マシンのデータイメージを ATDE(Atmark Techno Development Environment)と呼びます。ATDE の起動には仮想化ソフトウェアである VMware を使用します。ATDE のデータは、tar.xz 圧縮されています。環境に合わせたツールで展開してください。

仮想化ソフトウェアとして、VMware の他に Oracle VM VirtualBox が有名です。Oracle VM VirtualBox には以下の特徴があります。

- GPL v2(General Public License version 2)で提供されている [1]
- VMware 形式の仮想ディスク(.vmdk)ファイルに対応している

Oracle VM VirtualBox から ATDE を起動し、ソフトウェア開発環境として使用することができます。

ATDE は、バージョンにより対応するアットマークテクノ製品が異なります。本製品に対応している ATDE は、ATDE9 の v20221025 以降です。

ATDE9 は Debian GNU/Linux 11 (コードネーム bullseye) をベースに、Armadillo-IoT ゲートウェイ A6E のソフトウェア開発を行うために必要なクロス開発ツールや、Armadillo-IoT ゲートウェイ A6E の動作確認を行うために必要なツールが事前にインストールされています。

[1] バージョン 3.x までは PUEL(VirtualBox Personal Use and Evaluation License)が適用されている場合があります。

4.2.1. ATDE のセットアップ

4.2.1.1. VMware のインストール

ATDE を使用するためには、作業用 PC に VMware がインストールされている必要があります。VMware 社 Web ページ(http://www.vmware.com/)を参照し、利用目的に合う VMware 製品をインストールしてください。また、ATDE のアーカイブは tar.xz 圧縮されていますので、環境に合せたツールで展開してください。

VMware は、非商用利用限定で無償のものから、商用利用可能な有償のものまで複数の製品があります。製品ごとに異なるライセンス、エンドユーザー使用許諾契約書(EULA)が存在するため、十分に確認した上で利用目的に合う製品をご利用ください。

VMware や ATDE が動作しないことを未然に防ぐため、使用する VMware のドキュメントから以下の項目についてご確認ください。

- ホストシステムのハードウェア要件
- ホストシステムのソフトウェア要件
- ゲスト OS のプロセッサ要件

VMware のドキュメントは、VMware 社 Web ページ(http://www.vmware.com/)から取得することができます。

4.2.1.2. ATDE のアーカイブを取得

ATDE のアーカイブは Armadillo サイト(http://armadillo.atmark-techno.com)から取得可能です。

本製品に対応している ATDE のバージョンは ATDE9 v20221025 以降です。

作業用 PC の動作環境(ハードウェア、VMware、ATDE の対応アーキテクチャなど)により、ATDE が正常に動作しない可能性があります。VMware 社 Web ページ(http://www.vmware.com/)から、使用している VMware のドキュメントなどを参照して動作環境を確認してください。

4.2.1.3. ATDE のアーカイブを展開

ATDE のアーカイブを展開します。ATDE のアーカイブは、tar.xz 形式の圧縮ファイルです。

Windows での展開方法を「4.2.1.4. Windows で ATDE のアーカイブ展開する」に、Linux での展開方法を手順「4.2.1.5. Linux で tar.xz 形式のファイルを展開する」に示します。

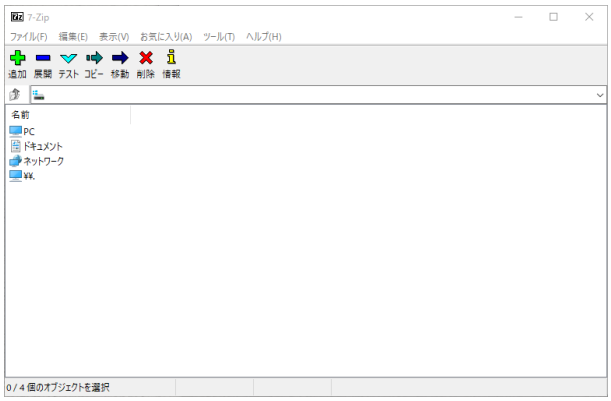
4.2.1.4. Windows で ATDE のアーカイブ展開する

1. 7-Zip のインストール

   7-Zip をインストールします。7-Zip は、圧縮解凍ソフト 7-Zip のサイト (http://sevenzip.sourceforge.jp)からダウンロード取得可能です。

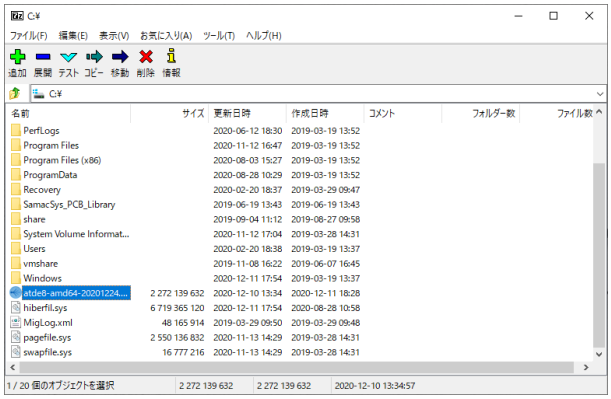
2. 7-Zip の起動

   7-Zip を起動します。



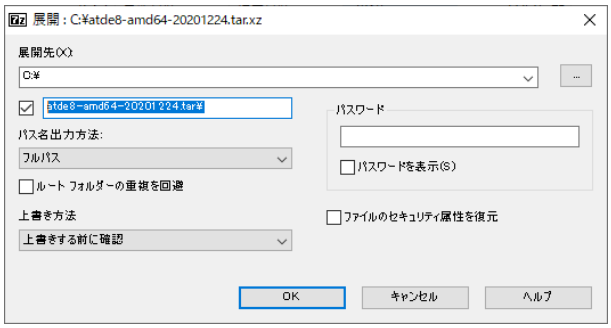
3. xz 圧縮ファイルの選択

   xz 圧縮ファイルを展開して、tar 形式のファイルを出力します。tar.xz 形式のファイルを選択して、「展開」をクリックします。



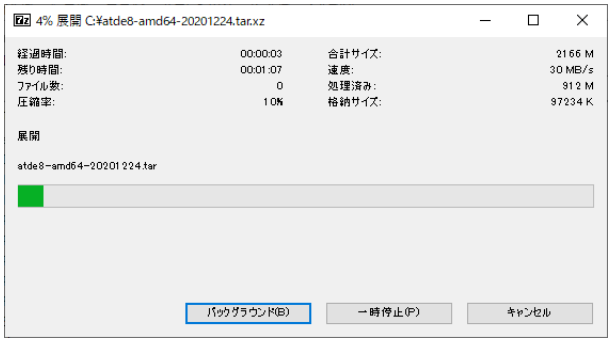
4. xz 圧縮ファイルの展開先の指定

   「展開先」を指定して、「OK」をクリックします。



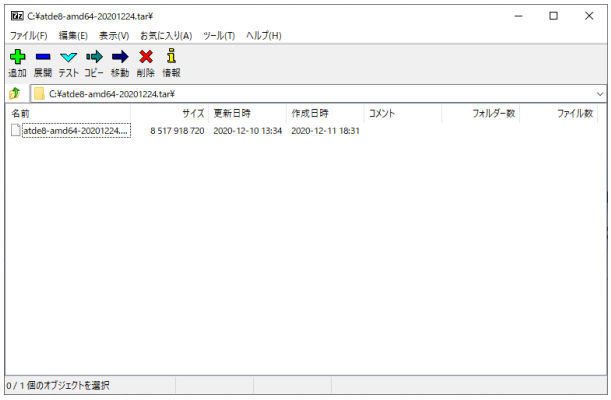
5. xz 圧縮ファイルの展開

   展開が始まります。



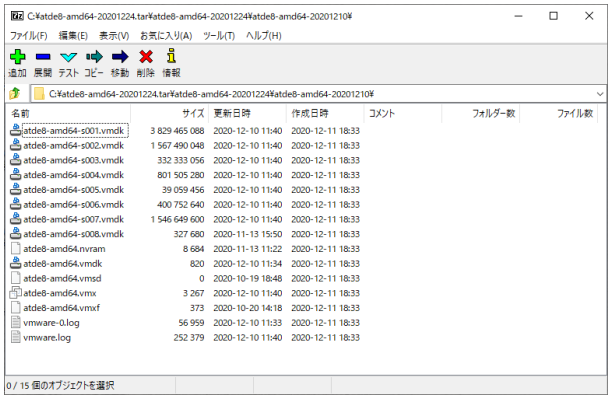
6. tar アーカイブファイルの選択

   xz 圧縮ファイルの展開が終了すると、tar 形式のファイルが出力されます。tar アーカイブファイルを出力したのと同様の手順で、tar アーカイブファイルから ATDE のデータイメージを出力します。tar 形式のファイルを選択して「展開」をクリックし、「展開先」を指定して、「OK」をクリックします。



7. 展開の完了確認

   tar アーカイブファイルの展開が終了すると、ATDE アーカイブの展開は完了です。「展開先」に指定したフォルダに ATDE のデータイメージが出力されています。



4.2.1.5. Linux で tar.xz 形式のファイルを展開する

1. tar.xz 圧縮ファイルの展開

   tar の xf オプションを使用して tar.xz 圧縮ファイルを展開します。

```
[PC ~]$ tar xf atde9-amd64-[VERSION].tar.xz
```

2. 展開の完了確認

   tar.xz 圧縮ファイルの展開が終了すると、ATDE アーカイブの展開は完了です。 **atde9-amd64-[VERSION]** ディレクトリに ATDE のデータイメージが出力されています。

```
[PC ~]$ ls atde9-amd64-[VERSION]/
atde9-amd64-s001.vmdk  atde9-amd64-s008.vmdk
atde9-amd64-s002.vmdk  atde9-amd64-s009.vmdk
atde9-amd64-s003.vmdk  atde9-amd64.nvram
atde9-amd64-s004.vmdk  atde9-amd64.vmdk
atde9-amd64-s005.vmdk  atde9-amd64.vmsd
atde9-amd64-s006.vmdk  atde9-amd64.vmx
atde9-amd64-s007.vmdk  atde9-amd64.vmxf
```

4.2.1.6. ATDE の起動

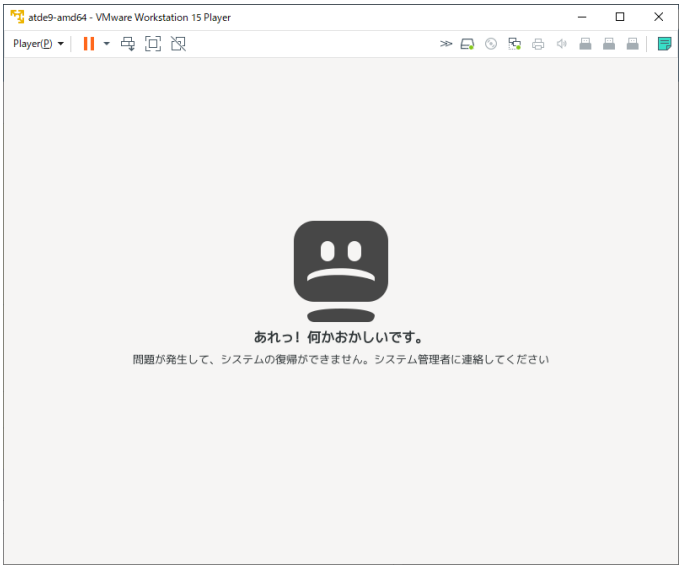
ATDE のアーカイブを展開したディレクトリに存在する仮想マシン構成(.vmx)ファイルを VMware 上で開くと、ATDE を起動することができます。ATDE9 にログイン可能なユーザーを、「表 4.1. ユーザー名とパスワード」に示します [2]。

表 4.1 ユーザー名とパスワード

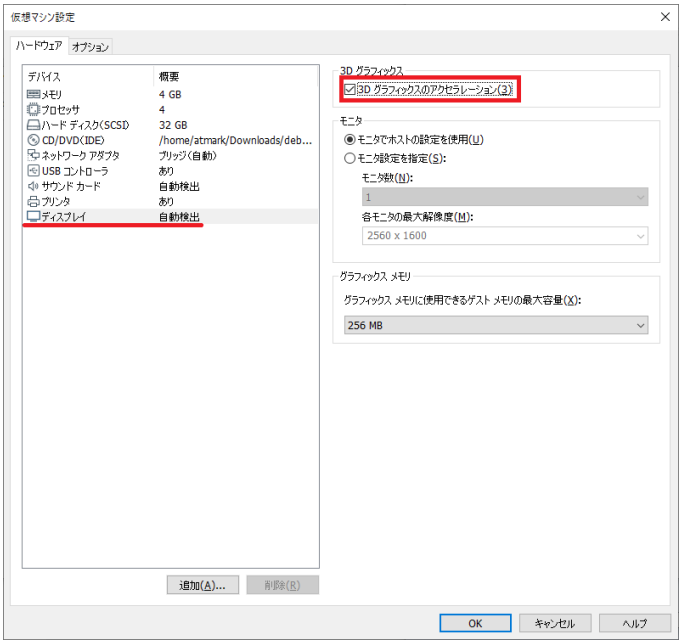
ユーザー名	パスワード	権限
atmark	atmark	一般ユーザー
root	root	特権ユーザー

ATDE を起動する環境によっては、GUI ログイン画面が表示されずに以下のようなエラー画面が表示される場合があります。

[2]特権ユーザーで GUI ログインを行うことはできません



この場合は、VMware の設定で「3D グラフィックスのアクセラレーション」を ON にした後、ATDE を起動すると正常に GUI ログイン画面が表示されます。設定箇所を以下に示します。



ATDE に割り当てるメモリおよびプロセッサ数を増やすことで、ATDE をより快適に使用することができます。仮想マシンのハードウェア設定の変更方法については、VMware 社 Web ページ (http://www.vmware.com/)から、使用している VMware のドキュメントなどを参照してください。

4.2.2. 取り外し可能デバイスの使用

VMware は、ゲスト OS (ATDE)による取り外し可能デバイス(USB デバイスや DVD など)の使用をサポートしています。デバイスによっては、ホスト OS (VMware を起動している OS)とゲスト OS で同時に使用することができません。そのようなデバイスをゲスト OS で使用するためには、ゲスト OS にデバイスを接続する操作が必要になります。

取り外し可能デバイスの使用方法については、VMware 社 Web ページ (http://www.vmware.com/)から、使用している VMware のドキュメントなどを参照してください。

Armadillo-IoT ゲートウェイ A6E の動作確認を行うためには、「表 4.2. 動作確認に使用する取り外し可能デバイス」に示すデバイスをゲスト OS に接続する必要があります。

表 4.2 動作確認に使用する取り外し可能デバイス

デバイス	デバイス名
USB シリアル変換 IC	Silicon CP2102N USB to UART Bridge Controller

4.2.3. コマンドライン端末(GNOME 端末)の起動

ATDE で、CUI (Character-based User Interface)環境を提供するコマンドライン端末を起動します。ATDE で実行する各種コマンドはコマンドライン端末に入力し、実行します。コマンドライン端末にはいくつかの種類がありますが、ここでは GNOME デスクトップ環境に標準インストールされている GNOME 端末を起動します。

GNOME 端末を起動するには、「図 4.1. GNOME 端末の起動」のようにデスクトップ左上のアプリケーションの「ユーティリティ」カテゴリから「端末」を選択してください。

図 4.1 GNOME 端末の起動

「図 4.2. GNOME 端末のウィンドウ」のようにウィンドウが開きます。

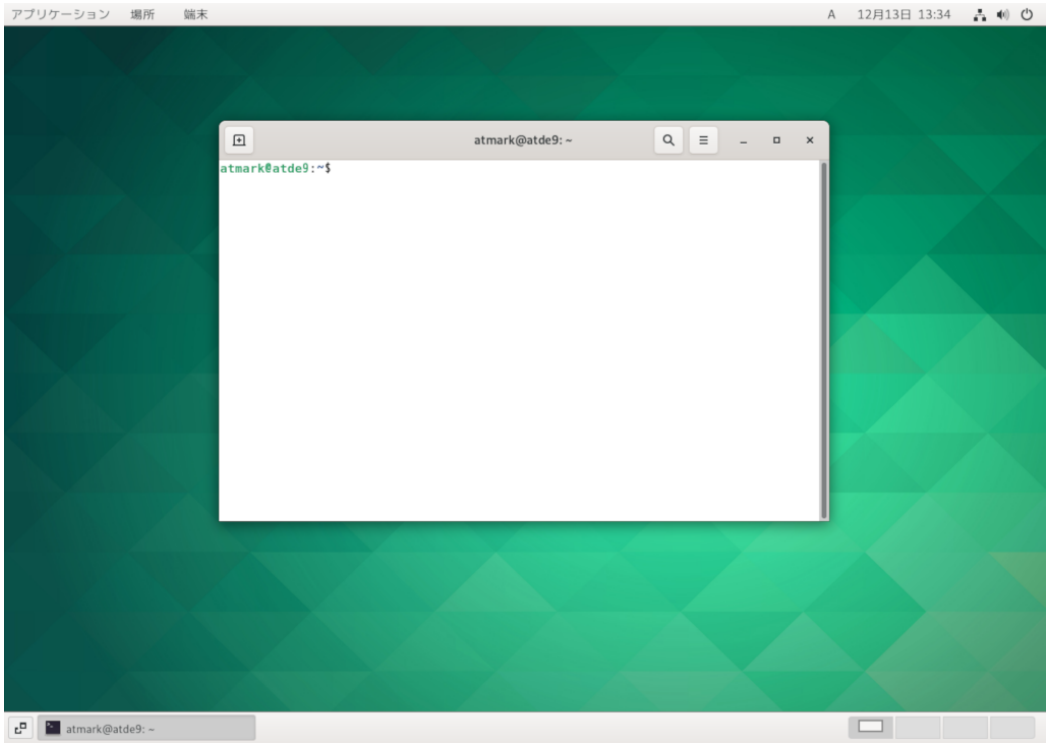


図 4.2 GNOME 端末のウィンドウ

4.2.4. シリアル通信ソフトウェア(minicom)の使用

シリアル通信ソフトウェア(minicom)のシリアル通信設定を、「表 4.3. シリアル通信設定」のように設定します。また、minicom を起動する端末の横幅を 80 文字以上にしてください。横幅が 80 文字より小さい場合、コマンド入力中に表示が乱れることがあります。

表 4.3 シリアル通信設定

項目	設定
転送レート	115,200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

1. 「図 4.3. minicom の設定の起動」に示すコマンドを実行し、minicom の設定画面を起動してください。

```
[ATDE ~]$ sudo LANG=C minicom --setup
```

図 4.3 minicom の設定の起動

2. 「図 4.4. minicom の設定」が表示されますので、「Serial port setup」を選択してください。

```
            +-----[configuration]------+
            | Filenames and paths      |
            | File transfer protocols  |
```

```
            | Serial port setup        |
            | Modem and dialing        |
            | Screen and keyboard      |
            | Save setup as dfl        |
            | Save setup as..          |
            | Exit                     |
            | Exit from Minicom        |
            +--------------------------+
```

図 4.4 minicom の設定

3. 「図 4.5. minicom のシリアルポートの設定」が表示されますので、A キーを押して Serial Device を選択してください。

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

図 4.5 minicom のシリアルポートの設定

4. Serial Device に使用するシリアルポートを入力して Enter キーを押してください。

シリアル通信用 USB ケーブル(A-microB)使用時のデバイスファイル確認方法

Linux でシリアル通信用 USB ケーブル(A-microB)を接続した場合、コンソールに以下のようなログが表示されます。ログが表示されなくても、dmesg コマンドを実行することで、ログを確認することができます。

```
usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
usb 2-2.1: New USB device found, idVendor=10c4, idProduct=ea60,
bcdDevice= 1.00
usb 2-2.1: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
usb 2-2.1: Product: CP2102N USB to UART Bridge Controller
usb 2-2.1: Manufacturer: Silicon Labs
usb 2-2.1: SerialNumber: 6a9681f80272eb11abb4496e014bf449
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial support registered for generic
usbcore: registered new interface driver cp210x
```

```
usbserial: USB Serial support registered for cp210x
usb 2-2.1: cp210x converter now attached to ttyUSB0
```

図 4.6 例. シリアル通信用 USB ケーブル(A-microB)接続時のログ

上記のログからシリアル通信用 USB ケーブル(A-microB)が ttyUSB0 に割り当てられたことが分かります。

5. F キーを押して Hardware Flow Control を No に設定してください。
6. G キーを押して Software Flow Control を No に設定してください。
7. キーボードの E キーを押してください。「図 4.7. minicom のシリアルポートのパラメータの設定」が表示されます。

```
+---------[Comm Parameters]----------+
|                                    |
|     Current: 115200 8N1            |
| Speed            Parity      Data  |
| A: <next>        L: None     S: 5  |
| B: <prev>        M: Even     T: 6  |
| C:   9600        N: Odd      U: 7  |
| D:  38400        O: Mark     V: 8  |
| E: 115200        P: Space          |
|                                    |
| Stopbits                           |
| W: 1             Q: 8-N-1          |
| X: 2             R: 7-E-1          |
|                                    |
|                                    |
| Choice, or <Enter> to exit?        |
+------------------------------------+
```

図 4.7 minicom のシリアルポートのパラメータの設定

8. 「図 4.7. minicom のシリアルポートのパラメータの設定」では、転送レート、データ長、ストップビット、パリティの設定を行います。
9. 現在の設定値は「Current」に表示されています。それぞれの値の内容は「図 4.8. minicom シリアルポートの設定値」を参照してください。

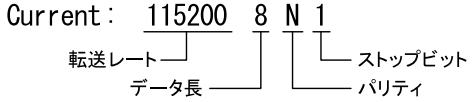


図 4.8 minicom シリアルポートの設定値

10. E キーを押して、転送レートを 115200 に設定してください。
11. Q キーを押して、データ長を 8、パリティを None、ストップビットを 1 に設定してください。
12. Enter キーを 2 回押して、「図 4.4. minicom の設定」に戻ってください。

13.「図 4.4. minicom の設定」から、「Save setup as dfl」を選択し、設定を保存してください。

14.「Exit from Minicom」を選択し、minicom の設定を終了してください。

minicom を起動させるには、「図 4.9. minicom 起動方法」のようにしてください。

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

図 4.9 minicom 起動方法

デバイスファイル名は、環境によって /dev/ttyS0 や /dev/ttyUSB1 など、本書の実行例とは異なる場合があります。

minicom がオープンする /dev/ttyS0 や /dev/ttyUSB0 といったデバイスファイルは、 root または dialout グループに属しているユーザーしかアクセスできません。

ユーザーを dialout グループに入れることで、以降、sudo を使わずに minicom で /dev/ttyUSB0 をオープンすることができます。

```
[ATDE ~]$ sudo usermod -aG dialout atmark
[ATDE ~]$ LANG=C minicom --wrap --device /dev/ttyUSB0
```

minicom を終了させるには、まず Ctrl-a に続いて q キーを入力します。その後、以下のように表示されたら「Yes」にカーソルを合わせて Enter キーを入力すると minicom が終了します。

```
+-----------------------+
| Leave without reset? |
|     Yes       No     |
+-----------------------+
```

図 4.10 minicom 終了確認

Ctrl-a に続いて z キーを入力すると、minicom のコマンドヘルプが表示されます。

4.3. インターフェースレイアウト

Armadillo-IoT ゲートウェイ A6E のインターフェースレイアウトです。一部のインターフェースを使用する際には、ケースを開ける必要があります。

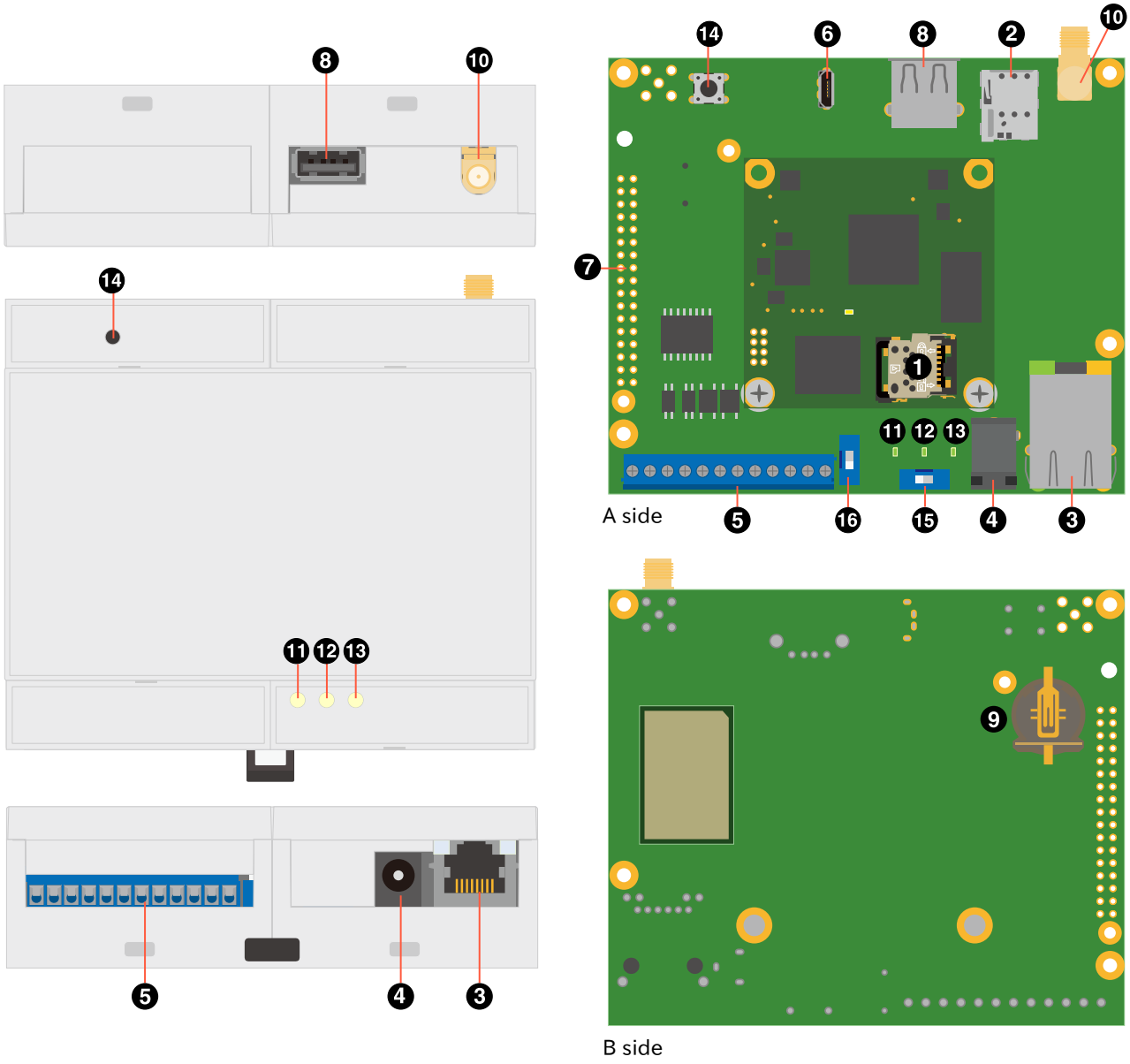


図 4.11 インターフェースレイアウト

表 4.4 インターフェース内容

番号	名称	形状	説明
1	SD インターフェース	microSD スロット	外部ストレージが必要な場合や、ブートローダーを破壊してしまった時の復旧等で使用します。microSD カードを挿入します。
2	nanoSIM インターフェース	nanoSIM スロット	LTE データ通信を利用する場合に使用します。nanoSIM カードを挿入します。
3	LAN インターフェース	RJ-45 コネクタ	有線 LAN を利用する場合に使用します。LAN ケーブルを接続します。
4	電源入力インターフェース	DC ジャック	Armadillo-IoT ゲートウェイ A6E への電源供給で使用します[a]。付属の AC アダプタ(12V/2A)を接続します。対応プラグは内径 2.1mm、外形 5.5mm です。

番号	名称	形状	説明
5	入出力インターフェース	端子台	絶縁入出力、RS485 通信する場合に使用します。Armadillo-IoT ゲートウェイ A6E への電源供給も可能です[a]。
6	USB コンソールインターフェース	USB micro B コネクタ	コンソール入出力を利用する場合に使用します。USB micro B ケーブルを接続します。
7	拡張インターフェース	ピンヘッダ 34 ピン (2.54mm ピッチ)	機能拡張する場合に使用します。2.54mm ピッチのピンヘッダを実装することができます。
8	USB インターフェース	USB 2.0 Type-A コネクタ	外部ストレージが必要な場合等に使用します。USB メモリ等を接続します。
9	RTC バックアップインターフェース	電池ボックス	リアルタイムクロックのバックアップ給電が必要な場合に使用します。対応電池は CR1220 等です。
10	LTE アンテナインターフェース	SMA コネクタ	LTE データ通信を利用する場合に使用します。付属のアンテナを接続します。
11	システム LED	LED(緑色、面実装)	電源の入力状態を表示する緑色 LED です。
12	アプリケーション LED	LED(緑色、面実装)	アプリケーションの状態を表示する緑色 LED です。
13	ワイヤレス WAN	LED(緑色、面実装)	LTE 通信の状態を表示する緑色 LED です。
14	ユーザースイッチ	タクトスイッチ	ユーザーが利用可能なタクトスイッチです。
15	起動デバイス設定スイッチ	DIP スイッチ	起動デバイスを設定する時に使用します。
16	RS485 終端抵抗設定スイッチ	DIP スイッチ	RS485 通信の終端抵抗を設定する時に使用します。

[a]DC ジャックと端子台の両方から同時に電源供給することはできません。

4.4. 接続方法

Armadillo-IoT ゲートウェイ A6E と周辺装置の接続例を次に示します。

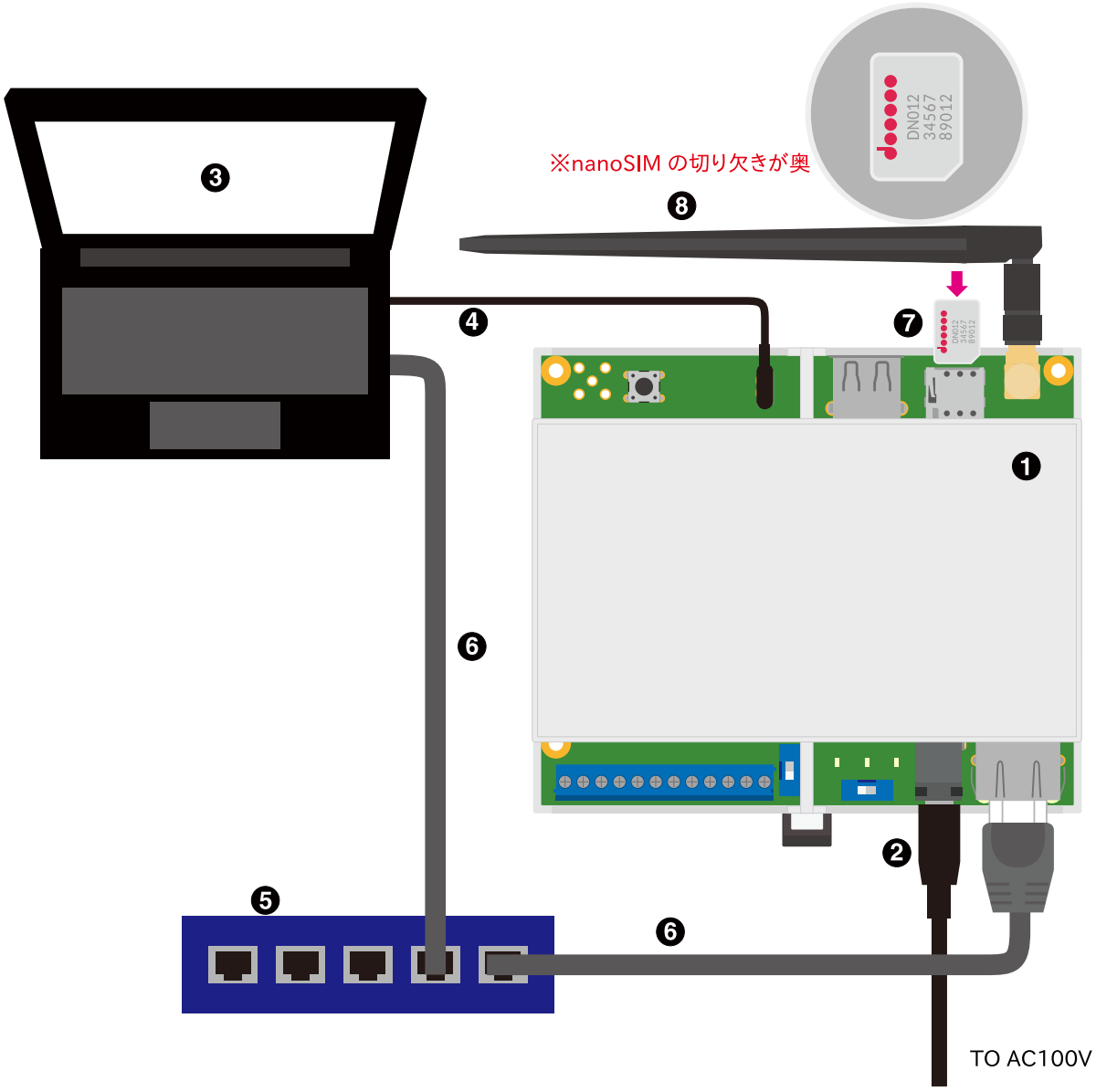


図 4.12 Armadillo-IoT ゲートウェイ A6E の接続例

1. Armadillo-IoT ゲートウェイ A6E
2. AC アダプタ(12V/2A)
3. 作業用 PC
4. シリアル通信用 USB ケーブル(A-microB)
5. LAN HUB
6. Ethernet ケーブル
7. nanoSIM カード

❽ LTE 用外付けアンテナ

4.5. 起動デバイス設定スイッチについて

起動デバイス設定スイッチを操作することで、起動デバイスを設定することができます。

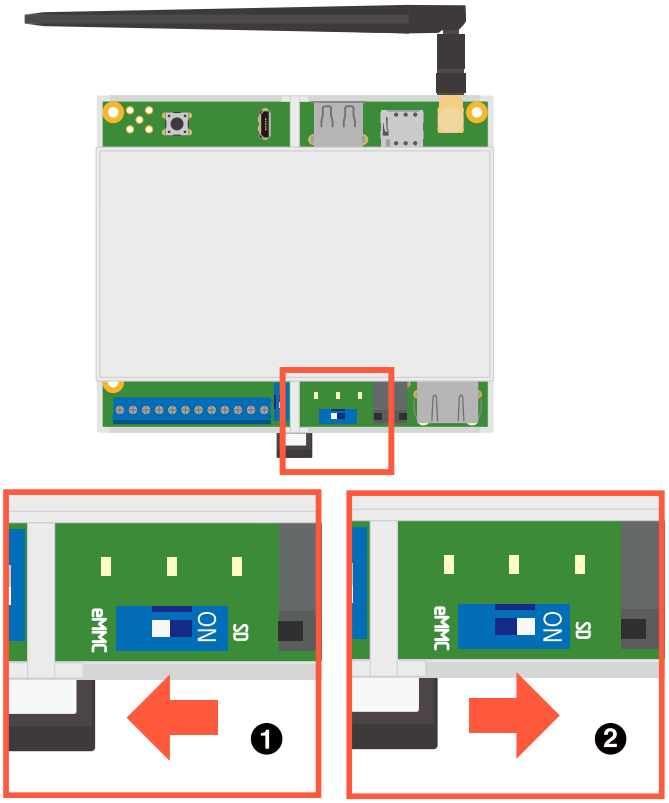


図 4.13 起動デバイス設定スイッチの操作

❶ 起動デバイスは eMMC になります。

❷ 起動デバイスは microSD になります。

起動デバイス設定スイッチの両脇の基板上に、白い文字で eMMC/SD とシルク記載しているので、操作の目印にご利用ください。

4.6. vi エディタの使用方法

vi エディタは、Armadillo に標準でインストールされているテキストエディタです。本書では、Armadillo の設定ファイルの編集などに vi エディタを使用します。

vi エディタは、ATDE にインストールされてる gedit や emacs などのテキストエディタとは異なり、モードを持っていることが大きな特徴です。vi のモードには、コマンドモードと入力モードがあります。

コマンドモードの時に入力した文字はすべてコマンドとして扱われます。入力モードでは文字の入力ができます。

本章で示すコマンド例は ATDE で実行するよう記載していますが、Armadillo でも同じように実行することができます。

4.6.1. vi の起動

vi を起動するには、以下のコマンドを入力します。

```
[ATDE ~]# vi [file]
```

図 4.14 vi の起動

`file` にファイル名のパスを指定すると、ファイルの編集(`file` が存在しない場合は新規作成)を行います。vi はコマンドモードの状態で起動します。

4.6.2. 文字の入力

文字を入力するにはコマンドモードから入力モードへ移行する必要があります。コマンドモードから入力モードに移行するには、「表 4.5. 入力モードに移行するコマンド」に示すコマンドを入力します。入力モードへ移行後は、キーを入力すればそのまま文字が入力されます。

表 4.5 入力モードに移行するコマンド

コマンド	動作
i	カーソルのある場所から文字入力を開始
a	カーソルの後ろから文字入力を開始

入力モードからコマンドモードに戻りたい場合は、ESC キーを入力することで戻ることができます。現在のモードが分からなくなった場合は、ESC キーを入力し、一旦コマンドモードへ戻ることにより混乱を防げます。

日本語変換機能を OFF に

vi のコマンドを入力する時は ATDE の日本語入力システム(Mozc)を OFF にしてください。日本語入力システムの ON/OFF は、半角/全角キーで行うことができます。

「i」、「a」それぞれのコマンドを入力した場合の文字入力の開始位置を「図 4.15. 入力モードに移行するコマンドの説明」に示します。

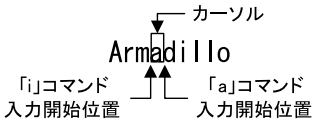


図 4.15 入力モードに移行するコマンドの説明

vi での文字削除

> コンソールの環境によっては BS(Backspace)キーで文字が削除できず、「^H」文字が入力される場合があります。その場合は、「4.6.4. 文字の削除」で説明するコマンドを使用し、文字を削除してください。

4.6.3. カーソルの移動

方向キーでカーソルの移動ができますが、コマンドモードで「表 4.6. カーソルの移動コマンド」に示すコマンドを入力することでもカーソルを移動することができます。

表 4.6 カーソルの移動コマンド

コマンド	動作
h	左に 1 文字移動
j	下に 1 文字移動
k	上に 1 文字移動
l	右に 1 文字移動

4.6.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.7. 文字の削除コマンド」に示すコマンドを入力します。

表 4.7 文字の削除コマンド

コマンド	動作
x	カーソル上の文字を削除
dd	現在行を削除

「x」コマンド、「dd」コマンドを入力した場合に削除される文字を「図 4.16. 文字を削除するコマンドの説明」に示します。

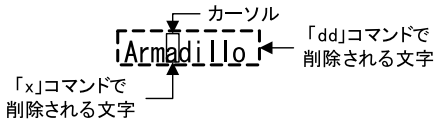


図 4.16 文字を削除するコマンドの説明

4.6.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.8. 保存・終了コマンド」に示します。

表 4.8 保存・終了コマンド

コマンド	動作
`:q!`	変更を保存せずに終了
`:w[file]`	ファイルを `file` に指定して保存
`:wq`	ファイルを上書き保存して終了

保存と終了を行うコマンドは「：」(コロン)からはじまるコマンドを使用します。"："キーを入力すると画面下部にカーソルが移り入力したコマンドが表示されます。コマンドを入力した後 Enter キーを押すことで、コマンドが実行されます。

5. 起動と終了

5.1. 起動

電源入力インターフェースに電源を接続すると Armadillo-IoT ゲートウェイ A6E が起動します。起動すると CON7 (USB コンソールインターフェース) から起動ログが表示されます。

Armadillo-IoT ゲートウェイ A6E の電源投入時点での起動デバイス設定スイッチ SW2 の状態によって起動モードが変化します。 詳しくは 「4.5. 起動デバイス設定スイッチについて」 を参照してください。

以下に起動ログの例を示します。

```
U-Boot 2020.04-at10(Oct 04 2022 - 11:22:32 +0900)

CPU:   i.MX6ULL rev1.1 at 396 MHz
Model: Atmark Techno Armadillo-IoT Gateway A6E
DRAM:  512 MiB
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
Saving Environment to MMC... Writing to MMC(1)... OK
switch to partitions #0, OK
mmc1 is current device
Net:
Warning: ethernet@2188000 using MAC address from ROM
eth0: ethernet@2188000
Normal Boot
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1 is current device
11659840 bytes read in 518 ms (21.5 MiB/s)
Booting from mmc ...
38603 bytes read in 21 ms (1.8 MiB/s)
Loading fdt boot/armadillo.dtb
43 bytes read in 14 ms (2.9 KiB/s)
1789 bytes read in 18 ms (96.7 KiB/s)
Applying fdt overlay: armadillo-iotg-a6e-ems31.dtbo
## Booting kernel from Legacy Image at 80800000 ...
   Image Name:   Linux-5.10.145-32-at
   Created:      2022-10-13   8:10:47 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    11659776 Bytes = 11.1 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83500000
```

```
   Booting using the fdt blob at 0x83500000
   Loading Kernel Image
   Loading Device Tree to 9ef2d000, end 9ef59fff ... OK

Starting kernel ...

[    0.601992] imx6ul-pinctrl 2290000.iomuxc-snvs: no groups defined in /soc/bus@2200000/iomuxc-
snvs@2290000

   OpenRC 0.44.10 is starting up Linux 5.10.145-32-at (armv7l)

 * Mounting /proc ... [ ok ]
 * Mounting /run ... * /run/openrc: creating directory
 * /run/lock: creating directory
 * /run/lock: correcting owner
 * Caching service dependencies ... [ ok ]
 * Starting atmark firstboot script ... * Mounting /sys ... * Remounting devtmpfs on /dev ... *
Starting rngd ... [ ok ]
 [ ok ]
 * Mounting security filesystem ... [ ok ]
 [ ok ]
 * Mounting config filesystem ... [ ok ]
 * Mounting /dev/mqueue ... * Mounting fuse control filesystem ... [ ok ]
 [ ok ]
 * Mounting /dev/pts ... [ ok ]
 * Mounting /dev/shm ... [ ok ]
udev                    | * Starting udev ... [ ok ]
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
Could not create partition 2 from 0 to 614399
Could not create partition 3 from 0 to 102399
Could not create partition 4 from 0 to 409599
Could not create partition 5 from 0 to 20479
Error encountered; not saving changes.
 * partitionning disk failed
fsck                    | * Checking local filesystems  ... [ ok ]
root                    | * Remounting filesystems ... [ ok ]
localmount              | * Mounting local filesystems ... [ ok ]
overlayfs               | * Preparing overlayfs over / ... [ ok ]
hostname                | * Setting hostname ... [ ok ]
sysctl                  | * Configuring kernel parameters ...udev-trigger
       | * Generating a rule to create a /dev/root symlink ... [ ok ]
 [ ok ]
udev-trigger            | * Populating /dev with existing devices through uevents ... [ ok ]
bootmisc                | * Migrating /var/lock to /run/lock ... [ ok ]
bootmisc                | * Creating user login records ... [ ok ]
bootmisc                | * Wiping /var/tmp directory ... [ ok ]
syslog                  | * Starting busybox syslog ...dbus
 | * /run/dbus: creating directory
 [ ok ]
dbus                    | * /run/dbus: correcting owner
dbus                    | * Starting System Message Bus ... [ ok ]
klogd                   | * Starting busybox klogd ... [ ok ]
networkmanager          | * Starting networkmanager ... [ ok ]
dnsmasq                 | * /var/lib/misc/dnsmasq.leases: creating file
dnsmasq                 | * /var/lib/misc/dnsmasq.leases: correcting owner
```

```
dnsmasq                 | * Starting dnsmasq ... [ ok ]
buttond                 | * Starting button watching daemon ... [ ok ]
reset_bootcount         | * Resetting bootcount in bootloader env ...Environment OK, copy 0
reset_bootcount         | [ ok ]
zramswap                | [ ok ]
zramswap                | * Creating zram swap device ...podman-atmark
   | * Starting configured podman containers ... [ ok ]
atmark-power-utils      | * Starting atmark-power-utils ... [ ok ]
chronyd                 | * Starting chronyd ... [ ok ]
 [ ok ]
local                   | * Starting local ... [ ok ]

Welcome to Alpine Linux 3.16
Kernel 5.10.145-32-at on an armv7l (/dev/ttymxc2)

armadillo login:
```

U-Boot プロンプト

ユーザースイッチ(SW1) を押しながら電源を投入すると、 U-Boot のプロンプトが表示されます。

```
U-Boot 2020.04-at10 (Oct 04 2022 - 11:22:32 +0900)

CPU:   i.MX6ULL rev1.1 at 396 MHz
Model: Atmark Techno Armadillo-IoT Gateway A6E
DRAM:  512 MiB
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
Saving Environment to MMC... Writing to MMC(1)... OK
switch to partitions #0, OK
mmc1 is current device
Net:   eth0: ethernet@2188000
Normal Boot
=>
```

5.2. ログイン

起動が完了するとログインプロンプトが表示されます。「root」か一般ユーザーの「atmark」でログインすることができます。

initial_setup.swu を適用しない場合、「root」ユーザーは初回ログイン時にパスワードを入力せずに新しいパスワードを促されます。「atmark」ユーザーは、初期状態ではロックされています。そのロックを解除するには、「root」ユーザーでログインし、 passwd atmark コマンドで「atmark」ユーザーのパスワードを設定してください。

設定するパスワードには大文字のアルファベット、小文字のアルファベット、0 から 9 までの数字、その他(記号・句読点など)を含める事ができます。

1. root でログイン

   初期パスワードを変更します。

```
armadillo login: root
You are required to change your password immediately (administrator enforced).
New password: ❶
Retype new password: ❷
Welcome to Alpine!
```

❶ 新しいパスワードを入力します

❷ 新しいパスワードを再入力します

2. atmark でログイン

初期状態でロックされてますので、root で一度パスワードを設定してからログインします。

```
armadillo:~# passwd atmark ❶
New password:
Retype new password:
passwd: password updated successfully
armadillo:~# persist_file /etc/shadow ❷
armadillo:~# exit

Welcome to Alpine Linux 3.16
Kernel 5.10.126-24-at on an armv7l (/dev/ttymxc2)

armadillo login: atmark
Password: ❸
Welcome to Alpine!
```

❶ atmark ユーザーのパスワード変更コマンド。「10.9.2. SWU イメージの作成」 を使用した場合には不要です

❷ パスワードファイルを永続化します。

❸ 設定したパスワードでログインすることができます。

Armadillo BaseOS ではルートファイルシステムに overlayfs を採用しており、そのままではシステムが OFF すると内容は消えてしまいます。そのため persist_file コマンドが用意されています。このコマンドを利用することでファイル単位で変更を反映することができます。パスワードを設定した後には以下のコマンドを実行してください。

```
[armadillo ~]# persist_file /etc/shadow
```

persist_file コマンドに関する詳細は「10.10.2. overlayfs と persist_file について」を参照してください。

5.3. 終了方法

安全に終了させる場合は、次のように poweroff コマンドを実行し、「reboot: Power down」と表示されたのを確認してから電源を切断します。

```
armadillo:~# poweroff
armadillo:~# zramswap                | * Deactivating zram swap device ...podman-atmark
| * Stopping all podman containers ...local
   | * Stopping local ... [ ok ]
 [ ok ]
atmark-power-utils       | * Stopping atmark-power-utils ...rngd
     | * Stopping rngd ...chronyd                 | * Stopping chronyd ...dnsmasq
| * Stopping dnsmasq ...buttond                 | * Stopping button watching daemon ... [ ok ]
 [ ok ]
 * start-stop-daemon: no matching processes found
 [ ok ]
 [ ok ]
atmark-power-utils       | [ ok ]
klogd                    | * Stopping busybox klogd ... [ ok ]
networkmanager           | * Stopping networkmanager ... [ ok ]
syslog                   | * Stopping busybox syslog ... [ ok ]
udev                     | * Stopping udev ... [ ok ]
dbus                   | * Stopping System Message Bus ...nm-dispatcher: Caught signal 15, shutting
down...
 [ ok ]
cgroups                  | * cgroups: waiting for podman-atmark (50 seconds)
 [ ok ]
localmount               | * Unmounting loop devices
localmount               | * Unmounting filesystems
localmount               | *   Unmounting /var/tmp ... [ ok ]
localmount               | *   Unmounting /var/app/volumes ... [ ok ]
localmount               | *   Unmounting /var/app/rollback/volumes ... [ ok ]
localmount               | *   Unmounting /var/lib/containers/storage_readonly ... [ ok ]
localmount               | *   Unmounting /var/log ... [ ok ]
localmount               | *   Unmounting /tmp ... [ ok ]
killprocs                | * Terminating remaining processes ...mount-ro
         | * Remounting remaining filesystems read-only ... *   Remounting / read only ... [ ok ]
mount-ro                 | [ ok ]
indicator_signals        | * Signaling external devices we are shutting down ... [ ok ]
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processes
Requesting system poweroff
[   99.211013] reboot: Power down
```

電源を再投入する際は、コンデンサに蓄えられた電荷を抜くため、電源を切断後、一定時間以上待つ必要があります。開発セット付属の AC アダプタの場合に必要な時間は以下のとおりです。

- DC プラグ側で電源を切断した場合：約 5 秒
- AC プラグ側で電源を切断した場合：約 1 分

コンデンサに蓄えられた電荷が抜ける前に電源を再投入した場合、電源シーケンスが守られず、起動しない等の動作不具合の原因となります。

6. ユーザー登録

アットマークテクノ製品をご利用のユーザーに対して、購入者向けの限定公開データの提供や大切なお知らせをお届けするサービスなど、ユーザー登録すると様々なサービスを受けることができます。サービスを受けるためには、「アットマークテクノ Armadillo サイト」にユーザー登録をする必要があります。

ユーザー登録すると次のようなサービスを受けることができます。

- 製品仕様や部品などの変更通知の閲覧・配信
- 購入者向けの限定公開データのダウンロード
- 該当製品のバージョンアップに伴う優待販売のお知らせ配信
- 該当製品に関する開発セミナーやイベント等のお知らせ配信

詳しくは、「アットマークテクノ Armadillo サイト」をご覧ください。

アットマークテクノ Armadillo サイト

https://armadillo.atmark-techno.com/

6.1. 購入製品登録

ユーザー登録完了後に、購入製品登録することで、「購入者向けの限定公開データ」をダウンロードすることができるようになります。

購入製品登録の詳しい手順は以下の URL をご参照ください。

Armadillo-IoT A6E 購入製品登録

https://armadillo.atmark-techno.com/armadillo-iot-a6e/register

7. 動作確認方法

本章では、ハードウェアの動作確認に使用するコマンドやその実行手順について説明します。

ハードウェアの動作確認以外が目的のコマンドや手順については「10. Howto」を参照してください。

7.1. ネットワーク

ここでは、ネットワークの設定方法について説明します。

7.1.1. 接続可能なネットワーク

Armadillo-IoT ゲートウェイ A6E は、Ethernet ポートと WLAN+BT コンボモジュールが搭載されています。Cat.M1 モデルには LTE モデムが搭載されています。Linux からは、それぞれ `eth0` 、 `ppp0` 、 `wlan0` に見えます。

表 7.1 ネットワークとネットワークデバイス

ネットワーク	ネットワークデバイス	出荷時の設定
Ethernet	`eth0`	DHCP
LTE (Cat.M1 モデルのみ)	`ppp0`	SIM / 料金プランに依存します
無線 LAN	`wlan0`	クライアントモード

7.1.2. ネットワークの設定方法

Armadillo-IoT ゲートウェイ A6E では、通常の Linux システムと同様、ネットワークインターフェースの設定は NetworkManager を使用します。NetworkManager はすべてのネットワーク設定をコネクションとして管理します。コネクションには「どのようにネットワークへ接続するか」、「どのようにネットワークを作成するか」を記述し、 `/etc/NetworkManager/system-connections/` に保存します。また、1 つのデバイスに対して複数のコネクションを保存することは可能ですが、1 つのデバイスに対して有効化にできるコネクションは 1 つだけです。

NetworkManager は、従来の `/etc/network/interfaces` を使った設定方法もサポートしていますが、本書では `nmcli` を用いた方法を中心に紹介します。

7.1.2.1. nmcli について

`nmcli` は NetworkManager を操作するためのコマンドラインツールです。「図 7.1. nmcli のコマンド書式」に `nmcli` の書式を示します。このことから、 `nmcli` は「オブジェクト (OBJECT) というものが存在し、それぞれのオブジェクトに対してコマンド (COMMAND) を実行する。」という書式でコマンドを入力することがわかります。また、オブジェクトそれぞれに `help` が用意されていることもここから読み取れます。

```
nmcli [ OPTIONS ] OBJECT { COMMAND | help }
```

図 7.1 nmcli のコマンド書式

7.1.3. nmcli の基本的な使い方

ここでは `nmcli` の、基本的な使い方を説明します。

7.1.3.1. コネクションの一覧表示

登録されているコネクションの一覧表示するには、「図 7.2. コネクションの一覧表示」に示すコマンドを実行します。[1]

```
[armadillo ~]# nmcli connection
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx  ethernet  eth0
```

図 7.2 コネクションの一覧表示

表示された NAME については、以降 [ID] として利用することができます。

7.1.3.2. コネクションの有効化・無効化

コネクションを有効化するには、「図 7.3. コネクションの有効化」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection up [ID]
```

図 7.3 コネクションの有効化

コネクションを無効化するには、「図 7.4. コネクションの無効化」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection down [ID]
```

図 7.4 コネクションの無効化

7.1.3.3. コネクションの作成

コネクションを作成するには、「図 7.5. コネクションの作成」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection add con-name [ID] type [type] ifname [interface name]
```

図 7.5 コネクションの作成

[ID] にはコネクションの名前(任意)、[type] には ethernet、wifi といった接続タイプ、[interfacename] にはインターフェース名(デバイス)を入力します。これにより /etc/NetworkManager/system-connections/ に[ID]の名前でコネクションファイルが作成されます。このファイルを vi などで編集し、コネクションを修正することも可能です。

Armadillo-IoT ゲートウェイ A6E を再起動したときにコネクションファイルが消えてしまわないように、persist_file コマンドで永続化する必要があります。persist_file コマンドに関する詳細は「10.10.2. overlayfs と persist_file について」 を参照してください。

```
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/<コネクションファイル名>
```

図 7.6 コネクションファイルの永続化

[1] nmcli connection show [ID] によって、より詳細な情報を表示することもできます。

別の Armadillo-IoT ゲートウェイ A6E からコネクションファイルをコピーした場合は、コネクションファイルのパーミッションを 600 に設定してください。600 に設定後、 nmcli c reload コマンドでコネクションファイルを再読込します。

```
[armadillo ~]# chmod 600 /etc/NetworkManager/system-connections/<コネクションファイル名>
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/<コネクションファイル名>
[armadillo ~]# nmcli c reload
```

swu イメージを使用してコネクションファイルのアップデートを行う場合は、swu イメージに含めるコネクションファイルのパーミッションを 600 に設定してから、swu イメージを作成してください。アップデート実行時には swu イメージ作成時のパーミッションが維持されるため、上記のコマンド実行手順は不要です。swu イメージに関しては 「10.9. Armadillo のソフトウェアをアップデートする」 を参考にしてください。

7.1.3.4. コネクションの削除

コネクションを削除するには、「図 7.7. コネクションの削除」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection delete [ID]
```

図 7.7 コネクションの削除

これにより /etc/NetworkManager/system-connections/ のコネクションファイルも同時に削除されます。コネクションの作成と同様に persist_file コマンドで永続化する必要があります。

```
[armadillo ~]# persist_file -d /etc/NetworkManager/system-connections/<コネクションファイル名>
```

図 7.8 コネクションファイル削除時の永続化

7.1.3.5. 固定 IP アドレスに設定する

「表 7.2. 固定 IP アドレス設定例」の内容に設定する例を、「図 7.9. 固定 IP アドレス設定」に示します。

表 7.2 固定 IP アドレス設定例

項目	設定
IP アドレス	192.0.2.10
マスク長	24
デフォルトゲートウェイ	192.0.2.1

```
[armadillo ~]# nmcli connection modify [ID] ¥
ipv4.method manual ipv4.addresses 192.0.2.10/24 ipv4.gateway 192.0.2.1
```

図 7.9 固定 IP アドレス設定

7.1.3.6. DHCP に設定する

DHCP に設定する例を、「図 7.10. DNS サーバーの指定」に示します。

```
[armadillo ~]# nmcli connection modify [ID] ipv4.method auto
```

図 7.10 DNS サーバーの指定

-ipv4.addresses のように、プロパティ名の先頭に "-" を付けることで設定したプロパティを削除することができます。反対に "+" を付けることでプロパティを追加することができます。

7.1.3.7. DNS サーバーを指定する

DNS サーバーを指定する例を、「図 7.11. DNS サーバーの指定」に示します。

```
[armadillo ~]# nmcli connection modify [ID] ipv4.dns 192.0.2.1
```

図 7.11 DNS サーバーの指定

7.1.3.8. コネクションの修正を反映する

有効化されているコネクションを修正した場合、かならず修正したコネクションを再度有効化してください。

```
[armadillo ~]# nmcli connection down [ID]
[armadillo ~]# nmcli connection up [ID]
```

図 7.12 コネクションの修正の反映

7.1.3.9. デバイスの一覧表示

デバイスの一覧(デバイス名、タイプ、状態、有効なコネクション)を確認するには、「図 7.13. デバイスの一覧表示」に示すコマンドを実行します。

```
[armadillo ~]# nmcli device
DEVICE  TYPE      STATE        CONNECTION
eth0    ethernet  connected    Wired connection 1
lo      loopback  unmanaged    --
```

図 7.13 デバイスの一覧表示

7.1.3.10. デバイスの接続

デバイスを接続するには、「図 7.14. デバイスの接続」に示すコマンドを実行します。

```
[armadillo ~]# nmcli device connect [ifname]
```

図 7.14 デバイスの接続

デバイスを接続するには、接続しようとしているデバイスの有効なコネクションが必要です。"Error: neither a valid connection nor device given" というメッセージが表示された場合には、`nmcli connection` などで有効なコネクションが存在するかを確認してください。

7.1.3.11. デバイスの切断

デバイスを切断するには、「図 7.15. デバイスの切断」に示すコマンドを実行します。

```
[armadillo ~]# nmcli device disconnect [ifname]
```

図 7.15 デバイスの切断

7.1.4. 有線 LAN の接続を確認する

有線 LAN で正常に通信が可能かを確認します。設定を変更した場合、必ず変更したインターフェースを再度有効化してください。

同じネットワーク内にある通信機器と PING 通信を行います。以下の例では、通信機器が「192.0.2.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping -c 3 192.0.2.20
PING 192.0.2.20 (192.0.2.20): 56 data bytes
64 bytes from 192.0.2.20: seq=0 ttl=64 time=3.056 ms
64 bytes from 192.0.2.20: seq=1 ttl=64 time=1.643 ms
64 bytes from 192.0.2.20: seq=2 ttl=64 time=1.633 ms

--- 192.0.2.20 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1.633/2.110/3.056 ms
```

図 7.16 有線 LAN の PING 確認

有線 LAN 以外のインターフェースが有効化されている場合、ルーティングの設定などにより、ネットワーク通信に有線 LAN が使用されない場合があります。確実に有線 LAN の接続確認をするために、有線 LAN 以外のインターフェースを無効化してください。

7.1.5. LTE (Cat.M1 モデルのみ)

本章では、Armadillo-IoT ゲートウェイ A6E に搭載されている LTE モジュールの使用方法について説明します。

Thales 製 LTE 通信モジュール EMS31-J はドコモ/KDDI/ソフトバンクそれぞれの相互接続性試験を完了しています。

7.1.5.1. LTE データ通信設定を行う前に

LTE データ通信を利用するには、通信事業者との契約が必要です。契約時に通信事業者から貸与された nanoSIM(UIM カード)と APN 情報を準備します。

Thales 製 EMS31-J 搭載モデルでの動作検証済み nanoSIM (料金プラン)に関しては、 Armadillo サイトの「Armadillo-IoT ゲートウェイ 動作確認済み SIM 一覧」を確認ください。

Armadillo-IoT ゲートウェイ 動作確認済み SIM 一覧 [https://armadillo.atmark-techno.com/howto/armadillo-iot-tested-sim]

Armadillo-IoT ゲートウェイ A6E の電源が切断されていることを確認してから nanoSIM(UIM カード)を取り付けてください。

本製品は、nanoSIM スロットを搭載しています。

標準/microSIM サイズの SIM カードを nanoSIM サイズにカットしたもの、サイズの異なるものを使用すると、nanoSIM スロットが故障する原因となります。これらを使用し本製品が故障した場合は、保証期間内であっても保証適用外となります。

nanoSIM(UIM カード)の切り欠きを挿入方向に向け、刻印面を上にして挿入してください。挿入位置などは、「図 4.12. Armadillo-IoT ゲートウェイ A6E の接続例」 を参照してください。

APN の設定を行うには、次に示す情報が必要です。() 内は Cat.M1 モデルでの設定可能な文字長です。この文字長を超える設定はできませんので、SIM の料金プランを選択する際にはご注意ください。

- APN(最大 99 文字)
- ユーザー名(最大 64 文字)
- パスワード(最大 64 文字)

・ 認証方式(PAP または CHAP)

・ PDP Type(IP のみをサポート)

7.1.5.2. LTE モデム EMS31-J 省電力などの設定

LTE モデム EMS31-J 起動時に設定する内容を、/etc/atmark/ems31-boot.conf ファイルに記載します。

/etc/atmark/ems31-boot.conf に設定できる内容を「表 7.3. ems31-boot.conf の設定内容」に示します。

ems31-boot.conf のフォーマットは以下の通りです。

・ パラメータは、「パラメータ名=値」のフォーマットで記載してください。

・ fix_profile の値のみダブルクォテーションで囲む必要があります。

・ 行頭に # が存在する場合、その行を無視します。

・ パラメーターが存在しない場合、その項目に関して何も設定をしません。

表 7.3 ems31-boot.conf の設定内容

パラメーター名	初期値	設定可能値	説明
fix_profile	"auto"	"docomojp","sbmjp","kddijp"	接続プロファイルの指定"auto" で接続できないときに、設定を変更すると接続できることがあります。
suspend	disable	enable または disable	サスペンドの有効無効
psm	3m,1m	disable または tau,act-time	Power Save Mode の設定
edrx	20.48,5.12	disable または pcl,ptw	eDRX の設定

PSM (Power Save Mode) の設定値を「表 7.4. psm の tau と act-time に設定可能な値」に示します。disable にしない場合、tau (Periodic TAU cycle (T3412)) は act_time (Active time (T3324)) より大きい値にする必要があります。

表 7.4 psm の tau と act-time に設定可能な値

パラメーター名	設定可能値
tau (s=秒,m=分,h=時間)	2s,4s,6s…62s,90s,120s,150s…930s,1m,2m,3m…31m,40m,50m,60m…310m, 1h,2h, 3h…31h,40h,50h,60h…310h
act-time (s=秒,m=分,h=時間)	2s,4s,6s…62s,1m,2m,3m…31m,36m,42m,48m…186m

eDRX (extended Discontinuous Reception) の設定値を「表 7.5. edrx の pcl と ptw に設定可能な値」に示します。disable にしない場合、pcl (Paging Cycle Length) は ptw (Paging Time Window eDRX) より大きい値にする必要があります。

表 7.5 edrx の pcl と ptw に設定可能な値

パラメーター名	設定可能値
pcl (秒)	5.12, 10.24, 20.48, 40.96, 61.44, 81.92, 102.4, 122.88, 143.36, 163.84, 327.68, 655.36, 1310.72, 2621.44
ptw (秒)	1.28, 2.56, 5.12, 6.40, 7.68, 8.96, 10.24, 11.52, 12.80, 14.08, 15.36, 16.64, 17.92, 19.20, 20.48

7.1.5.3. LTE のコネクションを作成する

「表 7.6. APN 情報設定例」の内容に設定する例を「図 7.17. LTE のコネクションの作成」に示します。

表 7.6 APN 情報設定例

項目	設定
APN	[apn]
ユーザー名	[user]
パスワード	[password]
ネットワークデバイス	[wwan]

ネットワークデバイス [wwan] は、「表 7.7. 通信モジュールのネットワークデバイス」 を参照ください。

表 7.7 通信モジュールのネットワークデバイス

通信モジュール	ネットワークデバイス
Thales 製 EMS31-J	ttyCommModem

```
[armadillo ~]# nmcli connection add type gsm ifname [wwan] apn [apn] user [user] password [password]
Connection 'gsm-[wwan]' (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx) successfully added.
```

図 7.17 LTE のコネクションの作成

コネクション設定を永続化するには、以下のコマンドを入力してください。設定を永続化すると、Armadillo 起動時に自動的にデータ接続を行うようになります。

同一インタフェースへの設定が複数存在する場合、 gsm-[wwan]-1.nmconnection など後ろに数値が付与されますので、「図 7.17. LTE のコネクションの作成」 入力時のメッセージで生成されたファイル名を確認した上で永続化を実施ください。

```
[armadillo ~]# persist_file /etc/NetworkManager/system-connections/gsm-[wwan].nmconnection
```

図 7.18 LTE のコネクションの設定の永続化

7.1.5.4. MCC/MNC を指定した LTE のコネクションを作成する

マルチキャリア SIM などを使用する際、MCC (Mobile Country Code) と MNC (Mobile Network Code) を指定してコネクションを作成すると LTE ネットワークに接続できることがあります。指定する場合は 「図 7.19. MCC/MNC を指定した LTE コネクションの作成」 に示すコマンドを実行してください。

[mccmnc] には 44010 などの数字を入力してください。実際に設定する値に関しては、ご契約の通信事業者へお問い合わせください。

```
[armadillo ~]# nmcli connection add type gsm ifname [wwan] apn [apn] user [user] password [password] ⏎
gsm.network-id [mccmnc]
```

図 7.19 MCC/MNC を指定した LTE コネクションの作成

7.1.5.5. PAP 認証を有効にした LTE のコネクションを作成する

LTE のコネクションの認証方式は、デフォルトで CHAP に設定されています。PAP 認証を有効にしたコネクションを作成する場合は「図 7.20. PAP 認証を有効にした LTE コネクションの作成」 に示すコマンドを実行してください。

```
[armadillo ~]# nmcli connection add type gsm ifname [wwan] apn [apn] user [user] password [password]
ppp.refuse-eap true ppp.refuse-chap true ppp.refuse-mschap true ppp.refuse-mschapv2 true
ppp.refuse-pap false
```

図 7.20 PAP 認証を有効にした LTE コネクションの作成

すでに LTE コネクションを作成済みの場合はコネクション設定を削除した後に、「図 7.20. PAP 認証を有効にした LTE コネクションの作成」を実施してください。

7.1.5.6. LTE コネクションを確立する

LTE コネクションの作成直後や設定変更後に再起動をせずにコネクションを確立するには、「図 7.21. LTE のコネクション確立」に示すコマンドを実行します。

```
[armadillo ~]# nmcli connection up gsm-[wwan]
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/x)
```

図 7.21 LTE のコネクション確立

7.1.5.7. LTE の接続を確認する

LTE で正常に通信が可能かを確認します。

「図 7.22. LTE の PING 導通確認」に示すコマンドで、アットマークテクノの Web サーバーと PING 通信を行います。VPN 接続を利用するなどインターネットに接続できない場合は、ネットワーク内の通信機器に読み替えてください。

```
[armadillo ~]# ping www.atmark-techno.com
```

図 7.22 LTE の PING 導通確認

LTE 以外のコネクションが有効化されている場合、ネットワーク通信に LTE が使用されない場合があります。確実に LTE の接続確認をする場合は、事前に LTE 以外のコネクションを無効化してください。

7.1.5.8. LTE コネクションを切断する

LTE コネクションを切断するには、「図 7.23. LTE コネクションを切断する」に示すコマンドを実行します。LTE コネクションを切断する前に、LTE 再接続サービスを停止しないと再接続処理が実行される為、事前に停止します。

```
[armadillo ~]# rc-service connection-recover stop ❶
connection-recover| * Stopping connection-recover ... [ ok ]
[armadillo ~]# nmcli connection down gsm-[wwan] ❷
```

図 7.23 LTE コネクションを切断する

❶ LTE 再接続サービスを停止します。

❷ LTE コネクションを切断します。

7.1.5.9. LTE のコネクション設定を編集する場合の注意事項

LTE の設定情報を **nmcli connection modify** コマンドで編集する場合、パスワード情報がリセットされます。「図 7.24. nmcli connection modify コマンドで LTE のパスワードを設定する」に示すコマンドを実行し、都度パスワードを再設定してください。

```
[armadillo ~]# nmcli connection modify gsm-[wwan] gsm.password [password]
```

図 7.24 nmcli connection modify コマンドで LTE のパスワードを設定する

7.1.5.10. LTE 再接続サービス

LTE 再接続サービスは、LTE のデータ接続の状態を定期的に監視し、切断を検出した場合に再接続を行うサービスです。

Cat.M1 モデルでは、LTE モデムの省電力動作のため、初期状態では LTE 再接続サービスを無効にしております。有効にする手順は、「図 7.30. LTE 再接続サービスを有効にする」を参照ください。LTE 再接続サービスを有効にした場合、定期的に ping 導通確認を実施するため、スリープ状態の LTE モデムが都度起床する、サスペンド状態の LTE モデムですと ping 導通が確認できないなど、制約が発生しますので、その辺りを考慮された上でのご利用をお願いします。

SIM カードが挿入されており、NetworkManager に有効な LTE コネクションの設定がされているとき、初期設定では 120 秒に一度コネクションの状態を監視します。オプションで SIM カードの認識ができないときに Armadillo の再起動を実施することも可能です。

コネクションが無効になっている場合、切断状態と判定しコネクションを有効にします。

コネクションが有効になっている場合、特定の宛先に PING を実行します。PING がエラーになったとき切断状態と判定し、コネクションの無効化・有効化を行うことで再接続を実施します。

コネクションの無効化・有効化による再接続を実施しても PING がエラーになる場合、電波のオン・オフまたは LTE モジュールの電源をオン・オフを実施して再接続を実施します。どちらを実施するかは設定ファイルの WWAN_FORCE_RESTART_COUNT に依存します。

WWAN_FORCE_RESTART_COUNT が初期値の 10 である場合、1 から 9 回目は電波のオン・オフを実施し、10 回目は LTE モジュールの電源オン・オフを実施します。それ以降も NG が続く場合、同じく 10 回に一度 LTE モジュールの電源オン・オフを実施します。

工場出荷状態で本サービスは有効化されており、システム起動時にサービスが自動的に開始されます。PING を実行する宛先は、初期設定ではで "8.8.8.8" です。ご利用の環境に合わせて設定ファイル(/etc/atmark/connection-recover/gsm-ttyMux0_connection-recover.conf)を適宜変更してください。

設定ファイルの概要を「表 7.8. 再接続サービス設定パラメーター」に示します。必要に応じて設定値を変更してください。

表 7.8 再接続サービス設定パラメーター

パラメーター名	初期値	意味	変更
PRODUCT_NAME	-	製品名	不可
CHECK_INTERVAL_SEC	120	監視周期(秒)	可
PING_DEST_IP	8.8.8.8	コネクション状態確認時 PING 送付先	可
DEVICE	-	ネットワークデバイス名	不可
TYPE	-	ネットワークタイプ	不可
NETWORK_IF	-	ネットワーク I/F 名	不可
FORCE_REBOOT	FALSE	TRUE に設定すると PING 導通チェック NG 時 Armadillo を再起動します。	可
REBOOT_IF_SIM_NOT_FOUND	FALSE	TRUE に設定すると SIM を検出できない時に Armadillo を再起動します。	可
WWAN_FORCE_RESTART_COUNT	10	PING 導通確認を設定した回数連続で失敗した場合モデムの再起動を実行します。設定した回数に満たない場合 、電波のオフ・オン実施のみで LTE 再接続を試みます。	可

設定ファイル(/etc/atmark/connection-recover/gsm-ttyMux0_connection-recover.conf)変更後、変更内容を永続化するには「図 7.25. LTE 再接続サービスの設定値を永続化する」に示すコマンドを実行してください。

```
[armadillo ~]# persist_file /etc/atmark/connection-recover/gsm-ttyMux0_connection-recover.conf
```

図 7.25 LTE 再接続サービスの設定値を永続化する

LTE 再接続サービスの状態を確認するには、「図 7.26. LTE 再接続サービスの状態を確認する」に示すコマンドを実行してください。

```
[armadillo ~]# rc-status | grep connection-recover
 connection-recover                                  [  started 00:43:02 (0) ]
```

図 7.26 LTE 再接続サービスの状態を確認する

LTE 再接続サービスを停止するには、「図 7.27. LTE 再接続サービスを停止する」に示すコマンドを実行してください。

```
[armadillo ~]# rc-service connection-recover stop
connection-recover| * Stopping connection-recover ... [ ok ]
```

図 7.27 LTE 再接続サービスを停止する

LTE 再接続サービスを開始するには、「図 7.28. LTE 再接続サービスを開始する」に示すコマンドを実行してください。

```
[armadillo ~]# rc-service connection-recover start
connection-recover| * Starting connection-recover ... [ ok ]
```

図 7.28 LTE 再接続サービスを開始する

独自に接続状態を確認するサービスを実装されるなどの理由で標準の LTE 再接続サービスが不要な場合、「図 7.29. LTE 再接続サービスを無効にする」に示す手順で再接続サービスを永続的に無効にできます。

```
[armadillo ~]# rc-service connection-recover stop ❶
connection-recover| * Stopping connection-recover ... [ ok ]
[armadillo ~]# rc-update del connection-recover default ❷
service connection-recover removed from runlevel default
[armadillo ~]# persist_file -rv /etc/runlevels/default/connection-recover ❸
```

図 7.29 LTE 再接続サービスを無効にする

❶ 再接続サービスを停止します。

❷ 再接続サービスを無効にします。

❸ サービスの設定ファイルを削除を永続化します。

LTE 再接続サービスを無効化した後、再度有効にする場合、「図 7.30. LTE 再接続サービスを有効にする」に示す手順を実行してください。

```
[armadillo ~]# rc-update add connection-recover default ❶
service connection-recover added to runlevel default
[armadillo ~]# rc-service connection-recover start ❷
connection-recover| * Starting connection-recover ... [ ok ]
[armadillo ~]# persist_file -rv /etc/runlevels/default/connection-recover ❸
```

図 7.30 LTE 再接続サービスを有効にする

❶ 再接続サービスを有効にします。

❷ 再接続サービスを開始します。

❸ サービスの設定ファイルを永続化します。

7.1.5.11. ModemManager - mmcli について

ここでは ModemMamager と mmcli について説明します。

Armadillo-IoT ゲートウェイ A6E にはネットワークを管理する NetworkManager とは別に、モデムを管理する ModemManager がインストールされています。ModemManager はモバイルブロードバンドデバイス(LTE モジュールなど)の操作および、接続状況の管理などを行います。

ModemManager のコマンドラインツールである **mmcli** を使用することで、LTE 通信の電波強度や SIM カードの情報(電話番号や IMEI など)を取得することが可能です。mmcli の詳しい使いかたについては **man mmcli** を参照してください。

ModemManager はモデムデバイスに応じたプラグインを選択して動作します。Cat.M1 モデルでは、cinterion-ems31 という名称のプラグインで動作しています。

7.1.5.12. mmcli - 認識されているモデムの一覧を取得する

認識されているモデムの一覧を取得するには、「図 7.31. 認識されているモデムの一覧を取得する」に示すコマンドを実行します。

```
[armadillo:~]# mmcli -L
    /org/freedesktop/ModemManager1/Modem/0 [Cinterion] EMS31-J
```

図 7.31 認識されているモデムの一覧を取得する

7.1.5.13. mmcli - モデムの情報を取得する

モデムの情報を取得するには、「図 7.32. モデムの情報を取得する」に示すコマンドを実行します。

```
armadillo:~# mmcli -m 0
  --------------------------------
  General  |                 path: /org/freedesktop/ModemManager[number1]/Modem/[number2]
           |           device id: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  --------------------------------
  Hardware |         manufacturer: Cinterion
           |                model: EMS31-J
           |    firmware revision: XXXXXXXXXXXXXXXXXXX
           |            supported: lte
           |              current: lte
           |         equipment id: XXXXXXXXXXXXXXX
: (省略)
```

図 7.32 モデムの情報を取得する

モデムの情報を取得するには、SIM カードが取り付けられている必要があります。正しく SIM カードが取り付けられていることを確認してください。

7.1.5.14. mmcli - SIM の情報を取得する

SIM の情報を取得するには、「図 7.33. SIM の情報を取得する」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m 0
: (省略)
SIM      |     primary sim path: /org/freedesktop/ModemManager1/SIM/[number] # [number] を次のコマ ⏎
ンドで使用
: (省略)

[armadillo ~]# mmcli -i [number]
  ------------------------------
```

```
  General    |              path: /org/freedesktop/ModemManager1/SIM/0
  -------------------------------
  Properties |            active: yes
             |              imsi: XXXXXXXXXXXXXX
             |             iccid: XXXXXXXXXXXXXXXXXX
             |       operator id: XXXXX
             |     operator name: XXXXXXXXXX
```

図 7.33 SIM の情報を取得する

7.1.5.15. mmcli - 回線情報を取得する

回線情報を取得するには、「図 7.34. 回線情報を取得する」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m 0
:（省略）
  Bearer   |               paths: /org/freedesktop/ModemManager1/Bearer/[number] # [number] を次の ⏎
コマンドで使用
:（省略）

[armadillo ~]# mmcli -b [number]
  ------------------------------------
  General            |           path: /org/freedesktop/ModemManager1/Bearer/[bearer number]
                     |           type: default
  ------------------------------------
  Status             |      connected: yes
                     |      suspended: XX
                     |    multiplexed: XX
                     |     ip timeout: XX
  ------------------------------------
  Properties         |            apn: XXXXXXXXXX
                     |        ip type: XXXXX
```

図 7.34 回線情報を取得する

7.1.6. 無線 LAN

本章では、Armadillo-IoT ゲートウェイ A6E に搭載されている無線 LAN モジュールの使用方法について説明します。

例として、WPA2-PSK(AES)のアクセスポイントに接続します。WPA2-PSK(AES)以外のアクセスポイントへの接続方法などについては、man nm-settings を参考にしてください。また、以降の説明では、アクセスポイントの ESSID を[essid]、パスフレーズを[passphrase]と表記します。

7.1.6.1. 無線 LAN アクセスポイントに接続する

無線 LAN アクセスポイントに接続するためには、次のようにコマンドを実行してコネクションを作成します。

```
[armadillo ~]# nmcli device wifi connect [essid] password [passphrase]
```

図 7.35 無線 LAN アクセスポイントに接続する

作成されたコネクションの ID は nmcli connection コマンドで確認できます。

```
[armadillo ~]# nmcli connection
NAME                UUID                                  TYPE      DEVICE
atmark-4f           e051a1df-6bd7-4bcf-9c71-461af666316d  wifi      wlan0
Wired connection 1  f147b8e8-4a17-312d-a094-8c9403007f6a  ethernet  --
```

図 7.36 無線 LAN のコネクションが作成された状態

7.1.6.2. 無線 LAN のコネクション設定を編集する場合の注意事項

無線 LAN の設定情報を **nmcli connection modify** コマンドで編集する場合、パスワード情報がリセットされます。「図 7.35. 無線 LAN アクセスポイントに接続する」に示すコマンドを実行し、都度パスワードを再設定してください。

7.1.6.3. 無線 LAN の接続を確認する

無線 LAN で正常に通信が可能か確認します。

同じネットワーク内にある通信機器と PING 通信を行います。以下の例では、通信機器が「192.0.2.20」という IP アドレスを持っていると想定しています。

```
[armadillo ~]# ping 192.0.2.20
```

図 7.37 無線 LAN の PING 確認

無線 LAN 以外のコネクションが有効化されている場合、ネットワーク通信に無線 LAN が使用されない場合があります。確実に無線 LAN の接続確認をする場合は、事前に無線 LAN 以外のコネクションを無効化してください。

7.1.7. BT

本章では、Armadillo-IoT ゲートウェイ A6E に搭載されている BT モジュールの使用方法について説明します。

例として、bluetoothctl コマンドを使用して周辺の bluetooth 機器のアドバタイジング・パケットを受信します。bluetoothctl コマンドを使用するには、 bluez パッケージが必要です。

```
[armadillo ~]# apk add bluez
```

図 7.38 bluez のインストール

インストール後、bluetooth.service を有効化します。

```
[armadillo ~]# service bluetooth start
```

図 7.39 bluetooth.service の有効化

bluetoothctl コマンドを使用し、 アドバタイジング・パケットをスキャンします。

```
[armadillo ~]# bluetoothctl
[bluetooth]#
Agent registered
[CHG] Controller [AA:AA:AA:AA:AA:AA] Pairable: yes
[bluetooth]# scan on
Discovery started
[CHG] Controller [AA:AA:AA:AA:AA:AA] Discovering: yes
[NEW] Device [BB:BB:BB:BB:BB:BB] [bluetooth Name]
[CHG] Device [BB:BB:BB:BB:BB:BB] RSSI: -66
[CHG] Device [BB:BB:BB:BB:BB:BB] RSSI: -73
```

図 7.40 bluetoothctl スキャン開始

スキャンを停止するには、scan off を実行します。

```
[bluetooth]# scan off
Discovery stopped
[CHG] Controller [AA:AA:AA:AA:AA:AA] Discovering: no
```

図 7.41 bluetoothctl スキャン停止

bluetoothctl を終了するには、exit を実行します。

```
[bluetooth]# exit
```

図 7.42 bluetoothctl 終了

7.2. ストレージ

Armadillo-IoT ゲートウェイ A6E でストレージとして使用可能なデバイスを次に示します。

表 7.9 ストレージデバイス

デバイス種類	ディスクデバイス	先頭パーティション	インターフェース
オンボード eMMC	/dev/mmcblk0	/dev/mmcblk0p1	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp2	なし	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp3	なし	オンボード
SD/SDHC/SDXC カード	/dev/mmcblk1	/dev/mmcblk1p1	microSD スロット(CON1)
USB メモリ	/dev/sd* [a]	/dev/sd*1	USB ホストインターフェース (CON9)

[a]USB ハブを利用して複数の USB メモリを接続した場合は、認識された順に sda 、sdb 、 sdc … となります。

GPP(General Purpose Partition)について

GPP は、eMMC の通常の記憶領域を割譲して eMMC 内部に作られた記憶領域です。 eMMC の通常の記憶領域とはアドレス空間が異なるため、/dev/mmcblk0 および /dev/mmcblk0p* に対してどのような書き込みを行っても /dev/mmcblk0gp* のデータが書き換わることはありません。

Armadillo-IoT ゲートウェイ A6E では、8 MiB の GPP を 4 つ作成しています。各領域の用途を「表 7.10. eMMC の GPP の用途」に示します。

表 7.10 eMMC の GPP の用途

ディスクデバイス	用途
/dev/mmcblk0gp0	ライセンス情報等の保存
/dev/mmcblk0gp1	予約領域
/dev/mmcblk0gp2	ユーザー領域
/dev/mmcblk0gp3	ユーザー領域

7.2.1. ストレージの使用方法

ここでは、SDHC カードを接続した場合を例にストレージの使用方法を説明します。以降の説明では、共通の操作が可能な場合に、SD/SDHC/SDXC カードを SD カードと表記します。

SDXC/microSDXC カードを使用する場合は、事前に「7.2.2. ストレージのパーティション変更とフォーマット」を参照してフォーマットを行う必要があります。これは、Linux カーネルが exFAT ファイルシステムを扱うことができないためです。通常、購入したばかりの SDXC/microSDXC カードは exFAT ファイルシステムでフォーマットされています。

Linux では、アクセス可能なファイルやディレクトリは、一つの木構造にまとめられています。あるストレージデバイスのファイルシステムを、この木構造に追加することを、マウントするといいます。マウントを行うコマンドは、 mount です。

mount コマンドの典型的なフォーマットは、次の通りです。

```
mount [-t fstype] device dir
```

図 7.43 mount コマンド書式

-t オプションに続く fstype には、ファイルシステムタイプを指定します。ファイルシステムタイプの指定は省略可能です。省略した場合、mount コマンドはファイルシステムタイプを推測します。この推測は必ずしも適切なものとは限りませんので、事前にファイルシステムタイプが分かっている場合は明示的に指定してください。FAT32 ファイルシステムの場合は vfat 、EXT3 ファイルシステムの場合は ext3 を指定します。

通常、購入したばかりの SDHC カードは FAT32 または exFAT ファイルシステムでフォーマットされています。

device には、ストレージデバイスのデバイスファイル名を指定します。microSD カードのパーティション 1 の場合は /dev/mmcblk1p1 、パーティション 2 の場合は /dev/mmcblk1p2 となります。

dir には、ストレージデバイスのファイルシステムをマウントするディレクトリを指定します。

microSD スロット (CON1) に SDHC カードを挿入し、以下に示すコマンドを実行すると、 /media ディレクトリに SDHC カードのファイルシステムをマウントすることができます。microSD カード内のファイルは、/media ディレクトリ以下に見えるようになります。

```
[armadillo ~]# mount -t vfat /dev/mmcblk1p1 /media
[armadillo ~]# ls /media
  :
  :
```

図 7.44 ストレージのマウント

ストレージを安全に取り外すには、アンマウントという作業が必要です。アンマウントを行うコマンドは、 umount です。オプションとして、アンマウントしたいデバイスがマウントされているディレクトリを指定します。

```
[armadillo ~]# umount /media
```

図 7.45 ストレージのアンマウント

7.2.2. ストレージのパーティション変更とフォーマット

通常、購入したばかりの SDHC カードや USB メモリは、一つのパーティションを持ち、FAT32 ファイルシステムでフォーマットされています。

パーティション構成を変更したい場合、 fdisk コマンドを使用します。 fdisk コマンドの使用例として、一つのパーティションで構成されている microSD カードのパーティションを、2 つに分割する例を「図 7.46. fdisk コマンドによるパーティション変更」に示します。一度、既存のパーティションを削除してから、新たにプライマリパーティションを二つ作成しています。先頭のパーティションには 100MByte、二つめのパーティションに残りの容量を割り当てています。先頭のパーティションは /dev/mmcblk1p1 、二つめは /dev/mmcblk1p2 となります。 fdisk コマンドの詳細な使い方は、man ページ等を参照してください。

```
[armadillo ~]# fdisk /dev/mmcblk1

Welcome to fdisk (util-linux 2.29.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): d
```

```
Selected partition 1
Partition 1 has been deleted.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-7744511, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-7744511, default 7744511): +100M

Created a new partition 1 of type 'Linux' and of size 100 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (206848-7744511, default 206848):
Last sector, +sectors or +size{K,M,G,T,P} (206848-7744511, default 7744511):

Created a new partition 2 of type 'Linux' and of size 3.6 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
[  447.905671]  mmcblk1: p1 p2
Syncing disks.
```

図 7.46 fdisk コマンドによるパーティション変更

FAT32 ファイルシステムでストレージデバイスをフォーマットするには、 mkfs.vfat コマンドを使用します。また、EXT2 や EXT3、 EXT4 ファイルシステムでフォーマットするには、mkfs.ext2 や mkfs.ext3 、 mkfs.ext4 コマンドを使用します。microSD カードのパーティション 1 を EXT4 ファイルシステムでフォーマットするコマンド例を、次に示します

```
[armadillo ~]# mkfs.ext4 /dev/mmcblk1p1
```

図 7.47 EXT4 ファイルシステムの構築

7.3. LED

Armadillo-IoT ゲートウェイ A6E の LED は GPIO で接続されているため、ソフトウェアで制御することができます。

利用しているデバイスドライバは LED クラスとして実装されているため、LED クラスディレクトリ以下のファイルによって LED の制御を行うことができます。LED クラスディレクトリと各 LED の対応を次に示します。

表 7.11 LED クラスディレクトリと LED の対応

LED クラスディレクトリ	インターフェース	デフォルトトリガ
/sys/class/leds/app/	アプリケーション LED	default-on
/sys/class/leds/yellow/	ユーザー LED 黄	none

以降の説明では、任意の LED を示す LED クラスディレクトリを /sys/class/leds/[LED]/ のように表記します。 [LED] の部分を適宜読みかえてください。

7.3.1. LED を点灯/消灯する

LED クラスディレクトリ以下の brightness ファイルへ値を書き込むことによって、LED の点灯/消灯を行うことができます。 brightness に書き込む有効な値は 0~255 です。

brightness に 0 以外の値を書き込むと LED が点灯します。

```
[armadillo ~]# echo 1 > /sys/class/leds/[LED]/brightness
```

図 7.48 LED を点灯させる

Armadillo-IoT ゲートウェイ A6E の LED には輝度制御の機能がないため、0(消灯)、1?255(点灯)の 2 つの状態のみ指定することができます。

brightness に 0 を書き込むと LED が消灯します。

```
[armadillo ~]# echo 0 > /sys/class/leds/[LED]/brightness
```

図 7.49 LED を消灯させる

brightness を読み出すと LED の状態が取得できます。

```
[armadillo ~]# cat /sys/class/leds/[LED]/brightness
```

図 7.50 LED の状態を表示する

7.3.2. トリガを使用する

Linux では、LED をある特定のタイミングで光らせることができます。これを「トリガ」と呼びます。LED クラスディレクトリ以下の trigger ファイルへ値を書き込むことによって LED の点灯/消灯にトリガを設定することができます。 trigger でサポートされている値は以下の通りです。

表 7.12 LED トリガの種類

設定	説明
none	トリガを設定しません
mmc0	eMMC のアクセスランプにします
mmc1	microSD スロットのアクセスランプにします

設定	説明
timer	任意のタイミングで点灯/消灯を行います。この設定にすることにより、LED クラスディレクトリ以下に delay_on, delay_off ファイルが出現し、それぞれ点灯時間, 消灯時間をミリ秒単位で指定します
heartbeat	心拍のように点灯/消灯を行います
default-on	主に Linux カーネルから使用します。LED が点灯します

trigger ファイルを読み出すとサポートしているトリガと、現在有効のトリガが表示されます。[] が付いているものが現在のトリガです。

```
[armadillo ~]# cat /sys/class/leds/[LED]/trigger
[none] rc-feedback bluetooth-power rfkill-any rfkill-none kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock
kbd-ctrlllock kbd-ctrlrlock rfkill0 rfkill1 timer oneshot heartbeat backlight gpio default-on mmc0
mmc1
```

図 7.51 対応している LED トリガを表示

以下のコマンドを実行すると、LED が 2 秒点灯、1 秒消灯を繰り返します。

```
[armadillo ~]# echo timer > /sys/class/leds/[LED]/trigger
[armadillo ~]# echo 2000 > /sys/class/leds/[LED]/delay_on
[armadillo ~]# echo 1000 > /sys/class/leds/[LED]/delay_off
```

図 7.52 LED のトリガに timer を指定する

7.4. ユーザースイッチ

Armadillo-IoT ゲートウェイ A6E のユーザースイッチのデバイスドライバは、インプットデバイスとして実装されています。インプットデバイスのデバイスファイルからボタンプッシュ/リリースイベントを取得することができます。

ユーザースイッチのインプットデバイスファイルと、各スイッチに対応したイベントコードを次に示します。

表 7.13 インプットデバイスファイルとイベントコード

ユーザースイッチ	インプットデバイスファイル	イベントコード
SW1	/dev/input/event0	28 (KEY_ENTER)

インプットデバイスは検出された順番にインデックスが割り振られます。USB デバイスなどを接続してインプットデバイスを追加している場合は、デバイスファイルのインデックスが異なる可能性があります。

7.4.1. イベントを確認する

ユーザースイッチのボタンプッシュ/リリースイベントを確認するために、ここでは evtest コマンドを利用します。 evtest を停止するには、Ctrl-c を入力してください。

```
[armadillo ~]# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 28 (KEY_ENTER)
Properties:
Testing ... (interrupt to exit)
Event: time 1662514165.249093, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0  ❶
Event: time 1662514165.249093, -------------- SYN_REPORT ------------
Event: time 1662514169.019097, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1  ❷
Event: time 1662514169.019097, -------------- SYN_REPORT ------------
```

図 7.53 ユーザースイッチ: イベントの確認

❶ SW1 のボタン **プッシュ** イベントを検出したときの表示

❷ SW1 のボタン **リリース** イベントを検出したときの表示

7.5. RS485

RS485 は、入出力インターフェース(CON6)の以下ピンを使用します。インターフェースの位置については「4.3. インターフェースレイアウト」、仕様については「14.7.3. RS485」 をご確認ください。

また、終端抵抗 120Ω の ON/OFF をスイッチで切り替えることができます。詳しくは「14.16. SW3(RS485 終端抵抗設定スイッチ)」 をご確認ください。

表 7.14 RS485 に対応する CON6 ピン番号

ピン番号	ピン名
10	DATA+
11	DATA-
12	GND

シリアルインターフェースのデバイスファイルは、/dev/ttymxc4 を使用します。

7.6. 接点入力

入出力インターフェース(CON6)のピン 4, ピン 5 を接点入力として使用できます。インターフェースの位置については「4.3. インターフェースレイアウト」、仕様については「14.7.1. 接点入力」 をご確認ください。

ソフトウェアからは GPIO として制御可能であり、対応する GPIO 番号を次に示します。

表 7.15 接点入力に対応する CON6 ピン番号

ピン番号	ピン名	GPIO チップ	GPIO 番号
4	DI1	gpiochip5	0
5	DI2	gpiochip5	1

7.6.1. 入力レベルの確認

gpioget コマンドを用いて入力レベルの確認ができます。"0"は LOW レベル、"1"は HIGH レベルを表わします。

```
[armadillo ~]# gpioget gpiochip5 [GPIO]
0
```

図 7.54 入力レベルの確認

7.7. 接点出力

入出力インターフェース(CON6)のピン 6/ピン 7、ピン 8/ピン 9 を接点出力として使用できます。インターフェースの位置については「4.3. インターフェースレイアウト」、仕様については「14.7.2. 接点出力」 をご確認ください。

ソフトウェアからは GPIO として制御可能であり、対応する GPIO 番号を次に示します。

表 7.16 接点出力に対応する CON6 ピン番号

ピン番号	ピン名	GPIO チップ	GPIO 番号
6	DO1A	gpiochip5	2
7	DO1B	gpiochip5	2
8	DO2A	gpiochip5	3
9	DO2B	gpiochip5	3

7.7.1. 出力レベルの設定

gpioset コマンドを用いて、出力レベルを設定することができます。出力レベルには "0" または "1" を設定します。"0"は LOW レベル、"1"は HIGH レベルを表わします。

```
[armadillo ~]# gpioset gpiochip5 [GPIO]=0
```

図 7.55 出力レベルを "0" に設定する場合

7.8. RTC

本章では、Armadillo-IoT ゲートウェイ A6E のリアルタイムクロック(RTC) の使用方法について説明します。

7.8.1. RTC に時刻を設定する

Linux の時刻には、Linux カーネルが管理するシステムクロックと、RTC が管理するハードウェアクロックの 2 種類があります。RTC に時刻を設定するためには、まずシステムクロックを設定します。その後に、ハードウェアクロックをシステムクロックと一致させる手順となります。

システムクロックは、date コマンドを用いて設定します。date コマンドの引数には、設定する時刻を[MMDDhhmmCCYY.ss] というフォーマットで指定します。時刻フォーマットの各フィールドの意味を次に示します。

表 7.17 時刻フォーマットのフィールド

フィールド	意味
MM	月
DD	日(月内通算)
hh	時
mm	分
CC	年の最初の 2 桁(省略可)
YY	年の最後の 2 桁(省略可)
ss	秒(省略可)

2022 年 3 月 2 日 12 時 34 分 56 秒に設定する例を次に示します。

```
[armadillo ~]# date
Sat Jan  1 09:00:00 JST 2000
[armadillo ~]# date 030212342022.56
Fri Mar  2 12:34:56 JST 2022
[armadillo ~]# date
Fri Mar  2 12:34:57 JST 2022
```

図 7.56 システムクロックを設定

システムクロックを設定後、ハードウェアクロックを hwclock コマンドを用いて設定します。

```
[armadillo ~]# hwclock ❶
2000-01-01 00:00:00.000000+09:00
[armadillo ~]# hwclock --utc --systohc ❷
[armadillo ~]# hwclock --utc ❸
2022-03-02 12:57:20.534140+09:00
```

図 7.57 ハードウェアクロックを設定

❶ 現在のハードウェアクロックを表示します。

❷ ハードウェアクロックを協定世界時(UTC)で設定します。

❸ ハードウェアクロックが UTC で正しく設定されていることを確認します。

8. 省電力・間欠動作機能

本章では、Armadillo-IoT ゲートウェイ A6E の 省電力・間欠動作機能や動作モード、状態遷移について説明します。

8.1. 動作モード・起床条件と状態遷移図

Armadillo-IoT ゲートウェイ A6E の動作モード・起床条件と状態遷移を次に示します。

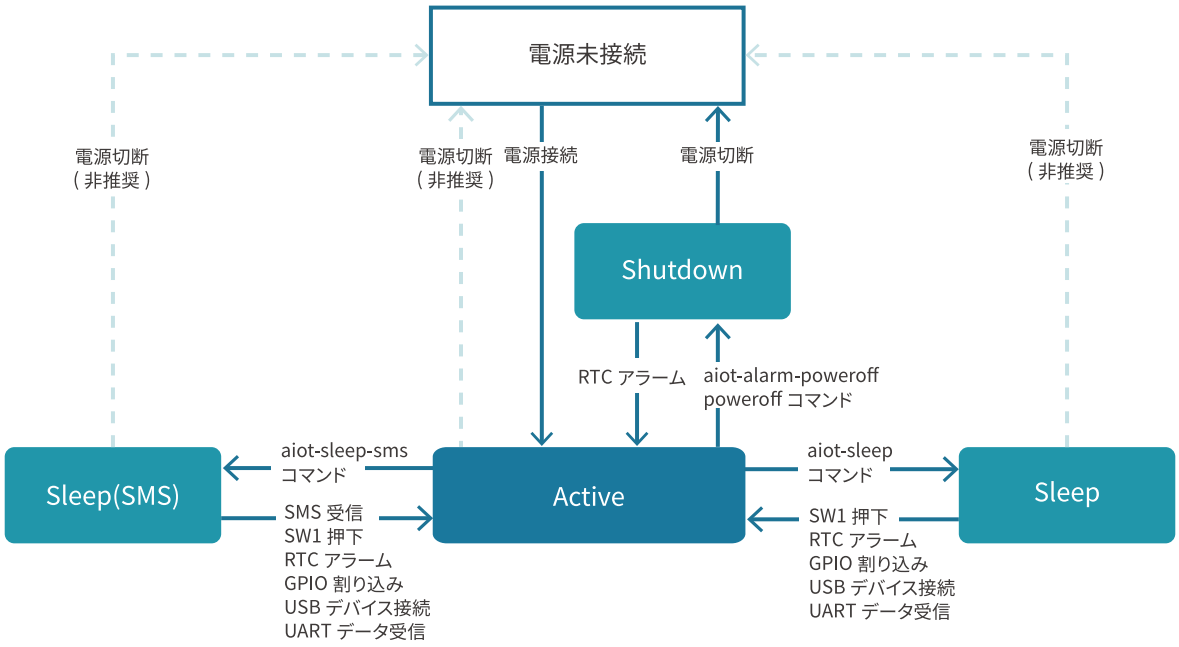


図 8.1 状態遷移図

8.1.1. 動作モード・起床条件

次に、各動作モードと利用することのできる起床条件について説明します。

8.1.1.1. アクティブモード

「CPU:動作」、「LTE-M:動作」 状態のモードです。

Armadillo-IoT ゲートウェイ A6E の電源投入後 Linux カーネルが起動し、まずはアクティブモードに遷移します。

任意のアプリケーションの実行や、外部センサー・デバイスの制御、LTE-M や Ethernet での通信が可能ですが、最も電力を消費するモードです。アクティブモードの時間をより短くすることで、消費電力を押さえることができます。

8.1.1.2. シャットダウンモード

「CPU:停止」、「LTE-M:停止」の状態であり最も消費電力を抑えることのできるモードです。

その反面、CPU を停止させ、Linux カーネルをシャットダウンしている状態であるため、アクティブモードに遷移する場合は Linux カーネルの起動分の時間がかかります。

シャットダウンモードからアクティブモードに遷移するには、RTC のアラーム割り込みを使用するか、一度電源を切断・再接続を行う必要があります。

8.1.1.3. スリープモード

「CPU:待機」、「LTE-M:停止」 状態のモードです。

CPU(i.MX6ULL)はパワーマネジメントの Suspend-to-RAM 状態になり、Linux カーネルは Pause の状態になります。シャットダウンモードと比較すると消費電力は高いですが、Linux カーネルの起動は不要であるため数秒程度でアクティブモードに遷移が可能です。ユーザスイッチの投下、RTC アラーム割り込み、GPIO 割り込み、USB デバイスの接続、UART によるデータ受信、によってアクティブモードへの遷移ができます。

8.1.1.4. スリープ(SMS 起床可能)モード

「CPU:待機」、「LTE-M:待機」 状態のモードです。

スリープモードとの違いは、SMS の受信によって、アクティブモードへの遷移も可能である点です。LTE-M:待機(PSM)の状態であるため、スリープモードよりも電力を消費します。

8.2. シャットダウンモードへの遷移と起床

シャットダウンモードへ遷移するには、poweroff コマンド、または aiot-alarm-poweroff コマンドを実行します。

8.2.1. poweroff コマンド

poweroff コマンドを実行してシャットダウンモードに遷移した場合、電源の切断・接続のみでアクティブモードに遷移が可能です。poweroff コマンドの実行例を次に示します。

```
[armadillo ~]# poweroff
[  OK  ] Stopped target Timers.
[  OK  ] Stopped Daily man-db regeneration.
[  OK  ] Stopped Daily rotation of log files.

※省略

[39578.876586] usb usb1: USB disconnect, device number 1
[39578.882754] ci_hdrc ci_hdrc.0: USB bus 1 deregistered
[39578.888133] reboot: Power down
```

8.2.2. aiot-alarm-poweroff コマンド

aiot-alarm-poweroff コマンドを実行することで、シャットダウンモードに遷移後、RTC のアラーム割り込みをトリガで起床（アクティブモードに遷移）することができます。コマンド書式を以下に示します。

```
[armadillo ~]# aiot-alarm-poweroff +[現在時刻からの経過秒数]
```

図 8.2 aiot-alarm-poweroff コマンド書式

シャットダウンモードに遷移し、300 秒後にアラーム割り込みを発生させるには、次のようにコマンドを実行します。

```
[armadillo ~]# aiot-alarm-poweroff +300
aiot-alarm-poweroff: alarm_timer +300 second
```

現在時刻からの経過秒数は 180 秒以上を指定する必要があります。

8.3. スリープモードへの遷移と起床

aiot-sleep コマンドを実行することで、スリープモードに遷移することができます。スリープモードからの起床（アクティブモードに遷移する）条件は、aiot-sleep コマンドを実行する前に aiot-set-wake-trigger コマンドで事前指定します。ユーザースイッチによる起床は標準で有効になっています。また、起床条件は OR 条件での設定が可能です。

8.3.1. RTC アラーム割り込み以外での起床

aiot-set-wake-trigger コマンドの書式と設定可能なパラメータを以下に示します。

```
[armadillo ~]# aiot-set-wake-trigger [TRIGGER] [enabled|disabled]
```

図 8.3 aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込み以外での起床のとき）

表 8.1 aiot-modem-control TRIGGER 一覧

TRIGGER	説明
usb	CON5(USB ホストインターフェース)に USB デバイスを挿抜したとき
uart3	CON3(シリアルインターフェース /dev/ttymxc2)にデータ受信があったとき
rs485	UART5(シリアルインターフェース /dev/ttymxc4)にデータ受信があったとき
gpio	GPIO 割り込みが発生したとき
sw1	SW1 が押下されたとき

コンソール(/dev/ttymxc2)から入力があった場合にスリープモードから起床するには、次に示すコマンドを実行します。

```
[armadillo ~]# aiot-set-wake-trigger uart3 enabled
aiot-set-wake-trigger: uart3 enabled

[armadillo ~]# aiot-sleep
aiot-sleep: Power Management suspend-to-ram
[ 1767.050404] PM: suspend entry (deep)
[ 1767.054019] PM: Syncing filesystems ...
[ 1767.236546] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full -
flow control rx/tx
[ 1767.428714] done.
[ 1767.431262] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 1767.439582] OOM killer disabled.
[ 1767.442834] Freezing remaining freezable tasks ... (elapsed 0.001
seconds) done.
[ 1767.451485] Suspending console(s) (use no_console_suspend to debug)
```

```
※ コンソールに入力

[ 1767.567686] OOM killer enabled.
[ 1767.570875] Restarting tasks ... done.
[ 1767.606048] PM: suspend exit
aiot-sleep: change mode CPU Idle
```

8.3.2. RTC アラーム割り込みでの起床

RTC アラーム割り込みでの起床を行う場合、パラメーター設定が異なります。なお、RTC を起床要因に使って間欠動作させる場合は、 「7.8. RTC」 を参考に、必ず RTC の日時設定を行ってください。

```
[armadillo ~]# aiot-set-wake-trigger rtc [enabled|disabled] <現在時刻からの経過秒数>
```

図 8.4 aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込みでの起床の場合）

現在時刻からの経過秒数 は 60 秒以上を指定する必要があります。

300 秒後に RTC アラーム割り込みを発生させ、スリープモードから起床させるコマンド実行例を以下に示します。

```
[armadillo ~]# aiot-set-wake-trigger rtc enabled +300
aiot-set-wake-trigger: rtc enabled
aiot-set-wake-trigger: alarm_timer +300 second

[armadillo ~]# aiot-sleep
aiot-sleep: Power Management suspend-to-ram
[ 1767.050404] PM: suspend entry (deep)
[ 1767.054019] PM: Syncing filesystems ...
[ 1767.236546] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full -
flow control rx/tx
[ 1767.428714] done.
[ 1767.431262] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 1767.439582] OOM killer disabled.
[ 1767.442834] Freezing remaining freezable tasks ... (elapsed 0.001
seconds) done.
[ 1767.451485] Suspending console(s) (use no_console_suspend to debug)

※ 約 300 秒待つ

[ 1767.567686] OOM killer enabled.
[ 1767.570875] Restarting tasks ... done.
[ 1767.606048] PM: suspend exit
aiot-sleep: change mode CPU Idle
```

8.3.3. 起床要因のクリア

すべての起床要因をクリアするには次に示すコマンドを実行します。ユーザースイッチによる起床設定は無効化できません。

```
[armadillo ~]# aiot-set-wake-trigger all disabled
aiot-set-wake-trigger: clear_all disabled
```

8.4. スリープ(SMS 起床可能)モードへの遷移と起床

aiot-sleep-sms コマンドを実行することで、スリープ(SMS 起床可能)モードに遷移することができます。スリープモードからの起床（アクティブモードに遷移する）条件は、aiot-sleep-sms コマンドを実行する前に aiot-set-wake-trigger コマンドで事前指定します。ユーザースイッチによる起床は標準で有効になっています。aiot-sleep-sms コマンドを実行した場合 SMS 受信による起床は強制的に有効になります。また、起床条件は OR 条件での設定が可能です。

aiot-sleep-sms コマンドの実行例を次に示します。

```
[armadillo ~]# aiot-sleep-sms
aiot-sleep-sms: Power Management suspend-to-ram
AT+CMGF=1
OK

AT^SIND=”message”,0
^SIND: message,0,0

OK

AT+CMGD=1,4
OK

AT+CMGL=”ALL”
OK

[ 3508.609638] PM: suspend entry (deep)

^SIND: message,1,0

OK

[ 3508.613982] PM: Syncing filesystems ... done.
[ 3508.637946] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 3508.646276] OOM killer disabled.
[ 3508.649527] Freezing remaining freezable tasks ... (elapsed 0.001
seconds) done.
[ 3508.658161] Suspending console(s) (use no_console_suspend to debug)

※ SMS 受信

[ 1767.567686] OOM killer enabled.
[ 1767.570875] Restarting tasks ... done.
[ 1767.606048] PM: suspend exit
aiot-sleep: change mode CPU Idle
```

8.5. 状態遷移トリガにコンテナ終了通知を利用する

動作中のコンテナの終了をトリガに、省電力状態のモードへの遷移を行うことができます。

コンテナの終了契機は「/etc/atmark/containers/*.conf ファイルの set_command で指定したコンテナ起動時に実行するコマンド」のプロセスが終了した時」となります。ファイルの詳細は 「10.5.1. コンテナの自動起動」 を参照してください。

遷移先の動作モードと起床条件は設定ファイルで指定し、コンテナが終了すると指定した動作モードへ遷移、指定した起床条件が発生すると省電力モードから復帰します。また、その際自動的にコンテナも開始します。

コンテナ終了時に遷移する動作モードと起床条件については、設定ファイル(/etc/conf.d/power-utils.conf)で指定します。設定ファイルは下記の通り、 TARGET , MODE , WAKEUP を指定します。

```
[armadillo ~]# cat /etc/conf.d/power-utils.conf
TARGET='a6e_gw_container'
MODE='NONE'
WAKEUP='SW1', 'USB', 'UART', 'GPIO', 'RTC:60'
```

以下は遷移する動作モードがシャットダウンモード、起床条件が RTC(300 秒後起床) のパターンです。なお、デフォルトでは省電力・間欠動作は OFF (MODE=NONE) となっています。

```
[armadillo ~]# cat /etc/conf.d/power-utils.conf
TARGET='a6e-gw-container'
MODE='SHUTDOWN'
WAKEUP='RTC:300'
```

設定ファイルの概要を以下に示します。

表 8.2 設定パラメーター

パラメーター名	意味
TARGET	状態遷移トリガの対象となるコンテナ名
MODE	遷移先の動作モード
WAKEUP	起床条件

表 8.3 遷移先の動作モード

モード名	設定値
省電力・間欠動作 OFF	NONE (初期値)
シャットダウンモード	SHUTDOWN
スリープモード	SLEEP

表 8.4 起床条件

起床条件	設定値
RTC	RTC:[コンテナ終了からの経過秒数[a]]
SW1 押下	SW1
GPIO 割り込み	GPIO
USB デバイス接続	USB
UART データ受信	UART

起床条件	設定値
SMS 受信	SMS

[a]現在時刻からの経過秒数は MODE が SHUTDOWN の場合は 300 秒以上、SLEEP の場合は 60 秒以上を指定する必要があります。

9. 開発の基本的な流れ

9.1. Armadillo への接続

9.1.1. シリアルコンソール

Armadillo-IoT ゲートウェイ A6E では CON7 (USB コンソールインターフェース)をシリアルコンソールとして使用できます。シリアル通信設定等については、「4.2.4. シリアル通信ソフトウェア(minicom)の使用」を参照してください。

ログイン方法については、「5.2. ログイン」を参照してください。

9.1.2. ssh

Armadillo-IoT ゲートウェイ A6E には openssh がインストールされていますが、デフォルトでは SSH サーバーが起動していません。

SSH サーバーを自動的に起動するようにするためには、以下のコマンドを実行してください。

```
[armadillo:~]# rc-update add sshd
 * service sshd added to runlevel default
[armadillo ~]# persist_file /etc/runlevels/default/sshd
[ 2819.277066] EXT4-fs (mmcblk0p1): re-mounted. Opts: (null)
[armadillo ~]# reboot
```

上記の例では、再起動後も設定が反映されるように、 `persist_file` コマンドで eMMC に設定を保存しています。

9.2. overlayfs の扱い

Armadillo BaseOS ではルートファイルシステムに overlayfs を採用しています。

その為、ファイルを変更した後 Armadillo の電源を切ると変更内容は保持されません。開発中などに rootfs の変更内容を保持するには、変更したファイルに対して `persist_file` コマンドを使用します。`persist_file` コマンドの詳細は 「10.10.2. overlayfs と persist_file について」 参照してください。

開発以外の時は安全のため、ソフトウェアアップデートによる更新を実行してください。アップデート手順に関しては 「10.9. Armadillo のソフトウェアをアップデートする」 を参照してください。

rootfs の内容を変更しても、ソフトウェアアップデートを実施した際に変更した内容が保持されない可能性があります。ソフトウェアアップデート実施後も変更内容を保持する手順に関しては 「10.9.6. swupdate_preserve_files について」 を参照してください。

9.3. アプリケーション開発パターン

Armadillo-IoT ゲートウェイ A6E でアプリケーションを開発する手順は、以下のパターンに分類されます。

・ゲートウェイアプリケーション開発：ゲートウェイコンテナを使用してアプリケーションを開発する

・ゲートウェイコンテナを利用せず、一からコンテナを作成する

各パターンにおける詳細な手順は 「9.4. ゲートウェイアプリケーション開発の流れ」 に記載しています。ゲートウェイコンテナに関しては 「9.3.1. ゲートウェイコンテナとは」 で説明します。

9.3.1. ゲートウェイコンテナとは

Armadillo-IoT ゲートウェイ A6E には、ゲートウェイコンテナがプリインストールされています。このコンテナを利用することで、インターフェースの操作やクラウドへのデータアップロードなどを簡単に行うことができます。

ゲートウェイコンテナを利用して実施できる内容は下記の通りです。

表 9.1 利用できるインターフェース・機能

インターフェース	機能
RS485 (ModbusRTU)	レジスタ読み出し
	レジスタ書き込み
接点入力 2ch	ポーリング監視
	エッジ検出
接点出力 2ch	指定レベル出力
アプリケーション LED	点灯/消灯操作
ユーザースイッチ	状態取得

表 9.2 利用できるクラウドベンダー・サービス

クラウドベンダー	クラウドサービス
AWS	AWS IoT Core
Azure	Azure IoT

インターフェースやクラウドサービスの選択はコンフィグ設定で行う事ができます。また、センサーデータのログ出力やネットワーク断時のキャッシュ機能にも対応しています。

利用方法や仕様については、「10.2. ゲートウェイコンテナを動かす」 を参照してください。

9.4. ゲートウェイアプリケーション開発の流れ

ここでは、 「9.3. アプリケーション開発パターン」 に記載した各パターンの手順を記載します。

9.4.1. ゲートウェイアプリケーション開発

ゲートウェイアプリケーション開発ではゲートウェイコンテナを利用します。以下のパターンに分類されます。

・ゲートウェイコンテナを改変せず利用する

・ゲートウェイアプリケーションを拡張する

・ゲートウェイコンテナイメージを改変する

9.4.1.1. ゲートウェイコンテナを改変せず利用する

ゲートウェイコンテナを改変せず利用する場合は、コードを書く必要はありません。各種設定を行うだけで、ゲートウェイアプリケーションを動作させることができます。設定、実行方法については 「10.1. ゲートウェイアプリケーションを開発する」 を参照してください。

9.4.1.2. ゲートウェイアプリケーションを拡張する

操作するインターフェースを追加したい場合や、インターフェースの制御仕様を変更したい場合は、所定のディレクトリにソースコードを配置することで実現することができます。詳細については 「10.3. ゲートウェイコンテナを拡張する」 を参照してください。

9.4.1.3. ゲートウェイコンテナイメージを改変する

基本的に、アットマークテクノではゲートウェイコンテナイメージの改変を推奨しておりません。もし改変を行う場合、以下の手順に沿って、対応することができます。なお、手順中のコマンドで指定するコンテナ名は a6e-gw-container となります。

ただし、ゲートウェイコンテナイメージの改変を行った場合、今後アットマークテクノが提供するゲートウェイコンテナのアップデートをそのまま適用することが出来なくなります。そのため、「10.3. ゲートウェイコンテナを拡張する」 に記載している方法をお勧めいたします。

1. 「9.4.2.1. Podman のデータを eMMC に保存する」 を実行

   このとき、起動しているコンテナがある場合は podman stop コマンドで停止します。(「図 10.35. コンテナを停止する実行例」)

2. ゲートウェイコンテナのコンフィグファイルを修正する

   ゲートウェイコンテナは、コンテナ起動時にコンテナアプリケーションの起動スクリプトを実行するように設定されています。そのため、下記の通り sleep infinity コマンドを実行して待ち続けるだけの設定に変更します。また、Armadillo Base OS ブート時に自動的に起動する設定となっているため、自動起動を無効にする設定も加えます。

   ```
   armadillo:~# cat /etc/atmark/containers/a6e-gw-container.conf
   :（省略）

   set_autostart no ❶

   #set_command sh /usr/bin/gw-app.sh ❷
   set_command sleep infinity ❸
   ```

   ❶ 自動起動の無効化

   ❷ 起動スクリプト実行部分をコメントアウト

   ❸ sleep infinity コマンドを実行するように修正

   設定後は persist_file コマンドで変更を保存してください。

```
[armadillo ~]# persist_file -P /etc/atmark/containers/a6e-gw-container.conf
```

3. 「10.4.2.9. 実行中のコンテナに接続する」 の通り、コンテナを起動・接続する
4. コンテナ内で開発を行う
5. 「10.4.2.5. コンテナの変更を保存する」 を実行

9.4.2. ゲートウェイコンテナを利用せず、一からコンテナを作成する

ゲートウェイコンテナを利用せずに、新たに一からコンテナを作成することもできます。次項より、その場合の手順を示します。

9.4.2.1. Podman のデータを eMMC に保存する

デフォルトでは、Podman のデータは tmpfs に保存されます。そのため、Armadillo を再起動するとデータは消えてしまいます。この挙動は、Armadillo の運用時を想定したものです。

eMMC への書き込みを最小限にする等の観点から、 Armadillo の運用時は、 Podman のデータは tmpfs に保存するのが適切です。eMMC への保存が必要な場合のみ SWUpdate または abos-ctrl で読み取り専用のイメージを保存します。

Armadillo 開発時のみ、 eMMC に Podman のデータが保存されるようにすることを推奨します。

eMMC に Podman のデータが保存されるようにするには、以下のコマンドを実行します。

```
[armadillo ~]# abos-ctrl podman-storage --disk
Creating configuration for persistent container storage
Create subvolume '/mnt/containers_storage'
[ 2145.288677] EXT4-fs (mmcblk0p1): re-mounted. Opts: (null)
[armadillo ~]# abos-ctrl podman-storage --status
Currently in disk mode, run with --tmpfs to switch
```

podman のストレージはコンテナのイメージやランタイムのデータのみです。

コンテナのデータをボリュームに入れたら消えません。詳しくは「10.4.2.5. コンテナの変更を保存する」 を参照してください。

9.4.2.2. ベースとなるコンテナを取得する

一からコンテナを作成する場合は、イメージの公開・共有サービスである Docker Hub [https://hub.docker.com/search?type=image&image_filter=official] からベースとなる OS を取得し、開発を行います。OS は alpine や debian 等があり、任意の環境でアプリケーションを作成することができます。目的に合わせて選択してください。

9.4.2.3. デバイスのアクセス権を与える

開発中のアプリケーションがデバイスを利用する場合は、コンテナにデバイスを渡す必要があります。

podman run コマンドに --device オプションでデバイスファイルを指定します。

--privileged オプションを指定するとすべてのセキュリティーメカニズムが無効になる為、全てのデバイスが利用できるようになります。このオプションを利用することは、セキュリティー上問題がある為、デバッグ用途でのみご利用ください。

9.4.2.4. アプリケーションを作成する

「10.4. アプリケーションコンテナを作成、実行する」 を参考にして、オリジナルのアプリケーションを開発します。

9.4.2.5. コンテナやデータを保存する

ログやデータベース、自分のアプリケーションのデータを保存する場合にボリュームを使ってください。実行中のコンテナイメージを保存するには podman commit コマンドで保存してください。詳しい手順は「10.4.2.5. コンテナの変更を保存する」 を参考にしてください。

9.5. アプリケーションコンテナの運用

「10.5. コンテナの運用」 を参考にしてください。

9.5.1. アプリケーションの自動起動

podman_start 用の設定ファイル (/etc/atmark/containers/*.conf) を作成します。その後、podman_start -a コマンドを実行するか、armadillo を再起動してコンテナが自動起動することを確認してください。コンテナの自動起動に関する詳しい説明は 「10.5.1. コンテナの自動起動」 を参考にしてください。

9.5.2. アプリケーションの送信

まず、コンテナをコンテナレジストリに送るか、 podman save コマンドを実行してアーカイブを作成します。

以下の例では ATDE に mkswu のキーを作成して、docker.io のイメージをこのまま使います。

手順の詳しい説明やオプションは 「10.9. Armadillo のソフトウェアをアップデートする」 を参考にしてください。

```
[ATDE ~]$ sudo apt update && sudo apt install mkswu
[ATDE ~]$ mkswu --init
: (省略)
[ATDE ~]$ cd mkswu
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/pull_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
[ATDE ~/mkswu]$ vi pull_container_nginx.desc
swdesc_option version=1

swdesc_pull_container "docker.io/nginx:alpine"
swdesc_files --extra-os nginx_start
```

```
[ATDE ~/mkswu]$ mkswu -o initial_setup_container.swu ¥
        initial_setup.desc pull_container_nginx.desc
```

ここで作成した initial_setup_container.swu ファイルを USB メモリに配置して、Armadillo-IoT ゲートウェイ A6E に刺すとインストールされます。

インストールが終了して再起動すると docker.io/nginx:alpine のコンテナを起動します。

9.5.3. インストール確認：初期化

購入状態で SWU をインストールできるか確認をするために、ソフトウェアの初期化を行います。

「10.8. Armadillo のソフトウェアの初期化」 を参照し、Armadillo Base OS を初期化してからアプリケーションをアップデートしてください。

9.5.4. アプリケーションのアップデート

アップデートを行う方法は以下の二通りです：

1. podman run コマンドで、差分アップデートを行う

   ここでは例として、アプリケーションのコンテナを myimage:1、アップデート後を myimage:2 とします。

   ```
   [armadillo ~]# podman run --name update myimage:1 sh -c "apk update && apk upgrade && apk ⏎
   cache --purge"
   [armadillo ~]# podman commit update myimage:2
   [armadillo ~]# podman rm update
   [armadillo ~]# podman_partial_image -b myimage:1 -o myimage2_update.tar myimage:2
   ```

   出来上がった myimage2_update.tar は普通のコンテナと同じように扱うことができます。myimage:1 が存在しない場合はエラーとなります。

   詳しいコンテナアップデートの手順は 「10.4.2.6. コンテナの自動作成やアップデート」 を参考にしてください。

   繰り返し差分アップデートをすると、イメージサイズが大きくなってしまいます。ストレージ容量が不足する場合は、次に示す手順でコンテナを新しく構築してください。

2. コンテナを新しく構築する

   ベースとなるコンテナをアップデートして、そのコンテナに自分のアプリケーションを入れます。

   差分アップデートと異なり共有部分が無い為、コンテナ全体を送る必要があります。

自動的にイメージを作る方法は 「10.4.2.6. コンテナの自動作成やアップデート」 を参考にしてください。

10. Howto

10.1. ゲートウェイアプリケーションを開発する

10.1.1. VSCode を用いた開発の流れ

ゲートウェイアプリケーションを開発する場合の流れは以下のようになります。

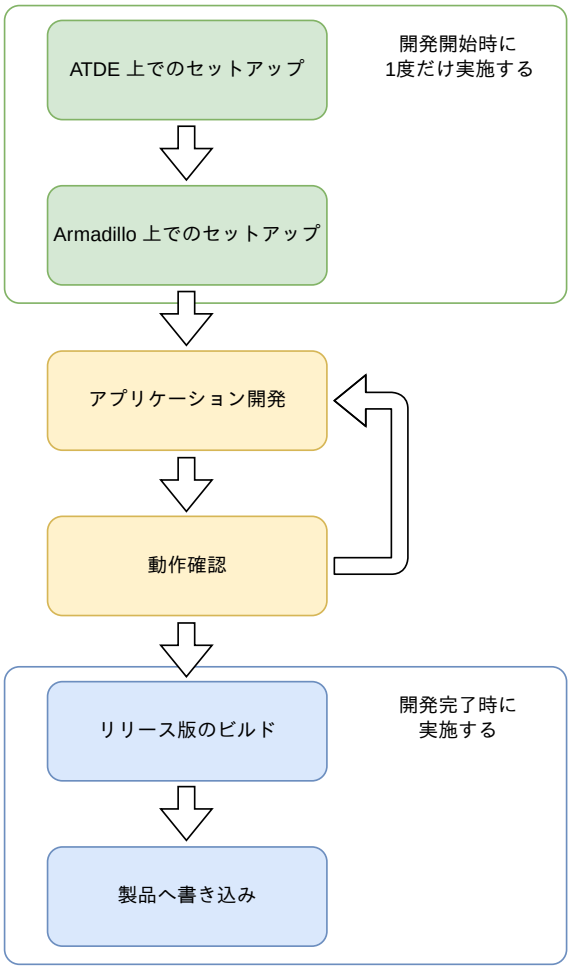


図 10.1 ゲートウェイアプリケーション開発の流れ

10.1.2. ATDE 上でのセットアップ

ここでは、開発開始時の ATDE 上でのセットアップ手順について説明します。ATDE をお使いでない場合は、先に 「4.2.1. ATDE のセットアップ」 を参照して ATDE のセットアップを完了してください。

10.1.2.1. ソフトウェアのアップデート

ATDE のバージョン v20230111 以上には、 VSCode がインストール済みのため新規にインストールする必要はありませんが、使用する前には最新版へのアップデートを行ってください。

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

図 10.2 ソフトウェアをアップデートする

10.1.2.2. ゲートウェイアプリケーションのダウンロード

gw-app-project-[VERSION].tar.gz という名前のファイルを こちら [https://download.atmark-techno.com/armadillo-iot-a6e/vscode-projects/]からダウンロードし、展開してください。以下は、ホームディレクトリ直下に展開する例です。

```
[ATDE ~]$ tar xzf gw-app-project-[VERSION].tar.gz
```

図 10.3 ゲートウェイアプリケーションを展開する

10.1.2.3. 初期設定

初期設定では主に Armadillo と SSH で接続するための秘密鍵と公開鍵の生成を行います。この手順は VSCode を使って行いますので、ゲートウェイアプリケーションディレクトリへ移動して VSCode を起動してください。VSCode を起動するには code コマンドを実行します。

```
[ATDE ~]$ cd gw-app-project-[VERSION]
[ATDE ~/gw-app-project-[VERSION]]$ code .
```

図 10.4 初期設定を行う

VSCode を起動すると、日本語化エクステンションのインストールを提案してくることがあります。その時に表示されるダイアログに従ってインストールを行うと VSCode を日本語化できます。

VSCode が起動したら [Terminal] メニューから [Run Task…] を選択し更に表示された項目から、[Generate development swu] を選択します。

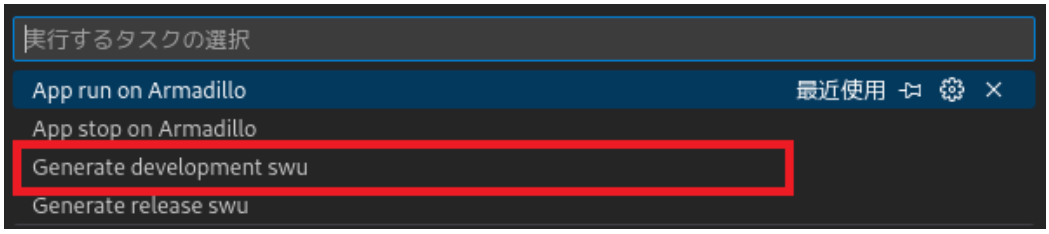


図 10.5 VSCode で初期設定用の swu を作成する

選択すると、 VSCode の下部に以下のようなターミナルが表示されます。

図 10.6 VSCode のターミナル

このターミナル上で以下のように入力してください。

```
 *  Executing task: ./scripts/setup_armadillo.sh

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): ❶
Enter same passphrase again:
Your identification has been saved in config/ssh/id_rsa
Your public key has been saved in config/ssh/id_rsa.pub
:(省略)

 *  Terminal will be reused by tasks, press any key to close it.  ❷
```

図 10.7 SSH 用の鍵を生成する

❶ パスフレーズを設定します。設定しない場合は何も入力せず Enter を押します。

❷ ここで何か任意のキーを押すとターミナルが閉じます。

Armadillo の初期設定を行う **scripts/setup_armadillo.swu** が作成されたことを確認してください。

10.1.2.4. ssh_config の編集

ATDE 上で操作するためにゲートウェイアプリケーションのディレクトリに入っている config/ssh_config ファイルを編集して IP アドレスを書き換えてください。

```
[ATDE ~/gw-app-project-[VERSION]]$ code config/ssh_config
Host Armadillo
    Hostname 0.0.0.0 ❶
    User root
    Port 22
    IdentityFile config/ssh/id_rsa
```

図 10.8 ssh_config を編集する

❶ Armadillo の IP アドレスに置き換えてください。

10.1.3. Armadillo 上でのセットアップ

10.1.3.1. 初期設定用 SWU イメージの書き込み

ここでは、開発開始時の Armadillo 上でのセットアップ手順について説明します。事前に 「10.9.2. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。その後、 「10.1.2.3. 初期設定」 で作成した **scripts/setup_armadillo.swu** を 「10.9.3. イメージのインストール」 を参照して Armadillo へインストールしてください。Armadillo にパスフレーズを設定した場合は、アプリケーションを Armadillo へ転送する時にパスフレーズの入力を求められることがあります。

10.1.4. ゲートウェイアプリケーション開発

ゲートウェイアプリケーションではゲートウェイコンテナを利用します。ゲートウェイコンテナについての詳細は 「9.3.1. ゲートウェイコンテナとは」 をご参照ください。

10.1.4.1. ゲートウェイアプリケーションの設定ファイルの編集

ゲートウェイアプリケーションの設定ファイルは `app/config` ディレクトリに配置されています。設定ファイルの詳細については 「10.2.5. 設定ファイルの編集」 をご参照ください。

10.1.4.2. ゲートウェイアプリケーションの開始

ゲートウェイアプリケーションのメインファイルは `app/src/main.py` です。

VSCode の [Terminal] メニューから [Run Task…] を選択し更に表示された項目から、[App run on Armadillo] を選択すると、変更内容を反映したゲートウェイアプリケーションが起動します。

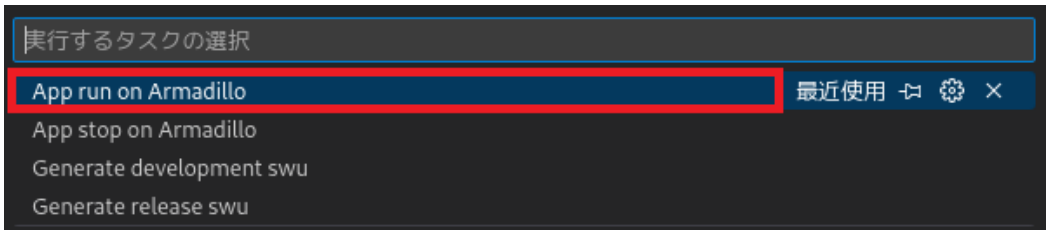


図 10.9 ゲートウェイアプリケーションを実行する

10.1.4.3. ゲートウェイアプリケーションの停止

VSCode の [Terminal] メニューから [Run Task…] を選択し更に表示された項目から、[App stop on Armadillo] を選択するとアプリケーションが停止します。

図 10.10 アプリケーションを停止する

10.1.5. ゲートウェイアプリケーションの拡張例

メインファイルを変更することで独自のアプリケーションを開始することも可能です。ゲートウェイアプリケーションの拡張例のファイルは **app/example** ディレクトリに配置してあります。実行する場合は

app/example ディレクトリのファイル一式を app/src ディレクトリにコピーしてください。拡張例のゲートウェイアプリケーションでは以下の動作を実行します。

- 5 秒毎に Count_value のカウントアップ
- Count_value が 100 に達すると 0 クリア

Count_value がカウントアップしていく様子はログファイルで確認できます。ゲートウェイアプリケーションのログについての詳細は 「10.2.9. ログ内容確認」 をご参照ください。

```
2023-01-26 11:05:35,115 <INFO> : {'data': {'Count_value': 0, 'timestamp': 1674698730}}
2023-01-26 11:05:45,150 <INFO> : {'data': {'Count_value': 1, 'timestamp': 1674698735}}
2023-01-26 11:05:45,165 <INFO> : {'data': {'Count_value': 2, 'timestamp': 1674698740}}
2023-01-26 11:05:45,175 <INFO> : {'data': {'Count_value': 3, 'timestamp': 1674698745}}
2023-01-26 11:05:55,202 <INFO> : {'data': {'Count_value': 4, 'timestamp': 1674698750}}
2023-01-26 11:05:55,215 <INFO> : {'data': {'Count_value': 5, 'timestamp': 1674698755}}
2023-01-26 11:06:05,242 <INFO> : {'data': {'Count_value': 6, 'timestamp': 1674698760}}
2023-01-26 11:06:05,255 <INFO> : {'data': {'Count_value': 7, 'timestamp': 1674698765}}
2023-01-26 11:06:15,282 <INFO> : {'data': {'Count_value': 8, 'timestamp': 1674698770}}
2023-01-26 11:06:15,295 <INFO> : {'data': {'Count_value': 9, 'timestamp': 1674698775}}
2023-01-26 11:06:25,323 <INFO> : {'data': {'Count_value': 10, 'timestamp': 1674698780}}
2023-01-26 11:06:25,335 <INFO> : {'data': {'Count_value': 11, 'timestamp': 1674698785}}
2023-01-26 11:06:35,362 <INFO> : {'data': {'Count_value': 12, 'timestamp': 1674698790}}
```

図 10.11 ログファイルの Count_value の出力例

ゲートウェイアプリケーションの拡張についての詳細は 「10.3. ゲートウェイコンテナを拡張する」をご参照ください。

10.1.6. リリース版のビルド

ここでは完成したゲートウェイアプリケーションをリリース版としてビルドする場合の手順について説明します。

ATDE 上で VSCode の [Terminal] メニューから [Run Task…] を選択し更に表示された項目から、[Generate release swu] を選択するとリリース版の SWU イメージが作成されます。

図 10.12 リリース版をビルドする

作成した SWU イメージは gw-app-project-[VERSION]/scripts/ ディレクトリ下に release_gw-app.swu というファイル名で保存されます。

10.1.7. 製品への書き込み

リリース版のゲートウェイアプリケーションを含んだ SWU イメージを Armadillo に展開します。事前に「10.9.2. SWU イメージの作成」 を参照して SWU の初期設定を行ってください。

リリース版のビルドで生成した SWU イメージを 「10.9.3. イメージのインストール」 を参照して Armadillo へインストールすると、Armadillo 起動時にアプリケーションも自動起動します。

10.2. ゲートウェイコンテナを動かす

Armadillo-IoT ゲートウェイ A6E にはゲートウェイコンテナがプリインストールされています。本章は、ゲートウェイコンテナを動かす方法について記載しています。

ゲートウェイコンテナは 「9.3.1. ゲートウェイコンテナとは」 に記載している通り、各インターフェースから取得するデータの設定や、接続するクラウドの情報を設定するだけで、コンテナ内で動作するアプリケーションを修正することなく、クラウドにデータを送信することができます。

10.2.1. ゲートウェイコンテナ利用の流れ

以下では、必要機器の接続やネットワークの設定は完了しているものとして説明を進めます。一連の流れは下記の通りです。

ゲートウェイコンテナでは AWS IoT Core と Azure IoT への接続をサポートしています。それぞれについて、データの可視化までを行うことが出来る環境を構築するためのテンプレートを提供しています。

1. ゲートウェイコンテナ起動確認
2. 接続先の クラウド 環境を構築 (クラウドにデータを送信する場合)
    a. AWS IoT Core
    b. Azure IoT Hub
3. コンフィグ 設定
    a. インターフェース設定
    b. 接続先クラウド設定
4. コンテナ起動・実行
5. コンテナ終了

10.2.2. ゲートウェイコンテナ起動確認

ゲートウェイコンテナは、デフォルトで Armadillo-IoT ゲートウェイ A6E に電源を入れると自動的に起動する設定となっています。Armadillo が起動し、ゲートウェイコンテナが起動・実行されると、 アプリケーション LED が点滅します。

10.2.3. 接続先の クラウド 環境を構築 (AWS)

AWS では、 AWS IoT Core と Amazon CloudWatch を組み合わせてデータの可視化を行います。本項では、 AWS 上で実施する設定を記載します。

手順中で使用するファイルは、Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) から予めダウンロードしておきます。

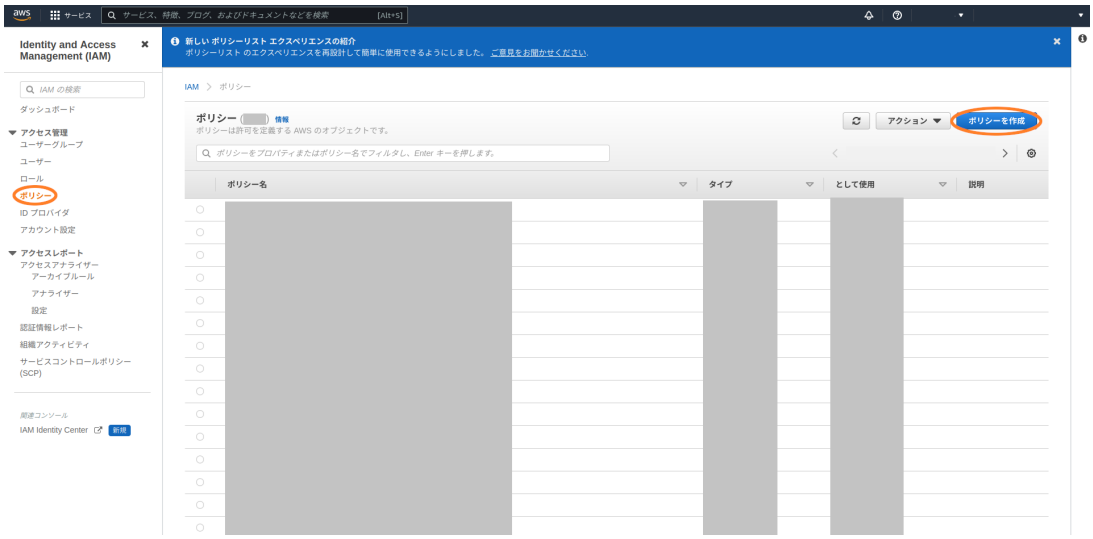
10.2.3.1. AWS アカウントを作成する

AWS アカウントの作成方法については、AWS 公式サイトの AWS アカウント作成の流れ https://aws.amazon.com/jp/register-flow/を参照してください。

10.2.3.2. IAM ユーザーを作成する

AWS IAM (Identity and Access Management) は、AWS リソースへのアクセスを安全に管理するためのウェブサービスです。IAM により、誰を認証(サインイン)し、誰にリソースの使用を承認する(アクセス許可を持たせる)かを管理することができます。

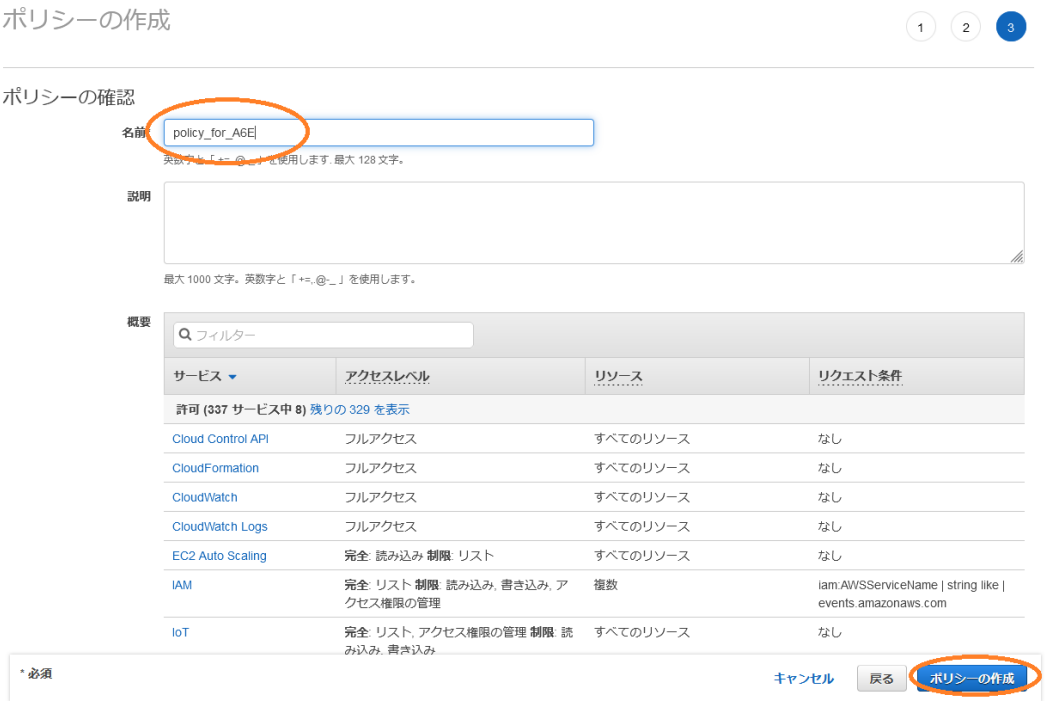
1. IAM へ移動し、「アクセス管理」→「ポリシー」を開き、「ポリシー作成」をクリックします。



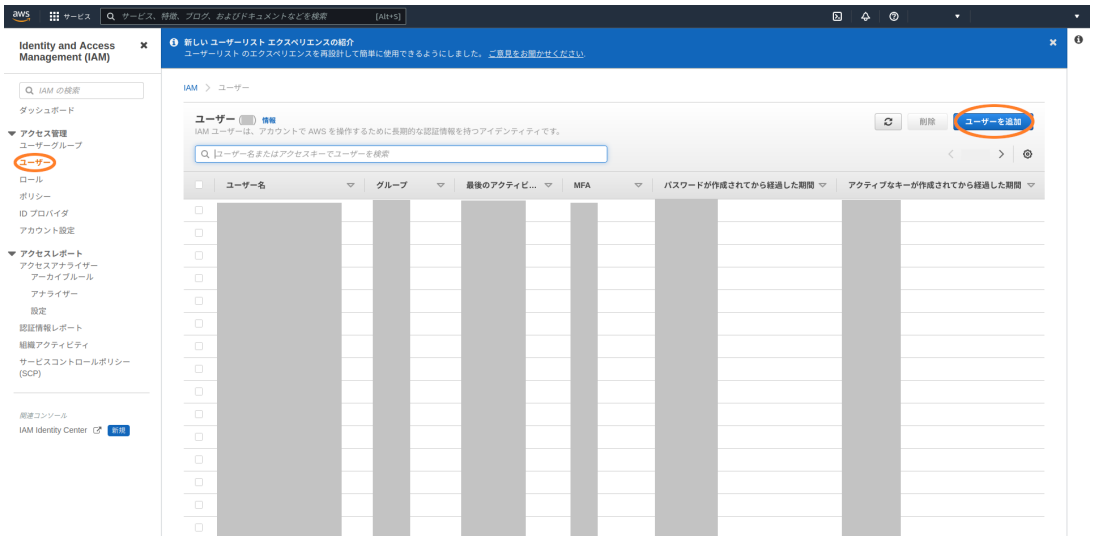
2. 「JSON」を選択し、「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) AWS フォルダ内の a6e_aws_iam_policy.json のファイルの内容を貼り付け、「次のステップ：タグ」をクリックします。

3. 何も選択せずに、「次のステップ：確認」をクリックします。

4. ポリシー名を入力し、「ポリシーの作成」をクリックします。ここでは、ポリシー名を "policy_for_A6E" としています。



5. IAM から、「アクセス管理」→「ユーザー」を開き、「ユーザーを追加」をクリックします。



6. 下記の通り入力、選択し、「次のステップ：アクセス権限」をクリックします。

   - ユーザー名を入力する
   - AWS 認証情報タイプは両方選択する
   - コンソールのパスワードは「自動生成パスワード」を選択する
   - パスワードのリセットが必要にチェックを入れる

7. 「既存のポリシーを直接アタッチ」をクリックし、先ほど作成したポリシーを選択して、「次のステップ：タグ」をクリックします。

8. 何も選択せずに、「次のステップ：確認」をクリックします。
9. 内容を確認し、「ユーザーの作成」をクリックします。
10. 「.csv のダウンロード」をクリックし、"new_user_credentials.csv" をダウンロードして、「閉じる」をクリックします。

10.2.3.3. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する

AWS IoT Core に登録する Thing 名は Armadillo のシリアル番号を使用します。環境設定時、パラメータに指定する必要があるため、下記のコマンドを実行しシリアル番号を取得します。

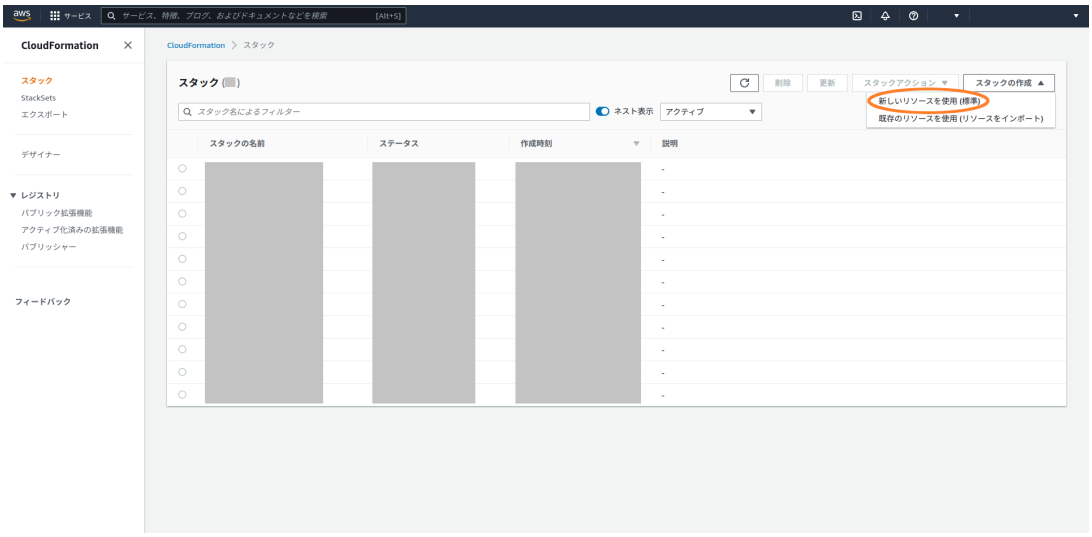
```
armadillo:~# hexdump -v -s 0xa0 -n 8 -e '/4 "%08X"' /sys/bus/nvmem/devices/imx-ocotp0/nvmem | cut
-c 5-
00CD11112222 ❶
```

❶ この場合、00CD11112222 がシリアル番号になります

10.2.3.4. AWS IoT Core と Amazon CloudWatch の設定を行う

AWS IoT Core に送信したデータを Amazon CloudWatch のダッシュボード上で可視化します。ここでは、CloudFormation を用いて AWS IoT Core と Amazon CloudWatch の設定を行います。

1. CloudFormation へ移動し、「スタックの作成」→「新しいリソースを使用(標準)」をクリックします。



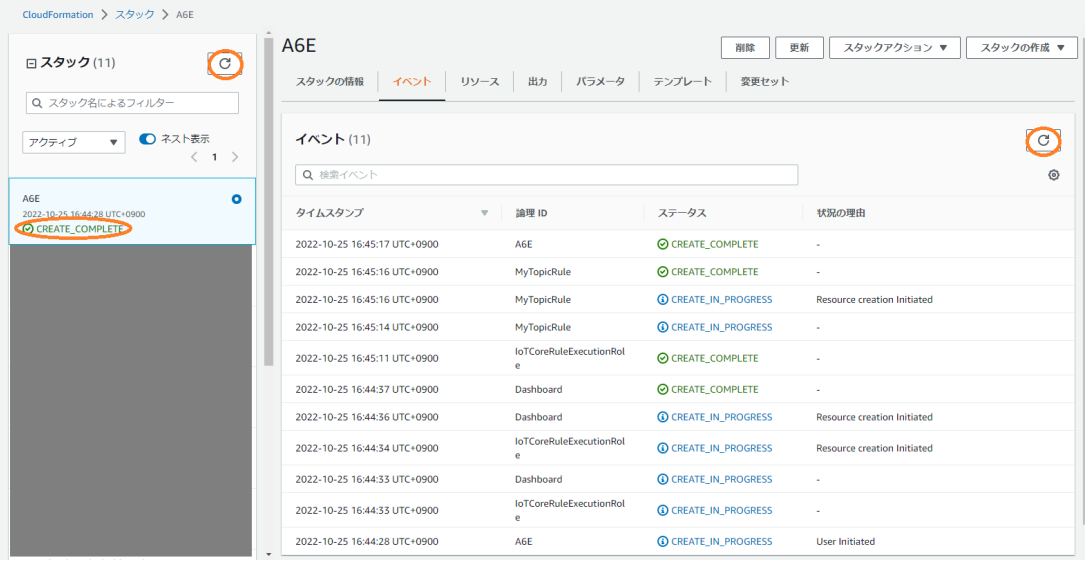
2. 「テンプレートファイルのアップロード」で「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) AWS フォルダ内の `a6e_aws_cfn_template.yml` を選択し、「次へ」をクリックします。

3. スタック名を入力します。また、「10.2.3.3. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する」で取得したシリアル番号をパラメータに指定し、「次へ」をクリックします。

4. そのまま「次へ」をクリックします。

5. チェックボックスを選択し、「スタックの作成」をクリックします。

6. 作成したスタックのステータスが"CREATE_COMPLETE" になったら作成完了です。



10.2.3.5. 設定に必要となるパラメータを取得する

「10.2.5.2. 接続先の クラウド 情報の設定」 で設定するパラメータを取得します。

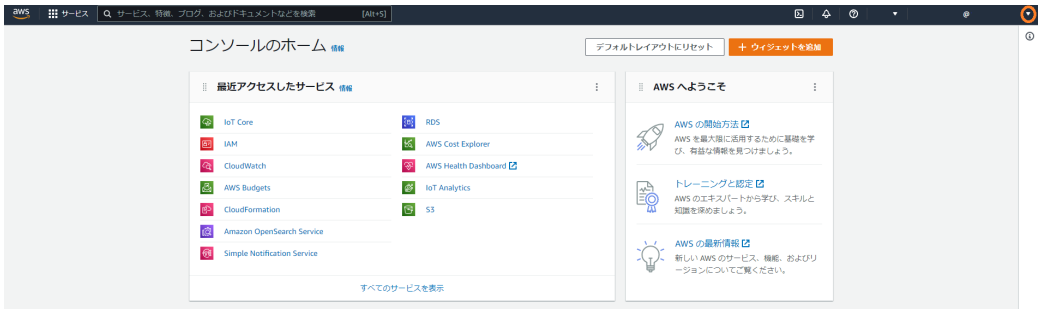
1. AWS IoT Core エンドポイント

1. IoT Core へ移動し、サイドバー下部にある設定をクリックします。

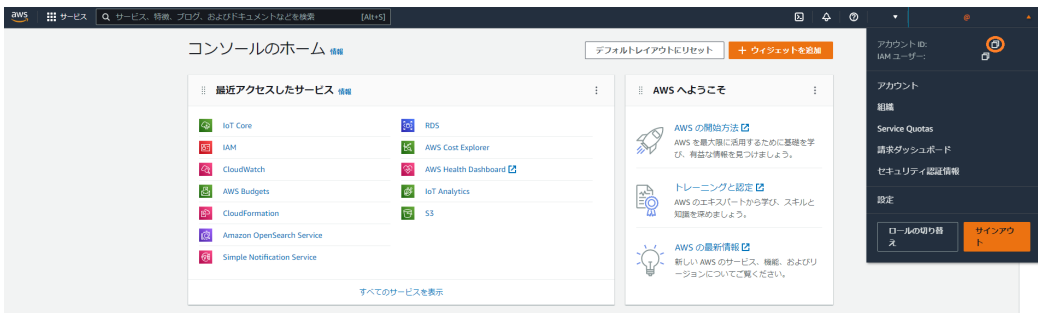
2. IoT Core エンドポイントが表示されます。

2. アカウント ID

1. AWS コンソール画面右上の ▼ をクリックします。



2. 下記画像の丸で囲んだマークをクリックすると、コピーすることができます。



10.2.4. 接続先の クラウド 環境を構築 (Azure)

Azure の場合は、 Azure IoT Hub にデータを送信します。本項では、 Azure portal 上で実施する設定を記載します。

手順中で使用するファイルは、Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) にアップロードしています。

10.2.4.1. Microsoft アカウントを作成する

Microsoft アカウントの作成については、Microsoft 公式ページ https://account.microsoft.com/ を参照してください。なお、サブスクリプションの設定も必要となります。

10.2.4.2. リソースグループを作成する

リソースグループの作成を行います。

1. Azure portal から [リソース グループ] を開き、[作成] を選択します。
2. サブスクリプションとリージョンを選択し、リソースグループ名を入力の後、[確認および作成] を選択します。

10.2.4.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う

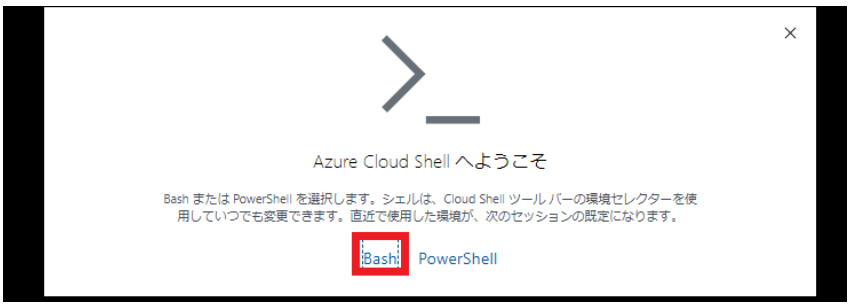
ここでは、データの送信先となる Azure IoT Hub と、デバイスプロビジョニングのヘルパーサービスである Azure IoT Hub Device Provisioning Service (以降、DPS と記載) の設定を行います。

以下の手順はアットマークテクノが提供する設定ファイルを用いて設定を行っていますが、Azure portal で作成した Azure IoT Hub / DPS に接続することも可能です。DPS の登録グループ機能を用いてデバイスプロビジョニングを行うため、以下のドキュメントを参考に、「Armadillo-IoT ゲートウェイ A6E クラウド設定データ」ファイル (a6e-gw-container-cloudsetting-[VERSION].zip) の Azure/cert 以下にあるファイルを中間証明書として登録してください。

https://learn.microsoft.com/ja-jp/azure/iot-dps/tutorial-custom-hsm-enrollment-group-x509?tabs=windows&pivots=programming-language-ansi-c#create-an-enrollment-group

1. Azure portal https://account.microsoft.com/ にサインインします。
2. Cloud Shell アイコンを選択し、 Azure Cloud Shell を起動します。

3. [Bash] を選択します。



4. ストレージアカウントの設定を行います。サブスクリプションを選択し、ストレージの作成をクリックすると自動的にストレージアカウントが作成されます。

5. Cloud Shell が起動したら、以下のコマンドで Armadillo-IoT ゲートウェイ A6E クラウド設定データをダウンロードします。

```
[Azure: ~]$ wget https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-a6e/
container/a6e-gw-container-cloudsetting-[VERSION].zip
[Azure: ~]$ unzip a6e-gw-container-cloudsetting-[VERSION].zip -d a6e-gw-container-cloud-
```

```
setting
[Azure: ~]$ cd a6e-gw-container-cloud-setting/Azure
```

図 10.13 Armadillo-IoT ゲートウェイ A6E クラウド設定データをダウンロードする

6. Cloud Shell 上でエディタを開き、コンフィグファイルを編集します。

```
[Azure: ~]$ code a6e_azure_create_hubdps.conf
# Common Config
resourceGroup="" ❶
certificateFilePath="./cert/SE050C1.pem"

# IoT Hub Config
iotHubName="" ❷
skuName="S1"
skuUnit=1
partitionCount=4

# DPS Config
provisioningServiceName="" ❸

# Certificate Config
certificateName="armadillo-iot-a6e-cert"
verified="true"

# Enrollment-group Config
groupName="armadillo-iot-a6e-group"
```

図 10.14 コンフィグファイルを編集する

❶ リソースグループを指定します

❷ 作成する Azure IoT Hub 名を入力します

❸ 作成する DPS 名を入力します

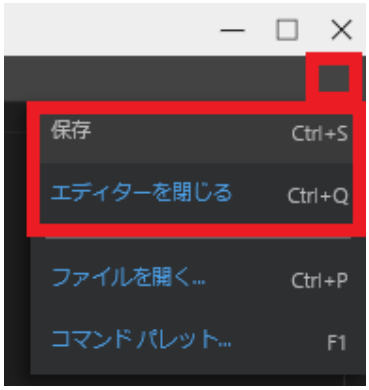
```
# Common Config
resourceGroup="armadillo"
certificateFilePath="./cert/SE050C1.pem"

# IoT Hub Config
iotHubName="armadillo-iothub"
skuName="S1"
skuUnit=1
partitionCount=4

# DPS Config
provisioningServiceName="armadillo-dps"

# Certificate Config
certificateName="armadillo-iot-a6e-cert"
verified="true"
```

```
# Enrollment-group Config
groupName="armadillo-iot-a6e-group"
```

図 10.15 コンフィグファイル設定例

Azure IoT Hub 名、 DPS 名はそれぞれグローバルで一意である必要があります。既に使用されている名称を指定した場合、エラーとなります。

コンフィグファイルの編集が終了したら、[保存] を行い、 [エディターを閉じる] を選択し、エディタを終了します。



7. 設定スクリプトを実行し、 Azure IoT Hub と DPS の設定を行います。

```
[Azure: ~]$ chmod +x a6e_azure_create_hubdps.sh
[Azure: ~]$ ./a6e_azure_create_hubdps.sh
Starting to create IoT Hub.
: (省略)
Starting to create DPS.
{
: (省略)
  "name": "xxxxx",
  "properties": {
: (省略)
    "idScope": "0ne12345678", ❶
: (省略)
  },
: (省略)
}
: (省略)
Starting to link between IoT Hub and DPS.
: (省略)
Starting to upload certificate.
: (省略)
Starting to create enrollment group.
: (省略)
Completed!
```

図 10.16 Azure IoT Hub と DPS の設定を実行する

❶ 環境設定時に使用するため、控えておきます

10.2.5. 設定ファイルの編集

利用したい内容に合わせて、 設定ファイルを編集します。設定内容はコンテナ起動時の内容が適用されるため、一度コンテナを終了させます。

```
[armadillo ~]# podman stop a6e-gw-container
a6e-gw-container
```

10.2.5.1. インターフェース設定

インターフェースの動作設定を行います。設定ファイルは /var/app/rollback/volumes/gw_container/config/sensing_mgr.conf です。

```
[DEFAULT]
; cloud_config=true or false
cloud_config=false
; send_cloud=true or false
send_cloud=false
; cache=true or false
cache=false
; send_interval[sec]
send_interval=10
; data_send_oneshot=true or false
data_send_oneshot=false
; wait_container_stop[sec]
wait_container_stop=0

[LOG]
file=true
stream=true

[CPU_temp]
; polling_interval[sec]
polling_interval=1
send_cloud=true

[DI1]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=

[DI2]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=

[DO1]
```

```
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=

[DO2]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=

[RS485_Data1]
;[RS485_Data1] ~ [RS485_Data4]
method=none
baudrate=
data_size=
; parity=none or odd or even
parity=
; stop=1 or 2
stop=
device_id=
func_code=
register_addr=
register_count=
; endian=little or big
endian=
; interval[sec]
interval=
; data_offset is option
data_offset=
; data_multiply is option
data_multiply=
; data_devider is option
data_devider=
```

図 10.17 /var/app/rollback/volumes/gw_container/config/sensing_mgr.conf のフォーマット

・全体動作設定

全体的な動作設定を行います。設定ファイル中の以下の箇所が該当します。

```
[DEFAULT]
; cloud_config=true or false
cloud_config=false
; send_cloud=true or false
send_cloud=false
; cache=true or false
cache=false
; send_interval[sec]
send_interval=10
; data_send_oneshot=true or false
data_send_oneshot=false
```

```
; wait_container_stop[sec]
wait_container_stop=0
```

表 10.1 [DEFAULT] 設定可能パラメータ

項目	概要	設定値	内容
cloud_config	クラウドからの設定変更を許容するか	true	許容する
		(デフォルト)false	無視する
send_cloud	クラウドにデータを送信するか	true	送信する
		(デフォルト)false	送信しない
cache	キャッシュ実施可否	true	キャッシュを実施する
		(デフォルト)false	キャッシュを実施しない
send_interval	データ送信間隔[sec]	1～10	この値に従って、クラウドへデータを送信する
data_send_oneshot	データ取得後コンテナを終了させるか	true	1 回データを取得し、コンテナを終了します。コンテナ終了通知をトリガに間欠動作を行う(「8.5. 状態遷移トリガにコンテナ終了通知を利用する」)場合は、この設定にする必要があります。
		(デフォルト)false	コンテナの実行を継続する (設定したインターバルでデータを取得する)
wait_container_stop	コンテナ終了までの待ち時間[sec]	0～60	data_send_oneshot が true の場合、クラウドへのデータ送信後、設定した時間 wait してからコンテナを終了する [a]

[a]現時点では 0 を設定してください

クラウドへのデータ送信は send_interval で指定した間隔毎に行います。値の取得間隔は、後述の通り各項目毎に指定することができます。値を取得するタイミングとクラウドへのデータ送信のタイミングを近くするためには、値の取得間隔より send_interval を短くするか、同じにすることを推奨します。

・ ログ出力

取得したデータのログを ログファイルに保存したり、コンソールに出力することが可能です。設定ファイル中の以下の箇所が該当します。

```
[LOG]
file=true
stream=true
```

表 10.2 [LOG] 設定可能パラメータ

項目	概要	設定値	内容
FILE_LOG	ログファイルに出力するか	(デフォルト)true	出力する
		false	出力しない

項目	概要	設定値	内容
STREAM_LOG	コンソールに出力するか	(デフォルト)true	出力する
		false	出力しない

・CPU_temp

CPU 温度読み出しに関する設定を行います。設定ファイル中の以下の箇所が該当します。

```
[CPU_temp]
; polling_interval[sec]
polling_interval=1
send_cloud=true
```

表 10.3 [CPU_temp] 設定可能パラメータ

項目	概要	設定値	内容
polling_interval	データ取得間隔[sec]	1～3600	この値に従って、CPU 温度を読み出します
send_cloud	クラウドデータ送信可否	(デフォルト)true	送信する
		false	送信しない

・接点入力

接点入力に関する設定を行います。設定ファイル中の以下の箇所が該当します。

```
[DI1]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=

[DI2]
; type=polling or edge
type=
; interval[sec]
interval=
; edge_type=falling or rising or both
edge_type=
```

表 10.4 [DI1,DI2] 設定可能パラメータ

項目	概要	設定値	内容
type	動作種別	(空欄) or none	接点入力状態取得を行わない
		polling	ポーリング
		edge	エッジ検出
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します
edge_type	エッジ検出設定	falling	立ち下がりエッジ
		rising	立ち上がりエッジ
		both	両方

・接点出力

接点出力に関する設定を行います。設定ファイル中の以下の箇所が該当します。「表 10.1. [DEFAULT] 設定可能パラメータ」 において、クラウドと通信しない場合はゲートウェイコンテナ起動後に設定した内容を出力します。クラウドと通信する場合は、「10.2.7.2. クラウドからのデバイス制御」 がトリガとなり、出力を開始します。

```
[DO1]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=

[DO2]
; output_state=high or low
output_state=
; output_time[sec]
output_time=
; output_delay_time[sec]
output_delay_time=
```

表 10.5 [DO1,DI2] 設定可能パラメータ

項目	概要	設定値	内容
output_state	出力状態	high	High
		low	Low
output_time	出力時間[sec]	1～3600	出力コマンド実行後に output_state で指定したレベルを出力する時間。 0 を指定すると永続的に出力します。
output_delay_time	出力遅延時間[sec]	0	出力コマンド実行後、指定した時間遅延して出力します。

・RS485

RS485 に関する設定を行います。設定ファイル中の以下の箇所が該当します。なお、RS485_Data1 から RS485_Data4 まで、4 個のデータについて設定することができます。デフォルトでは RS485_Data1 のみファイルに記載されているため、RS485_Data2, RS485_Data3, RS485_Data4 については適宜コピーして記載してください。

```
[RS485_Data1]
;[RS485_Data1] ~ [RS485_Data4]
method=none
baudrate=
data_size=
; parity=none or odd or even
parity=
; stop=1 or 2
stop=
device_id=
func_code=
register_addr=
register_count=
; endian=little or big
```

```
endian=
; interval[sec]
interval=
; data_offset is option
data_offset=
; data_multiply is option
data_multiply=
; data_devider is option
data_devider=
```

表 10.6 [RS485_Data1, RS485_Data2, RS485_Data3, RS485_Data4] 設定可能パラメータ

項目	概要	設定値	内容
method	通信種別	none	RS485 を利用しない
		rtu	Modbus-RTU
data_size	データサイズ	8	
baudrate	ボーレート	1200～38400[bps]	通信速度を指定します
parity	パリティビット	none	None
		odd	Odd
		even	Even
stop	ストップビット	1	1
		2	2
device_id	Modbus スレーブ機器のデバイス ID	0x01 ～ 0xF7	
func_code	ファンクションコード	0x03 or 0x04	
register_addr	レジスタアドレス	機器依存	値を読み出すレジスタのアドレスを指定
register_count	読み出しレジスタ数	1 or 2	一度に読み出すレジスタ数を指定
endian	エンディアン設定	little	リトルエンディアン
		big	ビッグエンディアン
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します
data_offset	読み出し値に加算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値に加算する値を指定します
data_multiply	読み出し値と乗算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と乗算する値を指定します
data_devider	読み出し値と除算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と除算する値を指定します

10.2.5.2. 接続先の クラウド 情報の設定

クラウド と連携する場合、接続先の クラウド の情報を入力する必要があります。設定ファイルは /var/app/rollback/volumes/gw_container/config/cloud_agent.conf です。

```
[CLOUD]
SERVICE = ;AWS or AZURE

[LOG]
FILE_LOG = true
STREAM_LOG = true

[AWS]
```

```
AWS_IOT_HOST =
AWS_IOT_REGION =
AWS_IOT_ACCOUNTID =
AWS_IOT_ENDPOINT =
AWS_IOT_CERT_FILE = /cert/device_cert.pem
AWS_IOT_POLICY_FILE = /config/aws_iot_policy.json
AWS_IOT_SHADOW_ENDPOINT =
AWS_IOT_CA_FILE = /cert/AmazonRootCA1.pem
AWS_IOT_PKCS11_PATH = /usr/lib/plug-and-trust/libsss_pkcs11.so
AWS_IOT_KEY_LABEL = sss:100100F0
AWS_ACCESS_KEY =
AWS_SECRET_KEY =
AWS_IOT_PORT = 443
AWS_IOT_PIN =

[AZURE]
AZURE_IOT_DEVICE_DPS_ENDPOINT = global.azure-devices-provisioning.net
AZURE_IOT_DEVICE_DPS_ID_SCOPE =
AZURE_IOT_KEY_FILE = /cert/key.pem
AZURE_IOT_CERT_FILE = /cert/device_cert.pem
```

図 10.18 /var/app/rollback/volumes/gw_container/config/cloud_agent.conf のフォーマット

・ 接続先の クラウドサービス 種別

ゲートウェイコンテナが接続するクラウドサービスの種別を指定します。設定ファイル中の以下の箇所が該当します。

```
[CLOUD]
SERVICE = ;AWS or AZURE
```

表 10.7 [CLOUD] 設定可能パラメータ

項目	概要	設定値	内容
SERVICE	接続先クラウドサービスを指定	AWS	AWS IoT Core に接続
		Azure	Azure IoT に接続

・ ログ出力

クラウド との接続状態や送受信したデータのログを ログファイルに保存したり、コンソールに出力することが可能です。設定ファイル中の以下の箇所が該当します。

```
[LOG]
FILE_LOG = true
STREAM_LOG = true
```

表 10.8 [CLOUD] 設定可能パラメータ

項目	概要	設定値	内容
FILE_LOG	ログファイルに出力するか	(デフォルト)true	出力する
		false	出力しない

項目	概要	設定値	内容
STREAM_LOG	コンソールに出力するか	(デフォルト)true	出力する
		false	出力しない

・AWS

ここでは、 AWS に接続する場合の設定内容を記載します。設定ファイル中の以下の箇所が該当します。

```
[AWS]
AWS_IOT_HOST =
AWS_IOT_REGION =
AWS_IOT_ACCOUNTID =
AWS_IOT_ENDPOINT =
AWS_IOT_CERT_FILE = /cert/device_cert.pem
AWS_IOT_POLICY_FILE = /config/aws_iot_policy.json
AWS_IOT_SHADOW_ENDPOINT =
AWS_IOT_CA_FILE = /cert/AmazonRootCA1.pem
AWS_IOT_PKCS11_PATH = /usr/lib/plug-and-trust/libsss_pkcs11.so
AWS_IOT_KEY_LABEL = sss:100100F0
AWS_ACCESS_KEY =
AWS_SECRET_KEY =
AWS_IOT_PORT = 443
AWS_IOT_PIN =
```

表 10.9 [CLOUD] 設定可能パラメータ

項目	概要	設定値・設定例	取得方法
AWS_IOT_HOST	IoT Core REST API エンドポイント(リージョンに準ずる)	(例) iot.ap-northeast-1.amazonaws.com	AWS IoT Core - コントロールプレーンエンドポイント [https://docs.aws.amazon.com/ja_jp/general/latest/gr/iot-core.html] から取得
AWS_IOT_REGION	リージョン	(例) ap-northeast-1	AWS リージョンエンドポイント[https://docs.aws.amazon.com/ja_jp/general/latest/gr/rande.html] から取得
AWS_IOT_ACCOUNTID	アカウント ID	(例) 111111111111	AWS マネジメントコンソール上から取得(参考:「10.2.3.5. 設定に必要となるパラメータを取得する」)
AWS_IOT_ENDPOINT	AWS IoT Core エンドポイント(リージョンに準ずる)	(例) https://iot.ap-northeast-1.amazonaws.com	AWS IoT Core - コントロールプレーンエンドポイント [https://docs.aws.amazon.com/ja_jp/general/latest/gr/iot-core.html] から取得
AWS_IOT_CERT_FILE	デバイス証明書ファイルパス	(デフォルト)/cert/device_cert.pem	変更不要
AWS_IOT_POLICY_FILE	AWS IoT Core ポリシーテンプレートファイルパス	(デフォルト)/config/aws_iot_policy.json	変更不要
AWS_IOT_SHADOW_ENDPOINT	AWS IoT Core エンドポイント	(例)xxxxxxxxx-ats.iot.ap-northeast-1.amazonaws.com	AWS IoT Core [設定] - [デバイスデータエンドポイント] から取得 (参考:「10.2.3.5. 設定に必要となるパラメータを取得する」)
AWS_IOT_CA_FILE	AWS IoT Core ルート CA ファイルパス	(デフォルト)/cert/AmazonRootCA1.pem	変更不要

項目	概要	設定値・設定例	取得方法
AWS_IOT_PKCS11_PATH	PKCS#11 ライブラリパス	(デフォルト)/usr/lib/plug-and-trust/libsss_pkcs11.so	変更不要
AWS_IOT_KEY_LABEL	利用する秘密鍵のラベル	(デフォルト)sss:100100F0	変更不要
AWS_ACCESS_KEY	アクセスキー	(例)AAAAAAAAAXXXXXXX	「10.2.3.2. IAM ユーザーを作成する」 でダウンロードした IAM ユーザー クレデンシャル情報
AWS_SECRET_KEY	シークレットキー	(例)sssssssddddddttttttttt	「10.2.3.2. IAM ユーザーを作成する」 でダウンロードした IAM ユーザー クレデンシャル情報
AWS_IOT_PORT	MQTT 接続ポート	(デフォルト)443	変更不要
AWS_IOT_PIN	PIN	-	指定不要

上記パラメータのうち、以下のパラメータは AWS IoT Core へのデバイス登録完了後クリアされます。デバイスを AWS IoT Core から削除した場合など再度デバイス登録を行いたい場合は、再度設定してください。

- AWS_IOT_ACCOUNTID
- AWS_ACCESS_KEY
- AWS_SECRET_KEY

- Azure

ここでは、 Azure に接続する場合の設定内容を記載します。設定ファイル中の以下の箇所が該当します。

```
[AZURE]
AZURE_IOT_DEVICE_DPS_ENDPOINT = global.azure-devices-provisioning.net
AZURE_IOT_DEVICE_DPS_ID_SCOPE =
AZURE_IOT_KEY_FILE = /cert/key.pem
AZURE_IOT_CERT_FILE = /cert/device_cert.pem
```

表 10.10 [CLOUD] 設定可能パラメータ

項目	概要	設定値・設定例	取得方法
AZURE_IOT_DEVICE_DPS_ENDPOINT	DPS エンドポイント	(デフォルト)global.azure-devices-provisioning.net	変更不要
AZURE_IOT_DEVICE_DPS_ID_SCOPE	Azure IoT Central ID スコープ	(例)0ne12345678	「図 10.16. Azure IoT Hub と DPS の設定を実行する」 で表示された内容を使用
AZURE_IOT_KEY_FILE	デバイスリファレンスキーファイルパス	(デフォルト)/cert/key.pem	変更不要
AZURE_IOT_CERT_FILE	デバイス証明書ファイルパス	(デフォルト)/cert/device_cert.pem	変更不要

10.2.6. コンテナ起動・実行

設定ファイルの修正が完了したら、コンテナを起動します。コンテナが起動すると、設定に従ってコンテナ内のアプリケーションが実行される仕組みとなっています。

```
[armadillo ~]# podman_start a6e-gw-container
Starting 'a6e-gw-container'
a3b719c355de677f733fa8208686c29424be24e57662d3972bc4131ab7d145ad
```

「表 10.1. [DEFAULT] 設定可能パラメータ」 でクラウドにデータを送信する設定を行った場合は、クラウド接続後、アプリケーション LED の状態が点滅から点灯に変化します。

10.2.6.1. Armadillo からクラウドに送信するデータ

Armadillo からクラウドに送信するデータは以下の通りです。

- デバイス情報

表 10.11 デバイス情報データー覧

項目	概要
DevInfo_SerialNumber	シリアル番号
DevInfo_LAN_MAC_Addr	LAN MAC アドレス
DevInfo_ABOS_Ver	Armadillo Base OS バージョン
DevInfo_Container_Ver	コンテナイメージバージョン

- CPU 温度

表 10.12 CPU 温度データー覧

項目	概要
CPU_temp	CPU 温度

- 接点入力

表 10.13 接点入力データー覧

項目	概要
DI1_polling	DI1 のポーリング結果
DI2_polling	DI2 のポーリング結果
DI1_edge	DI1 のエッジ検出結果
DI2_edge	DI2 のエッジ検出結果

- 接点出力

  クラウドに送信するデータはありません。

- RS485

表 10.14 RS485 データー覧

項目	概要
RS485_Data1	RS485_Data1 の読み出し値
RS485_Data2	RS485_Data2 の読み出し値
RS485_Data3	RS485_Data3 の読み出し値
RS485_Data4	RS485_Data4 の読み出し値

・ユーザースイッチ

表 10.15 ユーザースイッチ関連データ一覧

項目	概要
sw_state	ユーザースイッチの状態

クラウドにデータが届いているかどうかは、次項の方法で確認することができます。

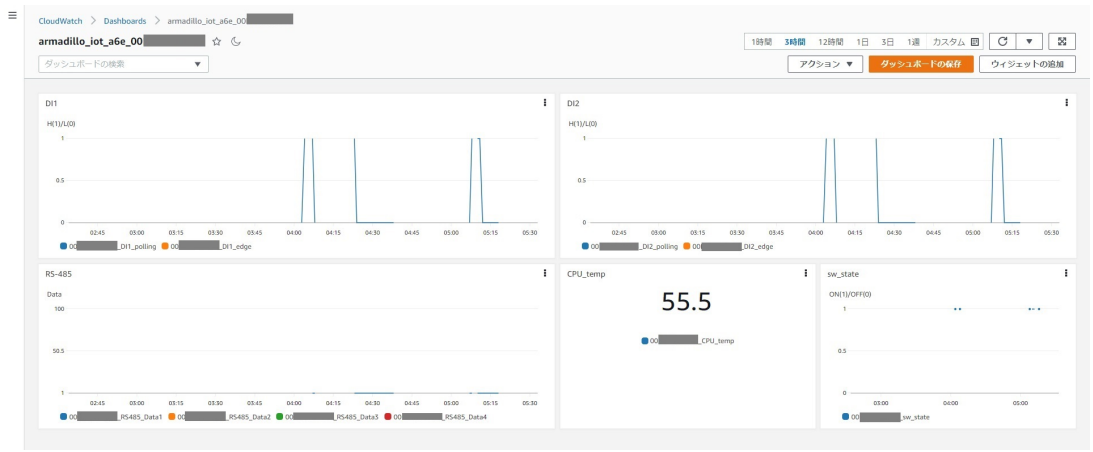
10.2.6.2. AWS 上でのデータ確認

Amazon CloudWatch ダッシュボードで、データが届いているかの確認を行う事ができます。

1. CloudWatch に移動し、「ダッシュボード」を選択します。

2. 「10.2.3.4. AWS IoT Core と Amazon CloudWatch の設定を行う」 で CloudWatch ダッシュボードが作成されています。ダッシュボード名は armadillo_iot_a6e_<シリアル番号> です。

3. ダッシュボード名をクリックすると、下記のような画面が表示されます。



・接点入力

・RS485

・CPU 温度

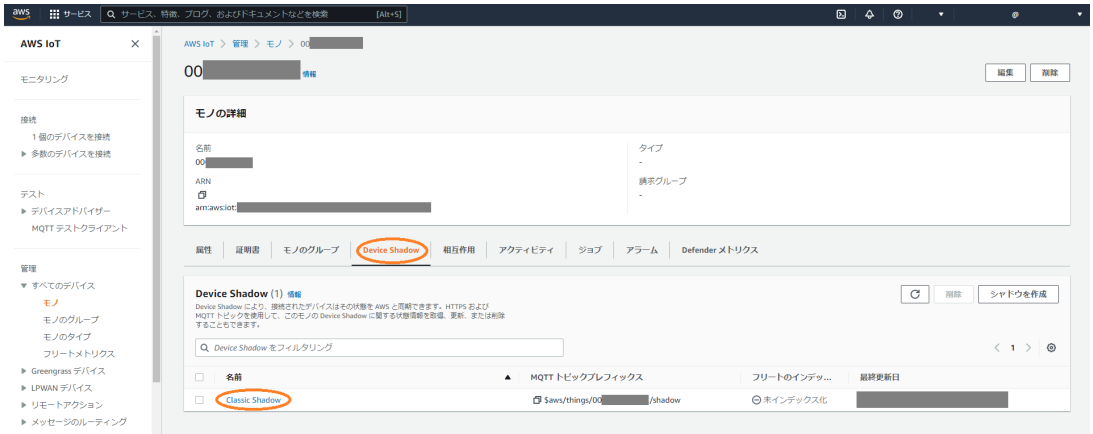
・ユーザースイッチ

また、実際にデバイスから届いているデータを確認する場合は、 AWS IoT Core の Device Shadow で確認を行います。

1. AWS IoT Core に移動し、「管理」→「すべてのデバイス」→「モノ」を選択します。

2. デバイスの名前は 「10.2.3.3. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する」で取得したシリアル番号で登録されています。

3. 「Device Shadow」の「Classic Shadow」を選択します。



4. 下記の通り、 Armadillo から送信されてきたデータを確認することができます。

10.2.6.3. Azure 上でのデータ確認

以下では可視化の手順を記載していますが、実際にデバイスから届いているデータを確認する場合は、Azure IoT Explorer を用いて確認することが可能です。詳細はこちらのドキュメント https://docs.microsoft.com/ja-jp/azure/iot-pnp/howto-use-iot-explorer をご参照ください。

Azure IoT Hub に登録されるデバイス ID は、デバイス認証に使用している証明書の CN となります。以下のコマンドで確認することが可能です。

```
[armadillo ~]# openssl x509 -noout -subject -in /var/app/rollback/
volumes/gw_container/cert/device_cert.pem | grep subject | awk '{print
$NF}'
```

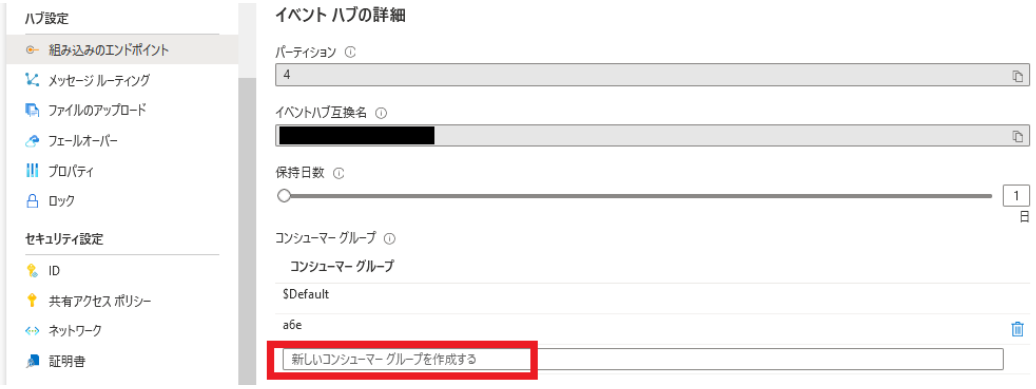
可視化の方法は様々ありますが、本書では一例として、Power BI を使用して Azure IoT Hub に送信したデータの可視化を行う方法を記載します。

以下の手順では、 「10.2.6.1. Armadillo からクラウドに送信するデータ」 のうち CPU_temp を例に記載します。

1. こちらのページで https://powerbi.microsoft.com/ja-jp/ Power BI アカウントを作成します。なお、 Pro アカウントでの登録が必要となります。

2. PowerBI にログインし、グループワークスペースを作成します。

3. Azure IoT Hub にコンシューマーグループを追加します。 Azure portal から [IoT Hub] を開き、「10.2.4.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 で作成した IoT Hub を選択します。[組み込みのエンドポイント] を選択し、[コンシューマーグループ] の下のテキストボックスに、新しいコンシューマーグループの名前を入力、保存します。



4. Azure IoT Hub のデータを Power BI のデータセットにルーティングする Azure Stream Analytics ジョブを作成します。

   Azure portal から [Stream Analytics ジョブ] を開き、 [Stream Analytics ジョブ] 概要ページで [作成] を選択します。

   [基本] タブに、 「表 10.16. Azure Stream Analytics ジョブ設定値」 の情報を入力し、 [確認と作成] を選択した後、 [作成] を選択して Stream Analytics ジョブを作成します。

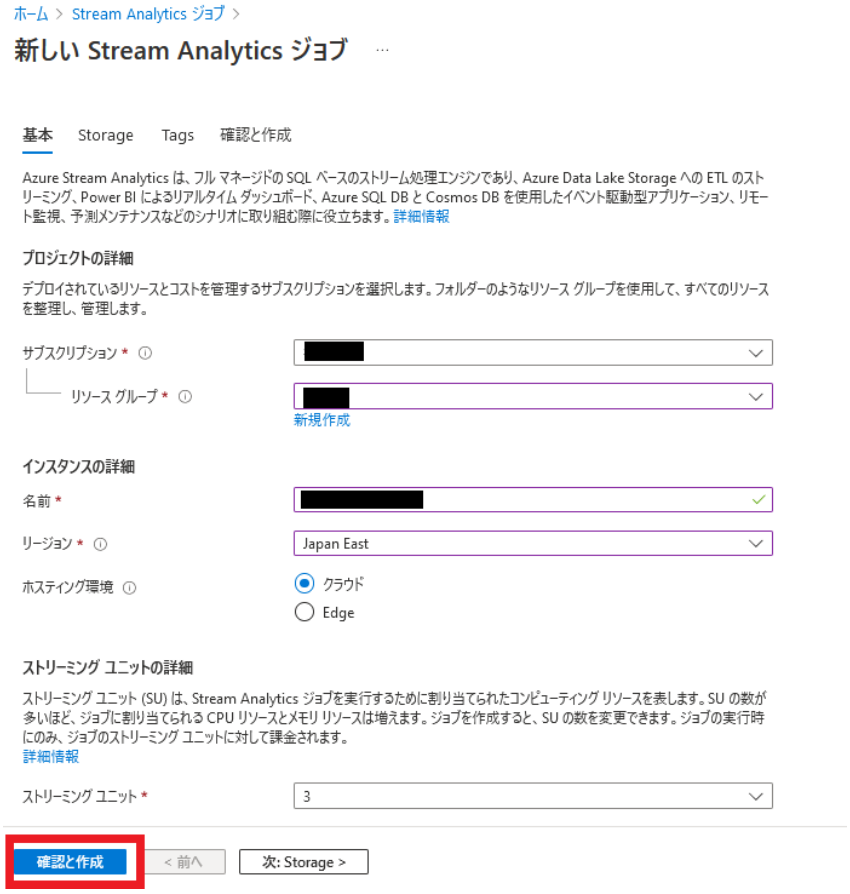


表 10.16 Azure Stream Analytics ジョブ設定値

項目	設定値
サブスクリプション	IoT Hub のサブスクリプション
リソースグループ	IoT Hub のサブスクリプション
名前	ジョブの名前(任意)
リージョン	IoT Hub のリージョン

5. Stream Analytics ジョブに入力を追加します。

   作成した Stream Analytics ジョブを開きます。

   [ジョブ トポロジ] - [入力] から [ストリーム入力の追加] を選択し、ドロップダウンリスト内の [IoT Hub] を選択します。

「表 10.17. Azure Stream Analytics ジョブ入力設定値」 の情報を入力し、それ以外の内容はデフォルトのまま [保存] を選択します。

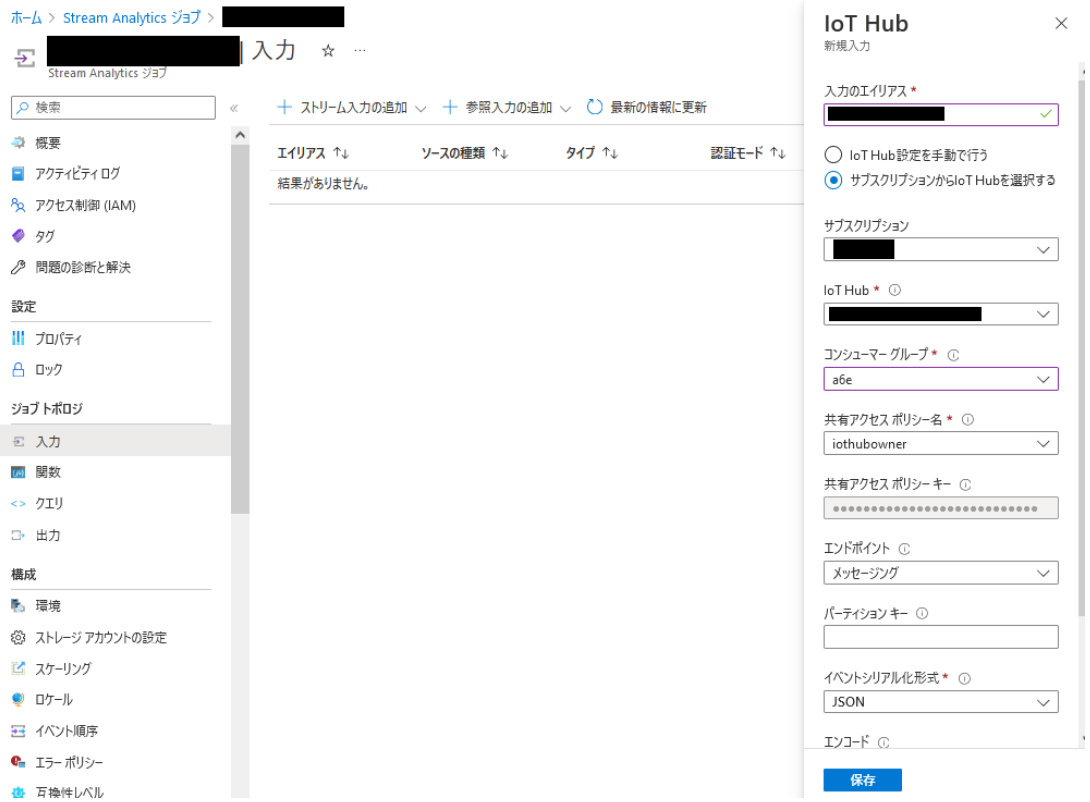


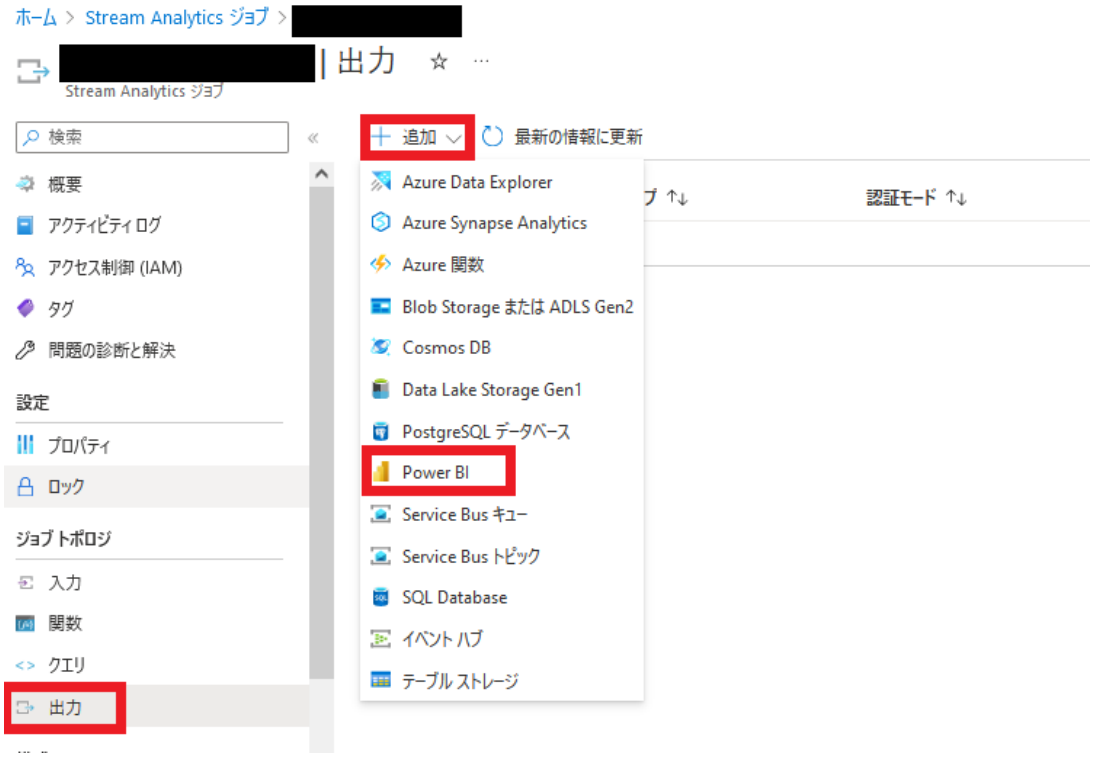
表 10.17 Azure Stream Analytics ジョブ入力設定値

項目	設定値
入力のエイリアス	一意の名前を入力
サブスクリプションから IoT Hub を選択する	選択
サブスクリプション	IoT Hub 用のサブスクリプション

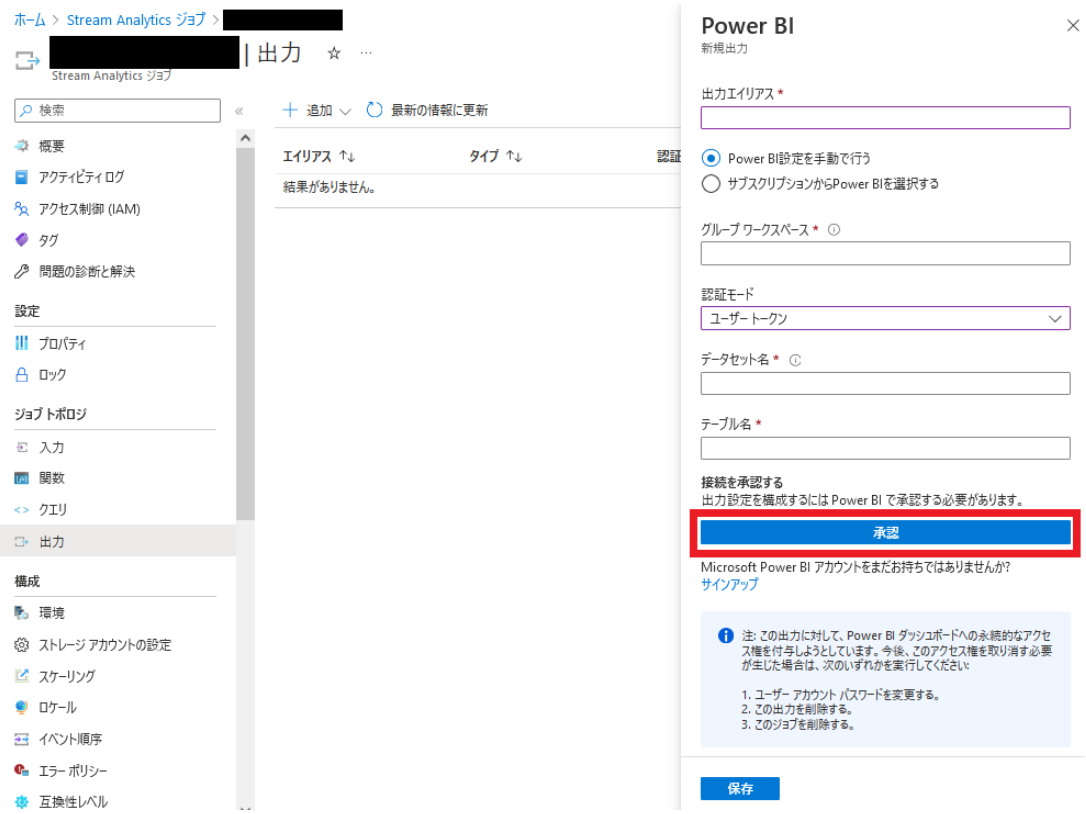
項目	設定値
IoT Hub	使用する IoT Hub
コンシューマーグループ	作成したコンシューマーグループを選択
共有アクセスポリシー名	iothubowner

6. Stream Analytics ジョブに出力を追加します。なお、複数の値を PowerBI で可視化する場合は、値の数分の出力設定が必要になります。

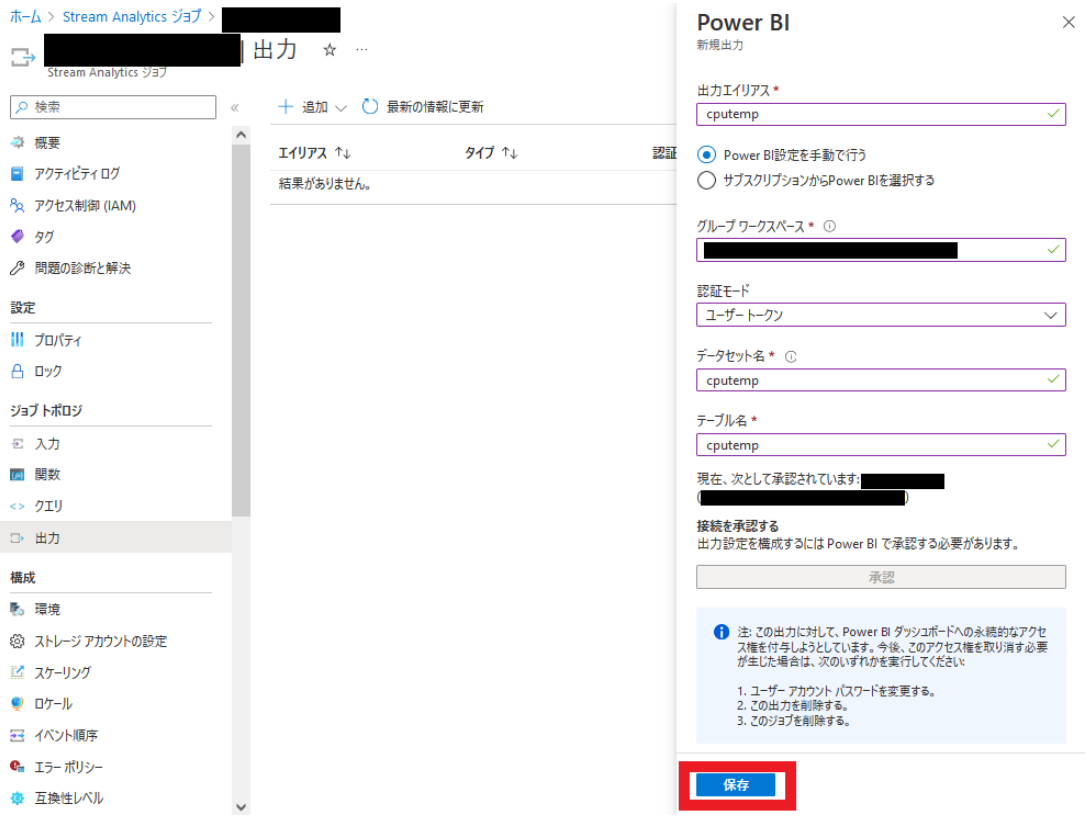
   [ジョブ トポロジ] - [出力] から [追加] を選択し、ドロップダウンリスト内の [Power BI] を選択します。



   [認証モード] で「ユーザートークン」を選択、[接続を承認する] の [承認] を選択し、Power BI アカウントにサインインします。

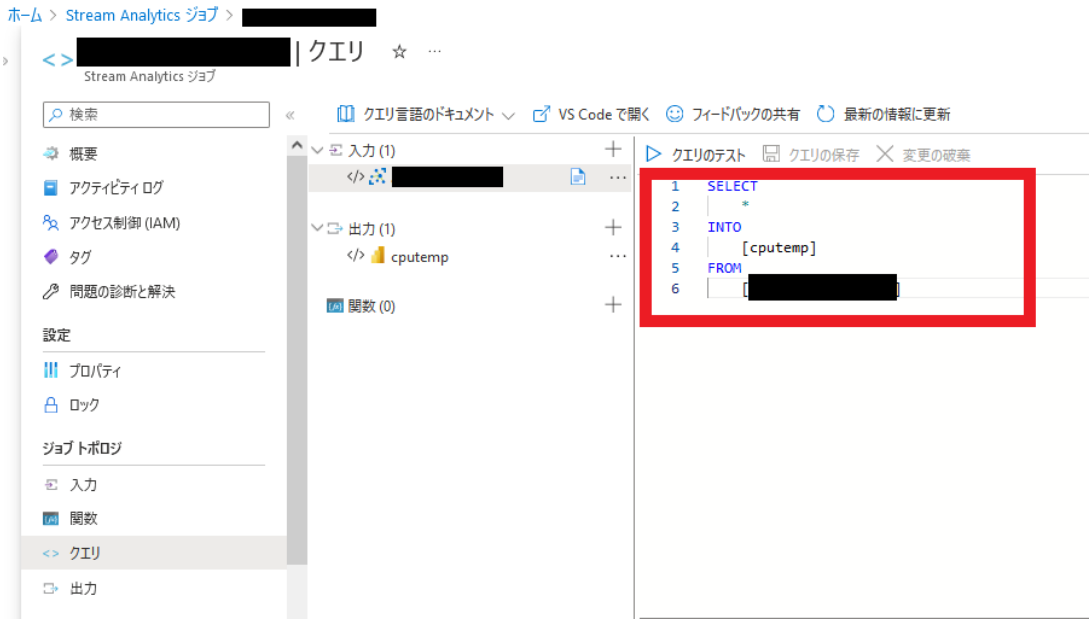


作成したグループワークスペースの ID を [グループワークスペース] に入力します。グループワークスペースの ID は、グループワークスペースの URL から取得することができます。[データセット名] と [テーブル名] は任意の値を指定してください。ここではそれぞれ `cputemp` を指定しています。情報登録完了後、 [保存] を選択します。



7. Stream Analytics ジョブのクエリを構成します。

   [ジョブ トポロジ] の [クエリ] を選択します。



   赤枠内にクエリを指定します。入力完了後、[クエリの保存] を選択してください。フォーマットは下記の通りです。 <パラメータ名> には、 「10.2.6.1. Armadillo からクラウドに送信するデータ」 の「項目」を指定してください。

```
SELECT
    <パラメータ名>,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [<ジョブ出力エイリアス名>]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE <パラメータ名> IS NOT NULL
```

これに従い、CPU_temp の場合は以下の通りとなります。

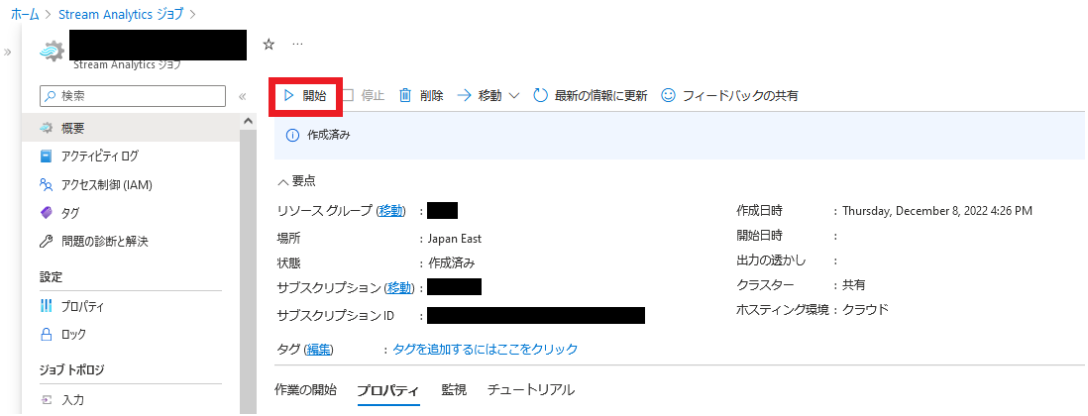
```
SELECT
    CPU_temp,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [cputemp]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE CPU_temp IS NOT NULL
```

なお、複数の出力がある場合は、クエリ入力欄に下記の通り複数のクエリを列挙してください。
INTO 句で指定するパラメータ(データセット名)が異なることに注意してください。

```
SELECT
    CPU_temp,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [cputemp]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE CPU_temp IS NOT NULL

SELECT
    DI1_polling,
    DATEADD(hour, 9, System.Timestamp) AS time,
    IoTHub.ConnectionDeviceId AS DeviceID
INTO
    [di1polling]
FROM
    [<ジョブ入力エイリアス名>] timestamp by dateadd(second, epoch, '1970-01-01T00:00:00Z')
WHERE DI1_polling IS NOT NULL
```

8. Stream Analytics ジョブを実行します。

   [概要] 画面で [開始] を選択します。

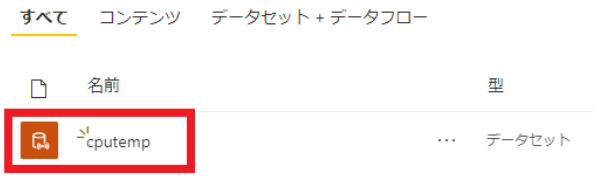


[ジョブの開始] 画面の [ジョブ出力の開始時刻] で [現在] が選択されていることを確認し、 [開始] を選択します。ジョブが正常に開始されると、[概要] 画面の [状態] が [実行中] に変わります。

9. ゲートウェイコンテナを停止している場合、下記のコマンドを実行しゲートウェイコンテナを開始します。

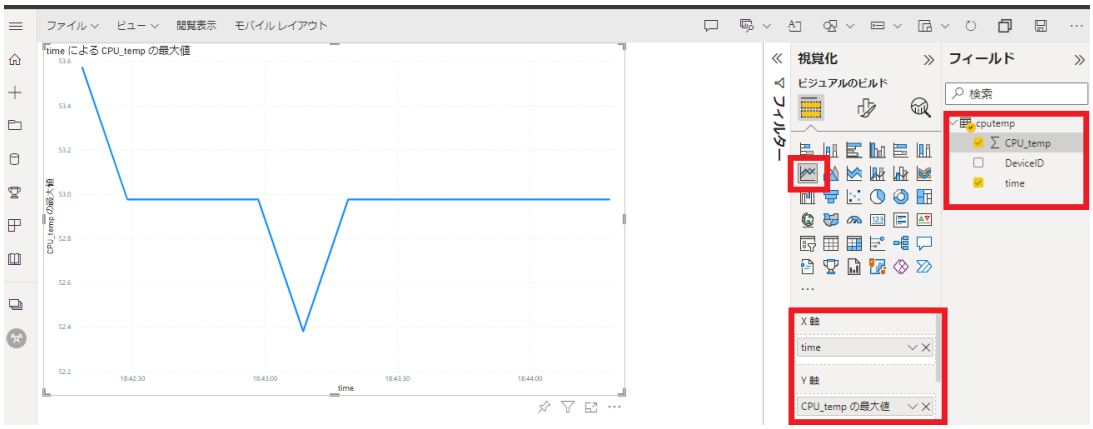
```
[armadillo ~]# podman_start a6e-gw-container
Starting 'a6e-gw-container'
a3b719c355de677f733fa8208686c29424be24e57662d3972bc4131ab7d145ad
```

10. PowerBI アカウントにサインインし、使用したワークスペースを右側のメニューから選択すると、 Stream Analytics ジョブ出力で指定した名称のデータセットが作成されています。



11. データセットの [レポートの作成] を選択します。

12. [視覚化] で [折れ線グラフ] を選択、X 軸に EventEnqueuedUtcTime 、 Y 軸に CPU_temp を指定することにより、グラフ化を行うことが出来ます。各設定を行った後、 [保存] すると、レポートが作成されます。



13. 複数のデータセットが存在している場合は、それぞれについてレポートの作成を行います。なお、各レポートを一括して表示したい場合はダッシュボード機能を選択してください。手順についてはこちらのドキュメント https://learn.microsoft.com/ja-jp/power-bi/create-reports/service-dashboard-create を参照してください。

10.2.7. クラウドからの操作

10.2.7.1. クラウドからのデータ設定

各インターフェースの設定については、「10.2.5.1. インターフェース設定」に記載している通り Armadillo 上の設定ファイルで行いますが、クラウドから設定値を変更することも可能です。

なお、クラウドからデータ設定を行うためには、「表 10.1. [DEFAULT] 設定可能パラメータ」の `cloud_config` を true に設定する必要があります。

設定を変更できる項目は以下の通りです。

- 接点入力設定
- 接点出力設定
- RS485 レジスタ読み出し

下記の手順でデータを設定します。

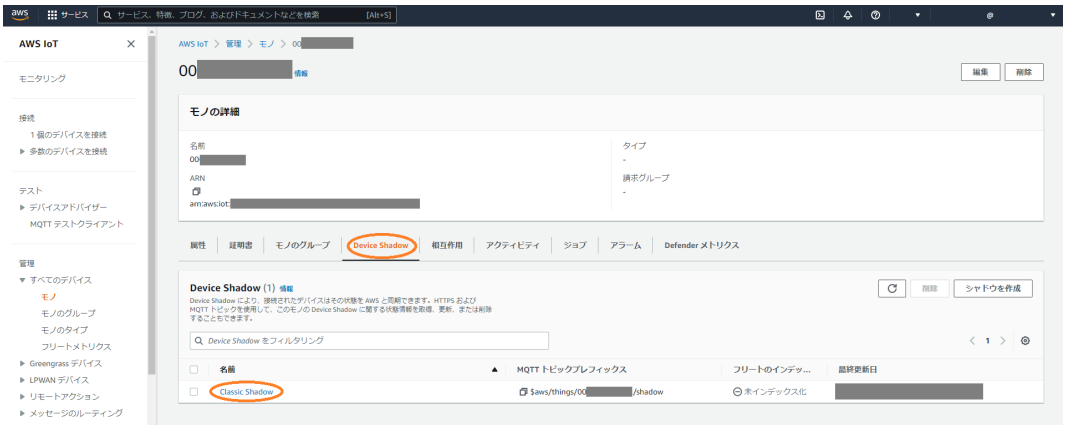
- AWS

  AWS IoT Core の Device Shadow を更新して設定を行います。

  1. AWS IoT Core に移動し、「管理」→「すべてのデバイス」→「モノ」を選択します。

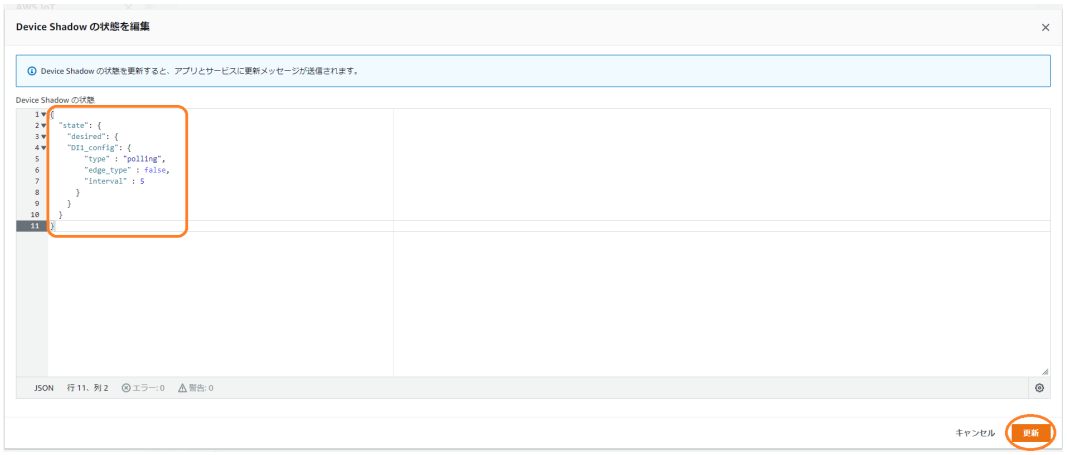
  2. デバイスの名前は「10.2.3.3. Armadillo-IoT ゲートウェイ A6E のシリアル番号を取得する」で取得したシリアル番号で登録されています。

  3. 「Device Shadow」の「Classic Shadow」を選択します。



4. Device Shadow ドキュメントの「編集」を選択します。

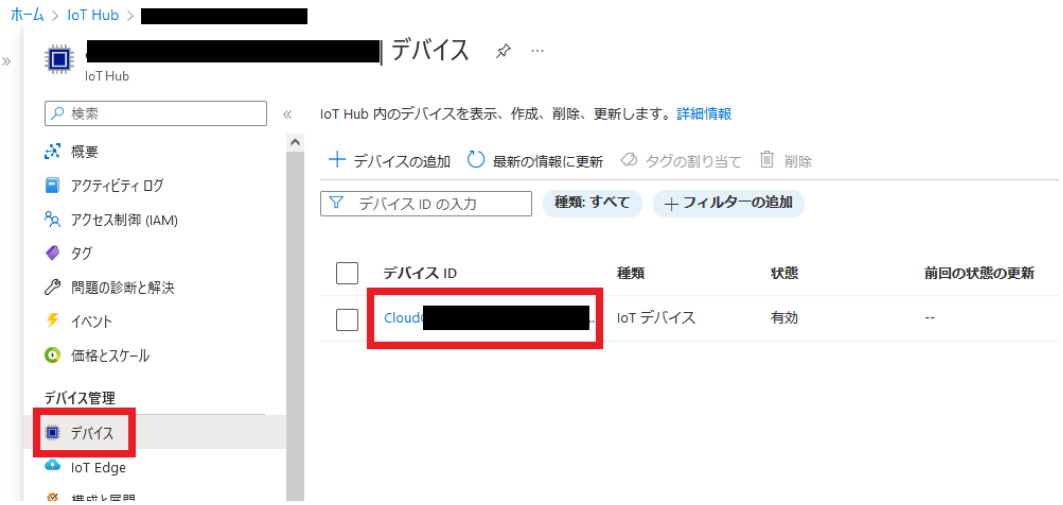
5. 入力画面が表示されるため、設定データを入力し「更新」をクリックします。



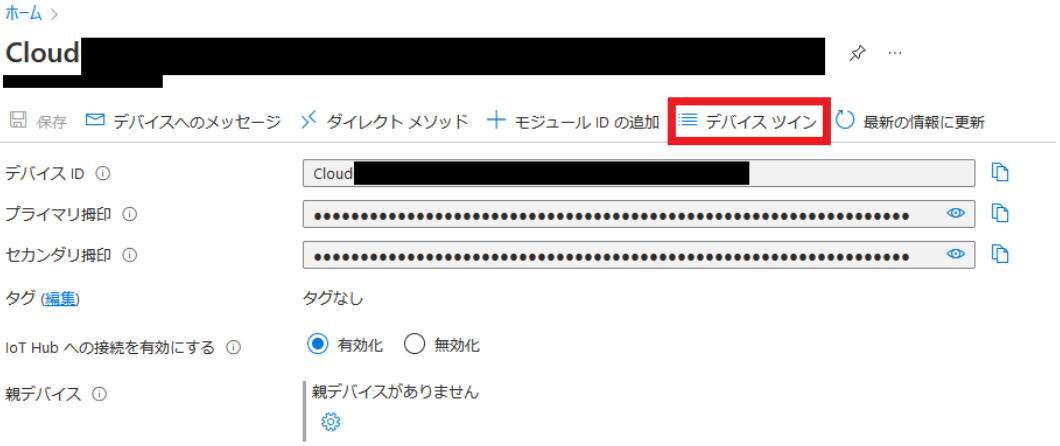
- Azure

  Azure IoT Hub のデバイスツインを更新して設定を行います。

  1. Azure portal から [IoT Hub] を開き、「10.2.4.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 で作成した IoT Hub を選択します。[デバイス] を選択し、一覧の中から該当するデバイス ID を選択します。



2. [デバイスツイン] を選択します。



3. デバイスツイン編集画面が表示されるため、設定データを入力し「保存」をクリックします。

各機能それぞれ、下記の通りのフォーマットとなっています。

・接点入力設定

表 10.18 接点入力設定値

項目	概要	設定値	内容
type	動作種別	(空欄) or none	接点入力状態取得を行わない
		polling	ポーリング
		edge	エッジ検出
edge_type	エッジ検出設定	falling	立ち下がりエッジ
		rising	立ち上がりエッジ
		both	両方
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "<制御ポート>_config": { ❶
        "type" : <polling or edge>,
        "edge_type" : <true or false>,
        "interval" : <読み出し間隔>
      }
    }
```

```
  }
}
```

❶ 制御ポートは DI1, DI2 のいずれかを指定してください

```
{
  "state": {
    "desired": {
    "DI1_config": {
        "type" : "polling",
        "edge_type" : false,
        "interval" : 5
      }
    }
  }
}
```

図 10.19 接点入力制御シャドウ設定例

・ Azure

フォーマットは下記の通りです。デバイスツインの "desired" プロパティに設定します。

```
{
  "properties": {
    "desired": {
    "<制御ポート>_config": { ❶
        "type" : <polling or edge>,
        "edge_type" : <true or false>,
        "interval" : <読み出し間隔>
      },
      :
    }
  }
}
```

❶ 制御ポートは DI1, DI2 のいずれかを指定してください

```
{
  "properties": {
    "desired": {
    "DI1_config": {
        "type" : "polling",
        "edge_type" : false,
        "interval" : 5
      },
```

図 10.20 接点入力制御デバイスツイン設定例

・ 接点出力設定

表 10.19 接点出力設定値

項目	概要	設定値	内容
output_state	出力状態	high	High
		low	Low
output_time	出力時間[sec]	1～3600	出力コマンド実行後に output_state で指定したレベルを出力する時間。 0 を指定すると永続的に出力します。
output_delay_time	出力遅延時間[sec]	0	出力コマンド実行後、指定した時間遅延して出力します。

・ AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "<制御ポート>_config": { ❶
        "output_state" : <high or low>,
        "output_time" : <出力時間>,
        "output_delay_time" : <出力遅延時間>
      }
    }
  }
}
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

```
{
  "state": {
    "desired": {
    "DO1_config": {
        "output_state" : "high",
        "output_time" : 10,
        "output_delay_time" : 10
      }
    }
  }
}
```

図 10.21 接点出力制御シャドウ設定例

・ Azure

フォーマットは下記の通りです。デバイスツインの "desired" プロパティに設定します。

```
{
  "properties": {
    "desired": {
    "<制御ポート>_config": { ❶
```

```
        "output_state" : <high or low>,
        "output_time" : <出力時間>,
        "output_delay_time" : <出力遅延時間>
      },
      :
    }
  }
}
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

```
{
  "properties": {
    "desired": {
      "DO1_config": {
        "output_state" : "high",
        "output_time" : 10,
        "output_delay_time" : 10
      },
```

図 10.22 接点出力制御デバイスツイン設定例

・RS485 レジスタ読み出し

表 10.20 RS485 レジスタ読み出し設定値

項目	概要	設定値	内容
method	通信種別	none	RS485 を利用しない
		rtu	Modbus-RTU
data_size	データサイズ	8	
baudrate	ボーレート	1200～38400[bps]	通信速度を指定します
parity	パリティビット	none	None
		odd	Odd
		even	Even
stop	ストップビット	1	1
		2	2
device_id	Modbus スレーブ機器のデバイス ID	0x01 ～ 0xF7	
func_code	ファンクションコード	0x03 or 0x04	
register_addr	レジスタアドレス	機器依存	値を読み出すレジスタのアドレスを指定
register_count	読み出しレジスタ数	1 or 2	一度に読み出すレジスタ数を指定
endian	エンディアン設定	little	リトルエンディアン
		big	ビッグエンディアン
interval	データ取得間隔[sec]	1～3600	この値に従って、値を読み出します
data_offset	読み出し値に加算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値に加算する値を指定します
data_multiply	読み出し値と乗算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と乗算する値を指定します
data_devider	読み出し値と除算する値	任意の値(整数値)	指定は任意です。読み出したレジスタ値と除算する値を指定します

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
      "RS485_Data<1～4>": { ❶
        "method" : <種別>,
        "baudrate" : <ボーレート>,
        "data_size": <データサイズ>,
        "parity" : <パリティ>,
        "stop" : <ストップビット>,
        "device_id" : <デバイス ID>,
        "func_code" : <ファンクションコード>,
        "register_addr" : <レジスタアドレス>,
        "register_count" : <読み出すレジスタ数>,
        "endian" : <エンディアン種別>,
        "interval" : <読み出し間隔>,
        "data_offset" : <データに加算する値>,
        "data_multiply" : <データに乗算する値>,
        "data_devider" : <データと除算する値>
      }
    }
  }
}
```

❶ 1～4 のいずれかを指定してください

```
{
  "state": {
    "desired": {
      "RS485_Data1": {
        "baudrate" : 9600,
        "parity" : "none",
        "stop" : 1,
        "device_id" : "01",
        "func_code" : "03",
        "register_addr" : "0000",
        "register_count" : 2,
        "endian" : "big",
        "interval" : 30,
        "data_offset" : 0,
        "data_multiply" : 0,
        "data_devider" : 0
      }
    }
  }
}
```

図 10.23 RS485 レジスタ読み出しシャドウ設定例

・Azure

フォーマットは下記の通りです。デバイスツインの "desired" プロパティに設定します。

```
{
  "properties": {
    "desired": {
      "RS485_Data<1～4>": { ❶
        "method" : <種別>,
        "baudrate" : <ボーレート>,
        "data_size": <データサイズ>,
        "parity" : <パリティ>,
        "stop" : <ストップビット>,
        "device_id" : <デバイス ID>,
        "func_code" : <ファンクションコード>,
        "register_addr" : <レジスタアドレス>,
        "register_count" : <読み出すレジスタ数>,
        "endian" : <エンディアン種別>,
        "interval" : <読み出し間隔>,
        "data_offset" : <データに加算する値>,
        "data_multiply" : <データに乗算する値>,
        "data_devider" : <データと除算する値>
      },
      :
    }
  }
}
```

❶ 1～4 のいずれかを指定してください

```
{
  "properties": {
    "desired": {
      "RS485_Data1": {
        "baudrate" : 9600,
        "parity" : "none",
        "stop" : 1,
        "device_id" : "01",
        "func_code" : "03",
        "register_addr" : "0000",
        "register_count" : 2,
        "endian" : "big",
        "interval" : 30,
        "data_offset" : 0,
        "data_multiply" : 0,
        "data_devider" : 0
      },
```

図 10.24 RS485 レジスタ読み出しデバイスツイン設定例

10.2.7.2. クラウドからのデバイス制御

以下について、クラウドからデバイスの動きを制御することができます。

- 接点出力制御
  - 接点出力開始/終了の制御
- RS485 レジスタ書き込み

  - 接続したデバイスのレジスタへの書き込み

- LED 点灯制御
  - アプリケーション LED の点灯 on/off 制御を行います

以下の手順で実行します。

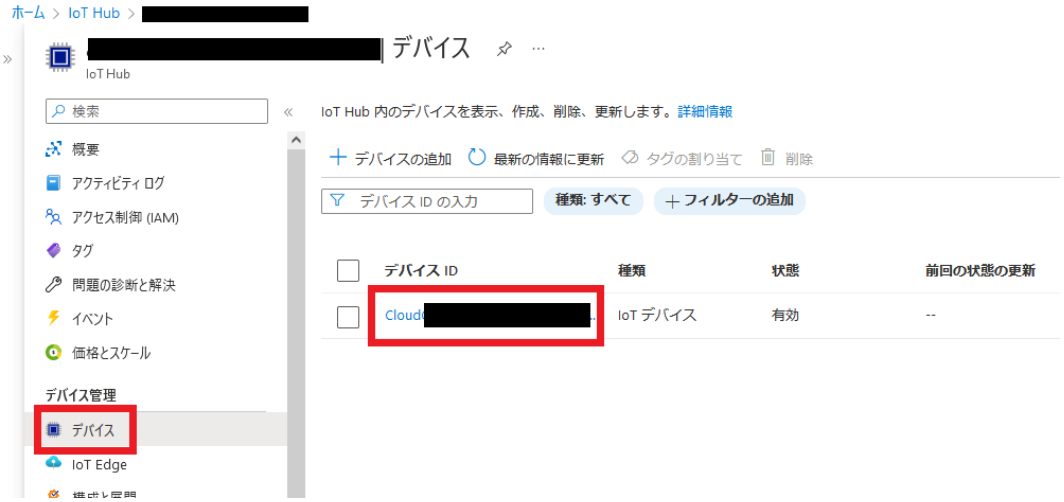
- AWS

  AWS IoT Core の デバイスシャドウを更新し、制御を行います。更新方法については 「10.2.7.1. クラウドからのデータ設定」 と同じです。

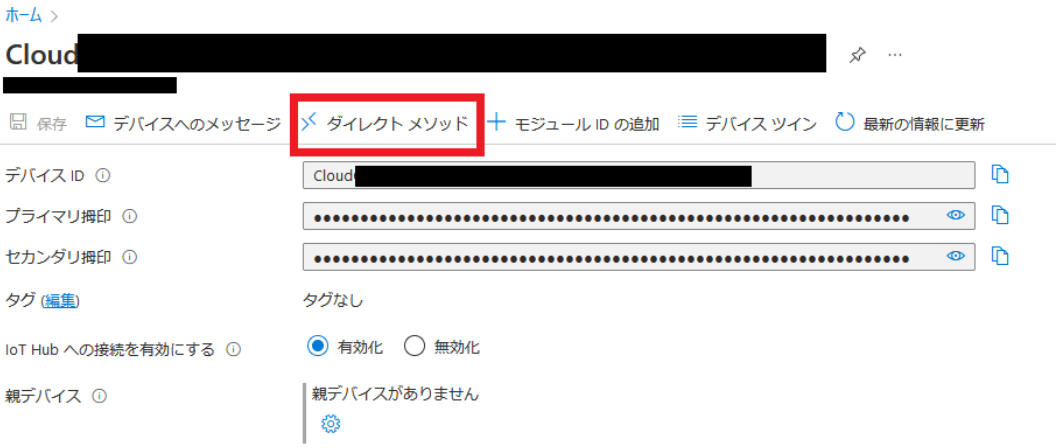
- Azure

  Azure IoT Hub のダイレクトメソッドを更新して設定を行います。

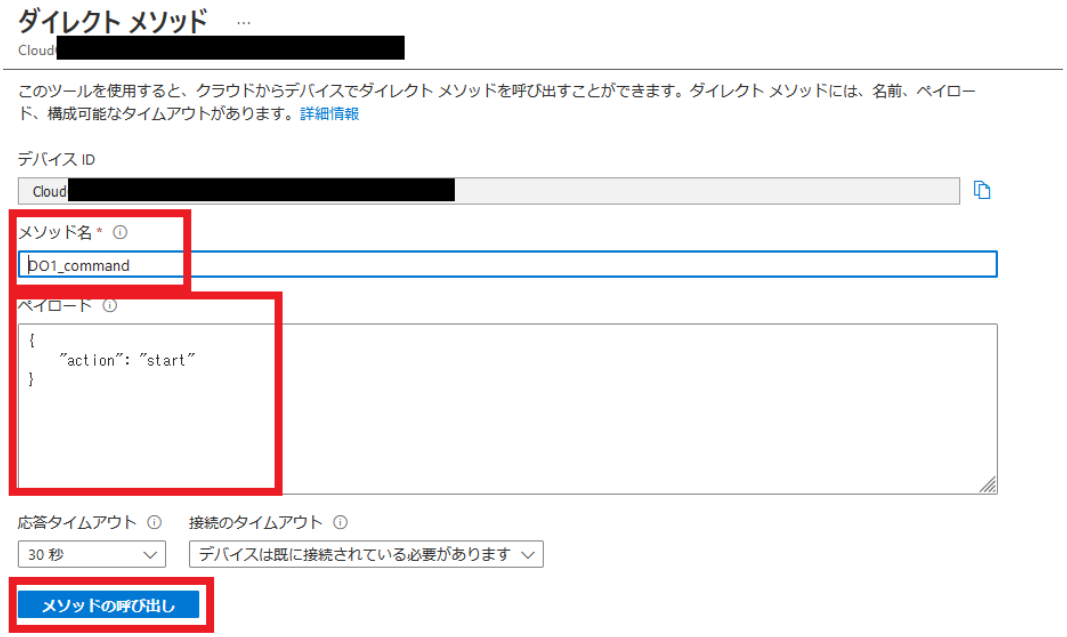
  1. Azure portal から [IoT Hub] を開き、「10.2.4.3. Azure IoT Hub と Azure IoT Hub Device Provisioning Service の設定を行う」 で作成した IoT Hub を選択します。[デバイス] を選択し、一覧の中から該当するデバイス ID を選択します。



  2. [ダイレクトメソッド] を選択します。



3. ダイレクトメソッド集画面が表示されるため、メソッド名とペイロードを入力し、「メソッドの呼び出し」をクリックします。



各機能それぞれ、下記の通りのフォーマットとなっています。

・接点出力制御

表 10.21 接点出力制御設定値

項目	概要	設定値	内容
action	制御状態	start	出力開始
		stop	出力停止

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "<制御ポート>_command": { ❶
        "action" : <start or stop>
      }
    }
  }
}
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

```
{
  "state": {
    "desired": {
    "DO1_command": {
```

```
        "action" : "start"
      }
    }
  }
}
```

図 10.25 接点出力制御シャドウ設定例

・ Azure

メソッド名とペイロードのフォーマットは下記の通りです。

・ メソッド名

```
<制御ポート>_command ❶
```

❶ 制御ポートは DO1, DO2 のいずれかを指定してください

・ ペイロード

```
{"action" : <start or stop> }
```

各パラメータの設定例は下記の通りです。

・ メソッド名

```
DO1_command
```

・ ペイロード

```
{"action" : "start"}
```

・ RS485 レジスタ書き込み

表 10.22 RS485 レジスタ書き込みシャドウ設定値

項目	概要	設定値	内容
method	通信種別	rtu	Modbus-RTU
data_size	データサイズ	8	
baudrate	ボーレート	1200～38400[bps]	通信速度を指定します
parity	パリティビット	none	None
		odd	Odd
		even	Even
stop	ストップビット	1	1
		2	2
device_id	Modbus スレーブ機器のデバイス ID	0x01 ～ 0xF7	
func_code	ファンクションコード	0x03 or 0x04	
register_addr	レジスタアドレス	機器依存	値を読み出すレジスタのアドレスを指定

項目	概要	設定値	内容
write_byte	レジスタに書き込む値	1	

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
      "RS485_command": {
        "method" : <種別>,
        "baudrate" : <ボーレート>,
        "data_size": <データサイズ>,
        "parity" : <パリティ>,
        "stop" : <ストップビット>,
        "device_id" : <デバイス ID>,
        "func_code" : <ファンクションコード>,
        "register_addr" : <レジスタアドレス>,
        "write_byte": <書き込む値>
      }
    }
  }
}
```

```
{
  "state": {
    "desired": {
      "RS485_command": {
        "method" : "rtu",
        "baudrate" : 19200,
        "data_size": 8,
        "parity" : "none",
        "stop" : 1,
        "device_id" : 1,
        "func_code" : 5,
        "register_addr" : 3,
        "write_byte": 1
      }
    }
  }
}
```

図 10.26 RS485 レジスタ書き込みシャドウ設定例

・Azure

メソッド名とペイロードのフォーマットは下記の通りです。

・メソッド名

```
RS485_command
```

・ペイロード

```
{
  "method" : <種別>,
  "baudrate" : <ボーレート>,
  "data_size": <データサイズ>,
  "parity" : <パリティ>,
  "stop" : <ストップビット>,
  "device_id" : <デバイス ID>,
  "func_code" : <ファンクションコード>,
  "register_addr" : <レジスタアドレス>,
  "write_byte": <書き込む値>
}
```

各パラメータの設定例は下記の通りです。

・メソッド名

```
RS485_command
```

・ペイロード

```
{
  "method" : "rtu",
  "baudrate" : 19200,
  "data_size": 8,
  "parity" : "none",
  "stop" : 1,
  "device_id" : 1,
  "func_code" : 5,
  "register_addr" : 3,
  "write_byte": 1
}
```

・LED

表 10.23 LED 点灯制御設定値

項目	概要	設定値	内容
state	点灯状態	on	点灯
		off	消灯

・AWS

フォーマットは下記の通りです。

```
{
  "state": {
    "desired": {
    "led_command": {
        "state" : <on or off>
      }
    }
  }
}
```

```
{
  "state": {
    "desired": {
    "led_command": {
        "state" : "off"
      }
    }
  }
}
```

図 10.27 LED 点灯制御シャドウ設定例

・ Azure

メソッド名とペイロードのフォーマットは下記の通りです。

・ メソッド名

```
led_command
```

・ ペイロード

```
{"state" : <on or off>}
```

各パラメータの設定例は下記の通りです。

・ メソッド名

```
led_command
```

・ ペイロード

```
{"state" : "off"}
```

10.2.8. コンテナの終了

podman_start で起動したゲートウェイコンテナを終了させる場合は、以下のコマンドを実行してください。

```
[armadillo ~]# podman stop a6e-gw-container
```

10.2.9. ログ内容確認

「10.2.5. 設定ファイルの編集」 でログファイルにログを出力する設定にした場合、インターフェース部とクラウド部にわかれて、それぞれ以下のファイルに出力されます。

・ インターフェース部

・ /var/app/volumes/gw_container/log/sensing_mgr.log

・ クラウド部

・ /var/app/volumes/gw_container/log/cloud_agent.log

ログファイルは自動的にローテートされるように設定されています。ローテートされると、各ファイルの末尾に番号が付与されます。なお、ファイル数が 10 を超えた場合は古いファイルから削除されます。

また、ログファイルの内容はテキストデータであり、以下のようなフォーマットになっています。

```
出力日時 ログレベル : メッセージ
```

図 10.28 ログファイルのフォーマット

10.2.10. ゲートウェイコンテナの構成

ゲートウェイコンテナは下記の通り構成されています。コンテナ内外関わらず、誤ってファイルを削除した場合はインストールディスクで初期化を行ってください。

起動スクリプト

コンテナ起動時、下記のスクリプトを実行します。

・ /usr/bin/gw-app.sh

ゲートウェイコンテナアプリケーション

ゲートウェイコンテナアプリケーションは下記に配置されています。

・ /usr/lib/python3.10/site-packages/atgateway/

ボリュームマウント

以下のパスをコンテナ内でマウントしています。

ホストパス	コンテナパス	概要
/var/app/rollback/volumes/gw_container/cert	/cert	デバイス認証関連ファイル
/var/app/rollback/volumes/gw_container/config	/config	ゲートウェイコンテナコンフィグファイル
/var/app/rollback/volumes/gw_container/src	/root/gw_container	ゲートウェイコンテナ main 関数
/var/app/volumes/gw_container/log	/log	ゲートウェイコンテナ ログ

10.3. ゲートウェイコンテナを拡張する

ゲートウェイコンテナのアプリケーションは Python で記述されており、起動スクリプトから実行されます。

アプリケーションを拡張したい場合は /var/app/rollback/volumes/gw_container/src/customize/ に main.py を配置すると、自動的にそちらが実行されるようになります。ゲートウェイコンテナアプリケーションのソースコードは以下にアップロードしているため、適宜参照してご利用ください。

Armadillo サイト - Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ

https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container

また、 Armadillo サイトの Howto やブログでも、手順を記載した記事を公開しています。

Armadillo サイト - Armadillo-IoT ゲートウェイ A6E

https://armadillo.atmark-techno.com/armadillo-iot-a6e

10.4. アプリケーションコンテナを作成、実行する

10.4.1. Podman - コンテナ仮想化ソフトウェア

10.4.1.1. Podman - コンテナ仮想化ソフトウェアとは

コンテナとはホスト OS 上に展開される仮想的なユーザ空間のことです。コンテナを使用することで複数の Armadillo-IoT ゲートウェイ A6E でも同一の環境がすぐに再現できます。ゲスト OS を必要としない仮想化であるため、アプリケーションの起動が素早いという特徴があります。

Podman とはこのようなコンテナを管理するためのソフトウェアであり、使用方法はコンテナ管理ソフトウェアの 1 つである Docker と互換性があります。

10.4.2. コンテナを操作する

この章では、コンテナ仮想化ソフトウェアの 1 つである Podman の基本的な使い方について説明します。Armadillo-IoT ゲートウェイ A6E で実行させたいアプリケーションとその実行環境自体を 1 つの Podman イメージとして扱うことで、複数の Armadillo-IoT ゲートウェイ A6E がある場合でも、全てのボード上で同一の環境を再現させることが可能となります。

この章全体を通して、イメージの公開・共有サービスである Docker Hub [https://hub.docker.com] から取得した、Alpine Linux のイメージを使って説明します。

10.4.2.1. イメージからコンテナを作成する

イメージからコンテナを作成するためには、`podman_start` コマンドを実行します。podman や docker にすでに詳しいかたは `podman run` コマンドでも実行できますが、ここでは 「10.5. コンテナの運用」で紹介するコンテナの自動起動の準備も重ねて `podman_start` を使います。イメージは Docker Hub [https://hub.docker.com] から自動的に取得されます。ここでは、簡単な例として "ls /" コマンドを実行するコンテナを作成します。

```
[armadillo ~]# vi /etc/atmark/containers/my_container.conf ❶
set_image docker.io/alpine
set_command ls /
[armadillo ~]# podman pull docker.io/alpine ❷
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
:（省略）
Writing manifest to image destination
Storing signatures
a6215f271958c760a2975a6765016044115dbae4b90f414eba3a448a6a26b4f6
[armadillo ~]# podman_start my_container ❸
Starting 'my_container'
b141e899b5ef7c9ec5434bda8f6a83d3e6bfc94f74bfb5dcef2a22041c71fdbf
[armadillo ~]# podman logs my_container ❹
```

```
bin
dev
:（省略）
usr
var
[armadillo ~]#
```

図 10.29 コンテナを作成する実行例

❶ コンテナのコンフィグを作成します。このファイルでは、コンテナのイメージやコマンド、デバイスへのアクセス権限を設定します。詳しい設定の説明には 「10.5.1. コンテナの自動起動」 を参照ください。

❷ コンテナのイメージを取得します。イメージが Armadillo に置いてない場合は「Error: docker.io/alpine: image not known」の様なエラーで失敗します。

❸ コンテナを起動します。これは Armadillo 起動時に自動的に起動されるコンテナと同じものになります。自動起動が不要な場合には `set_autostart no` で無効化できます。

❹ `podman logs` コマンドで出力を確認します。"ls /" を実行するだけの "my_container" という名前のコンテナが作成されました。コンテナが作成されると同時に "ls /" が実行され、その結果がログに残ります。ここで表示されているのは、コンテナ内部の "/" ディレクトリのフォルダの一覧です。

コンフィグファイルの直接な変更と `podman pull` によるコンテナの取得はデフォルト状態ではメモリ上でしか保存されません。ファイルは `persist_file` で必ず保存し、コンテナイメージは `abos-ctrl podman-storage --disk` で podman のストレージを eMMC に切り替えるか `abos-ctrl podman-rw` で一時的に eMMC に保存してください。運用中の Armadillo には直接に変更をせず、 swupdate でアップデートしてください。コンフィグファイルを保存して、 `set_autostart no` を設定しない場合は自動起動します。

`podman_start` でコンテナが正しく起動できない場合は `podman_start -v <my_container>` で `podman run` のコマンドを確認し、 `podman logs <my_container>` で出力を確認してください。

10.4.2.2. コンテナー覧を表示する

作成済みコンテナー覧を表示するためには podman ps コマンドを実行します。

```
[armadillo ~]# podman ps -a
CONTAINER ID  IMAGE                                COMMAND    CREATED         STATUS                     ⏎
PORTS    NAMES
d6de5881b5fb  docker.io/library/alpine:latest  ls /       12 minutes ago  Exited (0) 11 minutes      ⏎
ago          my_container
```

図 10.30 コンテナー覧の表示実行例

一覧表示により、コンテナ名やコンテナ ID を確認することができます。-a オプションを付けない場合は、動作中のコンテナのみ表示されます。podman ps コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman ps --help
```

図 10.31 podman ps --help の実行例

10.4.2.3. コンテナを起動する

作成済みのコンテナを起動するためには podman start コマンドを実行します。

```
[armadillo ~]# podman start my_container
podman start my_container
[ 3119.081068] IPv6: ADDRCONF(NETDEV_CHANGE): vethe172e161: link becomes ready
[ 3119.088214] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 3119.094812] cni-podman0: port 1(vethe172e161) entered blocking state
[ 3119.101231] cni-podman0: port 1(vethe172e161) entered disabled state
[ 3119.107745] device vethe172e161 entered promiscuous mode
[ 3119.113185] cni-podman0: port 1(vethe172e161) entered blocking state
[ 3119.119546] cni-podman0: port 1(vethe172e161) entered forwarding state
my_container
[ 3119.620731] cni-podman0: port 1(vethe172e161) entered disabled state
[ 3119.627696] device vethe172e161 left promiscuous mode
[ 3119.632762] cni-podman0: port 1(vethe172e161) entered disabled state
```

図 10.32 コンテナを起動する実行例

-a オプションを与えると、コンテナ内で実行されたアプリケーションの出力を確認できます。

```
[armadillo ~]# podman start -a my_container
[ 3150.303962] IPv6: ADDRCONF(NETDEV_CHANGE): vetha9ef8f8e: link becomes ready
[ 3150.311106] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 3150.317703] cni-podman0: port 1(vetha9ef8f8e) entered blocking state
[ 3150.324139] cni-podman0: port 1(vetha9ef8f8e) entered disabled state
[ 3150.330687] device vetha9ef8f8e entered promiscuous mode
[ 3150.336085] cni-podman0: port 1(vetha9ef8f8e) entered blocking state
[ 3150.342443] cni-podman0: port 1(vetha9ef8f8e) entered forwarding state
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
[ 3150.804164] cni-podman0: port 1(vetha9ef8f8e) entered disabled state
[ 3150.811249] device vetha9ef8f8e left promiscuous mode
[ 3150.816349] cni-podman0: port 1(vetha9ef8f8e) entered disabled state
```

図 10.33 コンテナを起動する実行例(a オプション付与)

ここで起動している my_container は、起動時に "ls /" を実行するようになっているので、その結果が出力されます。podman start コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman start --help
```

図 10.34 podman start --help 実行例

10.4.2.4. コンテナを停止する

動作中のコンテナを停止するためには podman stop コマンドを実行します。

```
[armadillo ~]# podman stop my_container
my_container
```

図 10.35 コンテナを停止する実行例

podman stop コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman stop --help
```

図 10.36 podman stop --help 実行例

10.4.2.5. コンテナの変更を保存する

コンテナに対して変更が行われた状態で、そのままコンテナを停止してしまうと変更が失なわれてしまいます。

変更を保存するには二つの方法があります。

1. podman commit コマンドで保存する。

   ```
   [armadillo ~]# podman commit my_container image_name:latest
   Getting image source signatures
   Copying blob f4ff586c6680 skipped: already exists
   Copying blob 3ae0874b0177 skipped: already exists
   Copying blob ea59ffe27343 done
   Copying config 9ca3c55246 done
   Writing manifest to image destination
   Storing signatures
   9ca3c55246eaac267a71731bad6bfe4b0124afcdd2b80c4f730c46aae17a88f3
   ```

   図 10.37 my_container を保存する例

   podman commit で保存する度に、変更が行なわれた差分が保存されます。繰り返し差分を保存すると、イメージサイズが大きくなってしまいます。ストレージ容量が不足する場合は、ベースとなる OS のイメージから作り直してください。

2. ボリュームを使用する。

   podman_start の add_volumes コマンドでコンテナに Armadillo Base OS のディレクトリをコンテナで使うことができます。保存するデータの性質によって、保存先を選択してください。. /var/app/volumes/myvolume: アップデートした場合はコピーされません。ログやデータベースなど、アプリケーションが作成し続けるようなデータの保存に向いています。. myvolume か /var/

app/rollback/volumes/myvolume: アップデートの際にコピーしてアップデートを行うので、アップデート中でも安全に使いつづけます。アプリケーションと一緒にアップデートするようなデータの保存に向いています。

コンテナを前のバージョンに戻した場合(ロールバック)、/var/app/rollback/volumes/ のデータの前のバージョンに戻ります。

その為、アプリケーションのバージョンに依存するようなデータは /var/app/rollback/volumes/ に入れることを推奨します。

mkswu の swdesc_files (--extra-os 無し) と podman_start` の add_volumes では、相対パスはそのディレクトリをベースにします。/var/app/rollback/volumes/myvolume は myvolume で簡潔に指定できます。

Copy-on-Write (CoW) について。

この二つの volumes ディレクトリは btrfs と呼ばれるファイルシステムに保存されています。btrfs ではデータは Copy on Write (CoW) を使ってデータ完全性を保証しますが、その保証にはコストがあります。

数百 MB のファイルに小さな変更を頻繁に行う場合 CoW を無効化することを推奨します。CoW を無効化されたファイルにチェックサムが入らなくなりますので、極端な場合以外に残してください。

```
[armadillo ~]# cd /var/app/volumes/
[armadillo /var/app/volumes]# mkdir database
[armadillo /var/app/volumes]# chattr +C database ❶
[armadillo /var/app/volumes]# echo example data > database/
example
[armadillo /var/app/volumes]# lsattr database/ ❷
---------------C---- database/example
```

図 10.38 chattr によって copy-on-write を無効化する例

❶ chattr +C でディレクトリに NoCow を設定します。これから作成されるファイルが NoCow で作成されます。すでに存在していたファイルに影響ないのでご注意ください。

❷ lsattr 確認します。リストの C の字があればファイルが「no cow」です。

10.4.2.6. コンテナの自動作成やアップデート

podman run, podman commit でコンテナを作成できますが、定期的にアップデートをする際にはコンテナの作成やアップデートを自動化できると便利です。

これを実現するために、Dockerfile と podman build を使います。この手順は Armadillo で実行可能です。

1. イメージを docker.io のイメージから作りなおします

```
[armadillo ~/podman-build]# cat Dockerfile
FROM docker.io/arm32v7/alpine:latest

# update & install dependencies (example: usbutils)
RUN apk upgrade && apk add usbutils && rm -f /var/cache/apk/*

# copy our application and set it to run on start
COPY my_application /my_application
ENTRYPOINT /my_application

[armadillo ~/podman-build]# podman build -t my_image:1 -t my_image:latest .
STEP 1: FROM docker.io/arm32v7/alpine:latest
STEP 2: RUN apk upgrade && apk add usbutils && rm -f /var/cache/apk/*
--> 234bf79175e
STEP 3: COPY my_application /my_application
--> 05ab31bb278
STEP 4: ENTRYPOINT /my_application
STEP 5: COMMIT my_image:latest
--> 590e3ba6d55
Successfully tagged localhost/my_image:1
Successfully tagged localhost/my_image:latest
590e3ba6d55f3e29bdef158d7283e9c4f7515567b2d3f978cfab2510dc02376b

[armadillo ~/podman-build]# podman save my_image:latest -o my_image_1.tar
```

図 10.39 podman build の実行例

2. イメージを前のバージョンからアップデートします

```
[armadillo podman-build-update]# cat Dockerfile
FROM localhost/my_image:latest

# update OS packages
RUN apk upgrade --no-cache

# update application
COPY my_application /my_application
$ podman build -t my_image:2 -t my_image:latest .
STEP 1: FROM localhost/my_image:latest
STEP 2: RUN apk upgrade --no-cache
--> cf1dc0d7296
STEP 3: COPY my_application /my_application
STEP 4: COMMIT my_image:latest
--> 9e9d9366072
Successfully tagged localhost/my_image:2
Successfully tagged localhost/my_image:latest
9e9d9366072751007b2e70544d76c46b95a7a5a02df658ef0fa3f7dcccf8850a
```

```
[armadillo podman-build-update]# podman save -o my_image_2.tar my_image:2
```

図 10.40 podman build でのアップデートの実行例

この場合、 podman_partial_image コマンドを使って、差分だけをインストールすることもできます。

```
[armadillo podman-build-update]# podman_partial_image -b my_image:1 ¥
        -o my_image_2_partial.tar my_image:2

[armadillo podman-build-update]# ls -lh
-rw-r--r-- 1 root root   88 Dec 21 15:24 Dockerfile
-rw-r--r-- 1 root root 9.4M Dec 21 15:26 my_image_1.tar
-rw-r--r-- 1 root root 9.4M Dec 21 15:26 my_image_2.tar
-rw-r--r-- 1 root root  51K Dec 21 15:26 my_image_2_partial.tar
```

作成した .tar アーカイブは 「10.9.5. mkswu の desc ファイル」 の swdesc_embed_container と swdesc_usb_container で使えます。

10.4.2.7. コンテナを削除する

作成済みコンテナを削除する場合は podman rm コマンドを実行します。

```
[armadillo ~]# podman rm my_container
d6de5881b5fb973227b84d1d74abf269ac3183aad7e18b7a9d85208632641d94
[armadillo ~]# podman ps -a
CONTAINER ID  IMAGE                           COMMAND    CREATED         STATUS                     ⏎
PORTS    NAMES
```

図 10.41 コンテナを削除する実行例

podman ps コマンドの出力結果より、コンテナが削除されていることが確認できます。podman rm コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman rm --help
```

図 10.42 $ podman rm --help 実行例

10.4.2.8. イメージを削除する

podman のイメージを削除するには podman rmi コマンドを実行します。イメージを削除するためには、そのイメージから作成したコンテナを先に削除しておく必要があります。podman rmi コマンドにはイメージ ID を指定する必要があるため、podman images コマンドで確認します。

```
[armadillo ~]# podman rm my_container
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE
docker.io/library/alpine  latest      02480aeb44d7  2 weeks ago  5.62 MB
[armadillo ~]# podman rmi 02480aeb44d7
```

```
Untagged: docker.io/library/alpine:latest
Deleted: 02480aeb44d78f1a44b8791af7edf7d6e1b18707397a1dfb3ff4f21c5ce4a44f
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE
```

図 10.43 イメージを削除する実行例

podman images コマンドの出力結果より、コンテナが削除されていることが確認できます。podman rmi コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman rmi --help
```

図 10.44 podman rmi --help 実行例

SWU で転送されたイメージは podman images で Read-Only として表示されますので、podman rmi を実行するとエラーとなります。その場合は abos-ctrl podman-rw rmi をご使用ください。 abos-ctrl podman-rw については 「10.5.5.2. イメージを eMMC に保存する方法」 を参照してください。

```
[armadillo ~]# podman images
REPOSITORY              TAG         IMAGE ID      CREATED      SIZE      R/O
docker.io/library/alpine  latest      02480aeb44d7  2 weeks ago  5.62
MB    true
[armadillo ~]# podman rmi docker.io/alpine
Error: cannot remove read-only image
"02480aeb44d78f1a44b8791af7edf7d6e1b18707397a1dfb3ff4f21c5ce4a44f"
[armadillo ~]# abos-ctrl podman-rw rmi docker.io/alpine
Untagged: docker.io/library/alpine:latest
Deleted:
02480aeb44d78f1a44b8791af7edf7d6e1b18707397a1dfb3ff4f21c5ce4a44f
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE
```

図 10.45 Read-Only のイメージを削除する実行例

10.4.2.9. 実行中のコンテナに接続する

実行中のコンテナに接続し、コンテナ内で指定したコマンドを実行するには podman exec コマンドを実行します。podman exec コマンドでコンテナ内部のシェルを起動すると、コンテナ内部を操作できるようになります。ここでは、sleep infinity コマンドを実行して待ち続けるだけのコンテナを作成し、そのコンテナに対して podman exec コマンドでシェルを起動する例を示します。

```
[armadillo ~]# vi /etc/atmark/containers/sleep_container.conf
set_image docker.io/alpine
set_command sleep infinity
add_args --init
[armadillo ~]# podman_start sleep_container
Starting 'test'
```

```
f62e7a666d7156d261905c8406c72fc271534fa29e69771c76f4f6660a2da41a
[armadillo ~]# podman exec -it sleep_container sh
[container ~]# ps
PID   USER     TIME  COMMAND
    1 root      0:00 /run/podman-init -- sleep infinity
    2 root      0:00 sleep infinity
    3 root      0:00 sh
    4 root      0:00 ps
```

図 10.46 コンテナ内部のシェルを起動する実行例

podman_start コマンドでコンテナを作成し、その後作成したコンテナ内で sh を実行しています。sh を実行すると、コンテナ内のプロンプトが表示されコンテナ内部を操作できるようになります。上記ではコンテナ内で、ps コマンドを実行しています。コンテナ作成時に実行した sleep と podman exec で実行した sh がプロセスとして存在していることが確認できます。

コンテナ内のシェルから抜ける時は exit コマンドを実行します。

```
[container ~]# exit
```

図 10.47 コンテナ内部のシェルから抜ける実行例

podman exec コマンドから抜けても、コンテナがまだ実行中です。コンテナを停止したい場合は podman stop sleep_container か podman kill sleep_container で停止して podman rm sleep_container でそのコンテナを削除してください。podman exec コマンドの詳細は --help オプションで確認できます。

```
[armadillo ~]# podman exec --help
```

図 10.48 podman exec --help 実行例

10.4.2.10. コンテナ間で通信をする

複数のコンテナを実行している環境で、それらのコンテナ間で通信を行う方法を示します。これにより、例えば SQL サーバを実行しているコンテナに対し別のコンテナから接続するといった使い方ができます。

コンテナには作成した時点でローカル IP アドレスが割り当てられるので、コンテナの名前かその IP アドレスで通信を行うことができます。

準備として、2 つのコンテナを作成します。

```
[armadillo ~]# vi /etc/atmark/containers/my_container_1.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# vi /etc/atmark/containers/my_container_2.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# podman_start my_container_1 my_container_2
Starting 'my_container_1'
```

```
cbe0802f4e2d2fec88f4e300dabeba3b48865359dc02cbd99375b1b38c2c28eb
Starting 'my_container_2'
```

図 10.49 コンテナを作成する実行例

コンテナに割り当てられた IP アドレスを確認するには podman inspect コマンドを実行します。

```
[armadillo ~]# podman inspect --format='{{.NetworkSettings.IPAddress}}' my_container_1
10.88.0.108
[armadillo ~]# podman inspect --format='{{.NetworkSettings.IPAddress}}' my_container_2
10.88.0.109
```

図 10.50 コンテナの IP アドレスを確認する実行例

これらの IP アドレスを使って、一方のコンテナからもう一方のコンテナへ対し ping コマンドで疎通確認を行うことができます。

```
[armadillo ~]# podman exec -it my_container_1 sh
[container ~]# ping -c 2 my_container_2
PING my_container_2 (10.88.0.109): 56 data bytes
64 bytes from 10.88.0.109: seq=0 ttl=42 time=0.144 ms
64 bytes from 10.88.0.109: seq=1 ttl=42 time=0.210 ms
--- my_container_2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.144/0.177/0.210 ms
[container ~]# ping -c 2 10.88.0.109
PING 10.88.0.109 (10.88.0.109): 56 data bytes
64 bytes from 10.88.0.109: seq=0 ttl=42 time=0.140 ms
64 bytes from 10.88.0.109: seq=1 ttl=42 time=0.138 ms
--- 10.88.0.109 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.138/0.139/0.140 ms
```

図 10.51 ping コマンドによるコンテナ間の疎通確認実行例

このように、my_container_1(10.88.0.108) から my_container_2(10.88.0.109) への通信が確認できます。

10.4.2.11. 開発時に有用な—privileged オプション

コンテナに、全権限と全てのデバイスへのアクセスを許可するオプション `--privileged` があります。このオプションを利用すると、コンテナに与えるべき最小の権限を洗い出す必要が無いため、開発時に有用です。

実運用の際、このオプションを利用することはセキュリティー上問題がある為、開発時にのみご利用ください。コンテナに必要な最低限の権限を与えることをおすすめします。

10.4.2.12. GPIO を扱う

コンテナ内で動作するアプリケーションから GPIO を扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/gpiochip**N** を渡す必要があります。以下は、/dev/gpiochip2 を渡して alpine イメージからコンテナを作成する例です。/dev/gpiochip**N** を渡すと、GPIO**N+1** を操作することができます。

```
[armadillo ~]# vi /etc/atmark/containers/gpio_example
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/gpiochip2
[armadillo ~]# podman_start gpio_example
Starting 'gpio_example'
956a0fecc48d5ea1210069910f7bb48b9e90b2dadb12895064d9776dae0360b5
```

図 10.52 GPIO を扱うためのコンテナ作成例

コンテナ内に入ってコマンドで GPIO を操作する例を以下に示します。この例では GPIO3_IO21 を操作しています。

```
[armadillo ~]# podman exec -it gpio_example sh
[container ~]# apk upgrade
[container ~]# apk add libgpiod
[container ~]# gpioget gpiochip2 21 ❶
0 ❷
[container ~]# gpioset gpiochip2 21=1 ❸
```

図 10.53 コンテナ内からコマンドで GPIO を操作する例

❶ GPIO 番号 21 の値を取得します。

❷ 取得した値を表示します。

❸ GPIO 番号 21 に 1(High) を設定します。

他にも、gpiodetect コマンドで認識している gpiochip をリスト表示できます。以下の例では、コンテナを作成する際に渡した /dev/gpiochip2 が認識されていることが確認できます。

```
[container ~]# gpiodetect
gpiochip2 [30220000.gpio] (32 lines)
```

図 10.54 gpiodetect コマンドの実行

gpioinfo コマンドでは、指定した gpiochip の詳細な情報を表示することができます。

```
[container ~]# gpioinfo gpiochip2
gpiochip2 - 32 lines:
        line   0:      unnamed         "?"  output  active-high [used]
        line   1:      unnamed       unused   input  active-High
        line   2:      unnamed       unused   input  active-high
        line   3:      unnamed       unused   input  active-high
        line   4:      unnamed       unused   input  active-high
        line   5:      unnamed       unused   input  active-high
        line   6:      unnamed       unused   input  active-high
        line   7:      unnamed       unused   input  active-high
        line   8:      unnamed       unused   input  active-high
        line   9:      unnamed       unused   input  active-high
        line  10:      unnamed       unused   input  active-high
```

```
        line  11:      unnamed       unused   input  active-high
        line  12:      unnamed       unused   input  active-high
        line  13:      unnamed       unused   input  active-high
        line  14:      unnamed       unused   input  active-high
        line  15:      unnamed       unused   input  active-high
        line  16:      unnamed       unused   input  active-high
        line  17:      unnamed       unused   input  active-high
        line  18:      unnamed       unused   input  active-high
        line  19:      unnamed       unused   input  active-high
        line  20:      unnamed       unused   input  active-high
        line  21:      unnamed       unused   input  active-high
        line  22:      unnamed       unused   input  active-high
        line  23:      unnamed       unused   input  active-high
        line  24:      unnamed       unused   input  active-high
        line  25:      unnamed       unused   input  active-high
        line  26:      unnamed       unused   input  active-high
        line  27:      unnamed       unused   input  active-high
        line  28:      unnamed       unused   input  active-high
        line  29:      unnamed       unused   input  active-high
        line  30:      unnamed       unused   input  active-high
        line  31:      unnamed       unused   input  active-high
```

図 10.55 gpioinfo コマンドの実行

C 言語プログラムから操作する場合は、GPIO 操作ライブラリである libgpiod を使用することができます。

10.4.2.13. I2C を扱う

コンテナ内で動作するアプリケーションから I2C を扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/i2c-N を渡す必要があります。以下は、/dev/i2c-1 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/i2c_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/i2c-1
[armadillo ~]# podman_start i2c_example
Starting 'i2c_example'
efa1eb129c1f036a709755f0d53b21a0f2a39307ecae32b24aac98c0b6567bf0
```

図 10.56 I2C を扱うためのコンテナ作成例

コンテナ内に入り、i2c-tools に含まれる i2cdetect コマンドを使ってスレーブアドレスを確認することができます。

```
[armadillo ~]# podman exec -it i2c_example sh
[container ~]# apk upgrade
[container ~]# apk add i2c-tools
[container ~]# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
30: -- -- -- -- -- -- -- -- -- -- -- -- -- UU -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- UU -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- 72 -- -- -- -- --
```

図 10.57 i2cdetect コマンドによる確認例

10.4.2.14. シリアルインターフェースを扱う

コンテナ内で動作するアプリケーションから RS-232C や RS-485 などのシリアル通信を行うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/ttymxcN を渡す必要があります。以下は、/dev/ttymxc0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/serial_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/ttymxc0
[armadillo ~]# podman_start serial_example
Starting 'serial_example'
3999f09d51253371cacffd68967c90fdd5250770888a82f59d7810b54fcc873e
```

図 10.58 シリアルインターフェースを扱うためのコンテナ作成例

コンテナ内に入り、setserial コマンドを使って現在の設定を確認することができます。

```
[armadillo ~]# podman exec -it serial_example sh
[container ~]# setserial -a /dev/ttymxc0
/dev/ttymxc0, Line 0, UART: undefined, Port: 0x0000, IRQ: 29
        Baud_base: 5000000, close_delay: 50, divisor: 0
        closing_wait: 3000
        Flags: spd_normal
```

図 10.59 setserial コマンドによるシリアルインターフェイス設定の確認例

10.4.2.15. USB を扱う

コンテナ内で動作するアプリケーションから USB 接続のデバイスを扱うための方法について示します。

・ USB シリアルデバイスを扱う

USB シリアルデバイスをコンテナ内から扱う場合には、Podman のイメージからコンテナを作成する際にホスト OS 側の /dev/ttyUSBN を渡す必要があります。以下は、 /dev/ttyUSB0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# podman run -itd --name=usb_example --device=/dev/ttyUSB0 docker.io/alpine /bin/sh
```

図 10.60 USB シリアルデバイスを扱うためのコンテナ作成例

コンテナ内に入り、setserial コマンドを使って現在の設定を確認することができます。

```
[armadillo ~]# podman exec -it usb_example /bin/sh
[container ~]# setserial -a /dev/ttyUSB0
/dev/ttyUSB0, Line 0, UART: unknown, Port: 0x0000, IRQ: 0
        Baud_base: 24000000, close_delay: 0, divisor: 0
        closing_wait: infinite
        Flags: spd_normal
```

図 10.61 setserial コマンドによる USB シリアルデバイス設定の確認例

・ USB カメラを扱う

USB カメラをコンテナ内から扱う場合には、Podman のイメージからコンテナを作成する際にホスト OS 側の /dev/videoN を渡す必要があります。以下は、 /dev/video3 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/usbcam_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/video3
[armadillo ~]# podman_start usbcam_example
Starting 'usbcam_example'
ffe06090b45826cc0b1c7710e9e850ba9521d36b70de4288d0dfe1fe91a35632
[armadillo ~]# podman exec -it usbcam_example sh
[container ~]# ls /dev/video3
/dev/video3
```

図 10.62 USB カメラを扱うためのコンテナ作成例

GStreamer などのマルチメディアフレームワークと組み合わせることで、USB カメラからの映像のキャプチャが可能となります。

・ USB メモリを扱う

ここでは、USB メモリを扱う方法について 2 つの例を示します。

・ ホスト OS 側でマウントした USB メモリをコンテナから扱う

あらかじめホスト OS 側でマウントしてある USB メモリをコンテナから扱う場合には、Podman のイメージからコンテナを作成する際にホスト OS 側で USB メモリをマウントしてるディレクトリを渡す必要があります。

```
[armadillo ~]# mount -t vfat /dev/sda1 /mnt
[armadillo ~]# echo test >> /mnt/sample.txt
[armadillo ~]# ls /mnt
sample.txt
```

図 10.63 USB メモリをホスト OS 側でマウントする例

上記の例では、USB メモリを /mnt にマウントしました。以下は、 /mnt を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/usbmem_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /mnt
[armadillo ~]# podman_start usbmem_example
Starting 'usbmem_example'
ef77d4bfd5b04f3b8b5ddcb5bfac321304fa64219a4b88c3130e45e5a14e1b3e
```

図 10.64 ホスト OS 側でマウント済みの USB メモリを扱うためのコンテナ作成例

ホスト OS 側の /mnt ディレクトリをコンテナ内の /mnt にマウントしています。これにより、コンテナ内からも /mnt ディレクトリを通して USB メモリを扱うことができます。

```
[armadillo ~]# podman exec -it usbmem_example sh
[container ~]# ls /mnt
sample.txt
[container ~]# cat /mnt/sample.txt
test
```

図 10.65 USB メモリに保存されているデータの確認例

・ USB メモリをコンテナ内からマウントする

USB メモリをコンテナ内からマウントして扱う場合には、Podman のイメージからコンテナを作成する際にホスト OS 側の /dev ディレクトリを渡すと同時に、適切な権限も渡す必要があります。以下は、/dev を渡して alpine イメージからコンテナを作成する例です。権限として SYS_ADMIN と SYS_RAWIO も渡しています。

```
[armadillo ~]# vi /etc/atmark/containers/usbmem_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_args --cap-add=SYS_ADMIN
add_devices /dev/sda1
[armadillo ~]# podman_start usbmem_example
Starting 'usbmem_example'
387a2256530e9b35b5361ca681a99fba8f46d78b6a6cb8ecd60096246b9198a8
```

図 10.66 USB メモリをマウントするためのコンテナ作成例

コンテナ内に入り、mount コマンドで USB メモリを /mnt にマウントし、保存されているデータを確認することができます。

```
[armadillo ~]# podman exec -it usbmem_example sh
[container ~]# mount /dev/sda1 /mnt
[container ~]# ls /mnt
sample.txt
[container ~]# cat /mnt/sample.txt
test
```

図 10.67 コンテナ内から USB メモリをマウントする例

・USB デバイスのホットプラグに対応する

通常、コンテナ内から USB デバイスを扱うためには、あらかじめ Armadillo-IoT ゲートウェイ A6E 本体に USB デバイスを接続した状態で、コンテナを起動する必要があります。コンテナ起動後に USB デバイスを接続して認識させるためには、/dev を volume としてコンテナ内にマウントする必要があります。以下は、 volume として /dev を渡して alpine イメージからコンテナを作成する例です。イベントの通知のために --net=host も渡す必要があります。

```
[armadillo ~]# vi /etc/atmark/containers/usbhotplug_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /dev
set_network host
[armadillo ~]# podman_start usbhotplug_example
Starting 'usbhotplug_example'
54b83ebf49db906af38501ff1fcb85eee8258a31d37ca5d0eed7ff6f9ed5b72c
```

図 10.68 USB デバイスのホットプラグに対応する例

10.4.2.16. RTC を扱う

コンテナ内から RTC を扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/rtcN を渡すと同時に、RTC への時刻の設定を行うための権限も渡す必要があります。以下は、/dev/rtc0 を渡して alpine イメージからコンテナを作成する例です。権限として SYS_TIME も渡しています。

```
[armadillo ~]# vi /etc/atmark/containers/rtc_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_args --cap-add=SYS_TIME
add_devices /dev/rtc0
[armadillo ~]# podman_start rtc_example
Starting 'rtc_example'
025209e0d96f43c2911239a8397b7002c3eaab057e031d8abb765df5707d75bd
```

図 10.69 RTC を扱うためのコンテナ作成例

コンテナ内に入り、hwclock コマンドで RTC の時刻表示と設定ができます。

```
[armadillo ~]# podman exec -it rtc_example sh
[container ~]# hwclock ❶
Thu Feb 18 05:14:37 2021  0.000000 seconds
[container ~]# date --set "2021-04-01 09:00:00" ❷
Thu Apr  1 09:00:00 UTC 2021
[container ~]# hwclock --systohc ❸
[container ~]# hwclock ❹
Thu Apr  1 09:00:28 2021  0.000000 seconds
```

図 10.70 hwclock コマンドによる RTC の時刻表示と設定例

❶ RTC に設定されている現在時刻を表示します。

❷ システム時刻を 2021 年 4 月 1 日 9 時 0 分 0 秒に設定します。

❸ システム時刻を RTC に反映させます。

❹ RTC に設定されている時刻が変更されていることを確認します。

10.4.2.17. 音声出力を行う

Armadillo-IoT ゲートウェイ A6E に接続したスピーカーなどの音声出力デバイスへコンテナ内から音声を出力するためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/snd を渡す必要があります。以下は、/dev/snd を渡して debian イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/snd_example.conf
set_image localhost/at-debian-image
set_command sleep infinity
add_devices /dev/snd
[armadillo ~]# podman_start snd_example
Starting 'snd_example'
b921856b504e9f0a3de2532485d7bd9adb1ff63c2e10bfdaccd1153fd36a3c1d
```

図 10.71 音声出力を行うためのコンテナ作成例

コンテナ内に入り、alsa-utils などのソフトウェアで音声出力を行えます。

```
[armadillo ~]# podman exec -it snd_example /bin/bash
[container ~]# apt update && apt upgrade
[container ~]# apt install alsa-utils ❶
[container ~]# /etc/init.d/alsa-utils start ❷
[container ~]# aplay -D hw:N,M [ファイル名] ❸
```

図 10.72 alsa-utils による音声出力を行う例

❶ alsa-utils をインストールします。

❷ alsa-utils を起動します。

❸ 指定したファイル名の音声ファイルを再生します。

aplay の引数にある、M は音声を出力したい CARD 番号、N はデバイス番号を表しています。CARD 番号とデバイス番号は、aplay コマンドに -l オプションを与えることで確認できます。

10.4.2.18. ユーザースイッチのイベントを取得する

Armadillo-IoT ゲートウェイ A6E にはユーザースイッチが実装されています。これらのスイッチのプッシュ/リリースイベントを取得するためには、Podman のイメージからコンテナを作成する際にホスト OS 側の /dev/input ディレクトリを渡す必要があります。以下は、/dev/input を渡して alpine イメージからコンテナを作成する例です。ここで渡された /dev/input ディレクトリはコンテナ内の /dev/input にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/sw_example.conf
set_image docker.io/alpine
set_command sleep infinity
```

```
add_devices /dev/input
[armadillo ~]# podman_start sw_example
Starting 'sw_example'
c0cd8b801883266197a3c20552b0e8b6c7dd473bb0b24e05bf3ecdb581c822b9
```

図 10.73 ユーザースイッチのイベントを取得するためのコンテナ作成例

コンテナ内に入り、evtest コマンドでイベントを確認できます。

```
[armadillo ~]# podman exec -it sw_example sh
[container ~]# apk upgrade
[container ~]# apk add evtest
[container ~]# evtest /dev/input/event1
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 28 (KEY_ENTER)
Properties:
Testing ... (interrupt to exit)
Event: time 1612849227.554456, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1  ❶
Event: time 1612849227.554456, -------------- SYN_REPORT ------------
Event: time 1612849229.894444, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0  ❷
Event: time 1612849229.894444, -------------- SYN_REPORT ------------
```

図 10.74 evtest コマンドによる確認例

❶ SW1 のボタン **プッシュ** イベントを検出したときの表示

❷ SW1 のボタン **リリース** イベントを検出したときの表示

10.4.2.19. LED を扱う

Armadillo-IoT ゲートウェイ A6E には LED が実装されています。これらの LED を扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側の /sys ディレクトリを渡す必要があります。以下は、/sys を渡して alpine イメージからコンテナを作成する例です。ここで渡された /sys ディレクトリはコンテナ内の /sys にマウントされます。

```
[armadillo ~]# vi /etc/atmark/containers/led_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /sys
[armadillo ~]# podman_start led_example
Starting 'led_example'
c770f76d7714f4cceb1229be2240382bded236c2c51bb6b866bc0098c2cb987a
```

図 10.75 LED を扱うためのコンテナ作成例

コンテナ内に入り、brightness ファイルに値を書き込むことで LED の点灯/消灯を行うことができます。0 を書き込むと消灯、0 以外の値 (1～255) を書き込むと点灯します。

```
[armadillo ~]# podman exec -it led_example sh
[container ~]# echo 0 > /sys/class/leds/app/brightness
[container ~]# echo 1 > /sys/class/leds/app/brightness
```

図 10.76 LED の点灯/消灯の実行例

10.4.3. 近距離通信を行う

この章では、コンテナ内から近距離通信デバイスを扱う方法について示します。

10.4.3.1. Bluetooth デバイスを扱う

コンテナ内から Bluetooth デバイスを扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/ttymxcN を渡すと同時にネットワークとして host を、権限として NET_ADMIN を渡す必要があります。/dev/ttymxcN は Bluetooth 通信で使用するように設定したシリアルデバイスを指定してください。以下は、/dev/ttymxc0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/bt_example.conf
set_image docker.io/alpine
set_command sleep infinity
set_network host
add_devices /dev/ttymxc0
add_args --cap-add=NET_ADMIN
[armadillo ~]# podman_start bt_example
Starting 'bt_example'
45fe1eb6b25529f0c84cd4b97ca1aef8451785fc9a87a67d54873c1ed45b70a4
```

図 10.77 Bluetooth デバイスを扱うためのコンテナ作成例

コンテナ内で必要なソフトウェアをインストールして、Bluetooth を起動します。btattach コマンドの引数にはコンテナ作成時に渡した ttymxc を設定してください。

```
[armadillo ~]# podman exec -it bt_example sh
[container ~]# apk upgrade
[container ~]# apk add bluez dbus
[container ~]# /usr/bin/dbus-daemon --system
[container ~]# /usr/lib/bluetooth/bluetoothd &
[container ~]# btattach -B /dev/ttymxc0 -S 115200 &
```

図 10.78 Bluetooth を起動する実行例

これにより、bluetoothctl で Bluetooth 機器のスキャンやペアリングなどが行えるようになります。以下に、bluetoothctl コマンドで周辺機器をスキャンしてペアリングを行う例を示します。

```
[container ~]# bluetoothctl
Agent registerd
[..CHG..] Controller XX:XX:XX:XX:XX:XX Pairable: yes
[bluetooth]# power on ❶
Changing power on succeeded
```

```
[..CHG..] Controller XX:XX:XX:XX:XX:XX Powered: yes
[bluetooth]# scan on ❷
Discovery started
[..CHG..] Controller XX:XX:XX:XX:XX:XX Discovering: yes
[..NEW..] Device AA:AA:AA:AA:AA:AA AA-AA-AA-AA-AA-AA
[..NEW..] Device BB:BB:BB:BB:BB:BB BB-BB-BB-BB-BB-BB
[..NEW..] Device CC:CC:CC:CC:CC:CC CC-CC-CC-CC-CC-CC
[..NEW..] Device DD:DD:DD:DD:DD:DD DD-DD-DD-DD-DD-DD
[..NEW..] Device EE:EE:EE:EE:EE:EE EE-EE-EE-EE-EE-EE
[bluetooth]# pair AA:AA:AA:AA:AA:AA ❸
[bluetooth]# exit ❹
[container ~]#
```

図 10.79 bluetoothctl コマンドによるスキャンとペアリングの例

❶ コントローラを起動します。

❷ 周辺機器をスキャンします。

❸ ペアリングしたい機器の MAC アドレスを指定してペアリングします。

❹ exit で bluetoothctl のプロンプトを終了します。

10.4.3.2. Wi-SUN デバイスを扱う

ここでは、Wi-SUN デバイスが UART で接続されている場合の例を示します。この場合、コンテナ内で動作するアプリケーションから Wi-SUN デバイスで通信を行うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/ttymxcN のうち、Wi-SUN と対応するものを渡す必要があります。以下は、/dev/ttymxc0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/wisun_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/ttymxc0
[armadillo ~]# podman_start wisun_example
Starting 'wisun_example'
ef9a5a2f7eee4236cb28c1fbf5090a6f0db9d6dfe7f3a34573867e0833dd3122
[armadillo ~]# podman exec -it wisun_example sh
[container ~]# ls /dev/ttymxc0
/dev/ttymxc0
```

図 10.80 Wi-SUN デバイスを扱うためのコンテナ作成例

コンテナ内から、/dev/ttymxc0 を使って Wi-SUN データの送受信ができるようになります。

10.4.3.3. EnOcean デバイスを扱う

ここでは、EnOcean デバイスが UART で接続されている場合の例を示します。この場合、コンテナ内で動作するアプリケーションから EnOcean デバイスで通信を行うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/ttymxcN のうち、EnOcean と対応するものを渡す必要があります。以下は、/dev/ttymxc0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/enocean_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_deivces /dev/ttymxc0
[armadillo ~]# podman_start enocean_example
Starting 'enocean_example'
a808b491a100f9078d8c72a7b36966d9182614f3657fe054fb8d7f87b0d4b31c
[armadillo ~]# podman exec -it enocean_example sh
[container ~]# ls /dev/ttymxc0
/dev/ttymxc0
```

図 10.81 EnOcean デバイスを扱うためのコンテナ作成例

コンテナ内から、/dev/ttymxc0 を使って EnOcean データの送受信ができるようになります。

10.4.4. ネットワークを扱う

この章では、コンテナ内のネットワークを扱う方法について示します。

10.4.4.1. コンテナの IP アドレスを確認する

基本的にコンテナの IP アドレスは Podman イメージからコンテナを作成したときに自動的に割り振られます。コンテナに割り振られている IP アドレスはホスト OS 側からは podman inspect コマンドを用いて、以下のように確認することができます。

```
[armadillo ~]# vi /etc/atmark/containers/net_example.conf
set_image docker.io/alpine
set_command sleep infinity
[armadillo ~]# podman_start net_example
Starting 'net_example'
48ae479af65445674323567c17c5418dd4624292351e061bd2bd8a0add4cf150
[armadillo ~]# podman inspect --format '{{ .NetworkSettings.IPAddress }}' net_example
10.88.0.17
```

図 10.82 コンテナの IP アドレス確認例

コンテナ内の ip コマンドを用いて確認することもできます。

```
[armadillo ~]# podman exec net_example ip addr show eth0
3: eth0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether xx:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 10.88.0.17/16 brd 10.88.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::40e5:98ff:feec:4b17/64 scope link
       valid_lft forever preferred_lft forever
```

図 10.83 ip コマンドを用いたコンテナの IP アドレス確認例

10.4.4.2. コンテナに固定 IP アドレスを設定する

podman はデフォルトで 10.88.0.0/16 を使います。

> 他に使用している IP アドレスと被った場合等はコンテナに別の IP アドレスを設定してください。

コンテナに固定 IP アドレスを設定するためには、最初にユーザ定義のネットワークを作成する必要があります。以下に 192.168.1.0/24 にユーザ定義のネットワークを作成する例を示します。

```
[armadillo ~]# vi /etc/atmark/containers/my_network.conf
set_type network
set_subnet 192.168.1.0/24
[armadillo ~]# podman_start my_network
Creating network 'my_network'
my_network
```

図 10.84 ユーザ定義のネットワーク作成例

コンテナを作成する際に、上記で作成したネットワークと設定したい IP アドレスを渡すことで、コンテナの IP アドレスを固定することができます。以下の例では、IP アドレスを 192.168.1.10 に固定します。

```
[armadillo ~]# vi /etc/atmark/containers/network_example.conf
set_image docker.io/alpine
set_command sleep infinity
set_network my_network
set_ip 192.168.1.10
[armadillo ~]# podman_start network_example
Starting 'network_example'
3ea8c9031bf833228908bd73d8929b1d543b189b436c218e0634e0d39409e100
```

図 10.85 IP アドレス固定のコンテナ作成例

コンテナの IP アドレスが、192.168.1.10 に設定されていることが確認できます。

```
[armadillo ~]# podman inspect --format '{{ .NetworkSettings.Networks.my_network.IPAddress }}'
network_example
192.168.1.10
```

図 10.86 コンテナの IP アドレス確認例

10.4.5. サーバを構築する

この章では、コンテナ内で様々なサーバを構築する方法について示します。この章で取り上げているサーバは alpine の apk コマンドでインストールすることが可能です。

10.4.5.1. HTTP サーバを構築する

ここでは、HTTP サーバとして Apache と lighttpd の 2 種類を使用する場合について説明します。

・ Apache を使用する

alpine イメージからコンテナを作成し、そのコンテナ内に Apache をインストールします。コンテナ作成の際に、ホスト OS の 8080 番ポートをコンテナ内の 80 番ポートに転送する指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/apache_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 8080:80
[armadillo ~]# podman_start apache_example
Starting 'apache_example'
ea0a1ed9c2fe170a6db02e480300467510f4e844900efb35c7a24cc1a8653af2
[armadillo ~]# podman exec -it apache_example sh
[container ~]# apk upgrade && apk add apache2
[container ~]# httpd
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
10.88.0.2. Set the 'ServerName' directive globally to suppress this message
```

図 10.87 コンテナに Apache をインストールする例

他の PC などの Web ブラウザから、ホスト OS の IP アドレスの 8080 番ポートに接続すると、動作確認用ページが表示されます。デフォルトでは、/var/www/localhost/htdocs ディレクトリにファイルを置くことで Web ブラウザから閲覧できます。Apache の詳細な設定は、/etc/apache2 ディレクトリにある設定ファイルを編集することで変更可能です。

・ lighttpd を使用する

alpine イメージからコンテナを作成し、そのコンテナ内に lighttpd をインストールします。コンテナ作成の際に、ホスト OS の 8080 番ポートをコンテナ内の 80 番ポートに転送する指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/lighttpd_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 8080:80
[armadillo ~]# podman_start lighttpd_example
Starting 'lighttpd_example'
fd7ea338d09c5e8962654ed54bba17fb6a9ed4fca1b344e350bbf8f943d2f12b
[armadillo ~]# podman exec -it lighttpd_example sh
[container ~]# apk upgrade && apk add lighttpd
[container ~]# echo "<html><body>It works!</body></html>" > /var/www/localhost/htdocs/index.html
[container ~]# lighttpd -f /etc/lighttpd/lighttpd.conf
```

図 10.88 コンテナに lighttpd をインストールする例

lighttpd はデフォルトでは動作確認用ページが用意されていないため、上記の手順では簡単なページを /var/www/localhost/htdocs ディレクトリの下に配置しています。他の PC などの Web ブラウザから、ホスト OS の IP アドレスの 8080 番ポートに接続すると表示されます。lighttpd の詳細な設定は、/etc/lighttpd ディレクトリにある設定ファイルを編集することで変更可能です。

10.4.5.2. FTP サーバを構築する

ここでは、FTP サーバとして vsftp を使用する場合について説明します。alpine イメージからコンテナを作成し、そのコンテナ内に vsftpd をインストールします。コンテナ作成の際に、FTP 通信で使用するポートについてホスト OS 側からコンテナ内のポートに転送する指定と、コンテナ内の環境変数として PASV_ADDRESS にホスト OS 側の IP アドレスの指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/ftp_example.conf
set_image docker.io/alpine
```

```
set_command sleep infinity
add_ports 21:21 21100-21110:21100-21110
add_args --env=PASV_ADDRESS=<ホストの IP アドレス>
[armadillo ~]# podman_start ftp_example
Starting 'ftp_example'
efcf1ba752c2db9ae1a33ac11af3be71d95ac7b737ce9734730ebca602e57796
[armadillo ~]# podman exec -it ftp_example sh
[container ~]# apk upgrade && apk add vsftpd
```

図 10.89 コンテナに vsftpd をインストールする例

コンテナ内にユーザアカウントを作成し、このユーザで ftp ログインできるようにします。

```
[container ~]# adduser atmark
Changing password for atmark
New password: （パスワードを入力）
Retype password: （パスワードを入力）
passwd: password for atmark changed by root
```

図 10.90 ユーザを追加する例

作成したユーザで ftp ログインできるように、vsftpd の設定ファイルを編集します。

```
[container ~]# sed -i -e 's/anonymous_enable=YES/#anonymous_enable=YES/g' /etc/vsftpd/vsftpd.conf
[container ~]# sed -i -e 's/#local_enable=YES/local_enable=YES/g' /etc/vsftpd/vsftpd.conf
[container ~]# sed -i -e 's/#write_enable=YES/write_enable=YES/g' /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_enable=YES" >> /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_min_port=21100" >> /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_max_port=21110" >> /etc/vsftpd/vsftpd.conf
[container ~]# echo "pasv_address=$PASV_ADDRESS" >> /etc/vsftpd/vsftpd.conf
```

図 10.91 設定ファイルの編集例

編集した設定ファイルを指定して vftpd を起動することにより、ftp 接続可能となります。ftp ログイン時のアカウントは前述の手順で作成したものを使用します。

```
[container ~]# vsftpd /etc/vsftpd/vsftpd.conf
```

図 10.92 vsftpd の起動例

10.4.5.3. Samba サーバを構築する

ここでは、Samba サーバの構築方法について説明します。alpine イメージからコンテナを作成し、そのコンテナ内に samba をインストールします。コンテナ作成の際に、samba で使用するポートについてホスト OS 側からコンテナ内のポートに転送する指定を行っています。

```
[armadillo ~]# vi /etc/atmark/containers/smb_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_ports 139:139 445:445
```

```
[armadillo ~]# podman_start smb_example
Starting 'smb_example'
6d81c01fe27b5a92ee6ea69de2f9a8dbb569d420c2f5f630ece1966c81824a1f
[armadillo ~]# podman exec -it smb_example sh
[container ~]# apk upgrade && apk add samba
```

図 10.93 コンテナに samba をインストールする例

コンテナ内にユーザアカウントを作成し、このユーザで samba にログインできるようにします。

```
[container ~]# adduser atmark
Changing password for atmark
New password: （パスワードを入力）
Retype password: （パスワードを入力）
passwd: password for atmark changed by root
[container ~]# pdbedit -a atmark
new password: （パスワードを入力）
retype new password: （パスワードを入力）
```

図 10.94 ユーザを追加する例

samba を起動すると、前述の手順で作成したユーザアカウントで他の PC などからログインすることができます。

```
[container ~]# smbd
```

図 10.95 samba の起動例

共有するディレクトリの指定などの詳細設定は /etc/samba/smb.conf ファイルを編集することで変更可能です。

10.4.5.4. SQL サーバを構築する

ここでは、RDMS として sqlite を使用する場合について説明します。alpine イメージからコンテナを作成し、そのコンテナ内に sqlite をインストールします。

```
[armadillo ~]# vi /etc/atmark/containers/sqlite_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_volumes /var/app/volumes/sqlite_db:/db
[armadillo ~]# podman_start sqlite_example
Starting 'sqlite_example'
114c5f1dbb7e81293dcb8fbe0c600b861626375b14cfe4023761acaa84fdcad1
[armadillo ~]# podman exec -it sqlite_example sh
[container ~]# apk upgrade && apk add sqlite
```

図 10.96 コンテナに sqlite をインストールする例

コンテナ内に入り、sqlite3 コマンドを実行すると sqlite のプロンプトが表示されデータベースの操作ができるようになります。

```
[container ~]# sqlite3 /db/mydb.sqlite
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite>
```

図 10.97 sqlite の実行例

10.4.6. 異常検知

この章では、コンテナ内で動作しているアプリケーションに何らかの異常が発生し停止してしまった際に、ソフトウェアウォッチドックタイマーを使って、システムを再起動する方法について示します。

10.4.6.1. ソフトウェアウォッチドッグタイマーを扱う

コンテナ内で動作するアプリケーションからソフトウェアウォッチドックタイマーを扱うためには、Podman のイメージからコンテナを作成する際にホスト OS 側のデバイスファイル /dev/watchdogN を渡す必要があります。以下は、/dev/watchdog0 を渡して alpine イメージからコンテナを作成する例です。

```
[armadillo ~]# vi /etc/atmark/containers/watchdog_example.conf
set_image docker.io/alpine
set_command sleep infinity
add_devices /dev/watchdog0
[armadillo ~]# podman_start watchdog_example
Starting 'watchdog_example'
a5d329cca49d60423ce4155d72a119b8049a03dbd1d0277817a253e96dce7bc7
```

図 10.98 ソフトフェアウォッチドッグタイマーを使うためのコンテナ作成例

ソフトウェアウォッチドックタイマーは、プログラム内からデバイスファイル /dev/watchdog0 を open した時点で起動します。コンテナ内に入ってソフトウェアウォッチドックタイマーを echo コマンドで起動する例を以下に示します。

```
[armadillo ~]# podman exec -it watchdog_example sh
[container ~]# echo > /dev/watchdog0
```

図 10.99 コンテナ内からソフトウェアウォッチドッグタイマーを起動する実行例

ソフトウェアウォッチドックタイマーを起動した後、/dev/watchdog0 に任意の文字を書き込むことでソフトウェアウォッチドッグタイマーをリセットすることができます。60 秒間任意の文字の書き込みがない場合は、システムが再起動します。

```
[armadillo ~]# podman exec -it watchdog_example sh
[container ~]# echo a > /dev/watchdog0
```

図 10.100 ソフトウェアウォッチドッグタイマーをリセットする実行例

ソフトウェアウォッチドックタイマーを停止したい場合は、/dev/watchdog0 に V を書き込みます。

```
[armadillo ~]# podman exec -it watchdog_example sh
[container ~]# echo V > /dev/watchdog0
```

図 10.101 ソフトウェアウォッチドッグタイマーを停止する実行例

10.5. コンテナの運用

10.5.1. コンテナの自動起動

Armadillo Base OS では、/etc/atmark/containers/*.conf ファイルに指定されているコンテナがブート時に自動的に起動します。nginx.conf の記載例を以下に示します。

```
[armadillo ~]# cat /etc/atmark/containers/nginx.conf
set_image docker.io/library/nginx:alpine
set_readonly no
add_ports 80:80
```

図 10.102 コンテナを自動起動するための設定例

.conf ファイルは以下のパラメータを設定できます。

- コンテナイメージの選択： **set_image [イメージ名]**

  イメージの名前を設定できます。

  例: set_image docker.io/debian:latest, set_image localhost/myimage

  イメージを rootfs として扱う場合に --rootfs オプションで指定できます。

  例: set_image --rootfs /var/app/volumes/debian

- ポート転送： **add_ports [ホストポート]:[コンテナポート]**

  設定したポートで外部からコンテナへのアクセスが可能となります。

  デフォルトは TCP で、UDP も /udp を付けて使えます。スペースで分けて複数のポートを設定することができます。

  以下の例では、ポート 80、443(web)、UDP の 69(tftp)にアクセスすることができ、コンテナのポート 22(ssh)にはポート 2222 からアクセスすることができます。

  例: add_ports 80:80 443:443 2222:22 69:69/udp

  pod を使う場合、このオプションは pod の設定にしないと有効になりませんのでご注意ください。

- デバイスファイル作成： **add_devices [ホストパス]:[コンテナパス]**

  コンテナでデバイスを作成して、使用可能となります。

コンテナパスを設定しない場合はホストと同じパスを使います。

複数のデバイスを作成したい場合はスペースで分けて設定してください。

例: add_devices /dev/galcore /dev/v4l/by-id/usb-046d_HD_Pro_Webcam_C920_78DA8CAF-video-index0:/dev/video3

ホストパスに「:」を含む場合は add_device "[ホストパス]" "[コンテナパス]" で追加できます。

例: add_device "/dev/v4l/by-path/XXXXX" "/dev/video3"

コンテナパスに「:」を含むようなパスは設定できません。

- ボリュームマウント： **add_volumes [ホストパス]:[コンテナパス]:[オプション]**

指定するパスをコンテナ内でマウントして、データの保存や共有することができます。

ホストパスは以下のどちらかを指定してください。

- /var/app/rollback/volumes/<folder> か <folder>:

  アップデートの際に新しくコピー（snapshot）した場合、コピー先のみ変更しますので、アップデート中でもこのデータを使うことができます。途中で電源が落ちた場合でも、このデータに影響はありません。

  SWUpdate でアップデートするデータに向いています。

- /var/app/volumes/<folder>: app パーティションに書きます。

  アップデートの際にコピーされませんので、アップデート中の新たな変更は更新されたコンテナ内のアプリケーションで見れます。

  ログやデータベースに向いています。

- /tmp/<folder>: 複数のコンテナでメモリファイルシステムを共有したい場合に使ってください。

- /opt/firmware: 学習能力に必要なファムウェアライブラリーのパス。

コンテナパスを設定しない場合はホストパスと同じパスを使います。

オプションは podman run の --volume のオプションになりますので、 ro (read-only), nodev, nosuid, noexec, shared, slave 等を設定できます。

例 ： add_volumes /var/app/volumes/database:/database: ロールバックされないデータを /database で保存します。

例: add_volumes assets:/assets:ro,nodev,nosuid /opt/firmware: アプリケーションのデータを /assets で読み取り、/opt/firmware のファームウェアを使えます。

「:」はホスト側のパスとコンテナのパスを別ける意味があるため、ファイル名やデバイス名に「:」を使うことはできません。

複数のコンテナでマウントコマンドを実行することがあれば、shared のフラグで起動後のマウントを共有することができます。

```
[armadillo ~]# cat /etc/atmark/containers/mounter.conf
set_image docker.io/alpine
add_args -ti
add_volumes /tmp/mnt:/mnt:shared ❶
add_args --cap-add SYS_ADMIN
add_device /dev/sda1
[armadillo ~]# cat /etc/atmark/containers/client.conf
set_image docker.io/alpine
add_volumes /tmp/mnt:/mnt:slave ❷
add_args -ti
[armadillo ~]# podman exec mounter mount /dev/sda1 /mnt ❸
[armadillo ~]# podman exec client ls /mnt ❹
file_on_usb
```

図 10.103 ボリュームを shared でサブマウントを共有する例

❶ マウントを行うコンテナに shared の設定とマウント権限 (SYS_ADMIN) を与えます。

❷ マウントを使うコンテナに slave だけを設定すれば一方にしか共有されません。

❸ USB デバイスをマウントします。

❹ マウントされたことを確認します。

- pod の選択： **set_pod［ポッド名］**

  「10.5.2. pod の作成」で作成した pod の名前を入れてコンテナを pod 内で起動します。

  例: `set_pod mypod`

- ネットワークの選択： **set_network［ネットワーク名］**

  この設定に「10.5.3. network の作成」で作成したネットワーク以外に `none` と `host` の特殊な設定も選べます。

  `none` の場合、コンテナに `localhost` しかないネームスペースに入ります。

  `host` の場合は OS のネームスペースをそのまま使います。

  例: `set_network mynetwork`

- IP アドレスの設定： **set_ip［アドレス］**

  コンテナの IP アドレスを設定することができます。

  例: set_ip 10.88.0.100

コンテナ間の接続が目的であれば、pod を使って localhost か pod の名前でアクセスすることができます。

- 読み取り専用設定： **set_readonly yes**

  コンテナ内からのファイルシステムへの書き込み許可を設定します。

  デフォルトで書き込み可能となっています。

  コンテナ内からのファイルシステムへの書き込みを禁止することで、tmpfs として使うメモリの消費を明示的に抑えることができますが、アプリケーションによっては読み込み専用のファイルシステムでは動作しない可能性もあります。

- イメージの自動ダウンロード設定： **set_pull［設定］**

  この設定を missing にすると、イメージが見つからない場合にイメージを自動的にダウンロードします。

  always にすると、イメージがすでにダウンロード済みでも起動前に必ず更新の確認を取ります。

  デフォルトでは never で、イメージが見つからない場合にエラーを表示します。

  例：set_pull missing か set_pull always

- コンテナのリスタート設定： **set_restart［設定］**

  コンテナが停止した時にリスタートさせます。

  podman kill か podman stop で停止する場合、この設定と関係なくリスタートしません。

  デフォルトで on-failure になっています。

  例: set_restart always か set_restart no

- 自動起動の無効化： **set_autostart no**

  手動かまたは別の手段で操作するコンテナがある場合、Armadillo の起動時に自動起動しないようにします。

  その場合、 podman_start <name> で起動させることができます。

  コンフィグに記載していないイメージはアップデートの際に削除されますので、そういったイメージに対して設定してください。

- 実行コマンドの設定： **set_command［コマンド］**

  コンテナを起動するときのコマンド。設定されなかった場合、コンテナイメージのデフォルトを使います。

  例: set_command /bin/sh -c ”echo bad example”

- podman run に引数を渡す設定： **add_args［引数］**

  ここまでで説明した設定項目以外の設定を行いたい場合は、この設定で podman run に直接引数を渡すことができます。

例：add_args --cap-add=SYS_TTY_CONFIG --env=XDG_RUNTIME_DIR=/run/xdg_home

10.5.2. pod の作成

podman_start で pod 機能を使うことができます。

pod を使うことで、複数のコンテナが同じネットワークネームスペースを共有することができます。同じ pod の中のコンテナが IP の場合 localhost で、 unix socket の場合 abstract path で相互に接続することができます。

```
[armadillo ~]# cat /etc/atmark/containers/mypod.conf
set_type pod
add_ports 80:80
set_infra_imager k8s.gcr.io/pause:3.5

[armadillo ~]# cat /etc/atmark/containers/nginx.conf
set_image docker.io/library/nginx:alpine
set_readonly no
set_pod mypod

[armadillo ~]# podman ps
CONTAINER ID  IMAGE                            COMMAND               CREATED      STATUS                  ⏎
PORTS               NAMES
0cdb0597b610  k8s.gcr.io/pause:3.5                                   2 hours ago  Up 2 hours ago          ⏎
0.0.0.0:80->80/tcp  5ba7d996f673-infra
3292e5e714a2  docker.io/library/nginx:alpine  nginx -g daemon o...  2 hours ago  Up 2 hours ago          ⏎
0.0.0.0:80->80/tcp  nginx
```

図 10.104 pod を使うコンテナを自動起動するための設定例

コンテナと同じく、 /etc/atmark/containers/[NAME].conf ファイルを作って、 set_type pod を設定することで pod を作成します。

pod を使う時にコンテナの設定ファイルに set_pod [NAME] の設定を追加します。

ネットワークネームスペースは pod を作成するときに必要なため、 ports, network と ip の設定は pod のコンフィグファイルに入れなければなりません。

ネットワーク設定の他に、 infra_image のオプションで pod のイメージも固める事ができます。この設定は set_type network の後しか使えませんので、set_type はファイルの最初のところに使ってください

必要であれば、他の podman pod create のオプションを add_args で設定することができます。

pod を使う時に podman が特殊な「infra container」も起動します（例の場合、 k8s.gcr.io/pause:3.5 を起動させました）

コンフィグレーションに pod を入れるアップデートの際に自動的に podman pull でイメージをダウンロードしますが、インターネットを使わせたくないアップデートがあれば swdesc_embed_container か swdesc_usb_container で入れてください。その場合、 infra_image の設定も使ってください。

10.5.3. network の作成

podman_start で podman の network も作成ことができます。

デフォルトの 10.88.0.0/16 が使えない場合、あるいはコンテナ同士で接続できないようにしたい場合は使ってください。

```
[armadillo ~]# cat /etc/atmark/containers/mynetwork.conf
set_type network
set_subnet 192.168.100.0/24

[armadillo ~]# cat /etc/atmark/containers/nginx.conf
set_image docker.io/library/nginx:alpine
set_ports 80:80
set_ip 192.168.100.10
set_network mynetwork

[armadillo ~]# podman ps
CONTAINER ID  IMAGE                           COMMAND               CREATED      STATUS                  ⏎
PORTS                 NAMES
3292e5e714a2  docker.io/library/nginx:alpine  nginx -g daemon o...  2 hours ago  Up 2 hours ago          ⏎
0.0.0.0:80->80/tcp  nginx
```

図 10.105 network を使うコンテナを自動起動するための設定例

コンテナと同じく、 /etc/atmark/containers/[NAME].conf ファイルを作って、 set_type network を設定することで network を作成します。

そのネットワークを使う時にコンテナの設定ファイルに set_network [NAME] の設定をいれます。

ネットワークのサブネットは set_subnet [SUBNET] で設定します。この設定は set_type network の後しか使えませんので、set_type はファイルの最初のところに使ってください

他の podman network create のオプションが必要であれば、 add_args で設定することができます。

10.5.4. コンテナからのコンテナ管理

podman では REST API による管理アクセスも可能です。

自分のコンテナから他のコンテナの管理が必要な場合に、ホストの podman サービスを有効にして、コンテナに /run/podman をボリュームマウントすれば podman --remote で管理できます。

podman_start をインストールすればそちらも --remote で使えます。

このオプションは Armadillo のホスト側の udev rules からコンテナを扱う時にも必要です。

10.5.5. コンテナの配布

コンテナの作成は 「10.4. アプリケーションコンテナを作成、実行する」を参考にしてください。

コンテナのイメージを配布する方法は大きく分けて二つあります：

1. インターネット上のリポジトリ（dockerhub 等）で登録してそこから配布する
2. SWUpdate のアップデートイメージを配布する

Podman のイメージをインストールする時に、一時データを大量に保存する必要があります。

swu イメージ内で組み込む時は 3 倍、pull や USB ドライブで分けてインストールすると転送するデータ量の 2 倍の空き容量が app パーティションに必要です。

アップデート時にアップデート前のコンテナが使われているのでご注意ください。

10.5.5.1. リモートリポジトリにコンテナを送信する方法

1. イメージをリモートリポジトリに送信する：

```
[armadillo ~]$ podman image push <localimage> docker://<registry>/<remoteimage>:<tag>
```

2. set_pull always を設定しないかぎり、SWUpdate でダウンロードの命令を送らないとアップデートを行いません。

   （mkswu については 「10.9. Armadillo のソフトウェアをアップデートする」 を参考にしてください）

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/pull_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
[ATDE ~/mkswu]$ cat pull_container_nginx.desc
swdesc_option version=1

swdesc_pull_container "docker.io/nginx:alpine"
swdesc_files --extra-os nginx_start
[ATDE ~/mkswu]$ mkswu pull_container_nginx.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
pull_container_nginx.swu を作成しました。
```

10.5.5.2. イメージを eMMC に保存する方法

Armadillo Base OS のデフォルトでは、Podman のデータは tmpfs に保存されます。

起動時にイメージを起動するにはイメージを eMMC に書き込む必要があります。想定の使い方では、以下の 「10.5.5.3. イメージを SWUpdate で転送する方法」 でコンテナのイメージを送信する場合には、 /var/lib/containers/storage_readonly の app パーティションのサブボリュームに展開します。

開発の時に、abos-ctrl podman-storage --disk を設定すると別の /var/log/containers/storage が作成されますが、SWUpdate で転送するイメージはそのまま readonly の方に書き込みます。

- abos-ctrl podman-rw

abos-ctrl podman-rw を使えば、read-only になっているイメージを扱う事ができます。

```
[armadillo ~]# podman images
REPOSITORY                TAG         IMAGE ID      CREATED            SIZE        R/O
[armadillo ~]# mount /dev/sda1 /mnt
[armadillo ~]# abos-ctrl podman-rw load -i /mnt/at-debian-image.tar
Getting image source signatures
Copying blob 63c098a71e7b done
Copying blob 837e73dd4d20 done
Copying blob a25086e65f63 done
Copying config b5a30f8581 done
Writing manifest to image destination
Storing signatures
Loaded image(s): localhost/at-debian-image:latest
[armadillo ~]# podman image list
REPOSITORY                TAG         IMAGE ID      CREATED      SIZE        R/O
localhost/at-debian-image  latest      b5a30f8581cc  2 hours ago  233 MB      true
```

図 10.106 abos-ctrl podman-rw の実行例

・ abos-ctrl podman-storage

abos-ctrl podman-storage はメモリとディスクの切り替えの他に、読み書きストレージから読み取り専用ストレージへのコピーもできます。

```
[armadillo ~]# podman pull docker.io/alpine ❶
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Copying blob f97344484467 done
Copying config 3d81c46cd8 done
Writing manifest to image destination
Storing signatures
3d81c46cd8756ddb6db9ec36fa06a6fb71c287fb265232ba516739dc67a5f07d
[armadillo ~]# abos-ctrl podman-storage ❷
There are images configured on development storage:
REPOSITORY                TAG         IMAGE ID      CREATED     SIZE
docker.io/library/alpine  latest      3d81c46cd875  3 days ago  5.56 MB

What should we do? ([N]othing (default), [C]opy, [R]emove)
copy ❸
Create a snapshot of '/mnt/boot_0/containers_storage' in '/mnt/new_storage'
Getting image source signatures
Copying blob 5b7df235d876 done
Copying config 3d81c46cd8 done
Writing manifest to image destination
Storing signatures
Delete subvolume (no-commit): '/mnt/new_storage'
Copying development images to readonly storage succeeded
Feel free to adjust the result with abos-ctrl podman-rw commands

Now freeing up original data...
Podman is in tmpfs mode ❹
[armadillo ~]# podman image list ❺
```

```
REPOSITORY                TAG         IMAGE ID      CREATED     SIZE        R/O
docker.io/library/alpine  latest      3d81c46cd875  3 days ago  5.56 MB     true
```

図 10.107 abos-ctrl podman-storage のイメージコピー例

❶ イメージを書き込み可能ストレージに取得します。

❷ abos-ctrl podman-storage をオプション無しで実行します。

❸ 書き込み可能ストレージにイメージがある場合に対応を聞かれます。今回はコピー（copy)します。

❹ abos-ctrl podman-storage にオプションを指定しなかったので、ストレージが tmpfs のままになります。すでに --disk で切り替えた場合にディスクのままでも可能です。

❺ コピーの確認します。イメージが読み取り専用（R/O, Read only）になりました。

SWUpdate でアップデートをインストールする際には、/var/lib/containers/storage_readonly ディレクトリの不要になったイメージを自動的に削除します。

自動起動させる予定がなくても、「10.5.1. コンテナの自動起動」を参考にして、/etc/atmark/containers/*.conf を使ってください。set_autostart no を設定することで自動実行されません。

10.5.5.3. イメージを SWUpdate で転送する方法

1. イメージをファイルに保存する：

```
[armadillo ~]$ podman image save -o <myimage>.tar <localimage>
```

2. ファイルを SWUpdate のイメージに入れる。

   二つのやり方があります：

   a. swu イメージ内に組み込む

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/embed_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
[ATDE ~/mkswu]$ cat embed_container_nginx.desc
swdesc_option version=1

swdesc_embed_container "nginx_alpine.tar"
swdesc_files --extra-os nginx_start
[ATDE ~/mkswu]$ podman pull --arch arm --variant v7 docker.io/nginx:alpine
[ATDE ~/mkswu]$ podman run --rm docker.io/nginx:alpine uname -m
armv7l
[ATDE ~/mkswu]$ podman save docker.io/nginx:alpine > nginx_alpine.tar
[ATDE ~/mkswu]$ mkswu embed_container_nginx.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
embed_container_nginx.swu を作成しました
```

   b. USB ドライブに保存する

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/usb_container_nginx.desc .
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/nginx_start .
[ATDE ~/mkswu]$ cat usb_container_nginx.desc
swdesc_option version=1

swdesc_usb_container "nginx_alpine.tar"
swdesc_files --extra-os nginx_start
[ATDE ~/mkswu]$ podman pull --arch arm --variant v7 docker.io/nginx:alpine
[ATDE ~/mkswu]$ podman run --rm docker.io/nginx:alpine uname -m
armv7l
[ATDE ~/mkswu]$ podman save docker.io/nginx:alpine > nginx_alpine.tar
[ATDE ~/mkswu]$ mkswu -o usb_container_nginx.swu usb_container_nginx.desc
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
以下のファイルを USB メモリにコピーしてください：
'/home/atmark/mkswu/usb_container_nginx.swu'
'/home/atmark/mkswu/nginx_alpine.tar'
'/home/atmark/mkswu/.usb_container_nginx/nginx_alpine.tar.sig'

usb_container_nginx.swu を作成しました。
```

10.6. Armadillo のソフトウェアをビルドする

ここでは、Armadillo-IoT ゲートウェイ A6E で使用するソフトウェアのビルド方法を説明します。

10.6.1. ブートローダーをビルドする

ここでは、ATDE 上で Armadillo-IoT ゲートウェイ A6E 向けのブートローダーイメージをビルドする方法を説明します。

1. ソースコードの取得

   Armadillo-IoT ゲートウェイ A6E ブートローダー [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/boot-loader] から「ブートローダー ソース」ファイル (u-boot-[VERSION].tar.gz) を「図 10.108. ブートローダーのソースコードをダウンロードする」に示す手順でダウンロードします。

   ```
   [ATDE ~]$ wget https://download.atmark-techno.com/armadillo-iot-a6e/bootloader/u-boot-
   [VERSION].tar.gz
   [ATDE ~]$ tar xf u-boot-[VERSION].tar.gz
   [ATDE ~]$ cd u-boot-[VERSION]
   ```

   図 10.108 ブートローダーのソースコードをダウンロードする

2. デフォルトコンフィギュレーションの適用

   「図 10.109. デフォルトコンフィギュレーションの適用」に示すコマンドを実行します。

   ```
   [ATDE ~/u-boot-[VERSION]]$ make ARCH=arm armadillo-iotg-a6e_defconfig
     HOSTCC  scripts/basic/fixdep
     HOSTCC  scripts/kconfig/conf.o
     YACC    scripts/kconfig/zconf.tab.c
     LEX     scripts/kconfig/zconf.lex.c
   ```

```
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

図 10.109 デフォルトコンフィギュレーションの適用

3. ビルド

   ブートローダーのビルドを実行するには、「図 10.110. ブートローダーのビルド」に示すコマンドを実行します。

```
[ATDE ~/u-boot-[VERSION]]$ make CROSS_COMPILE=arm-linux-gnueabihf-
:
: (省略)
:
  SHIPPED dts/dt.dtb
  FDTGREP dts/dt-spl.dtb
  CAT     u-boot-dtb.bin
  CFGS    u-boot-dtb.cfgout
  MKIMAGE u-boot-dtb.imx
  OBJCOPY u-boot.srec
  COPY    u-boot.bin
  SYM     u-boot.sym
  COPY    u-boot.dtb
  CFGCHK  u-boot.cfg
```

図 10.110 ブートローダーのビルド

4. ビルド結果の確認

   ビルドが終了しますと、イメージファイルが生成されます。確認には、「図 10.111. ブートローダーの結果確認」に示すコマンドを実行します。

```
[ATDE ~/u-boot-[VERSION]]$ ls u-boot-dtb.imx
u-boot-dtb.imx
```

図 10.111 ブートローダーの結果確認

10.6.2. Linux カーネルをビルドする

ここでは、Armadillo-IoT ゲートウェイ A6E 向けの Linux カーネルイメージをビルドする方法を説明します。

Armadillo-IoT ゲートウェイ A6E では、基本的には Linux カーネルイメージをビルドする必要はありません。「10.6.3. Alpine Linux ルートファイルシステムをビルドする」の手順を実施することで、標準の Linux カーネルイメージがルートファイルシステムに組み込まれます。

> 標準の Linux カーネルイメージは、アットマークテクノが提供する linux-at という Alpine Linux 用のパッケージに含まれています。
>
> カスタマイズした Linux カーネルイメージを利用するには、 「10.6.3. Alpine Linux ルートファイルシステムをビルドする」の手順の中で、a6e/packages から linux-at-a6e@atmark [mailto:linux-at-a6e@atmark] と記載された行を削除し、 a6e/resources/boot/ にイメージを配置する必要があります。

1. ソースコードの取得

   Armadillo-IoT ゲートウェイ A6E Linux カーネル [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/linux-kernel] から「Linux カーネル」ファイル (linux-at-[VERSION].tar) をダウンロードして、「図 10.112. Linux カーネルソースコードの展開」に示すコマンドを実行して展開します。

```
[ATDE ~]$ tar xf linux-at-[VERSION].tar
[ATDE ~]$ tar xf linux-at-[VERSION]/linux-[VERSION].tar.gz
[ATDE ~]$ cd linux-[VERSION]
```

図 10.112 Linux カーネルソースコードの展開

1. デフォルトコンフィギュレーションの適用

   「図 10.113. Linux カーネルデフォルトコンフィギュレーションの適用」に示すコマンドを実行します。

```
[ATDE ~/linux-[VERSION]]$ make ARCH=arm armadillo-iotg-a6e_defconfig
```

図 10.113 Linux カーネルデフォルトコンフィギュレーションの適用

2. Linux カーネルコンフィギュレーションの変更

   コンフィギュレーションの変更を行わない場合はこの手順は不要です。変更する際は、「図 10.114. Linux カーネルコンフィギュレーションの変更」に示すコマンドを実行します。

```
[ATDE ~]$ make ARCH=arm menuconfig
```

図 10.114 Linux カーネルコンフィギュレーションの変更

   コマンドを実行するとカーネルコンフィギュレーション設定画面が表示されます。カーネルコンフィギュレーションを変更後、"Exit"を選択して「Do you wish to save your new kernel configuration? (Press <ESC><ESC> to continue kernel configuration.)」で "Yes" を選択し、 カーネルコンフィギュレーションを確定します。

```
.config - Linux/arm 5.10.145 Kernel Configuration
──────────────────────────────────────────────────────────────────────────────
  ┌──────────────────────── Linux/arm 5.10.145 Kernel Configuration ────────────────────────┐
```

```
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable
  ┌──────────────────────────────────────────────────────────────────────────┐
  │          General setup  --->                                             │
  │          System Type  --->                                               │
  │          Bus support  --->                                               │
  │          Kernel Features  --->                                           │
  │          Boot options  --->                                              │
  │          CPU Power Management  --->                                      │
  │          Floating point emulation  --->                                  │
  │          Power management options  --->                                  │
  │          Firmware Drivers  --->                                          │
  │      [ ] ARM Accelerated Cryptographic Algorithms  ----                  │
  │          General architecture-dependent options  --->                    │
  │      [*] Enable loadable module support  --->                            │
  │      [*] Enable the block layer  --->                                    │
  │          IO Schedulers  --->                                             │
  │          Executable file formats  --->                                   │
  │          Memory Management options  --->                                 │
  │      [*] Networking support  --->                                        │
  │          Device Drivers  --->                                            │
  │          File systems  --->                                              │
  │          Security options  --->                                          │
  │      -*- Cryptographic API  --->                                         │
  │          Library routines  --->                                          │
  │          Kernel hacking  --->                                            │
  │                                                                          │
  └──────────────────────────────────────────────────────────────────────────┘
  ┌──────────────────────────────────────────────────────────────────────────┐
  │              <Select>    < Exit >    < Help >    < Save >    < Load >    │
  └──────────────────────────────────────────────────────────────────────────┘
```

図 10.115 Linux カーネルコンフィギュレーション設定画面

Linux Kernel Configuration メニューで"/"キーを押下すると、カーネルコンフィギュレーションの検索を行うことができます。カーネルコンフィギュレーションのシンボル名(の一部)を入力して"Ok"を選択すると、部分一致するシンボル名を持つカーネルコンフィギュレーションの情報が一覧されます。

3. ビルド

Linux カーネルをビルドするには、「図 10.116. Linux カーネルコンフィギュレーションの変更」に示すコマンドを実行します。

```
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x82000000 uImage
[ATDE ~/linux-[VERSION]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

図 10.116 Linux カーネルコンフィギュレーションの変更

4. ビルド結果の確認

   ビルドが正常終了すると、「図 10.117. Linux カーネルビルドしたファイルの確認」に示すイメージファイルが生成されます。

```
[ATDE ~/linux-[VERSION]]$ ls arch/arm/boot/uImage
arch/arm/boot/uImage
[ATDE ~/linux-[VERSION]]$ ls arch/arm/boot/dts/armadillo-iotg-a6e*.dtb*
arch/arm/boot/dts/armadillo-iotg-a6e-ems31.dtbo
arch/arm/boot/dts/armadillo-iotg-a6e.dtb
arch/arm/boot/dts/armadillo-iotg-a6e-lwb5plus.dtbo
```

図 10.117 Linux カーネルビルドしたファイルの確認

10.6.3. Alpine Linux ルートファイルシステムをビルドする

ここでは、build-rootfs を使って、 Alpine Linux ルートファイルシステムを構築する方法を説明します。

build-rootfs は、ATDE 上で Armadillo-IoT ゲートウェイ A6E 用の Alpine Linux ルートファイルシステムを構築することができるツールです。

10.6.3.1. デフォルトの Alpine Linux ルートファイルシステムを構築する

1. ルートファイルシステムのビルドに必要な Podman のインストール

   次のコマンドを実行します。

```
[ATDE ~]$ sudo apt install podman btrfs-progs xxhash
```

2. build-rootfs の入手

   Armadillo-IoT ゲートウェイ A6E 開発用ツール [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/tools] から 「Alpine Linux ルートファイルシステムビルドツール」 ファイル (build-rootfs-[VERSION].tar.gz) を次のようにダウンロードします。

```
[PC ~/]$ wget https://download.atmark-techno.com/armadillo-iot-a6e/tool/build-rootfs-
latest.tar.gz
[PC ~/]$ tar xf build-rootfs-[VERSION].tar.gz
[PC ~/]$ cd build-rootfs-[VERSION]
```

3. ビルド

   次のコマンドを実行します。

   パッケージをインターネット上から取得するため回線速度に依存しますが、ビルドには数分かかります。

```
[ATDE ~/build-rootfs-[VERSION]]$ sudo ./build_rootfs.sh -b a6e
use default(outdir=/home/atmark/git/build-rootfs)
use default(output=baseos-6e-ATVERSION.tar.zst)
```

```
:
: (略)
:
> Creating rootfs archive
-rw-r--r--    1 root     root     231700480 Oct 11 07:18 rootfs.tar
ERROR: No such package: .make-alpine-make-rootfs
=========================================
footprint[byte]  tarball[byte]  packages
      229904000       74942331  alpine-base coreutils chrony ...(省略)
=========================================
done.
```

ビルド時のログにエラー"ERROR: No such package: .make-alpine-make-rootfs"が出ていますが、正常時でも出力されるメッセージのため、問題はありません。

リリース時にバージョンに日付を含めたくないときは --release を引数に追加してください。

任意のパス、ファイル名で結果を出力することもできます。

```
[ATDE ~/build-rootfs-[VERSION]]$ ./build_rootfs.sh -b a6e ~/
alpine.tar.gz
:
: (略)
:
[ATDE ~/build-rootfs-[VERSION]]$ ls ~/alpine.tar.gz
~/alpine.tar.gz
```

4. ビルド結果の確認

   次のコマンドを実行します。

```
[ATDE ~/build-rootfs-[VERSION]]$ ls *tar.zst
baseos-6e-[VERSION].tar.zst
```

10.6.3.2. Alpine Linux ルートファイルシステムをカスタマイズする

alpine/build-rootfs ディレクトリ直下にある a6e ディレクトリ以下のファイルを変更し、build_rootfs.sh を実行することで、ルートファイルシステムをカスタマイズすることができます。

- install

    - resources/ ディレクトリ内のファイルを、 ルートファイルシステムにインストールするためのスクリプト

- resources/
    - ルートファイルシステムにインストールするファイルを含んだディレクトリ
- packages
    - ルートファイルシステムにインストールするパッケージのリスト
- fixup
    - パッケージのインストールや上記 install スクリプトの後に実行されるスクリプト
- ファイル/ディレクトリを追加する

a6e/resources/ 以下に配置したファイルやディレクトリは、そのままルートファイルシステムの直下にコピーされます。デフォルトでは、UID と GID は共に root、パーミッションは 0744(ディレクトリの場合は 0755)となります。

a6e/install を修正することで、ファイルの UID、GID、パーミッションを変更することができます。UID、GID を変更する場合は chown、パーミッションを変更する場合は chmod を利用してください。

- パッケージを変更する

a6e/packages を変更することで、ルートファイルシステムにインストールするパッケージをカスタマイズすることができます。

パッケージ名は 1 行に 1 つ書くことができます。パッケージ名は Armadillo 上で"apk add"の引数に与えることのできる正しい名前で記載してください。誤ったパッケージ名を指定した場合は、ルートファイルシステムのビルドに失敗します。

利用可能なパッケージは以下のページで検索することができます。

Alpine Linux Packages https://pkgs.alpinelinux.org/packages

Alpine Linux ルートファイルシステム使用した Armadillo で検索することもできます。

```
[armadillo ~]# apk list *ruby*
ruby-rmagick-4.1.2-r1 armhf {ruby-rmagick} (MIT)
ruby-concurrent-ruby-ext-1.1.6-r1 armhf {ruby-concurrent-ruby} (MIT)
ruby-net-telnet-2.7.2-r3 armhf {ruby} (Ruby AND BSD-2-Clause AND MIT)
:
:（省略）
:
ruby-mustache-1.1.1-r3 armhf {ruby-mustache} (MIT)
ruby-nokogiri-1.10.10-r0 armhf {ruby-nokogiri} (MIT)
```

10.7. SD ブートの活用

本章では、microSD カードから直接起動(以降「SD ブート」と表記します)する手順を示します。SD ブートを活用すると、microSD カードを取り替えることでシステムイメージを変更することができます。本章に示す手順を実行するためには、容量が 8Gbyte 以上の microSD カードを必要とします。

SD ブートを行った場合、ブートローダーの設定は **microSD カード** に保存されます。

10.7.1. ブートディスクの作成

1. ブートディスクイメージのビルドします

   「10.6.3. Alpine Linux ルートファイルシステムをビルドする」 で説明されているソースツリー alpine/build-rootfs にあるスクリプト `build_image` と 「10.6.1. ブートローダーをビルドする」でビルドした `u-boot-dtb.imx` を利用します。

```
[ATDE ~/build-rootfs-[VERSION]]$ sudo ./build_image.sh --board a6e ¥
          --boot ~/u-boot-[VERSION]/u-boot-dtb.imx
:（省略）
[ATDE ~/build-rootfs-[VERSION]]$ ls baseos-6e*img
baseos-6e-[VERSION].img
```

1. ATDE に microSD カードを接続します。詳しくは 「4.2.2. 取り外し可能デバイスの使用」 を参考にしてください。

2. microSD カードのデバイス名を確認します

```
[ATDE ~]$ ls /dev/sd?
/dev/sda  /dev/sdb
[ATDE ~]$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 7.22 GiB, 7751073792 bytes, 15138816 sectors
Disk model: SD/MMC
:（省略）
```

3. microSD カードがマウントされている場合、アンマウントします。

```
[ATDE ~]$ mount
:（省略）
/dev/sdb1 on /media/52E6-5897 type ext2
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=cp437,iocharset
=utf8,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
[ATDE ~]$ sudo umount /dev/sdb1
```

図 10.118 自動マウントされた microSD カードのアンマウント

4. ブートディスクイメージの書き込み

```
[ATDE ~]$ sudo dd if=~/build-rootfs-[VERSION]/baseos-6e-[VERSION].img ¥
                 of=/dev/sdb bs=1M oflag=direct status=progress
```

microSD カードの性能にもよりますが、書き込みには 5 分程度かかります。

microSD カードのパーティション構成は次のようになっています。

表 10.24 microSD カードのパーティション構成

パーティション	オフセット	サイズ	説明
-	0	10MiB	ブートローダー
1	10MiB	300MiB	A/B アップデートの A 面パーティション
2	310MiB	300MiB	A/B アップデートの B 面パーティション
3	610MiB	50MiB	ログ用パーティション
4	660MiB	200MiB	ファームウェア
5	860MiB	残り	アプリケーション用パーティション

gdisk で確認すると次のようになります。

```
[ATDE ~]$ sudo gdisk -l /dev/sdb
GPT fdisk (gdisk) version 1.0.6

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sdb: 60506112 sectors, 28.9 GiB
Model: VMware Virtual I
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 44B816AC-8E38-4B71-8A96-308F503238E3
Partition table holds up to 128 entries
Main partition table begins at sector 20448 and ends at sector 20479
First usable sector is 20480, last usable sector is 60485632
Partitions will be aligned on 2048-sector boundaries
Total free space is 0 sectors (0 bytes)

Number  Start (sector)    End (sector)  Size       Code  Name
   1           20480          634879   300.0 MiB   8300  rootfs_0
   2          634880         1249279   300.0 MiB   8300  rootfs_1
   3         1249280         1351679   50.0 MiB    8300  logs
   4         1351680         1761279   200.0 MiB   8300  firm
   5         1761280        60485632   28.0 GiB    8300  app
```

10.7.2. SD ブートの実行

「10.7.1. ブートディスクの作成」で作成したブートディスクから起動する方法を説明します。

1. Armadillo-IoT ゲートウェイ A6E に電源を投入する前に、ブートディスクを CON1(SD インターフェース)に挿入します。また、SW2 を 起動デバイスは microSD 側設定します。SW2 に関しては、「図 14.16. スイッチの状態と起動デバイス」 を参照ください。

2. 電源を投入します。

```
U-Boot 2020.04 (Oct 25 2022 - 10:37:29 +0900)

CPU:   i.MX6ULL rev1.1 at 396 MHz
Model: Atmark Techno Armadillo-IoT Gateway A6E Board
DRAM:  512 MiB
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... *** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Saving Environment to MMC... Writing to redundant MMC(1)... OK
switch to partitions #0, OK
mmc1 is current device
flash target is MMC:1
Net:
Warning: ethernet@2188000 using MAC address from ROM
eth0: ethernet@2188000
Fastboot: Normal
Normal Boot
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1 is current device
11660400 bytes read in 524 ms (21.2 MiB/s)
Booting from mmc ...
38603 bytes read in 22 ms (1.7 MiB/s)
Loading fdt boot/armadillo.dtb
## Booting kernel from Legacy Image at 80800000 ...

...中略...

Welcome to Alpine Linux 3.16
Kernel 5.10.149-1-at on an armv7l (/dev/ttymxc2)

armadillo login:
```

10.7.3. ゲートウェイコンテナのインストール

「10.7.1. ブートディスクの作成」で作成したブートディスクには、ゲートウェイコンテナが含まれていません。必要な場合は、ゲートウェイコンテナの SWU イメージを作成してインストールする必要があります。

1. 「10.9.2. SWU イメージの作成」 記載の手順で、最初の書き込み用の SWU イメージ initial_setup.swu を作成します。

2. ゲートウェイコンテナの SWU イメージを作成

   1. Armadillo-IoT ゲートウェイ A6E ゲートウェイコンテナ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/container] から「SWU イメージ作成アーカイブ」ファイル (a6e-gw-container-[version].tar.gz) を「図 10.119. ゲートウェイコンテナ SWU イメージアーカイブをダウンロードし、 SWU イメージを作成する」に示す手順でダウンロードし、 SWU イメージを作成します。

```
[ATDE ~]$ wget https://armadillo.atmark-techno.com/files/downloads/armadillo-iot-a6e/container/a6e-gw-container-[version].tar.gz ❶
[ATDE ~]$ mkdir a6e-gw-container-swu
[ATDE ~]$ tar xf a6e-gw-container-[version].tar.gz -C a6e-gw-container-swu
[ATDE ~]$ cd a6e-gw-container-swu
[ATDE ~]$ mkswu a6e-gw-container.desc ❷
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
以下のファイルを USB メモリにコピーしてください：
'/path/to/a6e-gw-container.swu'
'/path/to/a6e-gw-container-image-[version].tar'
'/path/to/.a6e-gw-container/a6e-gw-container-image-[version].tar.sig'
```

図 10.119 ゲートウェイコンテナ SWU イメージアーカイブをダウンロードし、 SWU イメージを作成する

   ❶ ゲートウェイコンテナ SWU イメージアーカイブをダウンロードします

   ❷ mkswu コマンドで SWU イメージを作成します

3. SWU イメージのインストール

   「10.9.3. イメージのインストール」 の手順に従い、最初の書き込み用の SWU イメージと、ゲートウェイコンテナ SWU イメージをインストールします。なお、必ず最初の書き込み用の SWU イメージを先にインストールするよう注意してください。

10.8. Armadillo のソフトウェアの初期化

microSD カードを使用し、Armadillo Base OS の初期化を行えます。

初期化を行っても、ファームウェアパーティション(mmcblk0p4)は変更されません。

10.8.1. インストールディスクの作成

インストールディスクは二つの種類があります：

- 初期化インストールディスク。Armadillo-IoT ゲートウェイ A6E インストールディスクイメージ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/disc-image] にある標準のイメージです。
- 開発が完了した Armadillo-IoT ゲートウェイ A6E をクローンするためのインストールディスク。

10.8.1.1. 初期化インストールディスクの作成

1. 512 MB 以上の microSD カードを用意してください。

2. 標準のインストールディスクイメージを使用する場合は、Armadillo-IoT ゲートウェイ A6E インストールディスクイメージ [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/disc-image] から「Armadillo Base OS」をダウンロードしてください。

   「10.6. Armadillo のソフトウェアをビルドする」 でビルドしたイメージを使用してインストールディスクを作成したい場合は、以下のコマンドを実行して、インストールディスクイメージを作成してください。

   ```
   [ATDE ~/build-rootfs-[VERSION]]$ sudo ./build_image.sh --board a6e
   : (省略)
   [ATDE ~/build-rootfs-[VERSION]]$ ls baseos-6e*img
   baseos-6e-[VERSION].img
   [ATDE ~/build-rootfs-[VERSION]]$ sudo ./build_image.sh --board a6e ¥
           --boot ~/u-boot-[VERSION]/u-boot-dtb.imx ¥
           --installer ./baseos-6e-[VERSION].img
   ```

   コマンドの実行が完了すると、baseos-6e-[VERSION]-installer.img というファイルが作成されていますので、こちらを使用してください。

3. ATDE に microSD カードを接続します。詳しくは「4.2.2. 取り外し可能デバイスの使用」を参考にしてください。

4. microSD カードのデバイス名を確認します

   ```
   [ATDE ~]$ ls /dev/sd?
   /dev/sda  /dev/sdb
   [ATDE ~]$ sudo fdisk -l /dev/sdb
   Disk /dev/sdb: 7.22 GiB, 7751073792 bytes, 15138816 sectors
   Disk model: SD/MMC
   : (省略)
   ```

5. microSD カードがマウントされている場合、アンマウントします。

   ```
   [ATDE ~]$ mount
   : (省略)
   /dev/sdb1 on /media/52E6-5897 type ext2
   (rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=cp437,iocharset
   =utf8,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
   [ATDE ~]$ sudo umount /dev/sdb1
   ```

6. ダウンロードしたファイルを展開し、img ファイルを microSD カードに書き込んでください。

   Linux PC の場合、以下のように microSD カードに書き込むことができます。

   ```
   [ATDE ~]$ unzip baseos-6e-installer-[VERSION].zip
   [ATDE ~]$ sudo dd if=baseos-6e-installer-[VERSION].img ¥
                   of=/dev/sdb bs=1M oflag=direct status=progress
   ```

また、Windows の場合、エクスプローラー等で Zip ファイルから img ファイルを取り出し、「Win32 Disk Imager」などを使用して microSD カードに書き込むことができます。

10.8.1.2. 開発が完了した Armadillo をクローンするインストールディスクの作成

1. microSD カードを用意してください。
2. 初期化インストールディスクをベースとしますので、「10.8.1.1. 初期化インストールディスクの作成」 でビルドした SD カードを使用できますが、用意されていなければ次のステップで自動的にダウンロードされます。
3. abos-ctrl make-installer を実行してください

```
[armadillo ~]# abos-ctrl make-installer
It looks like your SD card does not contain an installer image
Download base SD card image from https://armadillo.atmark-techno.com (~200MB) ? [y/N]
WARNING: it will overwrite your sd card!!
y
Downloading installer image
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  167M  100  167M    0     0   104M      0  0:00:01  0:00:01 --:--:--  104M
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    70  100    70    0     0   1441      0 --:--:-- --:--:-- --:--:--  1458
Writing baseos-6e-installer-3.15.4-at.6.img to SD card (442M)
439353344 bytes (439 MB, 419 MiB) copied, 134 s, 3.3 MB/s
421+0 records in
421+0 records out
441450496 bytes (441 MB, 421 MiB) copied, 134.685 s, 3.3 MB/s
Verifying written image is correct
436207616 bytes (436 MB, 416 MiB) copied, 46 s, 9.5 MB/s
421+0 records in
421+0 records out
441450496 bytes (441 MB, 421 MiB) copied, 46.8462 s, 9.4 MB/s
Checking and growing installer main partition
GPT data structures destroyed! You may now partition the disk using fdisk or
other utilities.
Setting name!
partNum is 0
The operation has completed successfully.
e2fsck 1.46.4 (18-Aug-2021)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
rootfs_0: 2822/102400 files (0.5% non-contiguous), 352391/409600 blocks
(1/1) Installing e2fsprogs-extra (1.46.4-r0)
Executing busybox-1.34.1-r5.trigger
OK: 202 MiB in 197 packages
resize2fs 1.46.4 (18-Aug-2021)
Resizing the filesystem on /dev/mmcblk1p1 to 15547884 (1k) blocks.
The filesystem on /dev/mmcblk1p1 is now 15547884 (1k) blocks long.

Currently booted on /dev/mmcblk0p1
Copying boot image
```

```
Copying rootfs
301989888 bytes (302 MB, 288 MiB) copied, 10 s, 30.1 MB/s
300+0 records in
300+0 records out
314572800 bytes (315 MB, 300 MiB) copied, 10.3915 s, 30.3 MB/s
Copying /opt/firmware filesystem
Copying appfs
At subvol app/snapshots/volumes
At subvol app/snapshots/boot_volumes
At subvol app/snapshots/boot_containers_storage
Cleaning up and syncing changes to disk...
Installer updated successfully!
```

10.8.2. インストールディスクを使用する

1. SW2(起動デバイス設定スイッチ)を ON にし、起動デバイスを microSD に設定します。
2. microSD カードを CON1 に挿入します。
3. 電源を投入すると、1 分程度で eMMC のソフトウェアの初期化が完了します。
4. 完了すると電源が切れます(SYS(システム LED)が消灯、コンソールに `reboot: Power down` が表示)。
5. 電源を取り外し、続いて SW2 を OFF に設定し、microSD カードを外してください。
6. 10 秒以上待ってから再び電源を入れると、初回起動時と同じ状態になります。

10.9. Armadillo のソフトウェアをアップデートする

Armadillo-IoT ゲートウェイ A6E では、開発・製造・運用それぞれに適した複数のソフトウェアアップデート方法を用意しています。本章では、それぞれのソフトウェアアップデート方法について説明します。

ソフトウェアアップデートを実現するソフトウェアの概要や仕様、用語については「13. ソフトウェア仕様」を参照してください。

10.9.1. SWU イメージとは

Armadillo Base OS ではソフトウェアアップデートのために OS やコンテナ等を格納するために SWU というイメージ形式を使います。

SWU イメージは `swupdate` (https://sbabic.github.io/swupdate/swupdate.html) によって Armadillo Base OS 上で検証とインストールが実行されます。SWU イメージを Armadillo に転送するための方法は、用途や状況に合わせて様々な方法を用意しています。例えば、USB メモリから読み取る、ウェブサーバーからダウンロードする、hawkBit という Web アプリケーションを使うなどです。

10.9.2. SWU イメージの作成

SWU イメージの作成には、mkswu というツールを使います。

mkswu に含まれる `mkswu` を実行すると、アップデート対象やバージョン等の情報を記載した `.desc` ファイルに含まれる命令を順次実行してイメージを作り上げます。

詳しくは「10.9.5. mkswu の desc ファイル」を参考にしてください。

1. mkswu の取得

```
[ATDE ~]$ sudo apt update && sudo apt install mkswu
```

インストール済みの場合は、以下のコマンドを実行し最新版への更新を行ってください。

```
[ATDE ~]$ sudo apt update && sudo apt upgrade
```

git のバージョンからアップデートする場合、 mkswu --import で以前使っていたコンフィグをロードしてください。

```
[ATDE ~/swupdate-mkimage]$ mkswu --import
コンフィグファイルを更新しました：/home/atmark/swupdate-mkimage/
mkswu.conf
/home/atmark/swupdate-mkimage/mkswu.conf のコンフィグファイルとそ
の鍵を
/home/atmark/mkswu にコピーします。
mkdir: ディレクトリ '/home/atmark/mkswu' を作成しました
'/home/atmark/swupdate-mkimage/swupdate.key' -> '/home/atmark/
mkswu/swupdate.key'
'/home/atmark/swupdate-mkimage/swupdate.pem' -> '/home/atmark/
mkswu/swupdate.pem'
/home/atmark/swupdate-mkimage/mkswu.conf のコンフィグファイルを
/home/atmark/mkswu/mkswu.conf にコピーしました。
mkswu でイメージ作成を試してから前のディレクトリを消してください。
```

2. 最初に行う設定

mkswu --init を実行して鍵や最初の書き込み用のイメージを生成します。作成する鍵は、swu パッケージを署名するために使用します。

過去に本手順を行っている場合、再度初回アップデート作業を行う必要はありません。再度アップデートを行う際には、Armadillo に配置した公開鍵に対応する秘密鍵でアップデートを行いますので、 「10.9.5. mkswu の desc ファイル」 を参考にしてください。

```
[ATDE ~]$ mkswu --init
mkdir: ディレクトリ '/home/atmark/mkswu' を作成しました
コンフィグファイルを更新しました：/home/atmark/mkswu/mkswu.conf
証明書の Common name を入力してください: [COMMON_NAME] ❶
証明書の鍵のパスワードを入力ください（4-1024 文字）❷
証明書の鍵のパスワード（確認）：
Generating an EC private key
writing new private key to '/home/atmark/mkswu/swupdate.key'
 -----
アップデートイメージを暗号化しますか？ (N/y) ❸
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) ❹
root パスワード: ❺
root パスワード（確認）：
```

```
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）：❻
atmark ユーザのパスワード（確認）：
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？
(y/N) ❼
/home/atmark/mkswu/initial_setup.swu を作成しました。

"/home/atmark/mkswu/initial_setup.swu" をそのまま使うことができますが、
モジュールを追加してイメージを再構築する場合は次のコマンドで作成してください：
  mkswu "/home/atmark/mkswu/initial_setup.desc" [他の.desc ファイル]

インストール後は、このディレクトリを削除しないように注意してください。
鍵を失うと新たなアップデートはデバイスの /etc/swupdate.pem
を修正しないとインストールできなくなります。

[ATDE ~]$ ls ~/mkswu
initial_setup.desc  initial_setup.swu  mkswu.conf
swupdate.aes-key    swupdate.key       swupdate.pem ❽
```

❶ COMMON_NAME には証明鍵の「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てください。

❷ 証明鍵を保護するパスフレーズを 2 回入力します。

❸ swu イメージ自体を暗号化する場合に「y」を入力します。詳細は 「10.9.8. SWUpdate と暗号化について」 を参考にしてください。

❹ アットマークテクノのアップデートをインストールしない場合は「n」を入力します。

❺ root のパスワードを 2 回入力します。

❻ atmark ユーザーのパスワードを 2 回入力します。何も入力しない場合はユーザーをロックします。

❼ 自動アップデートを無効のままで進みます。ここで「y」を入れると、定期的にアットマークテクノのサーバーからアップデートの有無を確認し、自動的にインストールします。

❽ 作成したファイルを確認します。「swupdate.aes-key」は暗号化の場合にのみ作成されます。

このイメージは初回インストール用の署名鍵を使って、作成した鍵とユーザーのパスワードを設定します。

インストール後にコンフィグの mkswu.conf と鍵の　swupdate.* をなくさないようにしてください。

このイメージに他の変更も入れれます。他の /usr/share/mkswu/examples/ ディレクトリにある.desc ファイルや「10.9.5. mkswu の desc ファイル」を参考にして、以下の例のように同じ swu にいくつかの.desc を組み込めます。

例えば、openssh を有効にします。

```
[ATDE ~/mkswu]$ cp -rv /usr/share/mkswu/examples/enable_sshd* .
:（省略）
'/usr/share/mkswu/examples/enable_sshd/root/.ssh/
authorized_keys'
    -> './enable_sshd/root/.ssh/authorized_keys'
'/usr/share/mkswu/examples/enable_sshd.desc' -> './
```

```
enable_sshd.desc'
[ATDE ~/mkswu]$ cp ~/.ssh/id_rsa.pub ¥
                  enable_sshd/root/.ssh/authorized_keys
[ATDE ~/mkswu]$ mkswu initial_setup.desc enable_sshd.desc
enable_sshd.desc を組み込みました。
initial_setup.swu を作成しました。
```

3. イメージのインストール

   「10.9.3. イメージのインストール」を参考に、作成したイメージをインストールしてください。

4. 次回以降のアップデート

   次回以降のアップデートは作成した証明鍵を使用して Armadillo-IoT ゲートウェイ A6E の SWU イメージを作成します。

   .desc ファイルの内容は /usr/share/mkswu/examples/ のディレクトリや「10.9.5. mkswu の desc ファイル」を参考にしてください。

10.9.3. イメージのインストール

イメージをインストールする方法として以下に示すような方法があります。

・USB メモリまたは SD カードからの自動インストール

Armadillo-IoT ゲートウェイ A6E に USB メモリを接続すると自動的にアップデートが始まります。アップデート終了後に Armadillo-IoT ゲートウェイ A6E は自動で再起動します。

USB メモリや SD カードを vfat もしくは ext4 形式でフォーマットし、作成した.swu のファイルをディレクトリを作らずに配置してください。

ATDE 上で USB メモリ/microSD カードのパーティションを作成・フォーマットする方法

https://armadillo.atmark-techno.com/howto/atde-partition-howto

```
[ATDE ~/mkswu]$ df -h
Filesystem           Size  Used Avail Use% Mounted on
:（省略）
/dev/sda1        15G  5.6G  9.1G  39% /media/USBDRIVE ❶
[ATDE ~/mkswu]$ cp initial_setup.swu /media/USBDRIVE/ ❷
[ATDE ~/mkswu]$ umount /media/USBDRIVE ❸
```

❶ USB メモリがマウントされている場所を確認します。

❷ ファイルをコピーします。

❸ /media/USBDRIVE をアンマウントします。コマンド終了後に USB メモリを取り外してください。

エラーの場合、/var/log/message に保存されます。例えば、コンソールで証明の間違ったイメージのエラーを表示します：

```
[armadillo ~]# tail /var/log/messages
Nov 19 10:48:42 user.notice swupdate-auto-update: Mounting sda0 on /mnt
Nov 19 10:48:42 user.notice swupdate-auto-update: Trying update /mnt/initial_setup.swu
Nov 19 10:48:42 user.info swupdate: START Software Update started !
Nov 19 10:48:42 user.err swupdate: FAILURE ERROR : Signature verification failed ❶
Nov 19 10:48:42 user.err swupdate: FAILURE ERROR : Compatible SW not found
Nov 19 10:48:42 user.err swupdate: FATAL_FAILURE Image invalid or corrupted. Not installing ...
```

❶ 証明が間違ったメッセージ。

・外部記憶装置からイメージのインストール（手動）

USB メモリや microSD カード等の外部記憶装置のルートディレクトリ以外に swu イメージを保存して、イメージのインストールを行います。ルートディレクトリに保存すると自動アップデートが行われますので、/var/log/messages を確認してください。

以下は外部記憶装置が/dev/mmcblk1p1(microSD カード）として認識された場合に、イメージのインストールを行う例です。

```
[armadillo ~]# mount /dev/mmcblk1p1 /mnt
[armadillo ~]# swupdate -i /mnt/swu/initial_setup.swu
SWUpdate v5f2d8be-dirty

Licensed under GPLv2. See source distribution for detailed copyright notices.

[INFO ] : SWUPDATE running :  [main] : Running on AGX4500 Revision at1
[INFO ] : SWUPDATE started :  Software Update started !
[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying current os over
[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers
[INFO ] : SWUPDATE running :  [read_lines_notify] : swupdate triggering reboot!
Killed
```

・ウェブサーバーからイメージのインストール（手動）

swu イメージをウェブサーバーにアップロードして、イメージのインストールを行います。以下は、http://server/initial_setup.swu のイメージをインストールする例です。

```
[armadillo ~]# swupdate -d '-u http://server/initial_setup.swu'
SWUpdate v5f2d8be-dirty

Licensed under GPLv2. See source distribution for detailed copyright notices.

[INFO ] : SWUPDATE running :  [main] : Running on AGX4500 Revision at1
[INFO ] : SWUPDATE running :  [channel_get_file] : Total download size is 25 kB.
[INFO ] : SWUPDATE started :  Software Update started !
[INFO ] : SWUPDATE running :  [read_lines_notify] : No base os update: copying current os over
[INFO ] : SWUPDATE running :  [read_lines_notify] : Removing unused containers
[INFO ] : SWUPDATE running :  [read_lines_notify] : swupdate triggering reboot!
Killed
```

・ ウェブサーバーからの定期的な自動インストール

swupdate-url を有効にしたら、定期的にチェックしてインストールします。以下はサービスの有効化とタイミングの設定の例です。

```
[armadillo ~]# rc-update add swupdate-url ❶
[armadillo ~]# persist_file /etc/runlevels/default/swupdate-url ❷
[armadillo ~]#
    echo https://download.atmark-techno.com/armadillo-iot-a6e/image/baseos-6e-latest.swu ¥
                    > /etc/swupdate.watch ❸
[armadillo ~]# echo 'schedule="0 tomorrow"' > /etc/conf.d/swupdate-url
[armadillo ~]# echo 'rdelay="21600"' >> /etc/conf.d/swupdate-url ❹
[armadillo ~]# persist_file /etc/swupdate.watch /etc/conf.d/swupdate-url ❺
```

❶ swupdate-url サービスを有効します。

❷ サービスの有効化を保存します。

❸ イメージの URL を登録します。一行ごとにイメージの URL を設定することができ、複数行にイメージの URL を設定することができます。

❹ チェックやインストールのスケジュールを設定します。

❺ 変更した設定ファイルを保存します。

USB メモリからのアップデートと同様に、ログは/var/log/messages に保存されます。

initial_setup のイメージを作成の際に /usr/share/mkswu/examples/enable_swupdate_url.desc を入れると有効にすることができます。

・ hawkBit を使用した自動インストール

hawkBit で Armadillo-IoT ゲートウェイ A6E を複数台管理してアップデートすることができます。以下の「10.9.4. hawkBit サーバーから複数の Armadillo に配信する」を参考にしてください。

10.9.4. hawkBit サーバーから複数の Armadillo に配信する

hawkBit サーバーを利用することで複数の Armadillo のソフトウェアをまとめてアップデートすることができます。

手順は次のとおりです。

1. コンテナ環境の準備

   Docker を利用すると簡単にサーバーを準備できます。Docker の準備については https://docs.docker.com/get-docker/ を参照してください。

   Docker の準備ができたら、要件に合わせてコンテナの設定を行います。

   ・ ATDE の場合

- apt update && apt install mkswu で最新のバージョンを確認してください。
- ポート転送も必要です。一番シンプルな、プロキシを使用しない場合は 8080、TLS を使う場合は 443 を転送してください。

  vmware を使う場合は vmware の NAT モードのネットワークを使用している仮想マシン上で Web サーバを構成する [https://kb.vmware.com/s/article/2006955?lang=ja]ページを参考にしてください。
- ホスト PC の IP アドレスを控えておいてください。

- ATDE 以外の場合
  - Armadillo-IoT ゲートウェイ A6E 開発用ツール [https://armadillo.atmark-techno.com/resources/software/armadillo-iot-a6e/tools] から「Hawkbit docker-compose コンテナ」 をダウンロードして展開してください。この場合、以下に /usr/share/mkswu/hawkbit-compose を使う際に展開先のディレクトリとして扱ってください。
  - docker がアクセスできるホストネームやアドレスを控えておいてください。

2. hawkBit サーバーの準備

   /usr/share/mkswu/hawkbit-compose/setup_container.sh を実行して、質問に答えてください。

   以下に簡単な（TLS を有効にしない）テスト用の場合と、TLS を有効にした場合の例を示します。

   setup_container.sh を一度実行した場合はデータのディレクトリにある setup_container.sh のリンクを実行して、ユーザーの追加等のオプション変更を行うこともできます。詳細は`--help`を参考にしてください。

```
[ATDE ~]$ /usr/share/mkswu/hawkbit-compose/setup_container.sh
docker-compose の設定ファイルと hawkBit のデータをどこに保存しますか？ [/home/atmark/hawkbit-compose] ❶
setup_container.sh へのリンクを /home/atmark/hawkbit-compose に作ります。
docker サービスに接続できませんでした。sudo でもう一度試します。
[sudo] atmark のパスワード: ❷
OK!
Hawkbit admin user name [admin] ❸
admin ユーザーのパスワード: ❹
パスワードを再入力してください:
追加の管理人アカウントのユーザーネーム（空にすると追加しません） ❺
hawkBit の「device」ユーザーを登録しますか？（自動登録用） [Y/n] ❻
device ユーザーのパスワード:
パスワードを再入力してください:
hawkBit の「mkswu」ユーザーを登録しますか？（swu のアップロード用） [Y/n] ❼
ユーザーにロールアウトの権限を与えますか？（インストール要求を出すこと） [Y/n] ❽
mkswu ユーザーのパスワード:
パスワードを再入力してください:
Setup TLS reverse proxy? [y/N] ❾

コンテナの設定が完了しました。docker-compose コマンドでコンテナの管理が可能です。
/home/atmark/hawkbit-compose/setup_container.sh を再び実行すると設定の変更が可能です。
hawkBit コンテナを起動しますか？ [Y/n] ❿
Creating network "hawkbit-compose_default" with the default driver
```

```
Pulling mysql (mysql:5.7)...
: (省略)
Creating hawkbit-compose_hawkbit_1 ... done
Creating hawkbit-compose_mysql_1   ... done
```

図 10.120 hawkBit コンテナの TLS なしの場合（テスト用）の実行例

❶ コンテナのコンフィグレーションとデータベースの場所を設定します。

❷ docker の設定によって sudo が必要な場合もあります。

❸ admin ユーザーのユーザー名を入力します。

❹ admin ユーザーのパスワードを二回入力します。

❺ 追加のユーザーが必要な場合に追加できます。

❻ examples/hawkbit_register.desc で armadillo を登録する場合に作っておいてください。詳細は 「10.9.4.2. SWU で hawkBit を登録する」 を参考にしてください。

❼ hawkbit_push_update でアップデートを CLI で扱う場合は、「Y」を入力してください。詳細は <sct.hawkbit_push_update>> を参照してください。

❽ hawkbit_push_update でアップデートを実行する場合は、「Y」を入力してください。

❾ ここでは http でテストのコンテナを作成するので、「N」のままで進みます。

❿ コンテナを起動します。初期化が終わったら <IP>:8080 でアクセス可能になります。

```
[ATDE ~]$ /usr/share/mkswu/hawkbit-compose/setup_container.sh
docker-compose の設定ファイルと hawkBit のデータをどこに保存しますか？ [/home/atmark/hawkbit-compose]
setup_container.sh へのリンクを /home/atmark/hawkbit-compose に作ります。
docker サービスに接続できませんでした。sudo でもう一度試します。
OK!
Hawkbit admin user name [admin]
admin ユーザーのパスワード:
パスワードを再入力してください:
パスワードが一致しません。
admin ユーザーのパスワード:
パスワードを再入力してください:
追加の管理人アカウントのユーザーネーム（空にすると追加しません）
hawkBit の「device」ユーザーを登録しますか？（自動登録用） [Y/n]
device ユーザーのパスワード:
パスワードを再入力してください:
hawkBit の「mkswu」ユーザーを登録しますか？（swu のアップロード用） [Y/n]
ユーザーにロールアウトの権限を与えますか？（インストール要求を出すこと） [Y/n]
mkswu ユーザーのパスワード:
パスワードを再入力してください:
Setup TLS reverse proxy? [y/N] y ❶
lighttpd が起動中で、リバースプロキシ設定と競合しています。
lighthttpd サービスを停止しますか？ [Y/n] ❷
Synchronizing state of lighttpd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable lighttpd
Removed /etc/systemd/system/multi-user.target.wants/lighttpd.service.
リバースプロキシの設定に証明書の domain name が必要です。
この domain はこのままデバイスからアクセスできる名前にしてください。
例えば、https://hawkbit.domain.tld でアクセスしたら hawkbit.domain.tld、
```

```
https://10.1.1.1 でしたら 10.1.1.1 にしてください。
証明書の domain name: 10.1.1.1 ❸
証明書の有効期限を指定する必要があります。Let's encrypt を使用する場合、
この値は新しい証明書が生成されるまでしか使用されないので、デフォルトの値
のままにしておくことができます。Let's encrypt を使用しない場合、
数年ごとに証明書を新しくすることが最も好ましです。
証明書の有効期間は何日間にしますか？ [3650] ❹
クライアントの TLS 認証を設定するために CA が必要です。
署名 CA のファイルパス（空にするとクライアント TLS 認証を無効になります） [] ❺
サーバーが直接インタネットにアクセス可能であれば、Let's Encrypt の証明書
を設定することができます。TOS への同意を意味します。
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf
certbot コンテナを設定しますか？ [y/N] ❻

/home/atmark/hawkbit-compose/data/nginx_certs/proxy.crt を /usr/local/share/ca-
certificates/ にコピーして、 update-ca-certificates を実行する必要があります。
この base64 でエンコードされたコピーを examples/hawkbit_register.sh の
SSL_CA_BASE64 に指定する手順が推奨されます。

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJlekNDQVNHZ0F3SUJBZ0lVQTByZ0cwcTJF
SFNnampmb0tUZWg3aGlaSVVVd0NnWUlLb1pJemowRUF3SXcKRXpFUk1BOEdBMVVFQXd3SU1UQXVN
UzR4TGpFd0hoY05Nakl3TXpJMU1EVXhOVFU0V2hjTk16SXdNekl5TURVeApOVFU0V2pBVE1SRXdE
d1lEVlFRRERBZ3hNQzR4TGpFdU1UQlpNQk1HQnlxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VICkEwSUFC
SDFFREhBN3NOTlFJUDlTdlhlUnNmWjl2dVVFWkRkMVE2TzViRlV2RTh4UjUwUlBCLzNlajMzd0VI
NEoKYmZqb296bEpXaExlSG5SbGZsaHExVDlKdm5TalV6QlJNQjBHQTFVZERnUVdCQlFBUmYvSkdT
dkVJek5xZ2JMNQpQamY2VGRpSk1EQWZCZ05WSFNNRUdEQVdnQlFBUmYvSkdTdkVJek5xZ2JMNVBq
ZjZUZGlKTURBUEJnTlZIUk1CCkFmOEVCVEFEQVFIL01Bb0dDQ3FHU0000OUJBTUNBMGdBTUVVQ0lD
Nis3ZzJlZk1SRXl0RVk5WDhDNC8vUEw1U1kKWUlGZHUxVFZiUEZrSlV0SUFpRUE4bm1VSnVQSFlz
SHg2N2ErZFRwSXZ1QmJUSG1KbWd6dUl3bTJ2RXppRnZRPQotLS0tLUVORCBDRVJUSUZJQ0FURS0t
LS0tCg== ❼


Let's encrypt の設定は後で足したい場合に setup_container.sh を--letsencrypt で実行してください。

コンテナの設定が完了しました。docker-compose コマンドでコンテナの管理が可能です。
/home/atmark/hawkbit-compose/setup_container.sh を再び実行すると設定の変更が可能です。
hawkBit コンテナを起動しますか？ [Y/n]
```

図 10.121 hawkBit コンテナの TLS ありの場合の実行例

❶ 今回は TLS を有効にするので、「y」を入力します。

❷ lighttpd サービスが起動している場合に聞かれます。不要なので、停止します。

❸ 証明書の common name を入力してください。ATDE の場合、ポート転送によってホストの IP アドレスで接続しますのでそのアドレスを入力します。Let's encrypt を使用する場合には外部からアクセス可能な DNS を入力してください。

❹ 証明書の有効期間を設定します。デフォルトでは 10 年になっています。Let's encrypt を使用する場合には使われていません。

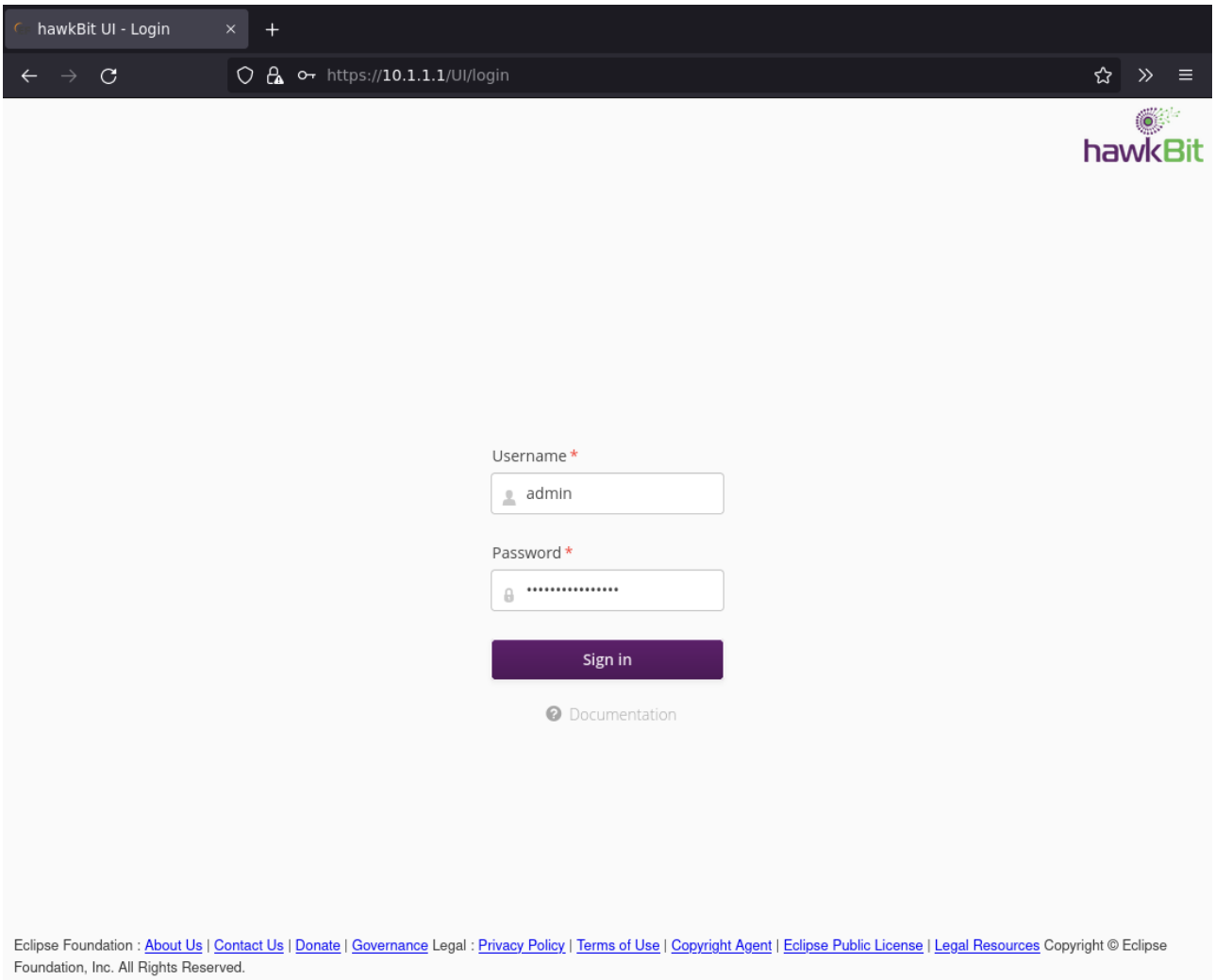
❺ クライアント側では x509 証明書で認証をとることができますが、この例では使用しません。

❻ Let's encrypt による証明書を作成できます。ATDE の場合は外部からのアクセスが難しいので、この例では使用しません。

❼ 自己署名証明書を作成したので、 Armadillo に設置する必要があります。この証明書の取扱いは 「10.9.4.2. SWU で hawkBit を登録する」 を参照してください。

3. hawkBit へのログイン

   作成したコンテナによって http://<サーバーの IP アドレス>:8080 か https://<サーバーのアドレス> にアクセスすると、ログイン画面が表示されます。

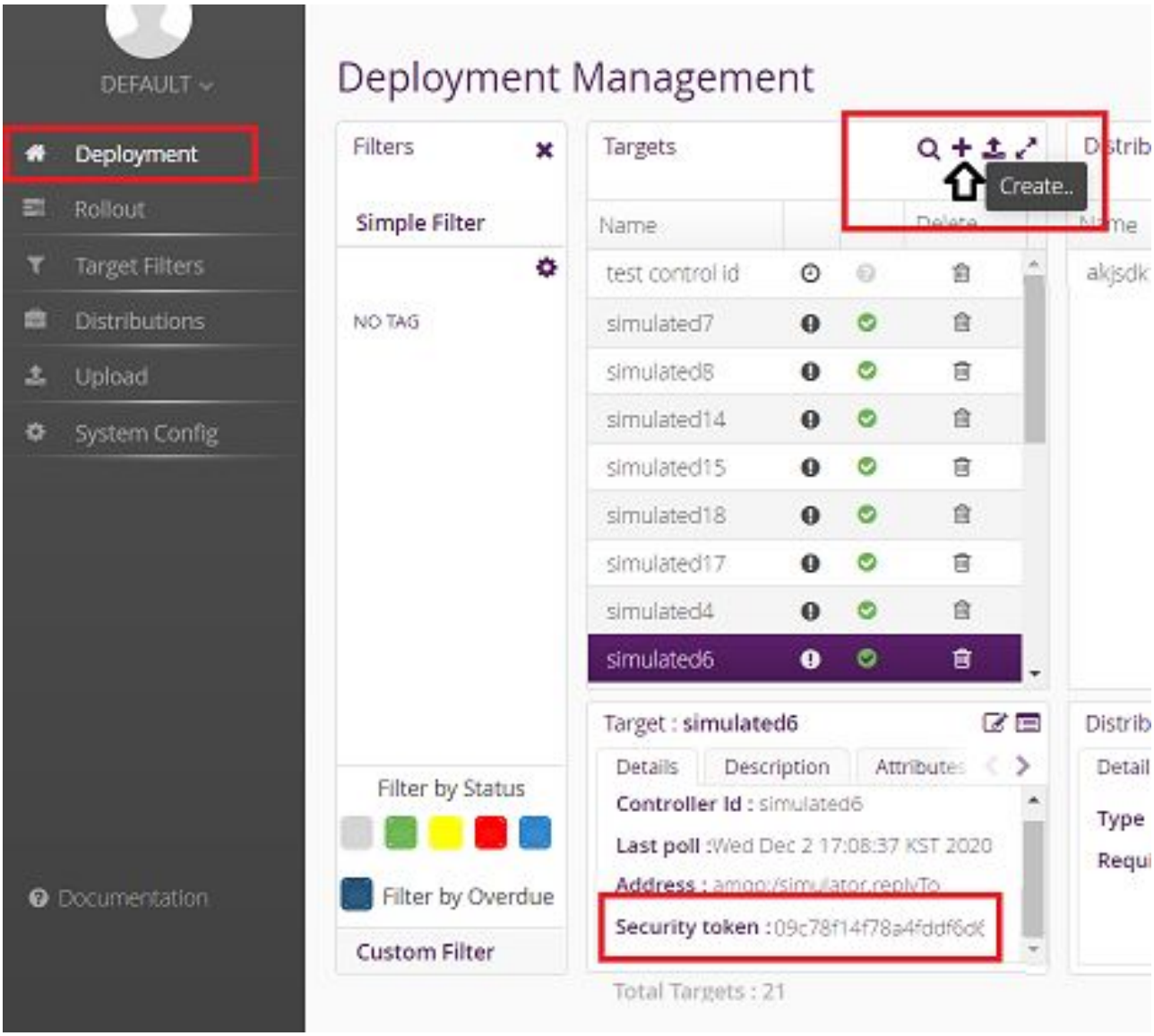


   デフォルトでは次のアカウントでログインできます。

ユーザー	admin
パスワード	admin

4. Armadillo を Target に登録する

   左側のメニューから Deployment をクリックして、Deployment の画面に移ります。



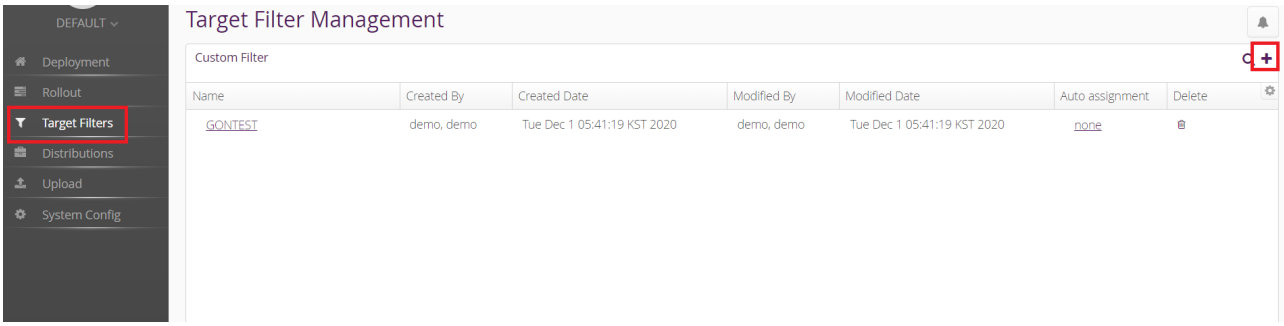
"+"をクリックして Target を作成します。

作成したターゲットをクリックすると、下のペインに "Security token:<文字列>" と表示されるので、<文字列>の部分をメモします。

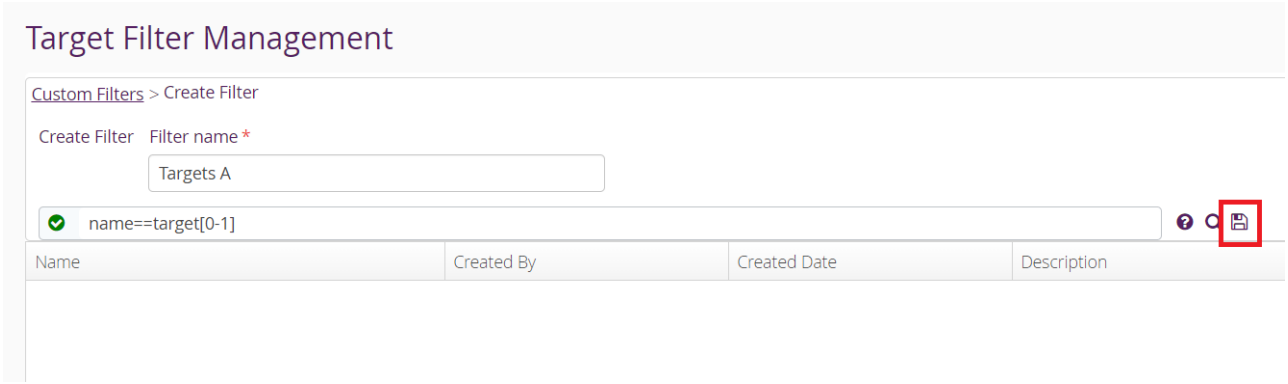
メモした<文字列>を Armadillo の /etc/swupdate.cfg に設定すると Hawkbit への接続認証が通るようになります。

5. Target Filter を作成する

左側のメニューから"Target Filters"をクリックして、Target Filters の画面に移ります。



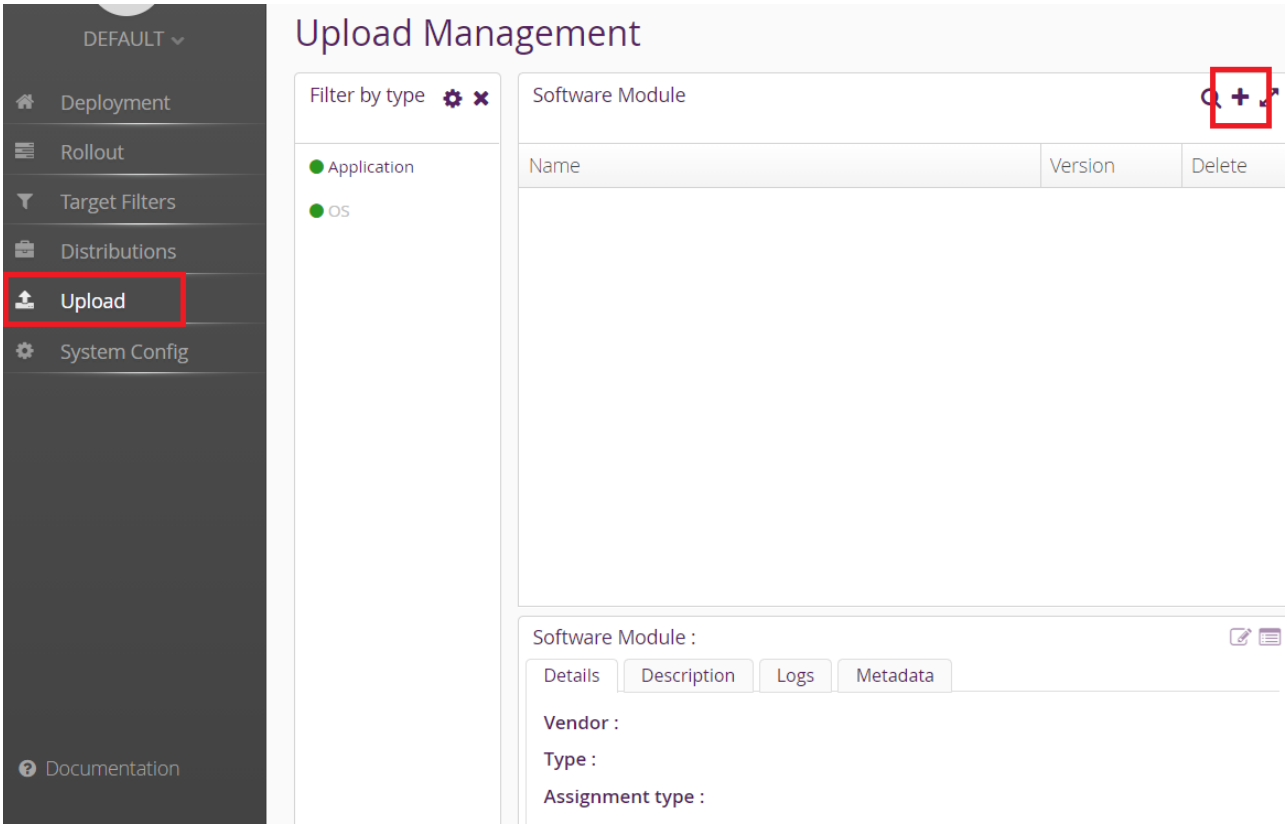
"+" をクリックして新規に Target Filter を作成します。



Filter name と フィルタリング条件を入力して保存します。

6. Software module を作成する

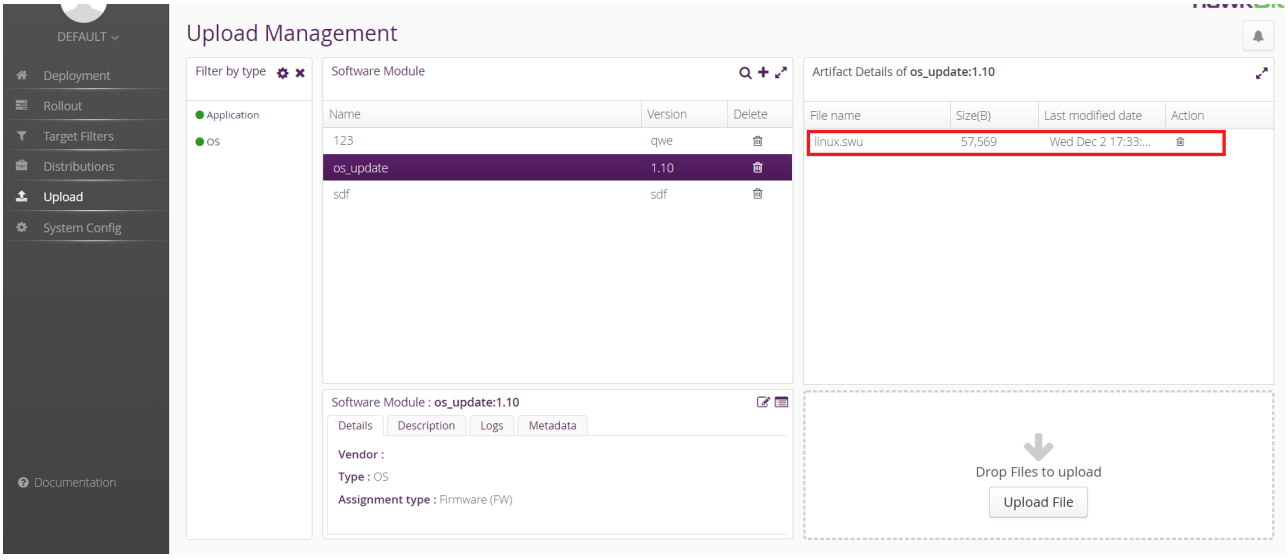
   左側のメニューから"Upload"をクリックして、Upload Management の画面に移ります。



"+" をクリックして Software module を作成します。type には OS/Application、version には任意の文字列を指定します。

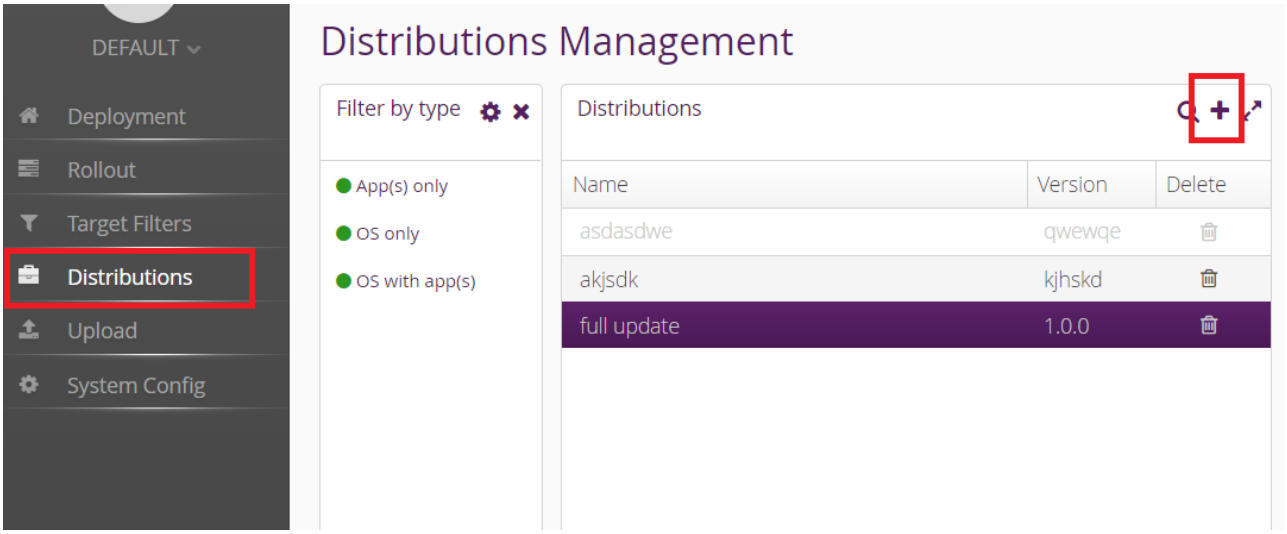
7. swu パッケージをアップロードして Software module に関連付ける

   先程作成した Software module を選択して、ハイライトされた状態で、"Upload File"ボタンをクリックするか、ファイルをドラッグアンドドロップしてアップロードします。

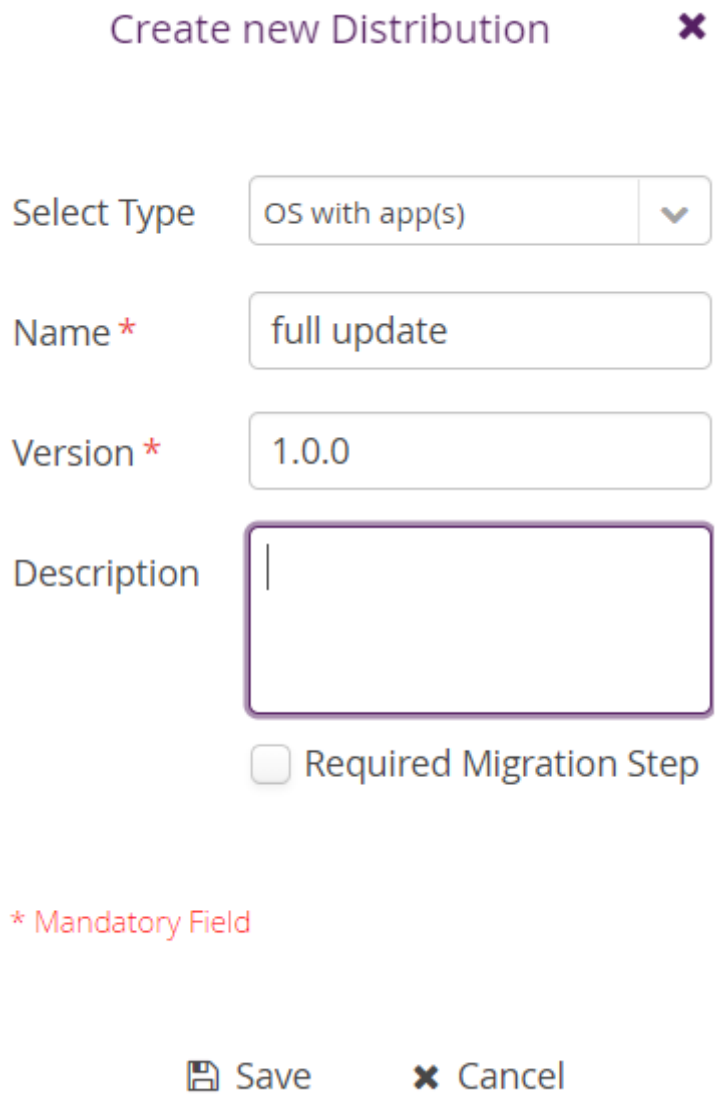


8. Distribution を作成して Software module を関連付ける

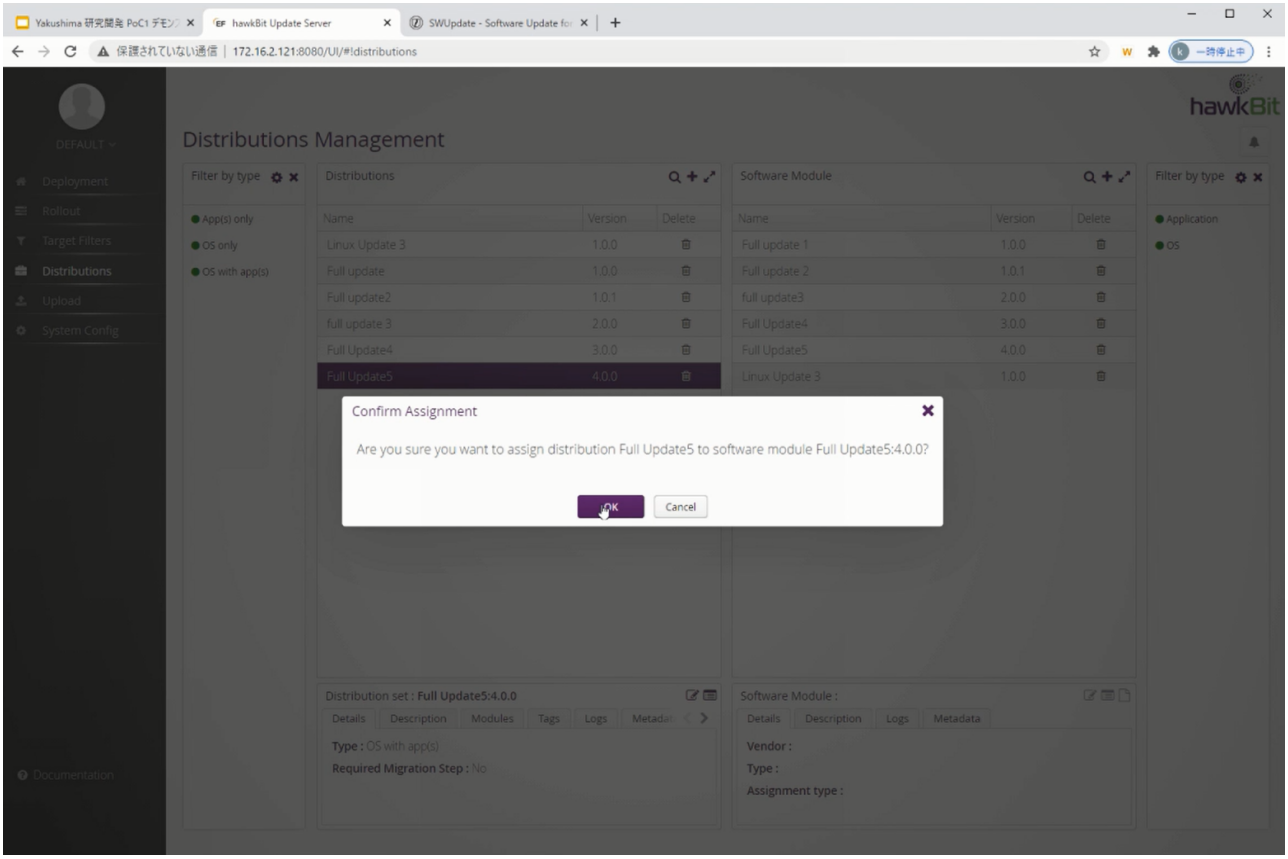
左側のメニューから"Distribution"をクリックして、Distribu てアップロードし tion Management の画面に移ります。



"+" をクリックして Distribution を作成します。type には OS/OSwithApp/Apps、version には任意の文字列を指定します。

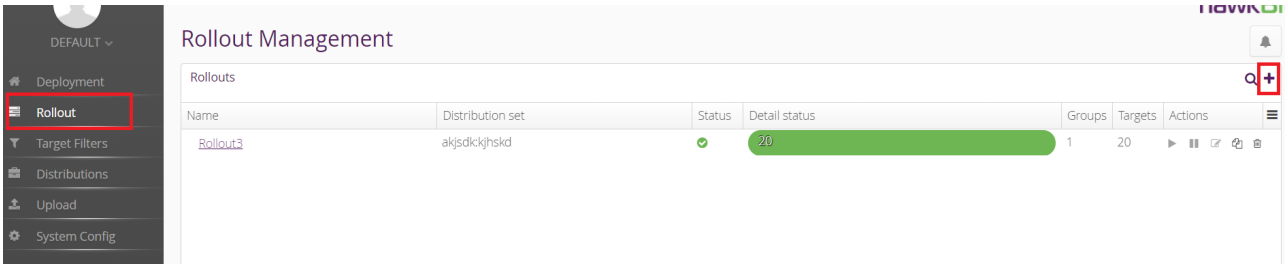


"Software module"のペインから先程作成した Software をドラッグして、作成した Distribution の上にドロップします。

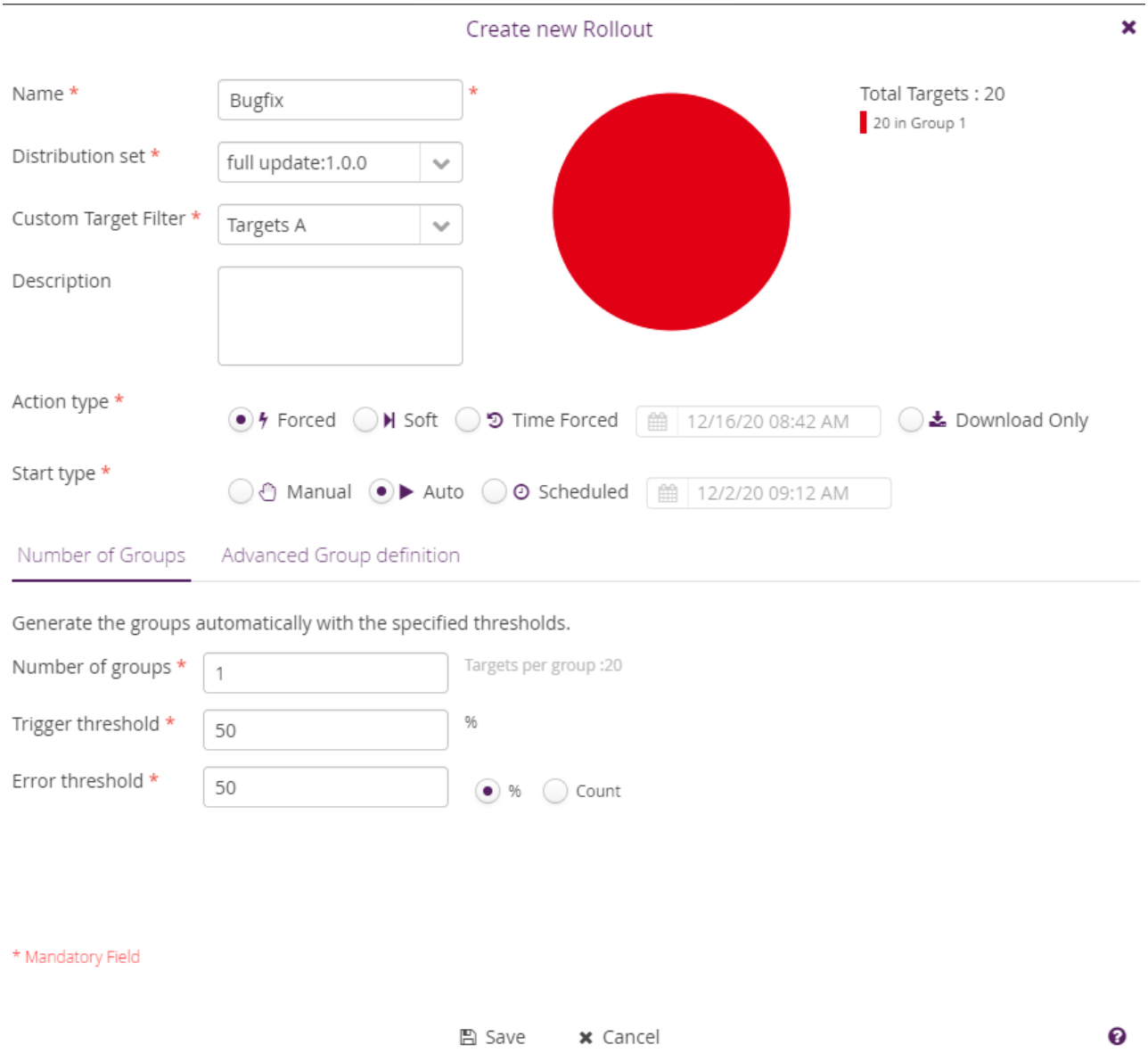


9. Rollout を作成してアップデートを開始する

   左側のメニューから"Rollout"をクリックして、Rollout Management の画面に移ります。



   "+"をクリックして Rollout を作成します。



項目	説明
Name	任意の文字列を設定します。
Distribution Set	先程作成した Distribution を選択します。
Custom Target Filter	先程作成した Target Filter を選択します。
Action Type	アップデート処理をどのように行うかを設定します。 ・Forced/Soft: 通常のアップデート ・Time Forced: 指定した時刻までにアップデートする ・Download only: ダウンロードのみ行う
Start Type	Rollout の実行をどのように始めるかを設定します。 ・Manual: 後で手動で開始する ・Auto: Target からのハートビートで開始する ・Scheduled: 決まった時間から開始する

10. アップデートの状態を確認する

    Rollout Management の画面の Detail Status で、各 Rollout のアップデートの状態を確認できます。

    アップデート中は黄色、アップデートが正常に完了すると緑色になります。

10.9.4.1. hawkBit のアップデート管理を CLI で行う

一つのアップデートを登録するには、hawkBit の Web UI で必要な手順が長いので CLI で行うことで効率よく実行できます。

サーバーの設定の段階では、「mkswu」のユーザーを作成する必要があります。作成していない場合は `setup_container.sh --add-user mkswu` で作成してください。

1. hawkbit_push_update の実行例

```
[ATDE ~/mkswu]$ ls enable_sshd.swu ❶
enable_sshd.swu

[ATDE ~/mkswu]$ hawkbit_push_update --help
Usage: /usr/bin/hawkbit_push_update [options] file.swu

rollout creation:
  --no-rollout: only upload the file without creating a rollout ❷
  --new: create new rollout even if there already is an existing one ❸
  --failed: Apply rollout only to nodes that previously failed update ❹

post action:
  --start: start rollout immediately after creation ❺

[ATDE ~/mkswu]$ hawkbit_push_update --start enable_sshd.swu ❻
Uploaded (or checked) image extra_os.sshd 1 successfully
Created rollout extra_os.sshd 1 successfully
Started extra_os.sshd 1 successfully
```

❶ この例ではあらかじめ作成されている enable_sshd.swu を hawkBit に登録します。

❷ --no-rollout を使う場合に SWU を「distribution」として登録します。デフォルトでは rollout も作成します。テストする際、デバイスがまだ登録されていなければ rollout の段階で失敗します。

❸ 同じ SWU で rollout を二回作成した場合にエラーが出ます。もう一度作成する場合は `--new` を使ってください。

❹ 一度 rollout をスタートして、 Armadillo で失敗した場合には失敗したデバイスだけに対応した rollout を作れます。

❺ 作成した rollout をすぐ実行します。このオプションには追加の権限を許可する必要があります。

❻ スタートまで行う実行例です。実行結果は Web UI で表示されます。

10.9.4.2. SWU で hawkBit を登録する

デバイスが多い場合は、SWU を一度作って armadillo を自己登録させることができます。

サーバーの設定の段階では、「device」のユーザーを作成する必要があります。作成していない場合は `setup_container.sh --add-user device` で作成してください。

1. hawkbit_register.desc で hawkBit の自己登録を行う例

```
[ATDE ~]$ cd mkswu/
```

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/hawkbit_register.* . ❶

[ATDE ~/mkswu]$ vi hawkbit_register.sh ❷
# Script configuration: edit this if required!
# user given here must have CREATE_TARGET,READ_TARGET_SECURITY_TOKEN permissions
HAWKBIT_USER=device
HAWKBIT_PASSWORD="CS=wC,zJmrQeeKT.3" ❸
HAWKBIT_URL=https://10.1.1.1 ❹
HAWKBIT_TENANT=default
# set custom options for suricatta block or in general in the config
CUSTOM_SWUPDATE_SURICATTA_CFG="" # e.g. "polldelay = 86400;"
CUSTOM_SWUPDATE_CFG=""
# set to non-empty if server certificate is invalid
SSL_NO_CHECK_CERT=
# or set to cafile that must have been updated first
SSL_CAFILE=
# ... or paste here base64 encoded crt content
SSL_CA_BASE64="
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJlakNDQVNHZ0F3SUJBZ0lVYTMvYXpNSHZ0
bFFnaFZnZDhIZWhMaEwxNm5Bd0NnWUlLb1pJemowRUF3SXcKRXpFUk1BOEdBMVVFQXd3SU1UQXVN
UzR4TGpFd0hoY05Nakl3TWpFNE1EVTFNakV6V2hjTk16SXdNakUyTURVMQpNakV6V2pBVE1SRXdE
d1lEVlFRRERBZ3hNQzR4TGpFdU1UQlpNQk1HQnlxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VICkEwSUFC
RFJGcnJVV3hHNnBHdWVoejRkRzVqYkVWTm5scHUwYXBHT1c3UVBPYUF4cWp1ZzJWYjk2UHNScWJY
Sk8KbEFDVVo2OStaMHk3clBqeDJHYnhDNms0czFHalV6QlJNQjBHQTFVZERnUVdCQlJtZzhxL2FV
OURRc3EvTGE1TgpaWFdkTHROUmNEQWZCZ05WSFNNRUdEQVdnQlJtZzhxL2FVOURRc3EvTGE1TlpY
V2RMdE5SY0RBUEJnTlZIUk1CCkFmOEVCVEFEQVFIL01Bb0dDQ3FHU000OUJBTUNBMGNBTUVRQ0lB
ZTRCQ0xKREpWZnFTQVdRcVBqNTFmMjJvQkYKRmVBbVlGY2VBMU45dE8rN0FpQXVvUEV1VGFxWjhH
UFYyRUg1UWdOMFRKS05SckJDOEtpNkZWcFlkRUowYWc9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0t
LS0tCg== ❺
"
# ... or add your own options if required
CURLOPT=-s

:（省略）

[ATDE ~/mkswu]$ cat hawkbit_register.desc ❻
:（省略）
swdesc_script hawkbit_register.sh --version extra_os.hawkbit 1

[ATDE ~/mkswu]$ mkswu hawkbit_register.desc ❼
hawkbit_register.swu を作成しました。

[ATDE ~/mkswu]$ mkswu initial_setup.desc hawkbit_register.desc ❽
hawkbit_register.desc を組み込みました。
initial_setup.swu を作成しました。
```

❶ hawkbit_register.sh と .desc ファイルをカレントディレクトリにコピーします。

❷ hawkbit_register.sh を編集して、設定を記載します。

❸ hawkBit の設定の時に入力した「device」ユーザーのパスワードを入力します。この例のパスワードは使用しないでください。

❹ hawkBit サーバーの URL を入力します。

❺ TLS を使用の場合に、コンテナ作成の時の証明書を base64 で入力します。

❻ hawkbit_register.desc の中身を確認します。hawkbit_register.sh を実行するだけです。

❼ SWU を作成して、initial_setup がすでにインストール済みの Armadillo にインストールできます。

❽ または、initial_setup.desc と合わせて hawkbit_register を含んだ initial_setup.swu を作成します。

10.9.5. mkswu の desc ファイル

.desc ファイルを編集すると、いくつかのコマンドが使えます。

例

```
[ATDE ~/mkswu]$ tree /usr/share/mkswu/examples/nginx_start
/usr/share/mkswu/examples/nginx_start
└── etc
    └── atmark
        └── containers
            └── nginx.conf

[ATDE ~/mkswu]$ cat /usr/share/mkswu/examples/usb_container_nginx.desc
swdesc_option version=1

swdesc_usb_container "nginx_alpine.tar" ❶
swdesc_files --extra-os "nginx_start" ❷
```

❶ nginx_alpine.tar ファイルに保存されたコンテナをインストールします。

❷ nginx_start ディレクトリの中身を転送します。

コマンドは書かれた順番でインストールされます。インストールするかどうかの判断はバージョンで行います:

```
swdesc_option component=<component>
swdesc_option version=<version>
か
swdesc_xxx --version <component> <version> [options]
```

- <component>は以下のどれかにしてください (デフォルトでは .desc ファイルのファイル名を使います)

  1. base_os: rootfs (Armadillo Base OS)を最初から書き込む時に使います。現在のファイルシステムは保存されていない。

     この場合、/etc/swupdate_preserve_files に載ってるファイルのみをコピーして新しい base OS を展開します。

     この component がないと現在の rootfs のすべてがコピーされます。

  2. extra_os.<文字列>: rootfs の変更を行う時に使います。<文字列> には任意の文字列を指定します。

     rootfs を変更を行う時に使います。 swdesc_* コマンドに --extra-os オプションを追加すると、 component に自動的に extra_os. を足します。

3. 〈文字列〉 (コンテナの名前などの任意の文字列) : rootfs の変更がないときに使います。

   この component を使うと rootfs の変更ができませんのでご注意ください。

・ アップデートを行う際にこのバージョンと現在のバージョンを比べてアップデートの判断を行います。

<component> がまだインストールされてなかった時や <version> が上がる時にインストールします。

デフォルトではダウングレードはできませんが、 --install-if=different オプションを追加することで <version> が変わる際にインストール可能になります。

アップデートの一部をインストールすることもありますので、複数の component で管理し、いくつかの古いバージョンに対応するアップデートも作成可能です。

以下のコマンドから使ってください

・ swdesc_tar と swdesc_files でファイルを転送します。

```
swdesc_tar [--dest <dest>] <tar_file>
swdesc_files [--dest <dest>] [--basedir <basedir>] ¥
             <file> [<more files>]
```

swdesc_tar の場合、予め用意されてある tar アーカイブをこのままデバイスで展開します。

--dest 〈dest〉 で展開先を選ぶことができます。デフォルトは / （--extra-os を含め、バージョンの component は base_os か extra_os.* の場合）か /var/app/rollback/volumes/ （それ以外の component)。後者の場合は /var/app/volumes と /var/app/rollback/volumes 以外は書けないので必要な場合に --extra-os を使ってください。

swdesc_files の場合、mkswu がアーカイブを作ってくれますが同じ仕組みです。

--basedir 〈basedir〉 でアーカイブ内のパスをどこで切るかを決めます。

・ 例えば、swdesc_files --extra-os --basedir /dir /dir/subdir/file ではデバイスに /subdir/file を作成します。

・ デフォルトは <file> から設定されます。ディレクトリであればそのまま basedir として使います。それ以外であれば親ディレクトリを使います。

・ swdesc_command や swdesc_script でコマンドを実行する

```
swdesc_command <command> [<more commands>]
swdesc_script <script>
```

アップデート先の環境でコマンドやスクリプトファイルを走らせます。

バージョンの component は base_os と extra_os 以外の場合、 /var/app/volumes と /var/app/rollback/volumes 以外は変更できないのでご注意ください。

コマンドの実行が失敗した場合、アップデートも失敗します。

・ swdesc_exec でファイルを配ってコマンドでそのファイルを使う

```
swdesc_exec <file> <command>
```

swdesc_command と同じくコマンドを走らせますが、<file> を先に転送してコマンド内で"$1"として使えます。

- swdesc_command_nochroot, swdesc_script_nochroot, swdesc_exec_nochroot で起動中のシステム上でコマンドを実行します。

  このコマンドは nochroot なしのバージョンと同じ使い方で、現在起動中のシステムに変更や確認が必要な場合にのみ使用してください。

  nochroot コマンドは確認を一切しないため、Armadillo が起動できない状態になる可能性もあります。充分にご注意ください。

  例が必要な場合は /usr/share/mkswu/examples/firmware_update.desc を参考にしてください。

- swdesc_embed_container, swdesc_usb_container, swdesc_pull_container で予め作成したコンテナを転送します。

```
swdesc_embed_container <container_archive>
swdesc_usb_container <container_archive>
swdesc_pull_container <container_url>
```

  例は「10.5.5. コンテナの配布」を参考にしてください。

- swdesc_boot で u-boot を更新します。

```
swdesc_boot <boot image>
```

  このコマンドだけにバージョンは自動的に設定されます。

コマンドの他には、設定変数もあります。以下の設定は /home/atmark/mkswu/mkswu.conf に設定できます。

- DESCRIPTION="<text>": イメージの説明、ログに残ります。
- PRIVKEY=<path>, PUBKEY=<path>: 署名鍵と証明書
- PRIVKEY_PASS=<val>: 鍵のパスワード（自動用）

  openssl の Pass Phrase をそのまま使いますので、pass:password, env:var や file:pathname のどれかを使えます。pass や env の場合他のプロセスに見られる恐れがありますので file をおすすめします。

- ENCRYPT_KEYFILE=<path>: 暗号化の鍵

以下のオプションも mkswu.conf に設定できますが、.desc ファイルにも設定可能です。swdesc_option で指定することで、誤った使い方した場合 mkswu の段階でエラーを出力しますので、必要な場合は使用してください。

- swdesc_option CONTAINER_CLEAR: インストールされたあるコンテナと /etc/atmark/containers/*.conf をすべて削除します。

  このオプションは簡単な初期化と考えてください。通常の運用では、不要になったイメージは自動的に削除されますのでこのオプションを設定する必要はありません。

以下のオプションは Armadillo 上の /etc/atmark/baseos.conf に、例えば MKSWU_POST_ACTION=xxx として設定することができます。

その場合に swu に設定されなければ /etc の設定で実行されますので、アットマークテクノが用意している Base OS のアップデートでも動作の変更は可能です。swu に特定のオプションが設定された場合は設定されたオプションが優先されますので、一時的な変更も可能です。

- swdesc_option POST_ACTION=container: コンテナのみのアップデート後に再起動を行いません。コンテナの中身だけをアップデートする場合、Armadillo-IoT ゲートウェイ A6E を再起動せずにコンテナだけを再起動させます。
- swdesc_option POST_ACTION=poweroff: アップデート後にシャットダウンを行います。
- swdesc_option POST_ACTION=wait: アップデート後に自動的に再起動は行われず、次回起動時にアップデートが適用されます。
- swdesc_option POST_ACTION=reboot: デフォルトの状態に戻します。アップデートの後に再起動します。
- swdesc_option NOTIFY_STARTING_CMD="command", swdesc_option NOTIFY_SUCCESS_CMD="command", swdesc_option NOTIFY_FAIL_CMD="command": アップデートをインストール中、成功した場合と失敗した場合に実行されるコマンドです。

  コマンドを実行する事で、アプリケーションやユーザーにアップデートを知らせることができます。

  LED で知らせる例を /usr/share/mkswu/examples/enable_notify_led.desc に用意してあります。

10.9.5.1. 例: sshd を有効にする

/usr/share/mkswu/examples/enable_sshd.desc を参考にします。

desc ファイルを編集する必要がありませんが自分の公開鍵を指定された場所に配置してください。

```
[ATDE ~/mkswu]$ cp -r /usr/share/mkswu/examples/enable_sshd* .
[ATDE ~/mkswu]$ cat enable_sshd.desc
swdesc_option component=extra_os.sshd version=1

# add your public key in enable_sshd/root/.ssh/authorized_keys
if [ -z "$SWDESC_TEST" ]; then
    grep -qE '^ssh-' enable_sshd/root/.ssh/authorized_keys ¥
        || error "Add your keys in enable_sshd/root/.ssh/authorized_keys"
fi
swdesc_files --dest /root enable_sshd/root ❶

swdesc_command "ssh-keygen -A" ¥ ❷
```

```
        "rc-update add sshd" ❸
[ATDE ~/mkswu]$ cp ~/.ssh/id_rsa.pub ¥
                        enable_sshd/root/.ssh/authorized_keys ❹
[ATDE ~/mkswu]$ mkswu enable_sshd.desc ❺
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
enable_sshd.swu を作成しました。
```

❶ 自分の公開鍵を転送します。デフォルトのオプションなので enable_sshd/root ディレクトリの中身をこのまま /root に転送されます。

❷ 再起動する度に新しいサーバーの鍵が変わらないように、アップデートの時に一回作成します。

❸ サービスを有効にします。

❹ 自分の公開鍵を指定された場所に配置します。

❺ イメージを作成します。パスワードは証明鍵のパスワードです。

10.9.5.2. 例: Armadillo Base OS アップデート

ここでは、「10.6. Armadillo のソフトウェアをビルドする」 でメインシステム向けのビルドで作成したファイルを使用します。

/usr/share/mkswu/examples/OS_update.desc を参考にします。

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/OS_update.desc update-[VERSION].desc
[ATDE ~/mkswu]$ vi update-[VERSION].desc
# uboot image can be generated with atmark imx-boot script
swdesc_uboot imx-boot_armadillo_x2 ❶

# base OS is a tar that will be extracted on a blank filesystem,
# after copying just a few key config files.
#
# OS updates are only installed if version is greater than previous update
# so if you install your own updates atmark-techno provided Armadillo Base OS
# updates might not get installed
swdesc_tar "baseos-6e-[VERSION].tar.zst" ¥ ❷
           --version base_os [VERSION] ❸
[ATDE ~/mkswu]$ mkswu update-[VERSION].desc ❹
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
update-[VERSION].swu を作成しました。
```

❶ 「10.6.1. ブートローダーをビルドする」でビルドしたイメージを使います。

❷ build-rootfs でビルドしたイメージを使います。

❸ バージョンが上がるときにしかインストールされませんので、現在の/etc/sw-versions を確認して適切に設定してください。

❹ イメージを作成します。パスワードは証明鍵の時のパスワードです。

10.9.5.3. 例: swupdate_preserve_files で Linux カーネル以外の Armadillo-IoT ゲートウェイ A6E 向けのイメージをインストールする方法

Armadillo-IoT ゲートウェイ A6E 向けのアップデートイメージに Linux カーネルが含まれています。

swupdate_preserve_files を使って、以下のコマンドでインストール後に現在のカーネルをコピーして更新させないようにします。

```
[armadillo ~]# echo 'POST /boot' >> /etc/swupdate_preserve_files
[armadillo ~]# echo 'POST /lib/modules' >> /etc/swupdate_preserve_files ❶
[armadillo ~]# persist_file /etc/swupdate_preserve_files ❷
```

❶ swupdate_preserve_files に /boot と /lib/modules を保存するように追加します。

❷ 変更した設定ファイルを保存します

/usr/share/mkswu/examples/kernel_update*.desc のように update_preserve_files.sh のヘルパーで、パスを自動的に /etc/swupdate_preserve_files に追加することができます。

```
[ATDE ~/mkswu]$ cp /usr/share/mkswu/examples/update_preserve_files.sh .
[ATDE ~/mkswu]$ cat example.desc
swdesc_script update_preserve_files.sh -- ¥
        "POST /boot" ¥
        "POST /lib/modules"
```

10.9.6. swupdate_preserve_files について

extra_os のアップデートで rootfs にファイルを配置することができますが、次の OS アップデートの際に削除される可能性があります。

デフォルトでは、 /etc/atmark と、 swupdate 、 sshd やネットワークの設定を保存しますがそれ以外はコピーされてません。

そうでないファイルを更新する際には /etc/swupdate_preserve_files に記載します。「10.9.5.3. 例: swupdate_preserve_files で Linux カーネル以外の Armadillo-IoT ゲートウェイ A6E 向けのイメージをインストールする方法」 を参考にしてください。

コピーのタイミングによって、以下のどれかを使用してください：

1. 単にファイルを記載する

   この場合、アップデートする前にファイルをコピーします。 baseos のイメージと同じ swu にアップデートしたいファイルを記載していても、このファイルが Armadillo Base OS に含まれないのであれば問題なくアップデートできます。

   例: echo "/root/.profile" >> /etc/swupdate_preserve_files

2. POST のキーワードの後に記載する

   この場合、アップデートの最後でコピーします。 Armadillo Base OS に含まれてるファイルであれば、インストール前にコピーしても保存されないのでコピーのタイミングをずらします。

   そのコピーが最後に行われるので、同じアップデートでファイルの変更ができません。アップデートを別けて、 baseos のイメージをインストールしてからこのファイルを更新することができます。

例: echo ”POST /etc/conf.d/podman-atmark” >> /etc/swupdate_preserve_files

10.9.7. SWU イメージの内容の確認

mkswu --show [file.swu] で SWU イメージの内容を確認することができます。

出力は desc ファイルに似ていますが、そのまま desc ファイルとして利用できませんので確認用としてお使いください。

```
[ATDE ~/mkswu]$ mkswu --show enable_sshd.swu
enable_sshd.swu

# built with mkswu 4.1

swdesc_files --dest /root enable_sshd/root
  --version extra_os.sshd 1
  (encrypted)

swdesc_command ssh-keygen -A && rc-update add sshd default
  --version extra_os.sshd 1
```

10.9.8. SWUpdate と暗号化について

mkswu --init の時に暗号化を有効にする場合は AES でファイルを暗号化します。

現在使われてる SWUpdate の暗号化はコマンドやメタデータを含む sw-description ファイルは暗号化されてません。そのため、通信の暗号化（HTTPS で送信するなど）を使うことを推奨します。

10.10. Armadillo Base OS の操作

Armadillo Base OS は Alpine OS をベースとして作られています。

このセクションでは Armadillo Base OS の機能を紹介します。

10.10.1. アップデート

Armadillo Base OS は SWUpdate によってアップデートすることができます。

アップデートする際には、rootfs ファイルシステムにインストールされたファイルをすべて消して、アップデートの中身と /etc/swupdate_preserve_files に記載されているファイルで新しい rootfs を作ります。

アップデートでファイルを削除してしまった場合に abos-ctrl mount-old で前のシステムを read-only でマウントして、削除されたファイルをコピーすることもできます。

10.10.2. overlayfs と persist_file について

Armadillo BaseOS ではルートファイルシステムに overlayfs を採用しています。

そのため、ファイルを変更した後 Armadillo の電源を切ると変更内容は保持されません。開発中などに rootfs の変更内容を保持するには、変更したファイルに対して persist_file コマンドを使用します。

開発以外の時は安全のため、ソフトウェアアップデートによる更新を実行してください。アップデート手順に関しては 「10.9. Armadillo のソフトウェアをアップデートする」 を参照してください。

rootfs の内容を変更しても、ソフトウェアアップデートを実施した際に変更した内容が保持されない可能性があります。

persist_file コマンドの概要を 「図 10.122. persist_file のヘルプ」 に示します。

```
[armadillo ~]# persist_file -h
Usage: /usr/bin/persist_file [options] file [more files...]

Mode selection:
  (none) single entry copy
  -d, --delete   delete file
  -l, --list     list content of overlay
  -a, --apk      apk mode: pass any argument after that to apk on rootfs
  -R, --revert   revert change: only delete from overlay, making it
                 look like the file was reverted back to original state

Copy options:
  -r, --recurse  recursive copy (note this also removes files!)
  -p, --preserve make the copy persist through baseos upgrade
                 by adding entries to /etc/swupdate_preserve_files
  -P, --preserve-post   same, but copy after upgrade (POST)

Delete options:
  -r, --recurse  recursively delete files

Common options:
  -v, --verbose  verbose mode for all underlying commands

Note this directly manipulates overlayfs lower directories
so might need a reboot to take effect
```

図 10.122 persist_file のヘルプ

1. ファイルの保存・削除手順例

```
[armadillo ~]# echo test > test
[armadillo ~]# persist_file -rv /root
'/root/test' -> '/mnt/root/test' ❶
'/root/.ash_history' -> '/mnt/root/.ash_history'
[armadillo ~]# rm -f test
[armadillo ~]# persist_file -rv /root
removed '/mnt/root/test' ❷
removed '/mnt/root/.ash_history' ❸
'/root/.ash_history' -> '/mnt/root/.ash_history'
```

図 10.123 persist_file 保存・削除手順例

❶ 追加・変更したファイルを rootfs へコピーします。

❷ -r を指定すると、ひとつ前の rm -f コマンドで削除したファイルが rootfs からも削除されますのでご注意ください。

❸ すでに rootfs に存在するファイルも一度削除してからコピーするため、このようなメッセージが表示されます。

2. ソフトウェアアップデート後も変更を維持する手順例

```
[armadillo ~]# vi /etc/conf.d/podman-atmark ❶
[armadillo ~]# persist_file -P /etc/conf.d/podman-atmark ❷
[armadillo ~]# tail -n 2 /etc/swupdate_preserve_files ❸
# persist_file 20211216
POST /etc/conf.d/podman-atmark
```

図 10.124 persist_file ソフトウェアアップデート後も変更を維持する手順例

❶ 何らかのファイルの内容を変更します。

❷ -P オプションを付与して persist_file を実行します。

❸ swupdate_preserve_files に追加されたことを確認します。

3. 変更ファイルの一覧表示例

```
[armadillo ~]# mkdir dir
[armadillo ~]# persist_file -l
directory          /
directory          /root
opaque directory   /root/dir ❶
whiteout           /root/test ❷
regular file       /root/.ash_history
directory          /etc
regular file       /etc/resolv.conf
directory          /var
symbolic link      /var/lock
: (省略)
```

図 10.125 persist_file 変更ファイルの一覧表示例

❶ rootfs のファイルを見せないディレクトリは opaque directory と表示されます。

❷ 削除したファイルは whiteout と表示されます。

4. パッケージをインストールする時は apk コマンドを使用してメモリ上にインストールできますが、persist_file コマンドで rootfs に直接インストールすることも可能です。

```
[armadillo ~]# persist_file -a add strace
(1/3) Installing fts (1.2.7-r1)
(2/3) Installing libelf (0.185-r0)
(3/3) Installing strace (5.14-r0)
Executing busybox-1.34.1-r3.trigger
OK: 251 MiB in 188 packages
Install succeeded, but might not work in the running system
Please reboot if installed program does not work ❶
[armadillo ~]# strace ls
: (省略)
```

```
exit_group(0)                           = ?
+++ exited with 0 +++
```

図 10.126 persist_file でのパッケージインストール手順例

❶ この例では Armadillo を再起動せずにインストールしたコマンドを使用できましたが、Armadillo の再起動が必要となるパッケージもありますので、その場合は Armadillo を再起動してください。

10.10.3. ロールバック状態の確認

Armadillo Base OS の ルートファイルシステムが壊れて起動できなくなった場合に自動的に前のバージョンで再起動します。

自分で確認する必要がある場合に abos-ctrl status でロールバックされてるかどうかの確認ができます。

必要な場合（例えば、自分のアプリケーションがアップデート直後に問題があった場合)、 abos-ctrl rollback で手動のロールバックも可能です。ロールバックにエラーがなければ、再起動してロールバックを完了します。

なお、/var/at-log/atlog に切り替えの際に必ずログを書きますので、調査の時に使ってください。

```
[armadillo ~]# cat /var/at-log/atlog
Mar 17 14:51:35 armadillo NOTICE swupdate: Installed update to /dev/mmcblk0p2: ¥
extra_os.sshd: unset -> 1, extra_os.initial_setup: unset -> 1
Mar 17 16:48:52 armadillo NOTICE swupdate: Installed update to /dev/mmcblk0p1: ¥
boot: 2020.04-at5 -> 2020.04-at6, base_os: 3.15.0-at.3 -> 3.15.0-at.4
Mar 17 17:42:15 armadillo NOTICE swupdate: Installed update to /dev/mmcblk0p2: ¥
other_boot: 2020.04-at5 -> 2020.04-at6, container: unset -> 1, extra_os.container: unset -> 1
```

図 10.127 /var/at-log/atlog の内容の例

10.10.4. ボタンやキーを扱う

buttond サービスを使用することで、ボタンやキー入力をトリガーとする処理を簡単に実装できます。

/etc/atmark/buttond.conf に BUTTOND_ARGS を指定することで、動作を指定することができます:

- -s <key> -a ”command”: 短押しの設定。キーを 1 秒以内に離せば短押しと認識し "command"を実行します。認識する時間は -t <time_ms> オプションで変更可能です。
- -l <key> -s ”command”: 長押しの設定。キーを 5 秒押し続けたタイミングで "command" を実行します。長押しと認識する時間は -t <time_ms> オプションで変更可能です。
- 1 つのキーに対して複数の設定が可能です。長押しの設定が複数ある場合、押したままの状態だと一番長い時間に設定されている "command" を実行します。途中でキーを離した場合は、キーを離した時間に応じた "command" を実行します。(例：buttond -s <key> -a "cmd1" -l <key> -t 2000 -a "cmd2" -l <key> -t 10000 -a "cmd3" <file> を実行した場合、1 秒以内に離すと "cmd1"、2 秒以上 10 秒以内に離すと "cmd2"、10 秒を越えたら "cmd3" を実行します)。
  - 短押し設定を複数指定する場合、時間の短い設定を先に指定してください。 0.5 秒、1 秒を設定したい場合、1 秒 → 0.5 秒の順番で指定すると 0.5 秒が無視されます。

以下にデフォルトを維持したままで SW1 の短押しと長押しにそれぞれの場合にコマンドを実行させます。

```
[armadillo ~]# vi /etc/atmark/buttond.conf ❶
BUTTOND_ARGS="$BUTTOND_ARGS -s prog1 -a 'date >> /tmp/shortpress'"
BUTTOND_ARGS="$BUTTOND_ARGS -l prog1 -t 5000 -a 'date >> /tmp/longpress'"
[armadillo ~]# persist_file /etc/atmark/buttond.conf ❷
[armadillo ~]# rc-service buttond restart ❸
buttond          | * Stopping button watching daemon ...                                [ ok ]
buttond          | * Starting button watching daemon ...                                [ ok ]
[armadillo ~]# cat /tmp/shortpress ❹
Tue Mar 22 17:16:42 JST 2022
Tue Mar 22 17:16:43 JST 2022
[armadillo ~]# cat /tmp/longpress
Tue Mar 22 17:16:48 JST 2022
```

図 10.128 buttond で SW1 を扱う

❶ buttond の設定ファイルを編集します。この例では、短押しの場合 /tmp/shotpress に、5 秒以上の長押しの場合 /tmp/longpress に日付を出力します。

❷ 設定ファイルを保存します。

❸ buttond サービスを再起動させます。ここでは再起動後短押しを 2 回、長押しを 1 回行ったとします。

❹ 押された回数を確認します。

USB キーボードや他の入力デバイスにも対応できます。

1. デバイスを接続してから、 buttond でデバイス名とキーコードを確認します。

```
[armadillo ~]# buttond -vvv /dev/input/* /dev/input/by-*/* ❶
Skipping directory /dev/input/by-id
Skipping directory /dev/input/by-path
[78972.042] /dev/input/event2 4 4 458976: non-keyboard event ignored
[78972.042] /dev/input/event2 LEFTCTRL (29) pressed: ignored ❷
[78972.042] /dev/input/by-id/usb-0566_3029-event-kbd 4 4 458976: non-keyboard event ignored
[78972.042] /dev/input/by-id/usb-0566_3029-event-kbd LEFTCTRL (29) pressed: ignored
[78972.042] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd 4 4 458976: ⏎
non-keyboard event ignored
[78972.042] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd LEFTCTRL ⏎
(29) pressed: ignored
[78972.130] /dev/input/event2 4 4 458976: non-keyboard event ignored
[78972.130] /dev/input/event2 LEFTCTRL (29) released: ignored
[78972.130] /dev/input/by-id/usb-0566_3029-event-kbd 4 4 458976: non-keyboard event ignored
[78972.130] /dev/input/by-id/usb-0566_3029-event-kbd LEFTCTRL (29) released: ignored
[78972.130] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd 4 4 458976: ⏎
non-keyboard event ignored
[78972.130] /dev/input/by-path/platform-xhci-hcd.1.auto-usb-0:1:1.0-event-kbd LEFTCTRL ⏎
(29) released: ignored
```

❶ buttond を -vvv で冗長出力にして、すべてのデバイスを指定します。

❷ 希望のキーを押すと、LEFTCTRL が三つのパスで認識されました。 一番安定する by-id のパスを控えておきます。

2. USB デバイスを外すこともありますので、-i (inotify) で管理されてる入力デバイスとして追加します。そうしないとデバイスを外したときに buttond が停止します。

```
[armadillo ~]# vi /etc/atmark/buttond.conf
BUTTOND_ARGS="$BUTTOND_ARGS -i /dev/input/by-id/usb-0566_3029-event-kbd"
BUTTOND_ARGS="$BUTTOND_ARGS -s LEFTCTRL -a 'podman_start button_pressed_container'"
[armadillo ~]# persist_file /etc/atmark/buttond.conf
[armadillo ~]# rc-service buttond restart
```

10.10.5. Armadillo Base OS 側の起動スクリプト

起動時に何かスクリプトを走らせるためにはコンテナとして実行することを推奨します。 「10.5.1. コンテナの自動起動」 を参照してください。

コンテナで実行不可能な場合に、「local」サービスを使うことができます: /etc/local.d ディレクトリに.start ファイルを置いておくと起動時に実行されて、 .stop ファイルは終了時に実行されます。

```
[armadillo ~]# vi /etc/local.d/date_test.start ❶
#!/bin/sh

date > /tmp/boottest
[armadillo ~]# chmod +x /etc/local.d/date_test.start ❷
[armadillo ~]# persist_file /etc/local.d/date_test.start ❸
[armadillo ~]# reboot
: (省略)
[armadillo ~]# cat /tmp/boottest ❹
Tue Mar 22 16:36:12 JST 2022
```

図 10.129 local サービスの実行例

❶ スクリプトを作ります。

❷ スクリプトを実行可能にします。

❸ スクリプトを保存して、再起動します。

❹ 実行されたことを確認します。

10.10.6. u-boot の環境変数の設定

u-boot の環境変数を変更するには /boot/uboot_env.d/ ディレクトリに環境変数が書かれた設定ファイルを配置します。

ファイルの構文は fw_setenv が扱うことができるもので、以下のとおりです：

- # で始まる行はコメントと扱われる為、無視されます。また、 環境変数への代入を示す = がない場合も無視されます。
- [変数]=[値] で変数を設定します。スペースや引用符を含め他の文字は有効ですので、変数の名前と値に不要な文字を入れないように注意してください。

・[変数]= で変数を消します。値がない場合に変数が消去されます。

このファイルによるアップデート内容は swupdate でアップデートする際に適用されます。

実行中のシステムに影響がありませんので、設定ファイルを swupdate で転送しない場合はファイル永続化後に fw_setenv -s /boot/uboot_env.d/[ファイル名] で変数を書き込んでください。

swupdate でファイルを転送した場合には、変数はすぐに利用されます。

```
[armadillo ~]# vi /boot/uboot_env.d/no_prompt ❶
# bootdelay を -2 に設定することで u-boot のプロンプトを無効化します
bootdelay=-2
[armadillo ~]# persist_file -v /boot/uboot_env.d/no_prompt ❷
'/boot/uboot_env.d/no_prompt' -> '/mnt/boot/uboot_env.d/no_prompt'
[armadillo ~]# fw_setenv -s /boot/uboot_env.d/no_prompt ❸
Environment OK, copy 0
[armadillo ~]# fw_printenv | grep bootdelay ❹
bootdelay=-2
```

図 10.130 uboot_env.d のコンフィグファイルの例

❶ コンフィグファイルを生成します。

❷ ファイルを永続化します。

❸ 変数を書き込みます。

❹ 書き込んだ変数を確認します。

mkswu バージョン 4.4 以降が必要です。必要な場合はアップデートしてください。

```
[ATDE ~]$ sudo apt update && sudo apt upgrade
```

書き方は、 /usr/share/mkswu/examples/uboot_env.desc を参考にしてください。

「10.6.1. ブートローダーをビルドする」 の際に u-boot のデフォルトを変更した場合や、u-boot のプロンプトで「setenv」や「saveenv」を実行しても、 /boot/uboot_env.d/00_defaults によって変更がアップデートの際にリセットされます。

00_defaults のファイルは Base OS の一部で更新されることもありますので、変更を望む場合は別のファイルを作って設定してください。ファイルはアルファベット順で処理されます。 00_defaults にある変数を後のファイルにも設定した場合はそのファイルの値だけが残ります。

10.10.7. Network Time Protocol (NTP, ネットワーク・タイム・プロトコル)

Armadillo Base OS では chronyd を使っています。

デフォルトの設定（使用するサーバーなど）は /etc/chrony/conf.d/ にあり、変更用に /etc/atmark/chrony.conf.d/ のファイルも読み込みます。/etc/atmark/chrony.conf.d ディレクトリに /etc/chrony/conf.d/ と同じファイル名の設定ファイルを置いておくことで、デフォルトのファイルを読まないようになります。

例えば、 NTP サーバーの設定は servers.conf に記載されてますので、変更する際はに/etc/atmark/chrony.conf.d/servers.conf のファイルに記載します：

```
[armadillo ~]# vi /etc/atmark/chrony.conf.d/servers.conf ❶
pool my.ntp.server iburst
[armadillo ~]# persist_file /etc/atmark/chrony.conf.d/servers.conf ❷
[armadillo ~]# rc-service chronyd restart ❸
chronyd          | * Stopping chronyd ... [ ok ]
chronyd          | * Starting chronyd ... [ ok ]
armadillo:~# chronyc sources ❹
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? my.ntp.server                 1   6     3     2    +88ms[  +88ms] +/-  173ms
```

図 10.131 chronyd のコンフィグの変更例

❶ コンフィグファイルを作ります。

❷ ファイルを保存します

❸ chronyd サービスを再起動します。

❹ chronyc で新しいサーバーが使用されていることを確認します。

10.11. Device Tree をカスタマイズする

at-dtweb を利用して Device Tree をカスタマイズする方法を説明します。at-dtweb では、Web ブラウザ上のマウス操作で dtbo ファイルおよび desc ファイルを生成することができます。カスタマイズの対象は拡張インターフェース(CON8)です。== at-dtweb のインストール

ATDE9 に at-dtweb パッケージをインストールします。

```
[ATDE ~]$ sudo apt update
[ATDE ~]$ sudo apt install at-dtweb
```

インストール済みの場合は、以下のコマンドを実行し最新版への更新を行ってください。

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

10.11.1. at-dtweb の起動

1. at-dtweb の起動開始

   at-dtweb の起動を開始するには、デスクトップ左上のアプリケーションの「システムツール」から「at-dtweb」を選択してください。

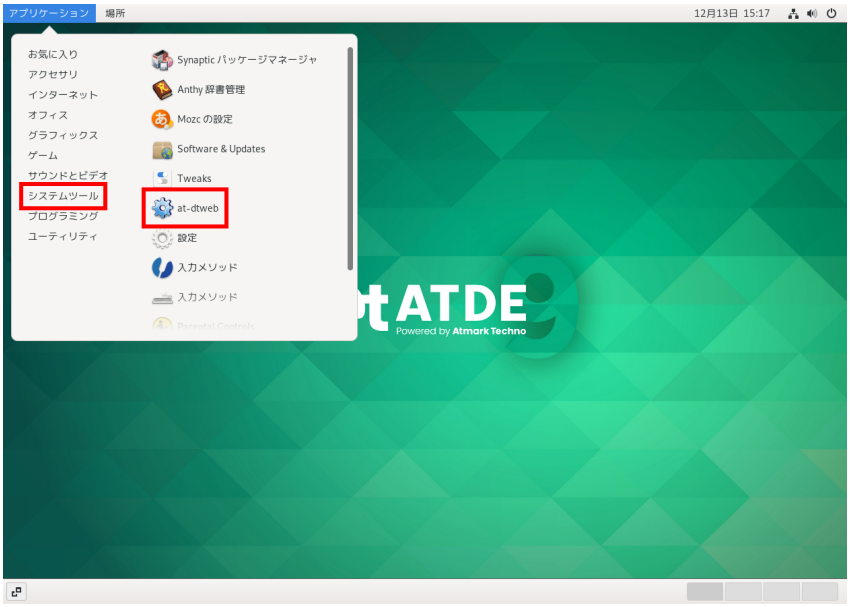


図 10.132 at-dtweb の起動開始

コマンドライン上からでも、at-dtweb コマンドで起動できます。

```
[ATDE ~]$ at-dtweb
```

1. ボードの選択

   ボードを選択します。Armadillo-IoT_A6E を選択して、「OK」をクリックします。

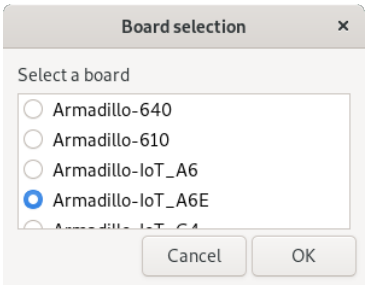


図 10.133 ボード選択画面

2. Linux カーネルディレクトリの選択

   Linux カーネルディレクトリを選択します。コンフィギュレーション済みの Linux カーネルディレクトリを選択して、「OK」をクリックします。

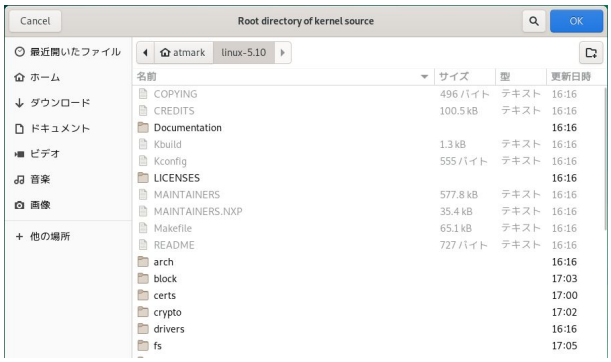


図 10.134 Linux カーネルディレクトリ選択画面

3. at-dtweb の起動完了

   at-dtweb が起動し、次のように画面が表示されます。

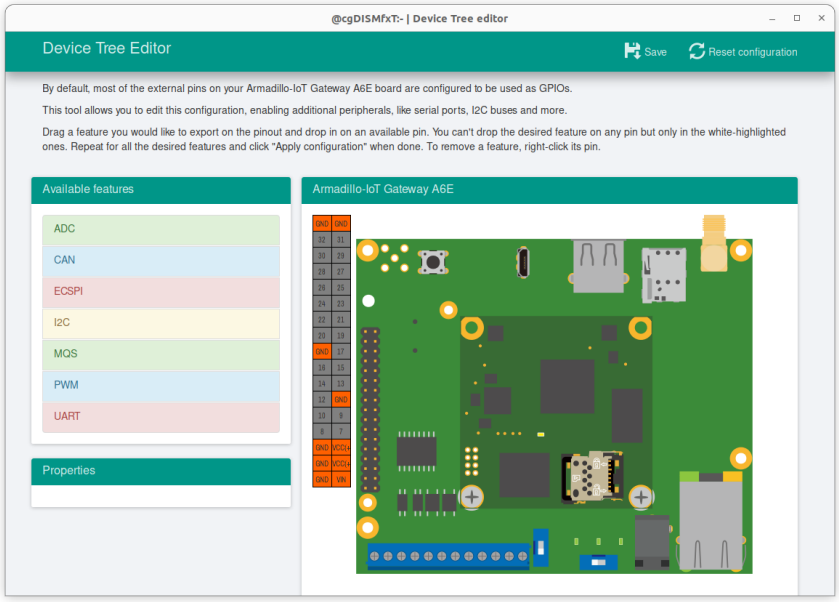


図 10.135 at-dtweb 起動画面

Linux カーネルは、事前にコンフィギュレーションされている必要があります。コンフィギュレーションの手順については「10.6. Armadillo のソフトウェアをビルドする」を参照してください。

10.11.2. Device Tree をカスタマイズ

10.11.2.1. 機能の選択

機能の選択は、ドラッグ&ドロップで行います。画面左上の「Available features」から有効にしたい機能をドラッグし、画面右側の「Armadillo-IoT Gateway A6E」の白色に変化したピンにドロップします。例として CON 8 8/9 ピンを UART 1 (RXD/TXD) に設定します。

何も機能が選択されていないピンには GPIO の機能が割り当てられます。

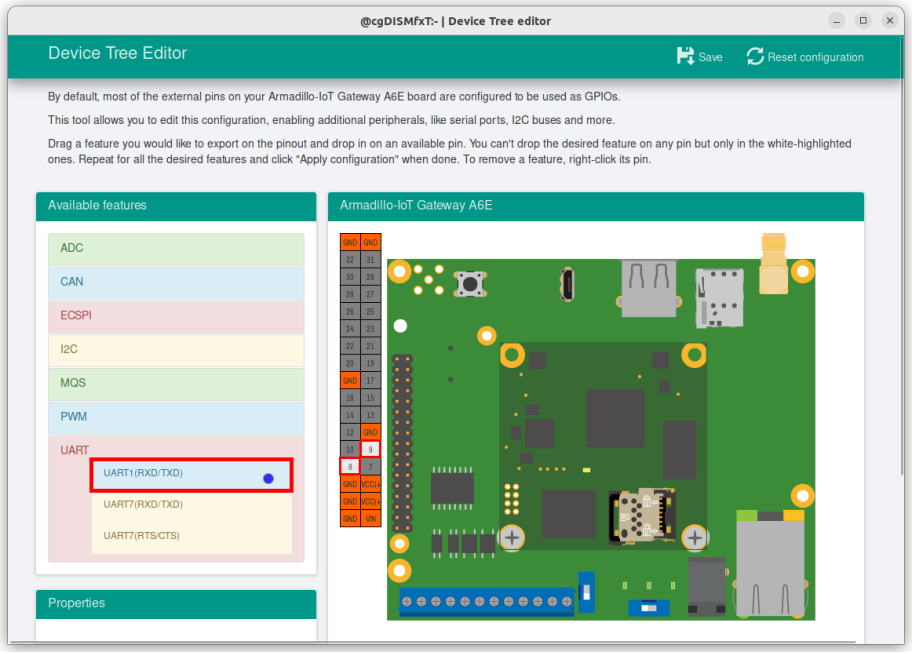


図 10.136 UART1(RXD/TXD) のドラッグ

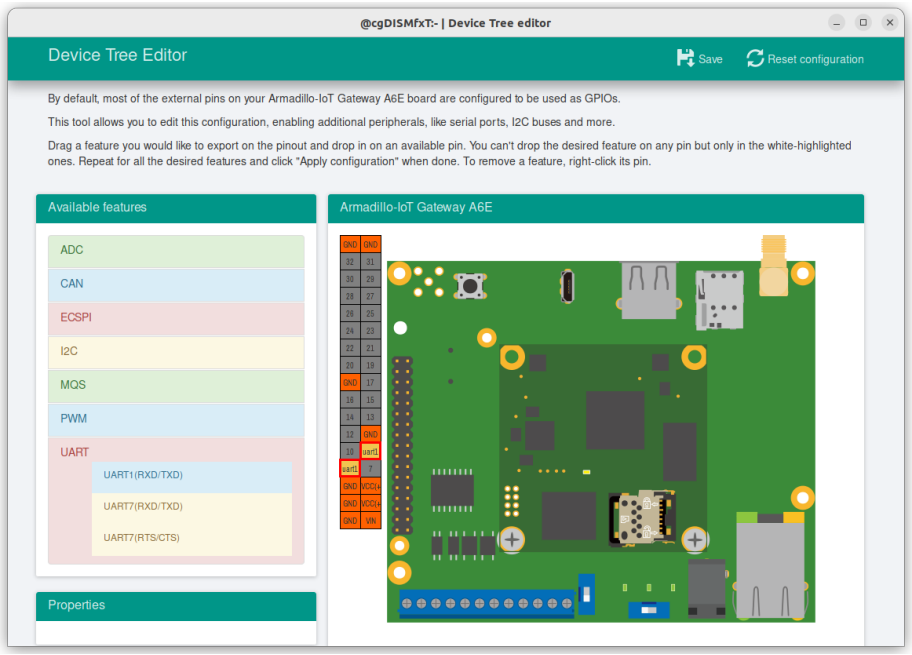


図 10.137 CON8 8/9 ピンへのドロップ

10.11.2.2. 信号名の確認

画面右側の「Armadillo-IoT Gateway A6E」にドロップして設定したピンを左クリックすると信号名が表示されます。どのピンがどの信号に対応しているのかを確認することができます。

例として UART1(RXD/TXD) の信号名を確認します。

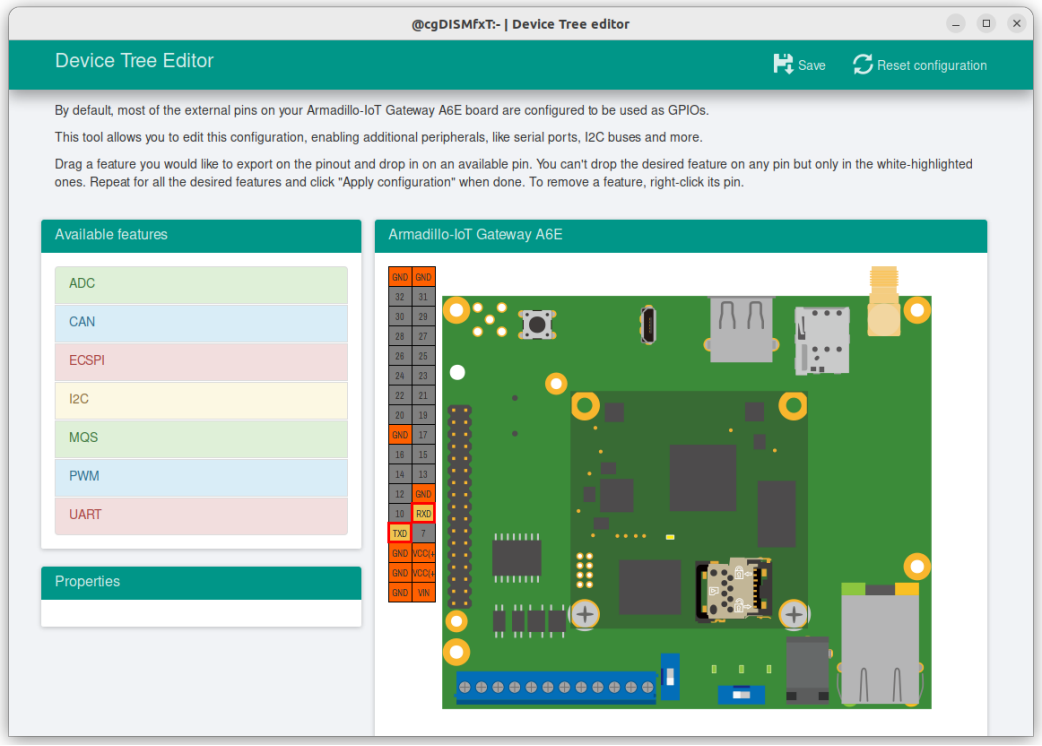


図 10.138 信号名の確認

再度ピンを左クリックすると機能名の表示に戻ります。

10.11.2.3. プロパティの設定

いくつかの機能にプロパティを設定することができます。画面右側の「Armadillo-IoT Gateway A6E」に選択した機能を左クリックすると、画面左下の「Properties」からプロパティを選択することができます。

例として CON8 12/13 ピンの I2C4(SCL/SDA) の clock_frequency プロパティを設定します。

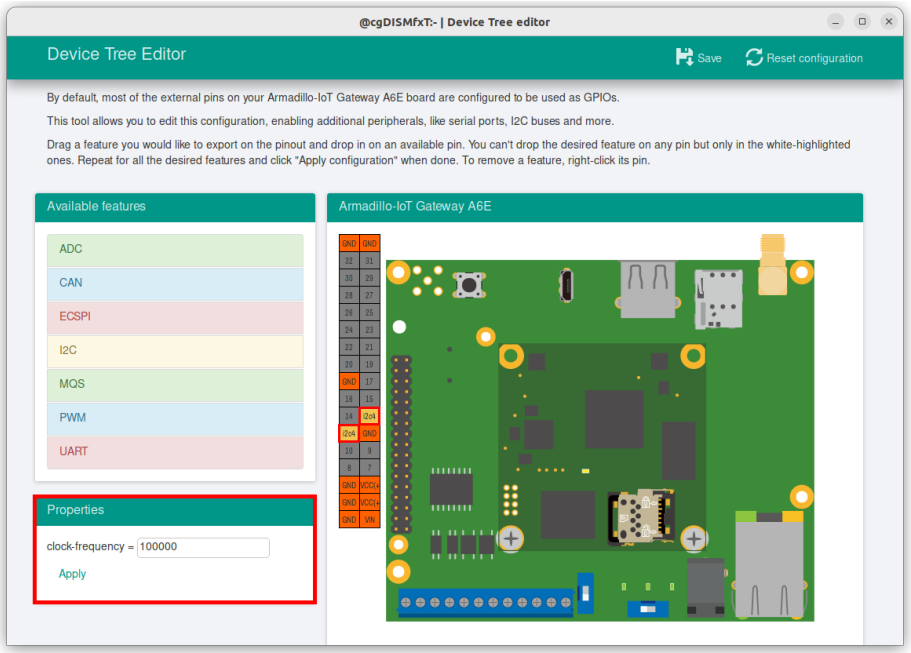


図 10.139 プロパティの設定

設定したプロパティを確定させるには「Apply」をクリックします。

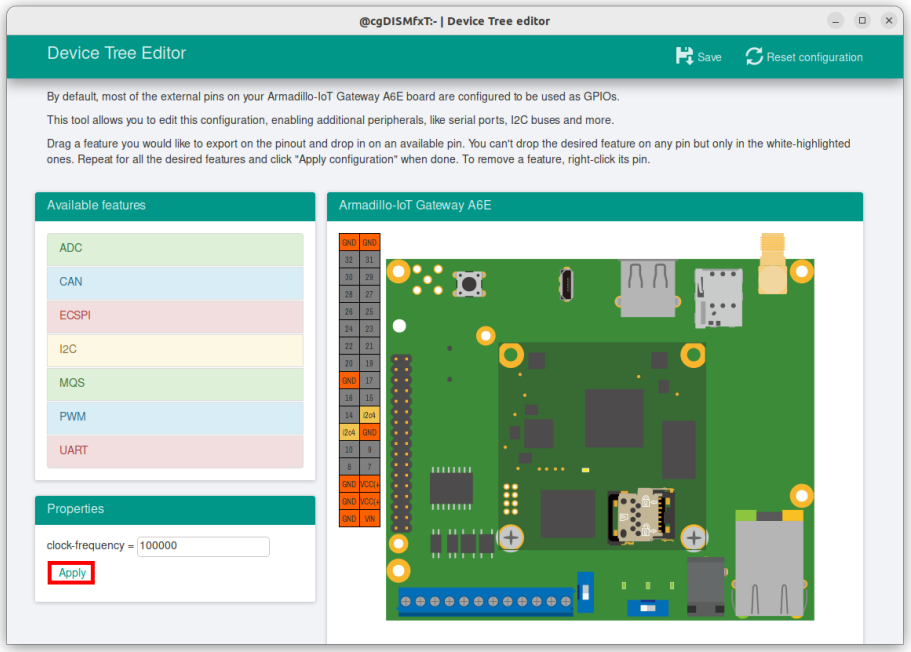


図 10.140 プロパティの保存

10.11.2.4. 機能の削除

全ての機能を削除する場合は、画面右上の「Reset configuration」をクリックします。機能ごとに削除する場合は、画面右側の「Armadillo-IoT Gateway A6E」のピンを右クリックして「Remove」をクリックします。

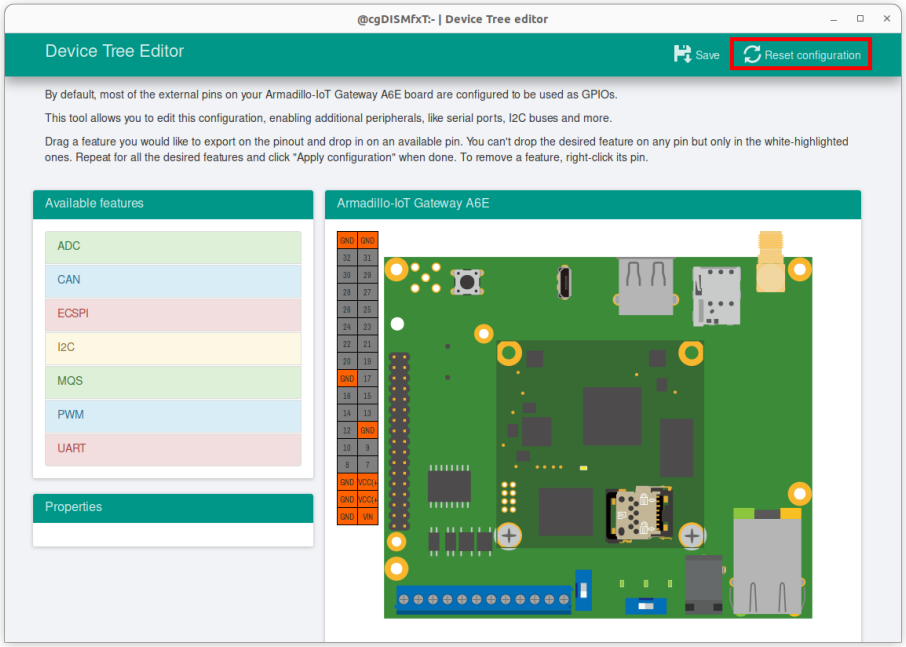


図 10.141 全ての機能の削除

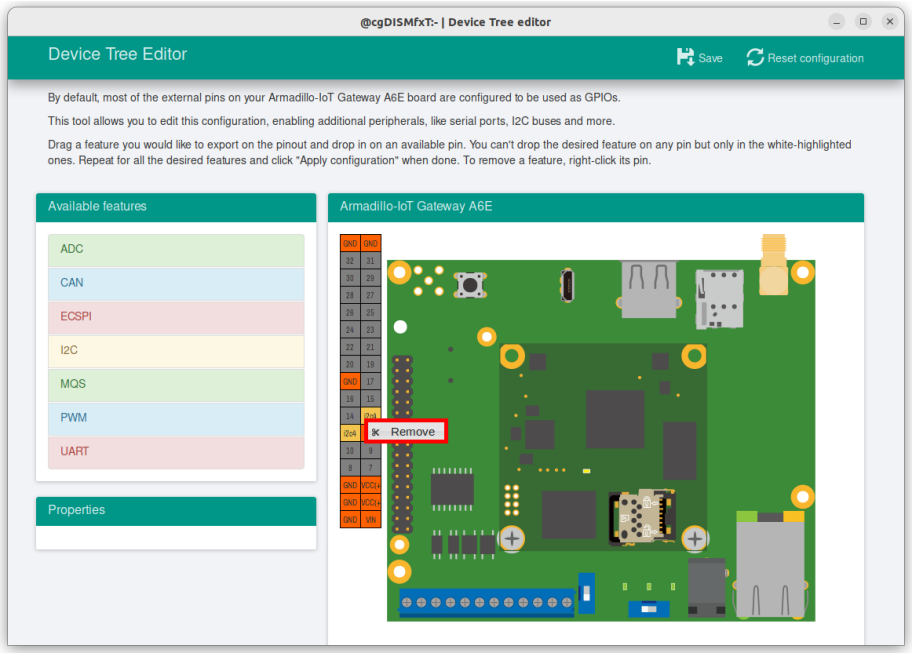


図 10.142 I2C4(SCL/SDA) の削除

10.11.2.5. dtbo/desc の生成

dtbo ファイルおよび desc ファイルを生成するには、画面右上の「Save」をクリックします。

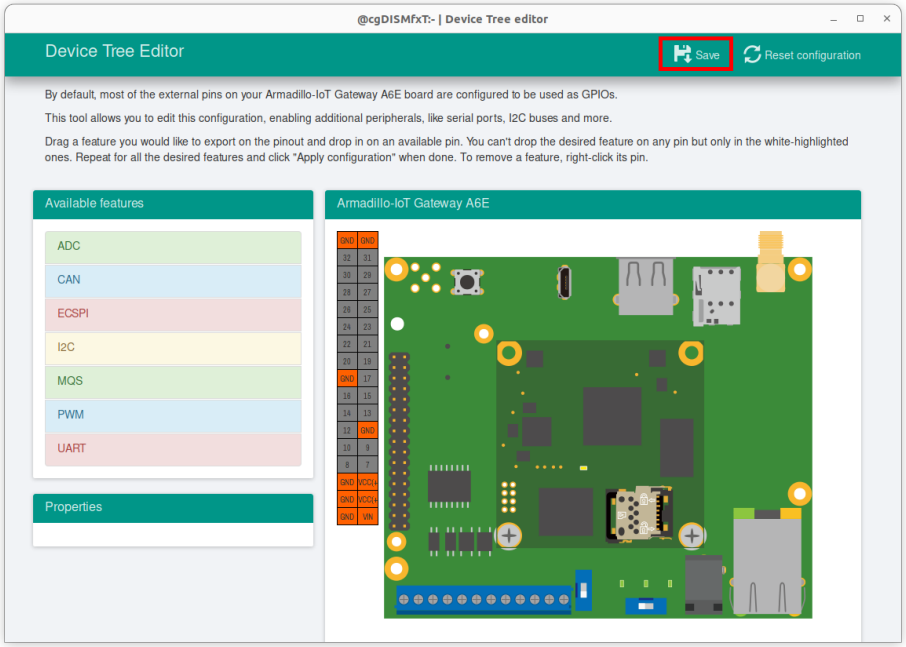


図 10.143 dtbo/desc ファイルの生成

以下の画面ようなメッセージが表示されると、dtbo ファイルおよび desc ファイルの生成は完了です。

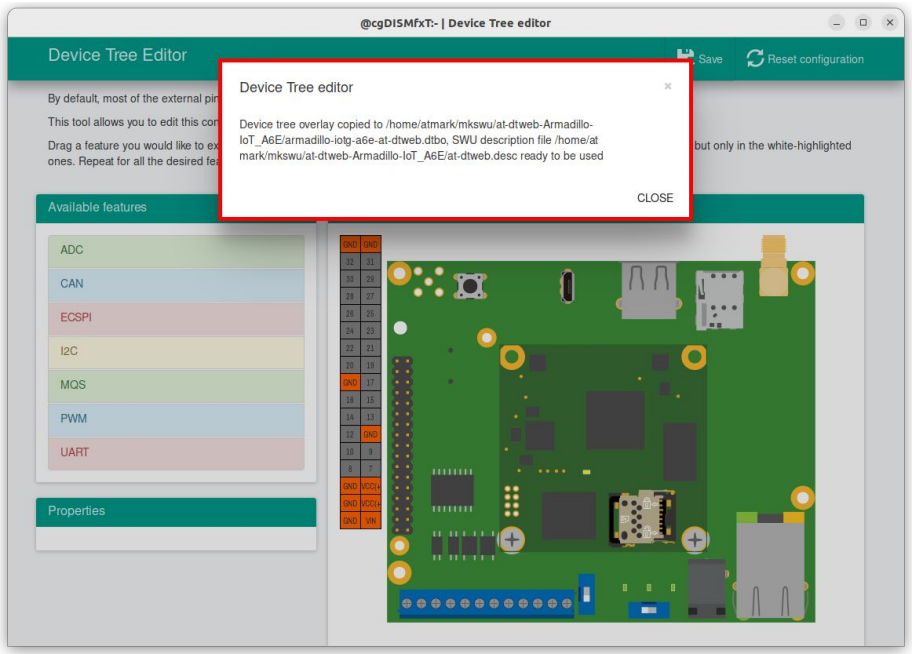


図 10.144 dtbo/desc の生成完了

ビルドが終了すると、ホームディレクトリ下の mkswu/at-dtweb-Armadillo-IoT_A6E/ディレクトリに、DTS overlays ファイル(dtbo ファイル)と desc ファイルが生成されます。Armadillo-IoT ゲートウェイ A6E 本体に書き込む場合は、mkswu コマンドで desc ファイルから SWU イメージを生成してアップデートしてください。

```
[ATDE ~]$ ls ~/mkswu/at-dtweb-Armadillo-IoT_A6E/
armadillo-iotg-a6e-at-dtweb.dtbo  update_overlays.sh
```

```
at-dtweb.desc                       update_preserve_files.sh
[ATDE ~]$ cd ~/mkswu/at-dtweb-Armadillo-IoT_A6E/
[ATDE ~/mkswu/at-dtweb-Armadillo-IoT_A6E]$ mkswu at-dtweb.desc ❶
Enter pass phrase for /home/atmark/mkswu/swupdate.key:
at-dtweb.swu を作成しました。
```

❶ SWU イメージを生成します。

SWU イメージを使ったアップデートの詳細は 「10.9. Armadillo のソフトウェアをアップデートする」を参照してください。

10.11.3. DTS overlays によるカスタマイズ

Device Tree は「DTS overlay」(dtbo) を使用することでも変更できます。

DTS overlay を使用することで、通常の dts の更新が自動的に入りつづける状態で dts の変更でしかできない設定を行うことができます。

/boot/overlays.txt に fdt_overlays を dtbo 名で設定することで、u-boot が起動時にその DTS overlay を通常の dtb と結合して起動します。

複数の DTS overlay を使う場合は以下の例のようにスペースで別けたファイル名を記載することができます。

```
[armadillo ~]# ls /boot/ ❶
armadillo-iotg-a6e-at-dtweb.dtbo  armadillo-iotg-a6e.dtb  uImage
armadillo-iotg-a6e-ems31.dtbo     armadillo.dtb           uboot_env.d
armadillo-iotg-a6e-lwb5plus.dtbo  overlays.txt

[armadillo ~]# persist_file /boot/armadillo-iotg-a6e-at-dtweb.dtbo ❷
[  441.860885] EXT4-fs (mmcblk0p1): re-mounted. Opts: (null)

[armadillo ~]# vi /boot/overlays.txt ❸
fdt_overlays=armadillo-iotg-a6e-ems31.dtbo armadillo-iotg-a6e-at-dtweb.dtbo

[armadillo ~]# persist_file -vp /boot/overlays.txt ❹
'/boot/overlays.txt' -> '/mnt/boot/overlays.txt'
Added "/boot/overlays.txt" to /etc/swupdate_preserve_files

[armadillo ~]# reboot ❺
:(省略)
Applying fdt overlay: armadillo-iotg-a6e-ems31.dtbo ❻
Applying fdt overlay: armadillo-iotg-a6e-at-dtweb.dtbo
:(省略)
```

図 10.145 /boot/overlays.txt の変更例

❶ at-dtweb で作成した dtbo ファイルを USB メモリや microSD カード等の外部記憶装置を用いて転送し、 /boot/ ディレクトリ下に配置します。

❷ 配置した dtbo ファイルを保存します。

❸ /boot/overlays.txt ファイルに「armadillo-iotg-a6e-at-dtweb.dtbo」を追加します。ファイルが存在しない場合は新規に作成してください。このファイルの詳細については 「10.11.3. DTS overlays によるカスタマイズ」 を参照してください。

❹ /boot/overlays.txt を保存し、アップデートの場合でも保存します。

❺ overlay の実行のために再起動します。

❻ シリアルコンソールの場合に、u-boot によるメッセージを確認できます。

10.11.3.1. 提供している DTS overlay

以下の DTS overlay を用意しています：

- armadillo-iotg-a6e-ems31.dtbo: LTE Cat.M1 モジュール搭載モデルで自動的に使用します。
- armadillo-iotg-a6e-lwb5plus.dtbo: WLAN+BT コンボモジュール搭載モデルで自動的に使用します。

10.12. eMMC のデータリテンション

eMMC は主に NAND Flash メモリから構成されるデバイスです。NAND Flash メモリには書き込みしてから 1 年から 3 年程度の長期間データが読み出されないと電荷が抜けてしまう可能性があります。その際、電荷が抜けて正しくデータが読めない場合は、eMMC 内部で ECC (Error Correcting Code) を利用してデータを訂正します。しかし、訂正ができないほどにデータが化けてしまう場合もあります。そのため、一度書いてから長期間利用しない、高温の環境で利用するなどのケースでは、データ保持期間内に電荷の補充が必要になります。電荷の補充にはデータの読み出し処理を実行し、このデータの読み出し処理をデータリテンションと呼びます。

Armadillo-IoT ゲートウェイ A6E に搭載の eMMC は、eMMC 自身にデータリテンション機能が備わっており、A6E に電源が接続されて eMMC に電源供給されている状態で、eMMC 内部でデータリテンション処理が自動実行されます。

10.13. SMS を利用する (Cat.M1 モデルのみ)

Armadillo-IoT ゲートウェイ A6E は、LTE モジュール を使用した SMS の送受信を行うことができます。SMS の送信、受信した SMS の確認および削除などの操作は ModemManager の mmcli コマンドで行うことができます。

本章では mmcli コマンドでの SMS の使用方法について説明します。

10.13.1. 初期設定

SMS が利用可能な SIM を挿入して Armadillo-IoT ゲートウェイ A6E の電源を入れると、ModemManager が必要な初期設定を行い、 SMS が利用可能になります。

SMS の受信は自動的に行われます。

「図 10.146. 言語設定」に示すコマンドを実行し、言語設定を行います。

```
[armadillo ~]# export LANG="ja_JP.UTF-8"
```

図 10.146 言語設定

10.13.2. SMS を送信する

SMS を作成するには、「図 10.147. SMS の作成」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m 0 --messaging-create-sms="number=[送信先電話番号],text='[SMS 本文]'"
```

図 10.147 SMS の作成

SMS の作成に成功すると、以下のように SMS 番号が表示されます。SMS 番号は送信時に使用します。

```
Successfully created new SMS: /org/freedesktop/ModemManager1/SMS/[SMS 番号]
```

図 10.148 SMS 番号の確認

「図 10.149. SMS の送信」に示すコマンドを実行し、SMS 送信を行います。 [SMS 番号] には、 SMS の作成時に表示された番号を指定します。

```
[armadillo ~]# mmcli -s [SMS 番号] --send
```

図 10.149 SMS の送信

10.13.3. SMS を受信する

SMS を送信可能な端末から Armadillo-IoT ゲートウェイ A6E に SMS を送信すると、 Armadillo-IoT ゲートウェイ A6E は自動的に SMS を受信します。

また、 EMS31-J の内蔵ストレージに 10 件 SMS を保存した状態で Armadillo-IoT ゲートウェイ A6E に SMS を送信した場合は、Armadillo-IoT ゲートウェイ A6E は受信を行いません。

受信を行うには、 EMS31-J の内蔵ストレージに保存している SMS を削除するか、他のストレージに移動する必要があります。

10.13.4. SMS 一覧を表示する

「図 10.150. SMS の一覧表示」のコマンドを実行することで、 SMS 一覧を表示できます。

末尾が "(sent)" となっているものが送信した SMS で "(received)" となっているものが受信した SMS です。

```
[armadillo ~]# mmcli -m 0 --messaging-list-sms
Found 7 SMS messages:
        /org/freedesktop/ModemManager1/SMS/0 (received)
        /org/freedesktop/ModemManager1/SMS/1 (received)
        /org/freedesktop/ModemManager1/SMS/2 (received)
        /org/freedesktop/ModemManager1/SMS/3 (received)
        /org/freedesktop/ModemManager1/SMS/4 (sent)
        /org/freedesktop/ModemManager1/SMS/5 (received)
        /org/freedesktop/ModemManager1/SMS/6 (sent)
```

図 10.150 SMS の一覧表示

10.13.5. SMS の内容を表示する

SMS の内容を表示するには、「図 10.151. SMS の内容を表示」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号]
  ----------------------------------
  Content    |              number: XXXXXXXXXX
             |                text: hello world
  ----------------------------------
  Properties |            PDU type: deliver
             |               state: received
             |             storage: me
             |                smsc: +XXXXXXXXXXX
             |           timestamp: XXXXXXXXXXX+XX
```

図 10.151 SMS の内容を表示

受信した SMS は自動的に LTE モジュールの内蔵ストレージに保存されます。Armadillo-IoT ゲートウェイ A6E に搭載されている、 EMS31-J は、最大 10 件まで SMS を保存することが可能です。

SMS の内容を表示した際の「storage: **me**」は、 LTE モジュールの内蔵ストレージに SMS が保存されていることを意味しています。

「storage: **sm**」と表示された場合、 SIM カードのストレージに SMS が保存されています。 SIM カードのストレージに保存できる SMS の件数は SIM カードによって異なります。

ストレージに保存されている SMS は、Armadillo-IoT ゲートウェイ A6E の電源を切断してもデータが保持されます。

10.13.6. SMS を削除する

SMS を削除するには、「図 10.152. SMS の削除」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -m 0 --messaging-delete-sms=[SMS 番号]
```

図 10.152 SMS の削除

10.13.7. SMS を他のストレージに移動する

SIM カードのストレージに SMS を移動するには、「図 10.153. SIM カードのストレージに SMS を移動」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号] --store-in-storage="sm"
```

図 10.153 SIM カードのストレージに SMS を移動

LTE モジュールの内蔵ストレージに SMS を移動するには、「図 10.154. LTE モジュールの内蔵ストレージに SMS を移動」に示すコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号] --store-in-storage="me"
```

図 10.154 LTE モジュールの内蔵ストレージに SMS を移動

10.14. 動作中の Armadillo の温度を測定する

この章では、Armadillo Base OS 搭載製品を組み込んだユーザー製品の熱設計時に役立つ温度プロファイラツールである「atmark-thermal-profiler」について紹介します。

10.14.1. 温度測定の重要性

Armadillo は製品ごとに動作温度範囲が設定されていますが、それらはあくまでも標準筐体に放熱材と共に取り付けて使用した場合の目安であり、実運用時には自作の筐体の使用や放熱の有無などで記載のスペック通りにならない場合があります。また、 Armadillo には CPU または SoC が特定の温度以上になると、自動的にシャットダウンするサーマルシャットダウン機能が搭載されています。そのため、現実的には Armadillo を組み込んだ製品を運用時と同等の環境で動作させつつ、実際に温度を計測して実運用時の CPU 及び SoC 温度がどの程度まで上がるか、サーマルシャットダウンは起こらないかを確かめる必要があります。

Armadillo Base OS 搭載製品では、動作中の Armadillo の各種温度等を取得し CSV 形式で出力する atmark-thermal-profiler を利用することができますので、温度測定に役立てることができます。

10.14.2. atmark-thermal-profiler をインストールする

atmark-thermal-profiler は apk パッケージで公開されていますので、apk add コマンドでインストールすることが可能です。

```
[armadillo ~]# apk upgrade
[armadillo ~]# apk add atmark-thermal-profiler
```

図 10.155 atmark-thermal-profiler をインストールする

atmark-thermal-profiler はデバッグ(開発)用途で温度情報を収集及び解析するツールです。atmark-thermal-profiler は、他の apk パッケージと同様に persist_file -a コマンドで永続的にインストールしておくことが可能ですが、ログの保存のために Armadillo が起動している間 eMMC への書き込みを続けるので、 Armadillo を組み込んだ製品の運用時に動かしたままにしておくことは推奨しません。

atmark-thermal-profiler を永続的にインストールする場合は、運用時には必ず削除してください。

10.14.3. atmark-thermal-profiler を実行・停止する

「図 10.156. atmark-thermal-profiler を実行する」に示すコマンドを実行することで、 atmark-thermal-profiler が動作を開始します。

```
[armadillo ~]# rc-service atmark-thermal-profiler start
```

図 10.156 atmark-thermal-profiler を実行する

「図 10.157. atmark-thermal-profiler を停止する」に示すコマンドを実行することで、 atmark-thermal-profiler が動作を停止します。

```
[armadillo ~]# rc-service atmark-thermal-profiler stop
```

図 10.157 atmark-thermal-profiler を停止する

10.14.4. atmark-thermal-profiler が出力するログファイルを確認する

atmark-thermal-profiler は、インストール直後から自動的に温度や CPU 負荷率、Load Average などの情報を 30 秒に 1 度の周期で集め、/var/log/thermal_profile.csv に追記していきます。

```
[armadillo ~]# head /var/log/thermal_profile.csv
DATE,ONESHOT,CPU_TMEP,SOC_TEMP,LOAD_AVE,CPU_1,CPU_2,CPU_3,CPU_4,CPU_5,USE_1,USE_2,USE_3,USE_4,USE_ ⏎
5
2022-11-30T11:11:05+09:00,0,54,57,0.24,/usr/sbin/rngd -b -p /run/rngd.pid -q -O jitter:buffer_size: ⏎
4133 -O jitter:refill_thresh:4133 -O jitter:thread_count:1,/usr/sbin/chronyd -f /etc/chrony/ ⏎
chrony.conf,[kworker/1:3H-kb],podman network inspect podman,/usr/sbin/NetworkManager -n,22,2,2,0,0,
:（省略）
```

図 10.158 ログファイルの内容例

thermal_profile.csv の 1 行目はヘッダ行です。各列についての説明を「表 10.25. thermal_profile.csv の各列の説明」に記載します。

表 10.25 thermal_profile.csv の各列の説明

ヘッダ	説明
DATE	その行のデータ取得日時です。 "年-月-日 T 時:分:秒+タイムゾーン" の形式で出力されます。
ONESHOT	この列が 1 の行のデータは、サーマルシャットダウンを含むシャットダウンが実行された時に取得されたことを示します。
CPU_TEMP	計測時点の CPU 温度を示します。単位は℃です。
SOC_TEMP	計測時点の SoC 温度を示します。単位は℃です。製品よっては非対応で、その場合は空白になります。
LOAD_AVE	計測時点から直近 1 分間の Load Average です。
CPU_1	計測時点の CPU 使用率 1 位のプロセスです。
CPU_2	計測時点の CPU 使用率 2 位のプロセスです。
CPU_3	計測時点の CPU 使用率 3 位のプロセスです。
CPU_4	計測時点の CPU 使用率 4 位のプロセスです。
CPU_5	計測時点の CPU 使用率 5 位のプロセスです。
USE_1	計測時点の CPU 使用率 1 位のプロセスの CPU 使用率です。
USE_2	計測時点の CPU 使用率 2 位のプロセスの CPU 使用率です。
USE_3	計測時点の CPU 使用率 3 位のプロセスの CPU 使用率です。
USE_4	計測時点の CPU 使用率 4 位のプロセスの CPU 使用率です。
USE_5	計測時点の CPU 使用率 5 位のプロセスの CPU 使用率です。

10.14.5. 温度測定結果の分析

atmark-thermal-profiler を使用して得られたログファイルの内容を分析してみます。

10.14.5.1. サーマルシャットダウン温度の確認

予め、使用している Armadillo が何℃でサーマルシャットダウンするか確認しておきます。ここでは、 Armadillo Base OS を搭載している Armadillo-IoT ゲートウェイ G4 を例とします。他の製品では得られる結果が異なる場合があることに注意してください。

```
[armadillo ~]# cat /sys/class/thermal/thermal_zone0/trip_point_1_temp
105000 ❶
[armadillo ~]# cat /sys/class/thermal/thermal_zone1/trip_point_1_temp
105000 ❷
```

図 10.159 サーマルシャットダウン温度の確認(Armadillo-IoT ゲートウェイ G4 を例に)

❶ CPU のサーマルシャットダウン温度です。ミリ℃で表記されているので、105℃でサーマルシャットダウンすることがわかります。

❷ SoC のサーマルシャットダウン温度です。ミリ℃で表記されているので、105℃でサーマルシャットダウンすることがわかります。

10.14.5.2. 温度測定結果のグラフ化

atmark-thermal-profiler が出力するログ(thermal_profile.csv)は CSV ファイルなので、各種表計算ソフトでインポートしてグラフ化することが可能です。これにより Armadillo 動作中の温度の変化が可視化され、得られる情報が見やすくなります。

「図 10.160. Armadillo-IoT ゲートウェイ G4 で取得した温度のグラフ」は Armadillo-IoT ゲートウェイ G4 上で一定期間 atmark-thermal-profiler を実行して取得した thermal_profile.csv を Google スプレッドシートでグラフ化したものです。例のために、途中で stress-ng コマンドを実行して CPU に負荷を与えた後、 stress-ng コマンドを停止して CPU と SoC の温度が下がるのを待った際のデータです。

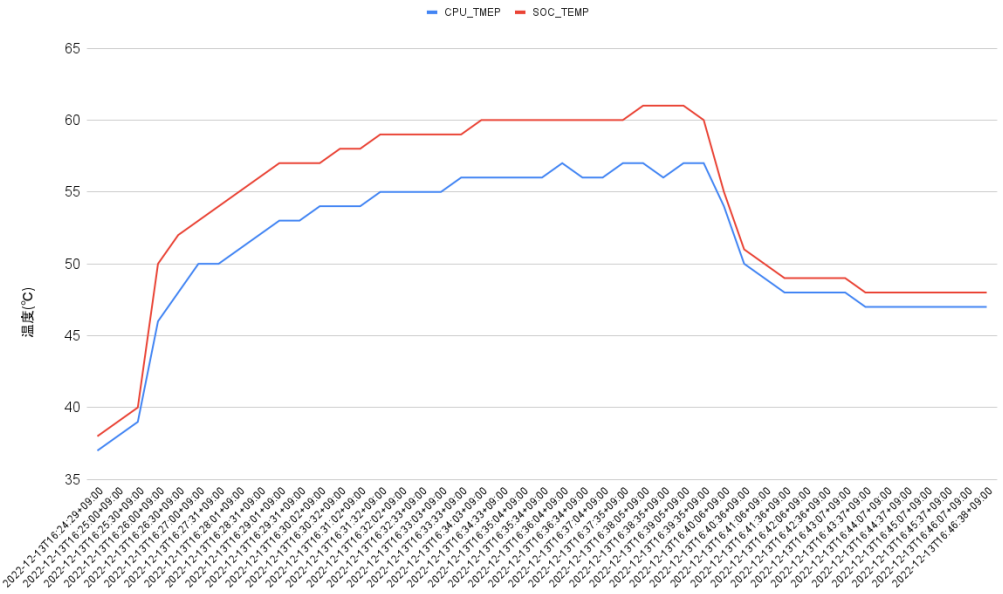


図 10.160 Armadillo-IoT ゲートウェイ G4 で取得した温度のグラフ

グラフの縦軸は温度(℃)で、横軸は時間です。青い線は CPU の温度、赤い線は SoC の温度を表しています。このグラフと、「10.14.5.1. サーマルシャットダウン温度の確認」で得たサーマルシャットダウン温度を見比べると、 CPU に負荷をかけた際であっても SoC の温度は 60℃ 前後ほどまでしか上がらず、この条件で動く Armadillo が温度的にどれほど余裕を持っているかをひと目で確認できます。

10.14.5.3. CPU 使用率の確認

atmark-thermal-profiler は、時間毎の温度だけでなく CPU 使用率と CPU 使用率の高いプロセスについても取得して記録します。CPU 使用率については thermal_profile.csv の CPU_1～CPU_5 列と、USE_1～USE_5 列を参照してください。各列について詳しくは「表 10.25. thermal_profile.csv の各列の説明」にまとまっています。

一般的に CPU 使用率が高くなると、 CPU 周辺の温度も高くなります。そのため、測定した温度が高い場合は、 CPU 使用率の高いプロセスに注目して、 CPU を無駄に使用している意図しない処理が行なわれていないかなどを確認することをおすすめします。

11. 動作ログ

11.1. 動作ログについて

Armadillo-IoT ゲートウェイ A6E ではシステムが出力するログの一部は、一般的な /var/log ディレクトリではなく、/var/at-log ディレクトリに出力されます。/var/at-log は、ルートファイルシステムとは別のパーティションになっているので、ルートファイルシステムに障害が発生した場合でも、/var/at-log のパーティションが無事であれば、ログファイルを取り出して、不具合等の解析に利用することができます。

11.2. 動作ログを取り出す

ログファイルは /var/at-log ディレクトリ内に atlog というファイル名で作成されているので、これを任意のディレクトリにコピーすることで取り出せます。もし、eMMC 上のルートファイルシステムが壊れてしまい起動できない場合は、microSD カードから起動することでログファイルを取り出すことができます。

11.3. ログファイルのフォーマット

ログファイルの内容はテキストデータであり、以下のようなフォーマットになっています。

```
日時 armadillo ログレベル 機能: メッセージ
```

図 11.1 動作ログのフォーマット

11.4. ログ用パーティションについて

ログ出力先である /var/at-log ディレクトリには、GPP である /dev/mmcblk0gp1 パーティションがマウントされています。このパーティションに論理的な障害が発生した場合は、/dev/mmcblk0gp1 のデータを /dev/mmcblk0gp2 にコピーし、/dev/mmcblk0gp1 は FAT ファイルシステムでフォーマットされます。このパーティションの障害チェックはシステム起動時に自動的に実行されます。

12. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-IoT ゲートウェイ A6E の Linux カーネル仕様について説明します。

12.1. デフォルトコンフィギュレーション

工場出荷時の Armadillo-IoT ゲートウェイ A6E に書き込まれている Linux カーネルは、デフォルトコンフィギュレーションが適用されています。デフォルトコンフィギュレーションが記載されているファイルは、Linux カーネルソースファイル (linux-v5.10-at[VERSION].tar.gz)に含まれる +arch/arm/configs/armadillo-iotg-a6e_defconfig です。

12.2. ブートパラメータ

ブートパラメータは、Linux カーネルに与えるパラメータのことです。ブートパラメータを指定することで Linux カーネルのいくつかの動作を変更することができます。工場出荷状態の Armadillo-IoT ゲートウェイ A6E の主要なブートパラメータを次に示します。

表 12.1 Linux カーネルの主要デフォルトブートパラメータ

ブートパラメータ	値	説明
コンソール	UART3	起動ログなどが出力されるイニシャルコンソールです。Device Tree で指定されます。
ルートファイルシステム	eMMC または microSD カード	起動デバイスに応じて適切に設定されます。U-Boot で指定されます。

同一のブートパラメータ(例えばルートファイルシステム)が複数の場所から与えられた場合は、次の優先順位で上書きされます。

1. U-Boot 環境変数の bootargs および optargs
2. Device Tree の chosen ノード内の bootargs および stdout-path

異なるブートパラメータが複数の場所から与えられた場合はマージされます。また、U-Boot および Device Tree から一切のブートパラメータが与えられない場合は、次がブートパラメータとして利用されます。

・ Linux カーネルコンフィギュレーションの CONFIG_CMDLINE

現在起動中の Linux カーネルのブートパラメータを確認するには、以下コマンドを実行します。

```
[armadillo]# cat /proc/cmdline
root=/dev/mmcblk0p2 rootwait
[armadillo]# cat /proc/device-tree/chosen/stdout-path
/soc/bus@2000000/spba-bus@2000000/serial@2020000
```

12.3. Linux ドライバー覧

Armadillo-IoT ゲートウェイ A6E で利用することができるデバイスドライバについて説明します。各ドライバで利用しているソースコードのうち主要なファイルのパスや、コンフィギュレーションに必要な情報、及びデバイスファイルなどについて記載します。

12.3.1. Armadillo-IoT ゲートウェイ A6E

Armadillo-IoT ゲートウェイ A6E のハードウェアの構成情報やピンのマルチプレクス情報、i.MX6ULL の初期化手順などが定義されています。

関連するソースコード
- arch/arm/mach-imx/
- arch/arm/boot/dts/armadillo-iotg-a6e.dts
- arch/arm/boot/dts/armadillo-610.dtsi
- arch/arm/boot/dts/imx6ull.dtsi
- arch/arm/boot/dts/imx6ul.dtsi

カーネルコンフィギュレーション

```
System Type --->
  [*] Freescale i.MX family  --->                          <ARCH_MXC>
    [*]   i.MX6 UltraLite support                          <SOC_IMX6UL>
```

12.3.2. UART

Armadillo-IoT ゲートウェイ A6E のシリアルは、i.MX6ULL の UART (Universal Asynchronous Receiver/Transmitter) を利用しています。Armadillo-IoT ゲートウェイ A6E の標準状態では、UART3 (CON7) を利用しています。

フォーマット
- データビット長: 7 or 8 ビット
- ストップビット長: 1 or 2 ビット
- パリティ: 偶数 or 奇数 or なし
- フロー制御: CTS/RTS or XON/XOFF or なし
- 最大ボーレート:4Mbps

UART3(CON7)は 4Mbps で利用することができません。USB シリアル変換 IC(CP2102N/Silicon Labs)の最大ボーレートが 3Mbps である為です。

関連するソースコード
- drivers/tty/serial/imx.c
- drivers/tty/serial/imx_earlycon.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/serial/fsl-imx-uart.yaml

デバイスファイル

シリアルインターフェース	デバイスファイル
UART2	/dev/ttymxc1 (LWB5+)
UART3	/dev/ttymxc2 (UART3)
UART4	/dev/ttymxc3 (EMS31-J)
UART5	/dev/ttymxc4 (RS485)

カーネルコンフィギュレーション

```
Device Drivers  --->
  Character devices  --->
    [*] Enable TTY                                          <TTY>
        Serial drivers  --->
          <*> IMX serial port support                  <SERIAL_IMX>
          <*>   Console on IMX serial port     <SERIAL_IMX_CONSOLE>
          [*] Earlycon on IMX serial port      <SERIAL_IMX_EARLYCON>
```

12.3.3. LTE

Armadillo-IoT ゲートウェイ A6E には、Quectel 製 EMS31-J が搭載されています。EMS31-J は、「12.3.8. USB ホスト」に示す OTG2 に接続されています。

デバイス
- /dev/ttyUSB0
- /dev/ttyUSB1
- /dev/ttyUSB2
  - ModemManager が /dev/ttyCommModem のシンボリックリンクを作成し AT コマンド用ポートとして使用します。
- /dev/ttyUSB3
- /dev/ttymxc3

関連するソースコード
- drivers/reset/reset-ec25.c
- drivers/net/ppp/ppp_generic.c
- drivers/net/ppp/ppp_async.c

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Network device support  --->                    <NETDEVICES>
     [*]   PPP (point-to-point protocol) support      <PPP>
     [*]     PPP support for async serial ports       <PPP_ASYNC>
  -*- Reset Controller Support  --->                <RESET_CONTROLLER>
     [*]   GPIO-based Reset Driver for Tales EMS31    <RESET_EMS31>
```

12.3.4. WLAN

Armadillo-IoT ゲートウェイ A6E には、Laird Connectivity 製 LWB5+ が搭載されています。LWB5+ の WLAN は「12.3.7. SD ホスト」に示す uSDHC2 に接続されています。

機能
- IEEE 802.11a/b/g/n/ac 準拠

- 最大通信速度: 49.5Mbps(理論値)
- 動作モード: インフラストラクチャモード(STA/AP), アドホックモード
- チャンネル(2.4GHz): 1-14
- チャンネル(5GHz): 36-48, 52-64, 100-140

LWB5+の最大通信速度は 433.3Mbps(802.11ac, 1x1 SISO, HT80, MCS9, SGI) ですが、「12.3.7. SD ホスト」に接続される為、49.5Mbps に制限されます。

ネットワークデバイス
- wlan0

関連するソースコード
- drivers/net/wireless/broadcom/brcm80211/brcmfmac/*

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Network device support  --->                    <NETDEVICES>
    [*]   Wireless LAN  --->                                <WLAN>
      [*]   Broadcom devices                  <WLAN_VENDOR_BROADCOM>
      [*]     Broadcom FullMAC WLAN driver              <BRCMFMAC>
      [*]     SDIO bus interface support for FullMAC driver
                                                        <BRCMFMAC_SDIO>
```

LWB5+ のファームウェアは、ATDE にインストールされている firmware-brcm80211 パッケージに含まれています。ファームウェアは Linux カーネルイメージ内に改変無く配置されます。firmware-ti-connectivity の著作権およびライセンス情報については、ATDE 上で/usr/share/doc/firmware-brcm80211/copyright を参照してください。

12.3.5. BT

Armadillo-IoT ゲートウェイ A6E には、Laird Connectivity 製 LWB5+ が搭載されています。LWB5+ の BT は「12.3.2. UART」に示す UART2 に接続されています。

機能 , デバイス
- hci0

関連するソースコード
- drivers/bluetooth/hci_bcm.c

カーネルコンフィギュレーション

```
[*] Networking support  --->                                <NET>
  [*]   Bluetooth subsystem support  --->                    <BT>
    Bluetooth device drivers  --->
      [*] Broadcom protocol support                <BT_HCIUART_BCM>
```

12.3.6. Ethernet

Armadillo-IoT ゲートウェイ A6E の Ethernet (LAN) は、i.MX6ULL の ENET(10/100-Mbps Ethernet MAC)を利用しています。

機能
- 通信速度: 100Mbps (100BASE-TX), 10Mbps (10BASE-T)
- 通信モード: Full-Duplex (全二重), Half-Duplex (半二重)
- Auto Negotiation サポート
- キャリア検知サポート
- リンク検出サポート

関連するソースコード
- drivers/net/ethernet/freescale/fec_main.c
- drivers/net/phy/smsc.c
- drivers/net/mdio/of_mdio.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/net/fsl-fec.txt
- Documentation/devicetree/bindings/net/smsc-lan87xx.txt
- Documentation/devicetree/bindings/net/net/ethernet-phy.yaml

ネットワークデバイス
- eth0

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Network device support  --->                    <NETDEVICES>
    [*]   Ethernet driver support  --->                 <ETHERNET>
      [*]   Freescale devices                <NET_VENDOR_FREESCALE>
     <*>     FEC ethernet controller (of ColdFire and some i.MX CPUs)
                                                              <FEC>
    -*-   PHY Device support and infrastructure  --->      <PHYLIB>
      [*]   SMSC PHYs                                     <SMSC_PHY>
```

12.3.7. SD ホスト

Armadillo-IoT ゲートウェイ A6E の SD ホストは、i.MX6ULL の uSDHC (Ultra Secured Digital Host Controller) を利用しています。Armadillo-IoT ゲートウェイ A6E では、LWB5+と SD インターフェース(Armadillo-610: CON1) が uSDHC2 を共用しています。そのため、どちらか一方しか利用することができません。Armadillo-IoT ゲートウェイ A6E の標準状態では、LWB5+が有効になっています。

機能
- カードタイプ: SD/SDHC/SDXC/SDIO
- バス幅: 1bit or 4bit
- スピードモード: Default Speed (24.75MHz), High Speed (49.5MHz)
- カードディテクトサポート

デバイスファイル
- /dev/mmcblk1 (SD インターフェース)

関連するソースコード
- drivers/mmc/host/sdhci-esdhc-imx.c
- drivers/mmc/host/sdhci-of-esdhc.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/mmc/fsl-imx-esdhc.yaml
- Documentation/devicetree/bindings/mmc/mmc-controller.yaml

カーネルコンフィギュレーション

```
Device Drivers  --->
  <*> MMC/SD/SDIO card support  --->                              <MMC>
    <*>   MMC block device driver                              <MMC_BLOCK>
    (32)    Number of minors per block device    <MMC_BLOCK_MINORS>
          *** MMC/SD/SDIO Host Controller Drivers ***
    <*>   Secure Digital Host Controller Interface support
                                                               <MMC_SDHCI>
    <*>     SDHCI platform and OF driver helper   <MMC_SDHCI_PLTFM>
    <*>       SDHCI OF support for the Freescale eSDHC controller
                                                     <MMC_SDHCI_OF_ESDHC>
    <*>       SDHCI support for the Freescale eSDHC/uSDHC i.MX
controller support
                                                   <MMC_SDHCI_ESDHC_IMX>
```

12.3.8. USB ホスト

Armadillo-IoT ゲートウェイ A6E の USB ホストは、i.MX6ULL の USB-PHY (Universal Serial Bus 2.0 Integrated PHY) および USB (Universal Serial Bus Controller) を利用しています。Armadillo-IoT ゲートウェイ A6E では、USB ホストインターフェース(CON9) が OTG1 を利用しています。OTG2 は EMS31-J に接続されています。

機能
- Universal Serial Bus Specification Revision 2.0 準拠
- Enhanced Host Controller Interface (EHCI)準拠
- 転送レート: USB2.0 High-Speed (480Mbps), Full-Speed (12Mbps), Low-Speed (1.5Mbps)

デバイスファイル
- メモリデバイスの場合は、デバイスを認識した順番で/dev/sdN (N は'a'からの連番)となります。
- I/O デバイスの場合は、ファンクションに応じたデバイスファイルとなります。

関連するソースコード
- drivers/usb/chipidea/
- drivers/usb/phy/phy-mxs-usb.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/usb/ci-hdrc-usb2.txt
- Documentation/devicetree/bindings/phy/mxs-usb-phy.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] USB support  --->                                  <USB_SUPPORT>
          *** USB Host Controller Drivers ***
    [*]   EHCI HCD (USB 2.0) support                  <USB_EHCI_HCD>
    [*]     Support for Freescale i.MX on-chip EHCI USB controller
```

```
                                                          <USB_EHCI_MXC>
    [*]   ChipIdea Highspeed Dual Role Controller     <USB_CHIPIDEA>
    [*]     ChipIdea host controller              <USB_CHIPIDEA_HOST>
          USB Physical Layer drivers  --->
      [*] Freescale MXS USB PHY support                 <USB_MXS_PHY>
```

12.3.9. リアルタイムクロック

Armadillo-IoT ゲートウェイ A6E のリアルタイムクロックは、Micro Crystal 製 RV-8803-C7 が搭載されておりこれを利用しています。 RV-8803-C7 は、「12.3.12. I2C」に示す I2C2 に接続されています。 i.MX6ULL の RTC 機能も存在します。

機能
- アラーム割り込みサポート

デバイスファイル
- /dev/rtc (/dev/rtc0 へのシンボリックリンク)
- /dev/rtc0 (RV-8803-C7)
- /dev/rtc1 (i.MX6ULL SNVS_HP Real Time Counter)

RTC が /dev/rtc0 となるよう、Device Tree でエイリアスを設定しています。そのため、i.MX6ULL の RTC 機能は /dev/rtc1 となります。エイリアスの設定は、 arch/arm/boot/dts/armadillo-iotg-a6e.dts で行っています。

関連するソースコード
- drivers/rtc/rtc-rv8803.c
- drivers/rtc/rtc-snvs.c

カーネルコンフィギュレーション

```
Device Drivers --->
  [*] Real Time Clock  --->                                 <RTC_CLASS>
    (rtc0)  RTC used to synchronize NTP adjustment
                                                  <RTC_SYSTOHC_DEVICE>
    <*>   Micro Crystal RV8803, Epson RX8900        <RTC_DRV_RV8803>
    <*>   Freescale SNVS RTC support                 <RTC_DRV_SNVS>
```

アラーム割り込みは、デバイスファイル経由で利用することができます。詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/admin-guide/rtc.rst)やサンプルプログラム(tools/testing/selftests/rtc/rtctest.c)を参照してください。

12.3.10. LED

Armadillo-IoT ゲートウェイ A6E に搭載されているソフトウェア制御可能な LED には、GPIO が接続されています。Linux では、GPIO 接続用 LED ドライバ (leds-gpio) で制御することができます。

sysfs LED クラスディレクトリ
- /sys/class/leds/led1
- /sys/class/leds/led2
- /sys/class/leds/led3

関連するソースコード
- drivers/leds/leds-gpio.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/leds/leds-gpio.yaml

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] LED Support  --->                                          <NEW_LEDS>
    <*>   LED Support for GPIO connected LEDs                    <LEDS_GPIO>
```

12.3.11. ユーザースイッチとイベント信号

Armadillo-IoT ゲートウェイ A6E に搭載されているユーザースイッチには、GPIO が接続されています。Linux では、ユーザー空間でイベント(Press/Release)を検出することができます。Linux では、GPIO 接続用キーボードドライバ(gpio-keys)で制御することができます。ユーザースイッチと信号には、次に示すキーコードが割り当てられています。

表 12.2 キーコード

ユーザースイッチ	キーコード	イベントコード
SW1	KEY_ENTER	28
EMS31-J RING0	KEY_WAKEUP	143

デバイスファイル
- /dev/input/by-path/platform-gpio-keys-event

関連するソースコード
- drivers/input/keyboard/gpio_keys.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/input/gpio-keys.yaml

カーネルコンフィギュレーション

```
Device Drivers  --->
  Input device support  --->
    -*- Generic input layer (needed for keyboard, mouse, ...)
                                                                 <INPUT>
      [*]   Keyboards  --->                                <INPUT_KEYBOARD>
      <*>   GPIO Buttons                                    <KEYBOARD_GPIO>
```

12.3.12. I2C

Armadillo-IoT ゲートウェイ A6E の I2C インターフェースは、i.MX6ULL の I2C(I2C Controller)を利用します。また、i2c-gpio を利用することで、I2C バスを追加することができます。Armadillo-IoT ゲートウェイ A6E で利用している I2C バスと、接続される I2C デバイスを次に示します。

表 12.3 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
0(I2C1)	0x08	PF3000 (PMIC)
1(I2C2)	0x20	TCA9534 (GPIO エキスパンダー)
	0x32	RV8803 (RTC)
	0x48	SE050(セキュアエレメント)

Armadillo-IoT ゲートウェイ A6E の標準状態では、CONFIG_I2C_CHARDEV が有効となっているためユーザードライバで I2C デバイスを制御することができます。ユーザードライバを利用する場合は、Linux カーネルで I2C デバイスに対応するデバイスドライバを無効にする必要があります。

機能
- 最大転送レート: 400kbps

デバイスファイル
- /dev/i2c-0 (I2C1)
- /dev/i2c-1 (I2C2)
- /dev/i2c-2 (I2C3)

関連するソースコード
- drivers/i2c/busses/i2c-imx.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/i2c/i2c-imx.yaml

カーネルコンフィギュレーション

```
Device Drivers  --->
  I2C support  --->
    -*- I2C support                                          <I2C>
      <*>   I2C device interface                       <I2C_CHARDEV>
      I2C Hardware Bus support  --->
        <*> IMX I2C interface                              <I2C_IMX>
```

12.3.13. GPIO

Armadillo-IoT ゲートウェイ A6E の GPIO は、i.MX6ULL の GPIO(General Purpose Input/Output)および、Texas Instruments 製 TCA9534(GPIO エキスパンダー)を利用しています。

関連するソースコード
- drivers/gpio/gpio-mxc.c
- drivers/gpio/gpio-pca953x.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/gpio/fsl-imx-gpio.yaml
- Documentation/devicetree/bindings/gpio/gpio-pca95xx.yaml

デバイスファイル

デバイスファイル	GPIO 番号
/dev/gpiochip0	0～31(GPIO1_IO00～GPIO1_IO31)
/dev/gpiochip1	32～53(GPIO2_IO00～GPIO2_IO21)
/dev/gpiochip2	64～92(GPIO3_IO00～GPIO3_IO28)
/dev/gpiochip3	96～124(GPIO4_IO00～GPIO4_IO28)
/dev/gpiochip4	128～139(GPIO5_IO00～GPIO5_IO11)
/dev/gpiochip5	504～511 [a] (TCA9534)

[a]GPIO エキスパンダーを追加した場合は、番号が異なる可能性があります。

sysfs GPIO クラスディレクトリ
- /sys/class/gpio/

sysfs GPIO クラスは旧バージョンの Linux カーネルとの互換性維持の為に残っています。新しくアプリケーションを開発する際の利用はおすすめしません。新しくアプリケーションを開発する場合は、libgpiod パッケージに含まれるアプリケーションまたは Linux カーネルのソースコードに含まれているサンプル(tools/gpio/)を参考にしてください。

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] GPIO Support  --->                                        <GPIOLIB>
    (512) Maximum number of GPIOs for fast path
                                              <GPIOLIB_FASTPATH_LIMIT>
    I2C GPIO expanders  --->
       [*] PCA95[357]x, PCA9698, TCA64xx, and MAX7310 I/O ports
                                                          <GPIO_PCA953X>
       [*]   Interrupt controller support for PCA953x
                                                    <GPIO_PCA953X_IRQ>
    Memory mapped GPIO drivers  --->
      -*- i.MX GPIO support                                     <GPIO_MXC>
```

12.3.14. パワーマネジメント

Armadillo-IoT ゲートウェイ A6E のパワーマネジメント機能は、Linux の SPM(System Power Management)および DPM(Device Power Management)を利用しています。パワーマネジメント状態を省電力モードに遷移させることにより、Armadillo-IoT ゲートウェイ A6E の消費電力を抑えることができます。パワーマネジメント状態を省電力モードに遷移させると、アプリケーションの実行は一時停止し、Linux カーネルはサスペンド状態となります。起床要因が発生すると、Linux カーネルのリジューム処理が行われた後、アプリケーションの実行を再開します。

sysfs ファイル　・ /sys/power/state

関連するソースコード　・ kernel/power/

カーネルコンフィギュレーション

```
Power management options  --->
  [*] Suspend to RAM and standby                              <SUSPEND>
  -*- Device power management core functionality                   <PM>
```

Armadillo-IoT ゲートウェイ A6E が対応するパワーマネジメント状態と、/sys/power/state に書き込む文字列の対応を次に示します。

表 12.4 対応するパワーマネジメント状態

パワーマネジメント状態	文字列	説明
Suspend-to-RAM	mem	Suspend-to-Idle よりも消費電力を抑えることができる
Suspend-to-Idle	freeze	suspend-to-ram よりも短時間で復帰することができる

起床要因として利用可能なデバイスは次の通りです。

UART3(CON7)　起床要因　データ受信

有効化

```
[armadillo ~]# echo enabled > /sys/class/tty/
ttymxc2/power/wakeup
```

RS485(UART5)　起床要因　データ受信

	有効化	[armadillo ~]# echo enabled > /sys/class/tty/ttymxc4/power/wakeup ⏎
USB ホストインターフェース (CON9)	起床要因	USB デバイスの挿抜
	有効化	[armadillo ~]# echo enabled > /sys/devices/platform/soc@0/2184000.usb/power/wakeup ⏎ [armadillo ~]# echo enabled > /sys/bus/usb/devices/usb1/power/wakeup ⏎
RTC(i.MX6ULL)	起床要因	アラーム割り込み
	有効化	[armadillo ~]# echo enabled > /sys/bus/platform/devices/20cc000.snvs¥:snvs-rtc-lp/power/wakeup ⏎
RTC(RV8803)	起床要因	アラーム割り込み
	有効化	[armadillo ~]# echo enabled > /sys/bus/i2c/devices/1-0032/power/wakeup ⏎
EMS31-J RING0 (Ring Indicator)	起床要因	SMS 受信
	有効化	デフォルトで有効化されています

13. ソフトウェア仕様

13.1. SWUpdate

13.1.1. SWUpdate とは

SWUpdate は、デバイス上で実行されるプログラムで、ネットワーク/ストレージ経由でデバイスのソフトウェアを更新することができます。Stefano Babic, DENX software engineering, Germany によってオープンソースで開発が進められています。

Armadillo-IoT ゲートウェイ A6E では、SWUpdate を利用することで次のような機能を実現しています。

- A/B アップデート(アップデートの 2 面化)
- リカバリーモード
- ソフトウェアの圧縮、暗号化、署名付与
- Web サーバー機能
- hawkBit への対応
- ダウングレードの禁止

13.1.2. swu パッケージ

swu パッケージは、SWUpdate 独自のソフトウェアの配布フォーマットです。SWUpdate では、1 回のアップデートは 1 つの swu パッケージで行われます。

swu パッケージには、次のような様々なものを含めることができます。

- アップデート対象のイメージファイル
- アップデート対象のイメージファイルのチェックサム
- アップデート前後に実行するスクリプト
- 書き込み先ストレージの情報
- U-Boot 環境変数の書き換え情報
- ソフトウェアのバージョン情報
- etc…

13.1.3. A/B アップデート(アップデートの 2 面化)

A/B アップデートは、Flash メモリにパーティションを 2 面確保し、アップデート時には交互に利用する仕組みです。

常に使用していない方のパーティションを書き換えるため次の特徴を持ちます。(※以下で、○は長所、×は短所を意味します。)

- ○ アップデートによって動作中のソフトウェアは破壊されない
- ○ 書き込みが電源断などで中断後しても、すぐに復帰出来る
- ○ 機器が動作中に書き込みが出来る
- × 使用 Flash メモリ量が増える

13.1.4. ロールバック（リカバリー）

システムが起動できなくなった際に、自動的にアップデート前のシステムにロールバックします。

ロールバック状態の確認は 「10.10.3. ロールバック状態の確認」 を参照してください。

ロールバックする条件は次の通りです:

- rootfs にブートに必要なファイルが存在しない場合（/boot/lmage, /boot/armadillo.dtb)
- 3 回起動を試して「bootcount」サービスが 1 度も起動できなかった場合は、次の起動時にロールバックします。

  bootcount 機能は uboot の「upgrade_available」変数で管理されています。bootcount 機能を利用しないようにするには、「10.10.6. u-boot の環境変数の設定」 を参照して変数を消します。
- ユーザーのスクリプトなどから、「abos-ctrl rollback」コマンドを実行した場合。

ロールバックが実行されると `/var/at-log/atlog` にログが残ります。

13.2. hawkBit

13.2.1. hawkBit とは

hawkBit は、サーバー上で実行されるプログラムで、ネットワーク経由でデバイスのソフトウェアを更新(配信)することができます。

hawkBit は次のような機能を持っています。

- ソフトウェアの管理
- デバイスの管理
  - デバイス認証 (セキュリティトークン、証明書)
  - デバイスのグループ化
- アップデート処理の管理
  - 進捗のモニタリング
  - スケジューリング、強制アップデート
- RESTful API での直接操作

13.2.2. データ構造

hawkBit は、配信するソフトウェアを次のデータ構造で管理します。

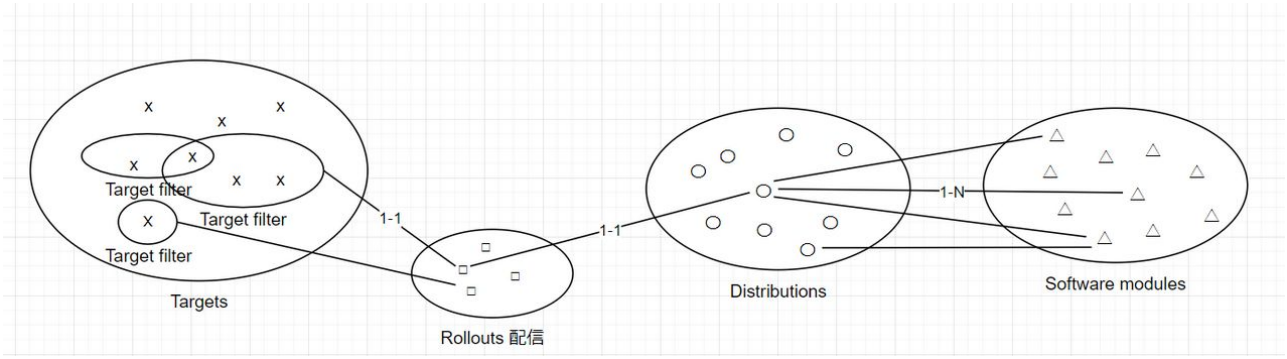


図 13.1 hawkBit が扱うソフトウェアのデータ構造

14. ハードウェア仕様

14.1. 電気的仕様

14.1.1. 絶対最大定格

表 14.1 絶対最大定格

項目	記号	Min.	Max.	単位	備考
電源電圧	VIN	-0.3	26.4	V	CON5,CON6
入出力電圧(GPIO 信号)	VI,VO	-0.3	OVDD+0.3	V	CON8(OVDD=VCC_3.3V)
入出力電圧(RS485 信号)	VI_RS485 VO_RS485	-8.0	12.5	V	CON6(DATA+,DATA-)
入力電圧(接点入力)	VI_DI	-26.4	26.4	V	CON6(DI1,DI2,COM)
出力耐圧(接点出力)	Voff_DO	-60	60	V	CON6(DO1A,DO1B,DO2A,DO2B)
RTC バックアップ電源電圧	RTC_BAT	-0.3	5.5	V	CON10
動作温度範囲	Topr	-20	60	℃	結露なきこと

絶対最大定格は、あらゆる使用条件や試験状況において、瞬時でも超えてはならない値です。上記の値に対して余裕をもってご使用ください。

14.1.2. 推奨動作条件

表 14.2 推奨動作条件

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	VIN	8	12	26.4	V	CON5,CON6
RTC バックアップ電源電圧	RTC_BAT	2.4	3	3.6	V	CON10,対応電池 : CR1220 等

14.1.3. 入出力仕様

14.1.3.1. 電源入力仕様

表 14.3 電源入力仕様

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	VIN	8	12	26.4	V	CON5,CON6

14.1.3.2. 電源出力仕様

表 14.4 電源出力仕様

項目	記号	Min.	Typ.	Max.	単位	備考
5V 電源電圧	VCC_5V	4.75	5	5.25	V	CON8
3.3V 電源電圧	VCC_3.3V	3.102	3.3	3.498	V	CON8

項目	記号	Min.	Typ.	Max.	単位	備考
USB VBUS 電圧	USB_OTG1_VBUS	4.75	5	5.25	V	CON9

14.1.3.3. 入出力インターフェース(CON6)の入出力仕様

表 14.5 入出力インターフェース(CON6)の入出力仕様

接点入力	**入力インピーダンス**	4.7 kΩ
	入力 ON 電流	2.0 mA 以上
	入力 OFF 電流	0.2 mA 以下
	応答時間	1ms 以内
	入力電圧	最大 26.4 V
接点出力	**定格電圧**	最大 48 V
	定格電流	最大 500 mA
	応答時間	2ms 以内
	出力形式	無極性
絶縁耐圧		2kV

14.1.3.4. 拡張インターフェース(CON8)の入出力仕様

表 14.6 拡張インターフェース(CON8)の入出力仕様(OVDD = VCC_3.3V)

項目	記号	Min.	Max.	単位	備考
ハイレベル出力電流	IOH	OVDD-0.15	OVDD	mA	IOH = -0.1mA, -1mA
ローレベル出力電流	IOL	0	0.15	mA	IOL = 0.1mA, 1mA
ハイレベル入力電圧[a]	VIH	0.7xOVDD	OVDD	V	-
ローレベル入力電圧[a]	VIL	0	0.3xOVDD	V	-
入力リーク電流(no Pull-up/Pull-down)	IIN	-1	1	μA	-
Pull-up 抵抗(5kΩ)	-	4	6	kΩ	-
Pull-up 抵抗(47kΩ)	-	37.6	56.4	kΩ	-
Pull-up 抵抗(100kΩ)	-	80	120	kΩ	-
Pull-down 抵抗(100kΩ)	-	80	120	kΩ	-

[a]オーバーシュートとアンダーシュートは 0.6V 以下でかつ 4ns を超えないようにしてください。

14.1.4. 電源回路の構成

Armadillo-IoT ゲートウェイ A6E の電源回路の構成は「図 14.1. 電源回路の構成」のとおりです。

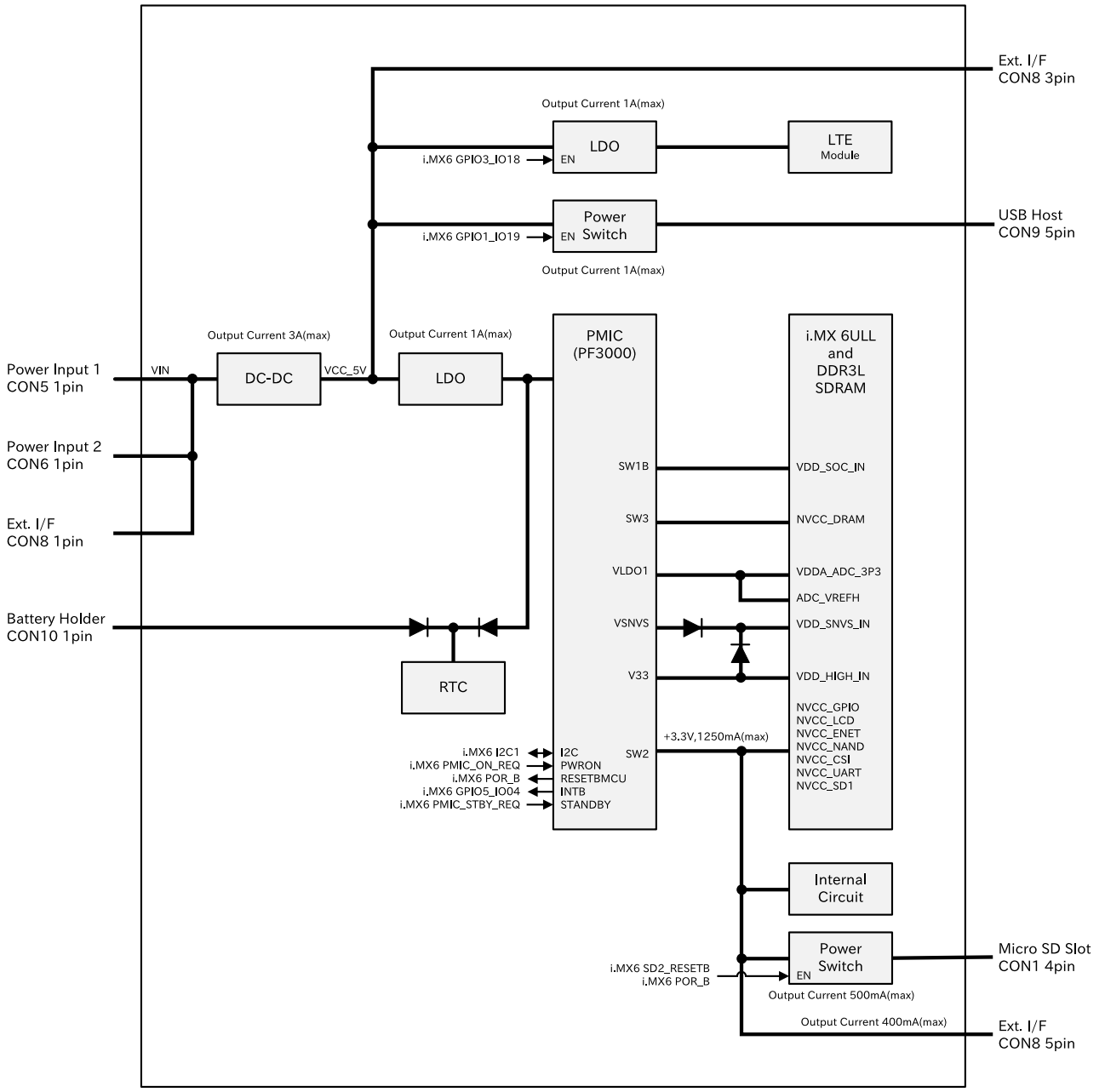


図 14.1 電源回路の構成

入力電圧(VIN)を電源 IC で各電圧に変換し、内部回路および各インターフェースに供給しています。各インターフェースやスイッチング・レギュレータの最大出力電流値を超えないように、外部機器の接続、供給電源の設計を行なってください。

外部インターフェースへの電源は GPIO によりオンオフ制御できるようになっており、不要な場合はオフすることで、省電力化が可能です。

14.2. インターフェース仕様

Armadillo-IoT ゲートウェイ A6E のインターフェース仕様について説明します。

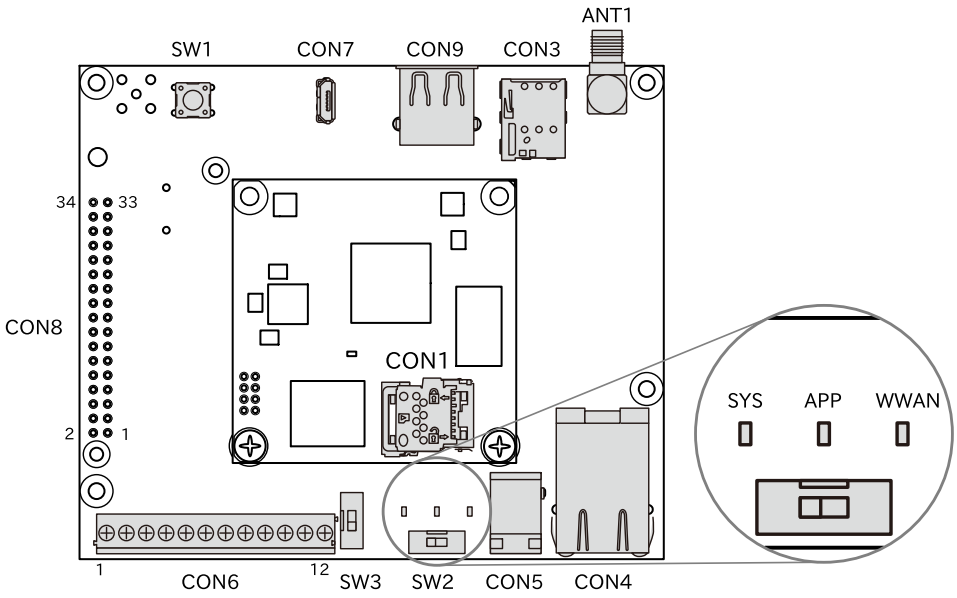


図 14.2 Armadillo-IoT ゲートウェイ A6E のインターフェース 表面

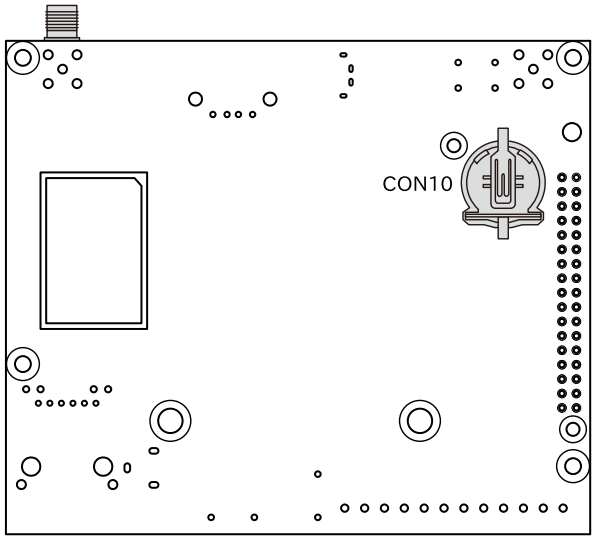


図 14.3 Armadillo-IoT ゲートウェイ A6E のインターフェース 裏面

表 14.7 Armadillo-IoT ゲートウェイ A6E インターフェース一覧

部品番号	インターフェース名	型番	メーカー
CON1	SD インターフェース	SDHK-8BNS-K-303-TB(HF)	J.S.T.Mfg.
CON3	nanoSIM インターフェース	SF72S006VBDR2500	Japan Aviation Electronics Industry
CON4	LAN インターフェース	08B0-1X1T-36-F	Bel Fuse Inc.
CON5	電源入力インターフェース	PJ-102AH	CUI
CON6	入出力インターフェース	1-1776275-2	TE Connectivity
CON7	USB コンソールインターフェース	ZX80-B-5P(30)	HIROSE ELECTRIC
CON8	拡張インターフェース	A1-34PA-2.54DSA(71)	HIROSE ELECTRIC
CON9	USB インターフェース	SS-52100-001	Bel Fuse Inc.

部品番号	インターフェース名	型番	メーカー
CON10	RTC バックアップインターフェース	BH-44C-5	Adam Tech
ANT1	LTE アンテナインターフェース 1	S-037-TGG	COSMTEC RESOURCES CO., LTD
SYS	システム LED	SML-D12M1WT86	ROHM
APP	アプリケーション LED	SML-D12M1WT86	ROHM
WWAN	ワイヤレス WAN LED	SML-D12M1WT86	ROHM
SW1	ユーザースイッチ	SKHHDJA010	ALPS ELECTRIC
SW2	起動デバイス設定スイッチ	DS01-254-S-01BE	CUI
SW3	RS485 終端抵抗設定スイッチ	DS01-254-S-01BE	CUI

「表 14.7. Armadillo-IoT ゲートウェイ A6E インターフェース一覧」には、搭載可能な代表型番を、部品の実装、未実装を問わず記載しており、実際に搭載されている部品と違う場合があります。お手元の製品に搭載されている部品型番や部品の実装、未実装の情報については、Armadillo サイト [https://armadillo.atmark-techno.com/]からダウンロードできる納入仕様書および変更履歴表にてご確認ください。

14.3. CON1(SD インターフェース)

CON1 はハイスピード(最大クロック周波数: 49.5MHz)に対応した SD インターフェースです。信号線は i.MX6ULL の SD ホストコントローラ(uSDHC2)に接続されています。

SD カードに供給される電源は i.MX6ULL の NAND_ALE ピン(GPIO4_IO10)で制御が可能です。High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

CON1 は活線挿抜に対応していません。microSD カードの挿抜は、電源を切断してから行ってください。

表 14.8 CON1 信号配列

ピン番号	ピン名	I/O	説明
1	DAT2	In/Out	SD データバス(bit2)、i.MX6ULL の NAND_DATA02 ピンに接続
2	CD/DAT3	In/Out	SD データバス(bit3)、i.MX6ULL の NAND_DATA03 ピンに接続
3	CMD	In/Out	SD コマンド/レスポンス、i.MX6ULL の NAND_WE_B ピンに接続
4	VDD	Power	電源(VCC_3.3V)
5	CLK	Out	SD クロック、i.MX6ULL の NAND_RE_B ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	SD データバス(bit0)、i.MX6ULL の NAND_DATA00 ピンに接続
8	DAT1	In/Out	SD データバス(bit1)、i.MX6ULL の NAND_DATA01 ピンに接続

14.3.1. microSD カードの挿抜方法

1. 上からカバーを軽く押し、約 1.2mm スライドさせて、ロックを解除します。

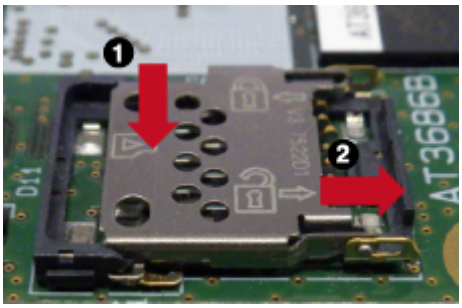


図 14.4 カバーのロックを解除する

2. カバーを開けます。

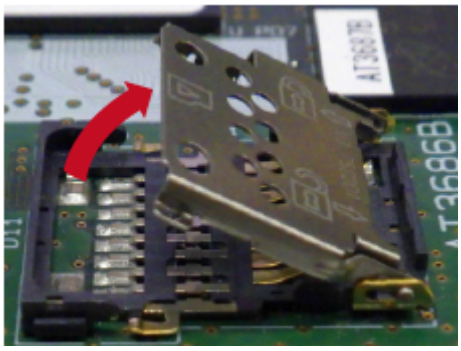


図 14.5 カバーを開ける

> カバーは過度な力で回転させたり、回転方向以外の方向へ力を加えると、破損の原因となりますので、ご注意ください。

3. 任意の角度までトレイを開いた状態で、microSD カードを挿抜します。

図 14.6 microSD カードの挿抜

microSD カード挿入方向については、カバーに刻印されているカードマークを目安にしてください。

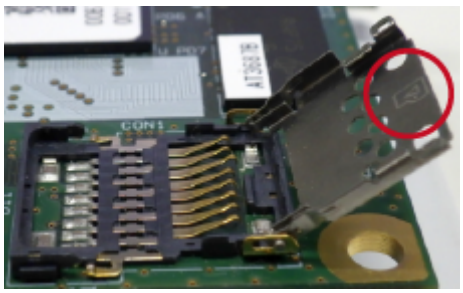


図 14.7 カードマークの確認

4. カバーを閉めます。

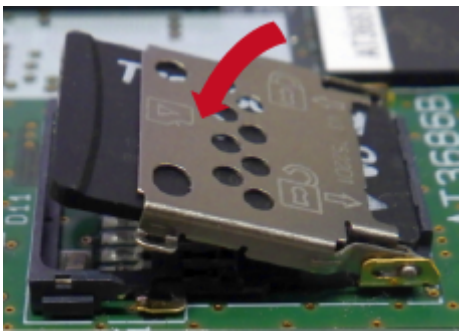


図 14.8 カバーを閉める

5. 上からカバーを軽く押し、約 1.2mm スライドさせて、ロックします。

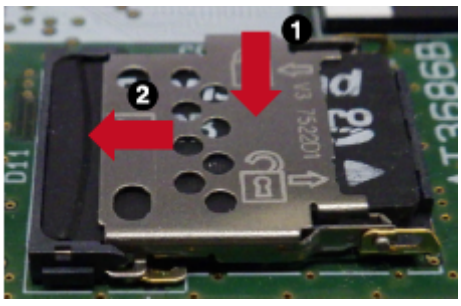


図 14.9 カバーをロックする

microSD カード装着後のカードの抜き取り手順は挿入時と同じです。

14.4. CON3(nanoSIM インターフェース)

CON3 は LTE データ通信時に利用する、nanoSIM カード用インターフェースです。

表 14.9 CON3 信号配列

ピン番号	ピン名	I/O	説明
C1	SIM_VCC	Power	SIM 電源、LTE モジュールの CCVCC に接続
C2	SIM_RST	Out	SIM リセット、LTE モジュールの CCRST に接続
C3	SIM_CLK	Out	SIM クロック、LTE モジュールの CCCLK に接続
C5	GND	Power	電源(GND)
C6	SIM_VPP	-	未接続
C7	SIM_I/O	In	SIM データ、LTE モジュールの CCIO に接続

14.5. CON4(LAN インターフェース)

CON4 は 10BASE-T/100BASE-TX に対応した LAN インターフェースです。カテゴリ 5 以上の Ethernet ケーブルを接続することができます。AUTO-MDIX 機能を搭載しており、ストレートケーブルまたはクロスケーブルを自動認識して送受信端子を切り替えます。

信号線は Ethenet PHY(LAN8720AI-CP/Microchip Technology) を経由して i.MX6ULL の Ethernet コントローラ(ENET1)に接続されています。

表 14.10 CON4 信号配列

ピン番号	ピン名	I/O	説明
1	TX+	In/Out	送信データ(+)
2	TX-	In/Out	送信データ(-)
3	RX+	In/Out	受信データ(+)
4	-	-	5 ピンと接続後に 75Ω 終端
5	-	-	4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	受信データ(-)
7	-	-	8 ピンと接続後に 75Ω 終端
8	-	-	7 ピンと接続後に 75Ω 終端

表 14.11 CON4 LAN LED の動作

名称(色)	状態	説明
LAN スピード LED(緑)	消灯	10Mbps で接続されている、もしくは Ethernet ケーブル未接続
	点灯	100Mbps で接続されている
LAN リンクアクティビティ LED(黄)	消灯	リンクが確立されていない
	点灯	リンクが確立されている
	点滅	リンクが確立されており、データを送受信している

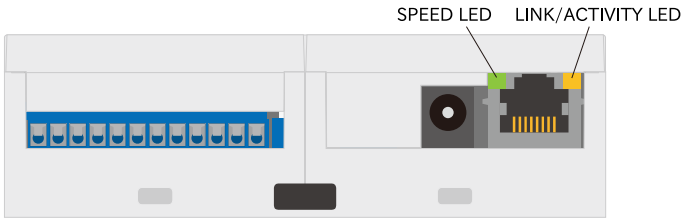


図 14.10 CON4 LAN LED

14.6. CON5(電源入力インターフェース)

CON5 は電源入力用のインターフェースです。DC ジャックが実装されており、「図 14.11. AC アダプタの極性マーク」と同じ極性マークのある AC アダプタが使用できます。対応プラグは内径 2.1mm、外形 5.5mm のものとなります。

図 14.11 AC アダプタの極性マーク

CON5、CON6 の電源ライン(VIN)は接続されており、同時に電源を供給することはできません。

CON5 から電源供給する場合、AC アダプタの DC プラグを DC ジャックに接続してから、AC プラグをコンセントに挿してください。

14.7. CON6(入出力インターフェース)

CON6 は以下の機能をもった入出力インターフェースです。

- 電源入力
- 接点入力 x 2
- 接点出力 x 2
- RS485(2 線式)

端子台を実装しています。接続可能な電線については、「表 14.13. CON6 接続可能な電線」をご確認ください。

表 14.12 CON6 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源入力(+)
2	GND	Power	電源入力(GND)
3	COM	In	接点入力プラスコモン
4	DI1	In	接点入力 1
5	DI2	In	接点入力 2
6	DO1A	-	接点出力 1A
7	DO1B	-	接点出力 1B
8	DO2A	-	接点出力 2A
9	DO2B	-	接点出力 2B
10	DATA+	In/Out	送受信データ(+)RS485 トランシーバの A/Y ピンに接続
11	DATA-	In/Out	送受信データ(-)RS485 トランシーバの B/Z ピンに接続
12	GND	Power	電源入力(GND)

表 14.13 CON6 接続可能な電線

規格	UL	IEC
電線範囲	26～18 AWG	0.12～0.9mm^2
被覆剥き長さ	5～6mm	
使用可能フェルール端子	型番：MFL25-5BE メーカー：ミスミ	
推奨ねじ締めトルク	0.28Nm	

電線の先端に予備半田しないでください。正しい接続ができなくなります。

端子台に電線を接続する際、端子台に過度な力を加えないでください。端子台が破損する恐れがあります。

14.7.1. 接点入力

接点入力部はフォトカプラによる絶縁入力(電流シンク出力)となっています。入力部を駆動するために電源は、外部から供給する必要があります。

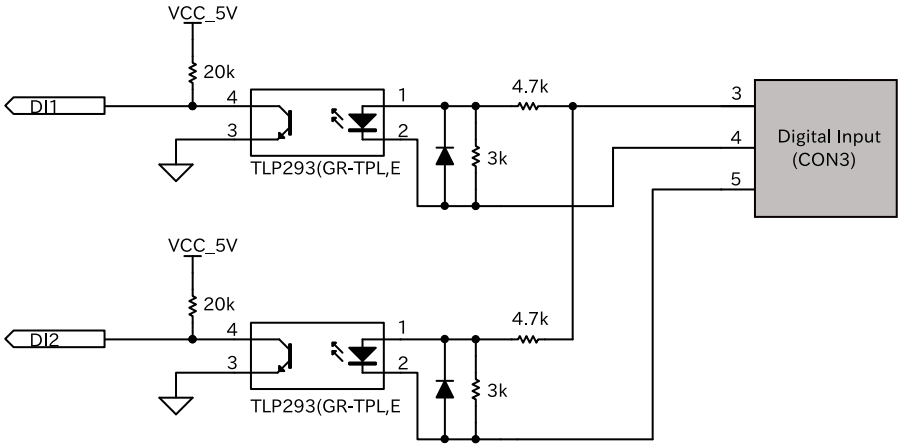


図 14.12 CON6 接点入力周辺回路

14.7.2. 接点出力

接点出力部はフォトリレーによる絶縁出力(無極性)となっています。出力部を駆動するためには外部に電源が必要となります。出力 1 点につき最大電流 500mA(定格 48V)まで駆動可能です。

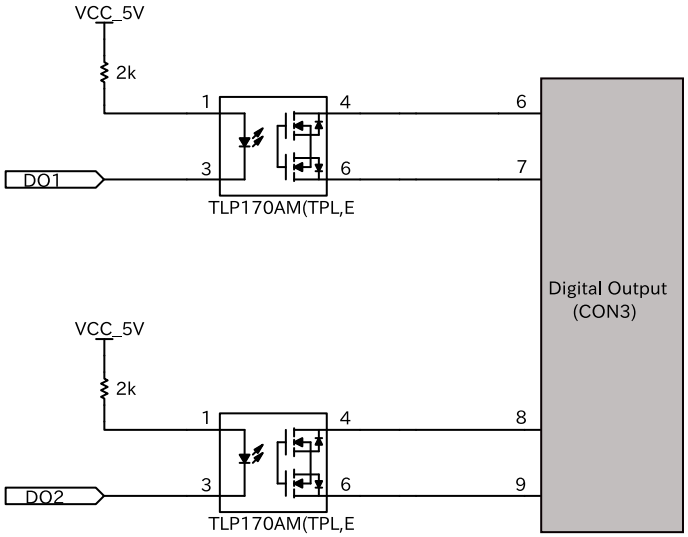


図 14.13 CON6 接点出力周辺回路

14.7.3. RS485

2 線式の RS485 インターフェースです。RS485 終端抵抗設定スイッチ(SW3)で終端抵抗(120Ω)の ON/OFF が可能です。

・ 最大データ転送レート：5Mbps

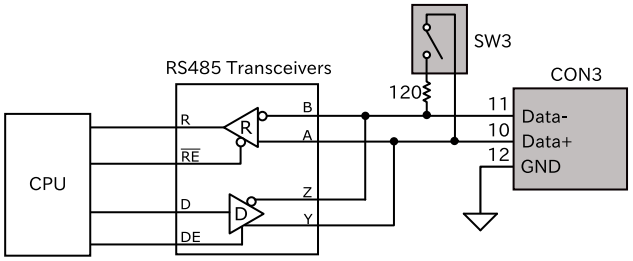


図 14.14 CON6 RS485 トランシーバ周辺回路

14.8. CON7(USB コンソールインターフェース)

CON7 は USB コンソール用インターフェースです。

信号線は USB シリアル変換 IC(CP2102N/Sillicon Labs)を経由して i.MX6ULL の UART コントローラ(UART3)に接続されています。

表 14.14 CON7 信号配列

ピン番号	ピン名	I/O	説明
1	VBUS_CNSL	Power	電源(VBUS_CNSL)
2	CNSL_USB_D-	In/Out	コンソール用 USB のマイナス側信号、USB シリアル変換 IC に接続
3	CNSL_USB_D+	In/Out	コンソール用 USB のプラス側信号、USB シリアル変換 IC に接続
4	CNSL_USB_ID	-	未接続
5	GND	Power	電源(GND)

14.9. CON8(拡張インターフェース)

CON8 は機能拡張用のインターフェースです。複数の機能(マルチプレクス)を持つ、i.MX6ULL の信号線が接続されており、GPIO、UART、I2C、SPI、CAN、PWM 等の機能を拡張することができます。

拡張できる機能の詳細につきましては、Armadillo サイト [https://armadillo.atmark-techno.com/]からダウンロードできる「Armadillo-IoT ゲートウェイ A6E マルチプレクス表」をご参照ください。

表 14.15 CON8 搭載コネクタと対向コネクタ例

名称	型番	メーカー	備考
搭載コネクタ	A1-34PA-2.54DSA(71)	HIROSE ELECTRIC	許容電流 3A(端子 1 本あたり)
対向コネクタ	6130xx21821 [a]	Wurth Elektronik	-

[a]xx にはピン数が入ります。

表 14.16 CON8 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源出力(VIN)
2	GND	Power	電源(GND)
3	VCC_5V	Power	電源出力(VCC_5V)
4	GND	Power	電源(GND)
5	VCC_3.3V	Power	電源出力(VCC_3.3V)
6	GND	Power	電源(GND)
7	GPIO1_IO01	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO01 ピンに接続
8	GPIO1_IO02	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO02 ピンに接続
9	GPIO1_IO03	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO03 ピンに接続
10	GPIO1_IO04	In/Out	拡張入出力、i.MX6ULL の GPIO1_IO04 ピンに接続
11	GND	Power	電源(GND)
12	GPIO1_IO20	In/Out	拡張入出力、i.MX6ULL の UART2_TX_DATA ピンに接続
13	GPIO1_IO21	In/Out	拡張入出力、i.MX6ULL の UART2_RX_DATA ピンに接続
14	GPIO3_IO05	In/Out	拡張入出力、i.MX6ULL の LCD_DATA00 ピンに接続、基板上で 10kΩ プルダウンされています。
15	GPIO3_IO06	In/Out	拡張入出力、i.MX6ULL の LCD_DATA01 ピンに接続、基板上で 10kΩ プルアップ(VCC_3.3V)されています。
16	GPIO3_IO07	In/Out	拡張入出力、i.MX6ULL の LCD_DATA02 ピンに接続、基板上で 10kΩ プルダウンされています。
17	GPIO3_IO08	In/Out	拡張入出力、i.MX6ULL の LCD_DATA03 ピンに接続、基板上で 10kΩ プルダウンされています。
18	GND	Power	電源(GND)
19	GPIO3_IO10	In/Out	拡張入出力、i.MX6ULL の LCD_DATA05 ピンに接続、起動デバイス設定スイッチ(SW2)が eMMC 側に設定されている時 10kΩ プルアップ(VCC_3.3V)、SD 側に設定されている時 10kΩ プルダウンされます。
20	GPIO3_IO11	In/Out	拡張入出力、i.MX6ULL の LCD_DATA06 ピンに接続、基板上で 10kΩ プルアップ(VCC_3.3V)されています。
21	GPIO3_IO12	In/Out	拡張入出力、i.MX6ULL の LCD_DATA07 ピンに接続、基板上で 10kΩ プルダウンされています。
22	GPIO3_IO13	In/Out	拡張入出力、i.MX6ULL の LCD_DATA08 ピンに接続、基板上で 10kΩ プルダウンされています。
23	GPIO3_IO14	In/Out	拡張入出力、i.MX6ULL の LCD_DATA09 ピンに接続、基板上で 10kΩ プルダウンされています。

ピン番号	ピン名	I/O	説明
24	GPIO3_IO15	In/Out	拡張入出力、i.MX6ULL の LCD_DATA10 ピンに接続、基板上で 10kΩ プルダウンされています。
25	GPIO3_IO16	In/Out	拡張入出力、i.MX6ULL の LCD_DATA11 ピンに接続、起動デバイス設定スイッチ(SW2)が eMMC 側に設定されている時 10kΩ プルダウン、SD 側に設定されている時 10kΩ プルアップ(VCC_3.3V)されます。
26	GPIO3_IO20	In/Out	拡張入出力、i.MX6ULL の LCD_DATA15 ピンに接続、基板上で 10kΩ プルダウンされています。
27	GPIO3_IO21	In/Out	拡張入出力、i.MX6ULL の LCD_DATA16 ピンに接続、基板上で 10kΩ プルダウンされています。
28	GPIO3_IO22	In/Out	拡張入出力、i.MX6ULL の LCD_DATA17 ピンに接続、基板上で 10kΩ プルダウンされています。
29	GPIO4_IO25	In/Out	拡張入出力、i.MX6ULL の CSI_DATA04 ピンに接続
30	GPIO4_IO26	In/Out	拡張入出力、i.MX6ULL の CSI_DATA05 ピンに接続
31	GPIO4_IO27	In/Out	拡張入出力、i.MX6ULL の CSI_DATA06 ピンに接続
32	GPIO4_IO28	In/Out	拡張入出力、i.MX6ULL の CSI_DATA07 ピンに接続
33	GND	Power	電源(GND)
34	GND	Power	電源(GND)

14.10. CON9(USB インターフェース)

CON9 は USB 2.0 に対応した USB インターフェースです。

信号線は i.MX6ULL の USB コントローラ(USB OTG1)に接続されています。

USB デバイスに供給される電源(USB_OTG1_VBUS)は i.MX6ULL の UART1_RTS_B ピン(GPIO1_IO19)で制御しており、High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

- データ転送モード
  - High Speed(480Mbps)
  - Full Speed(12Mbps)
  - Low Speed(1.5Mbps)

表 14.17 CON9 信号配列

ピン番号	ピン名	I/O	説明
1	USB1_VBUS	Power	電源(USB_OTG1_VBUS)、i.MX6ULL の USB_OTG1_VBUS ピンに接続
2	USB1_DN	In/Out	USB1 のマイナス側信号、i.MX6ULL の USB_OTG1_DN ピンに接続
3	USB1_DP	In/Out	USB1 のプラス側信号、i.MX6ULL の USB_OTG1_DP ピンに接続
4	GND	Power	電源(GND)

14.11. CON10(RTC バックアップインターフェース)

CON10 はリアルタイムクロックのバックアップ用インターフェースです。長時間電源が切断されても時刻データを保持させたい場合にご使用ください。

CON10 には CR1220、BR1220 等の電池を接続することができます。リアルタイムクロックの時刻保持時の平均消費電流は、データシート上、240nA(Typ.)です。

温度補償タイプのリアルタイムクロックを実装しており、平均月差は周囲温度-20℃～60℃で 8 秒(参考値)です。

RTC の時間精度は周囲温度に大きく影響を受けますので、使用温度での十分な特性の確認をお願いいたします。

電池を電池ホルダーに装着する際は、異物の挟み込みや不完全な装着がないか、十分な確認をお願いいたします。

表 14.18 CON10 信号配列

ピン番号	ピン名	I/O	説明
1	RTC_BAT	Power	リアルタイムクロックのバックアップ用電源入力(RTC_BAT)
2	GND	Power	電源(GND)

14.12. ANT1(LTE アンテナインターフェース 1)

ANT1 は LTE データ通信時に利用する、アンテナコネクタです。SMA オス端子のアンテナを接続することができます。アンテナコネクタの形状は「図 14.15. ANT1 接続可能なアンテナコネクタ形状」のとおりです。

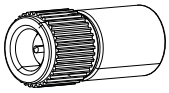


図 14.15 ANT1 接続可能なアンテナコネクタ形状

LTE モジュールメーカーにより、技適認証取得済みのアンテナについて抜粋したリストを Armadillo サイト [https://armadillo.atmark-techno.com/]で公開しています。付属のアンテナ以外をご検討の際に、ご活用ください。

当社にて全てのアンテナの動作を確認したものではありませんので、通信性能の評価については、ユーザー様自身にて実施いただくようお願いいたします。

14.13. SYS、APP、WWAN(LED)

SYS、APP、WWAN は電源の入力状態やシステム状態、アプリケーション状態を表示する LED です。LED の動作仕様については、「表 14.20. LED 状態と製品状態の対応について」をご確認ください。

表 14.19 LED 信号配列

部品番号	名称(色)	説明
SYS	システム LED(緑)	電源(VCC_3.3V)の入力状態を表示、i.MX6ULL の UART2_CTS_B ピン (GPIO1_IO22)に接続 (Low: 消灯、High: 点灯)

部品番号	名称(色)	説明
APP	アプリケーション LED(緑)	アプリケーションの状態を表示、i.MX6ULL の UART2_RTS_B ピン(GPIO1_IO23)に接続 (Low: 消灯、High: 点灯)
WWAN	ワイヤレス WAN LED(緑)	LTE 通信の状態を表示、i.MX6ULL の UART1_RX_DATA ピン(GPIO1_IO17)に接続 (Low: 消灯、High: 点灯)

表 14.20 LED 状態と製品状態の対応について

LED 状態\LED 名称	SYS	APP	WWAN
OFF	電源 OFF	アプリ起動不可	SIM 未検出または認識中、または LTE モデム未検出
ON	電源 ON	アプリ起動可能	LTE 接続済み
Blink Slow	シャットダウン中	アプリ起動完了	SIM 検出、LTE 未接続 [a]
Blink Fast	アップデート中	アプリエラー	SIM 検出、LTE 未接続、電波品質が低い [a]

[a]LTE コネクションが未作成、設定間違いの場合もこの状態となります

14.14. SW1(ユーザースイッチ)

SW1 はユーザー側で自由に利用できる押しボタンスイッチです。

表 14.21 SW1 信号配列

部品番号	名称	説明
SW1	ユーザースイッチ	i.MX6ULL の JTAG_MOD ピンに接続 (Low: 押されていない状態、High: 押された状態)

14.15. SW2(起動デバイス設定スイッチ)

SW2 は起動デバイス設定スイッチです。SW2 を操作することで、起動デバイスを設定することができます。

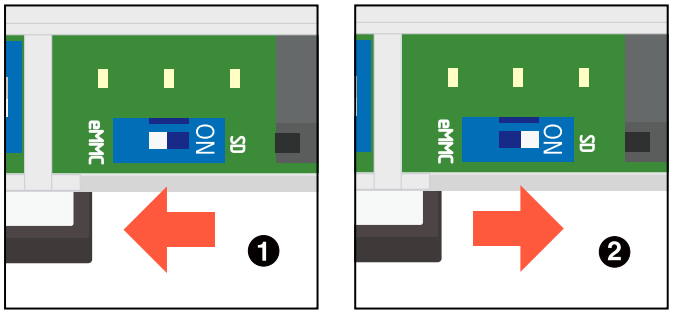


図 14.16 スイッチの状態と起動デバイス

❶ 起動デバイスは eMMC になります。

❷ 起動デバイスは microSD になります。

14.16. SW3(RS485 終端抵抗設定スイッチ)

SW3 は RS485 の終端抵抗設定スイッチです。SW3 を操作することで、終端抵抗 120Ω の ON/OFF を切り替えることができます。

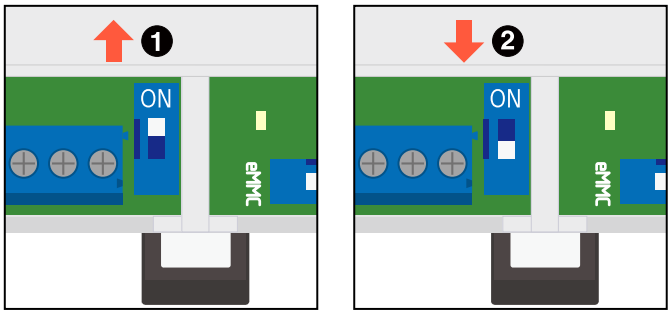


図 14.17 スイッチの状態と終端抵抗の ON/OFF

❶ 終端抵抗 120Ω が ON になります。

❷ 終端抵抗 120Ω が OFF になります。

終端は RS485 の信号線の最遠端で行います。Armadillo-IoT A6E が最遠端になる場合は終端抵抗を ON にしてください。

14.17. 形状図

14.17.1. 筐体形状図

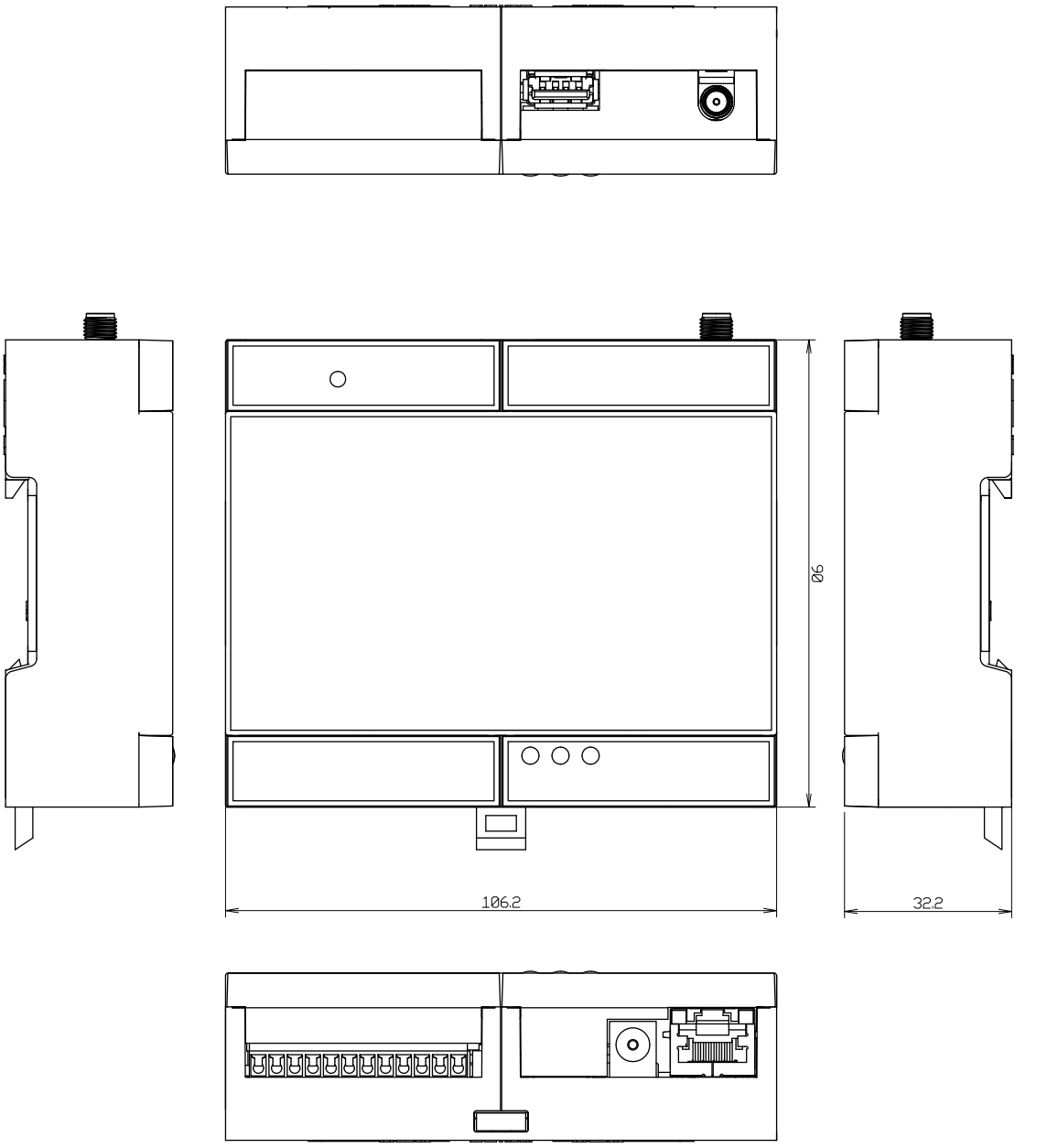


図 14.18 筐体形状

14.17.2. 基板形状図

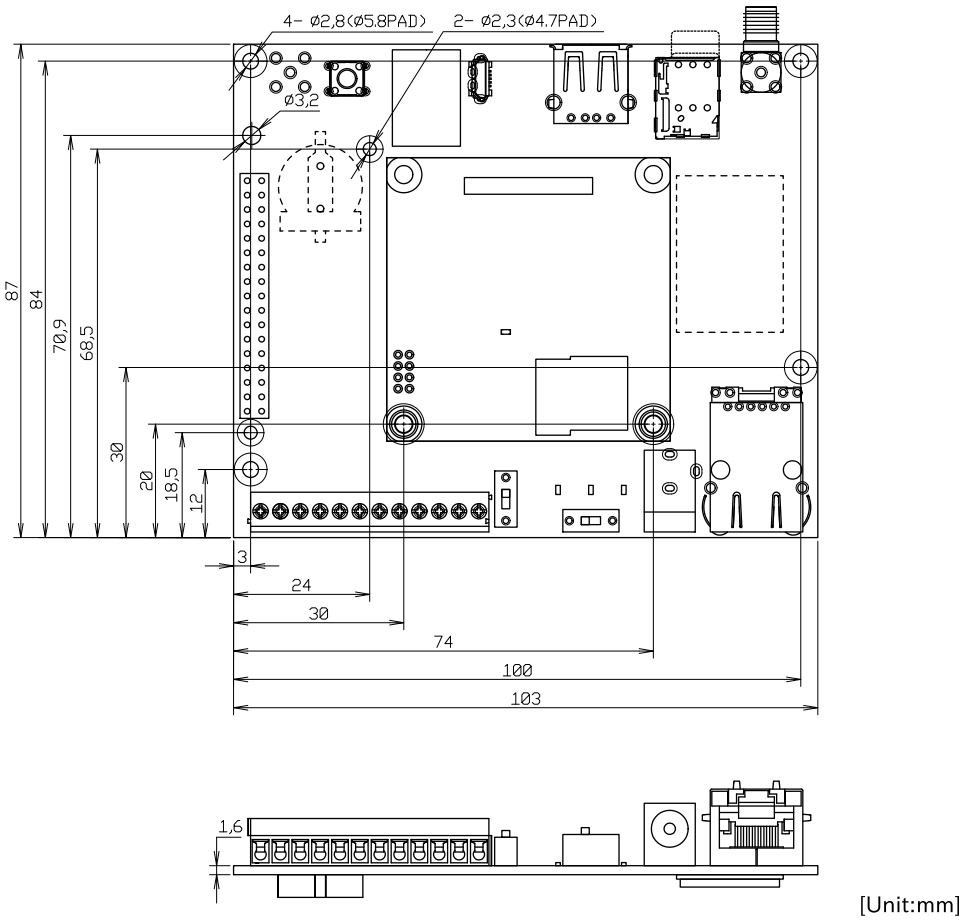


図 14.19 基板形状および固定穴寸法

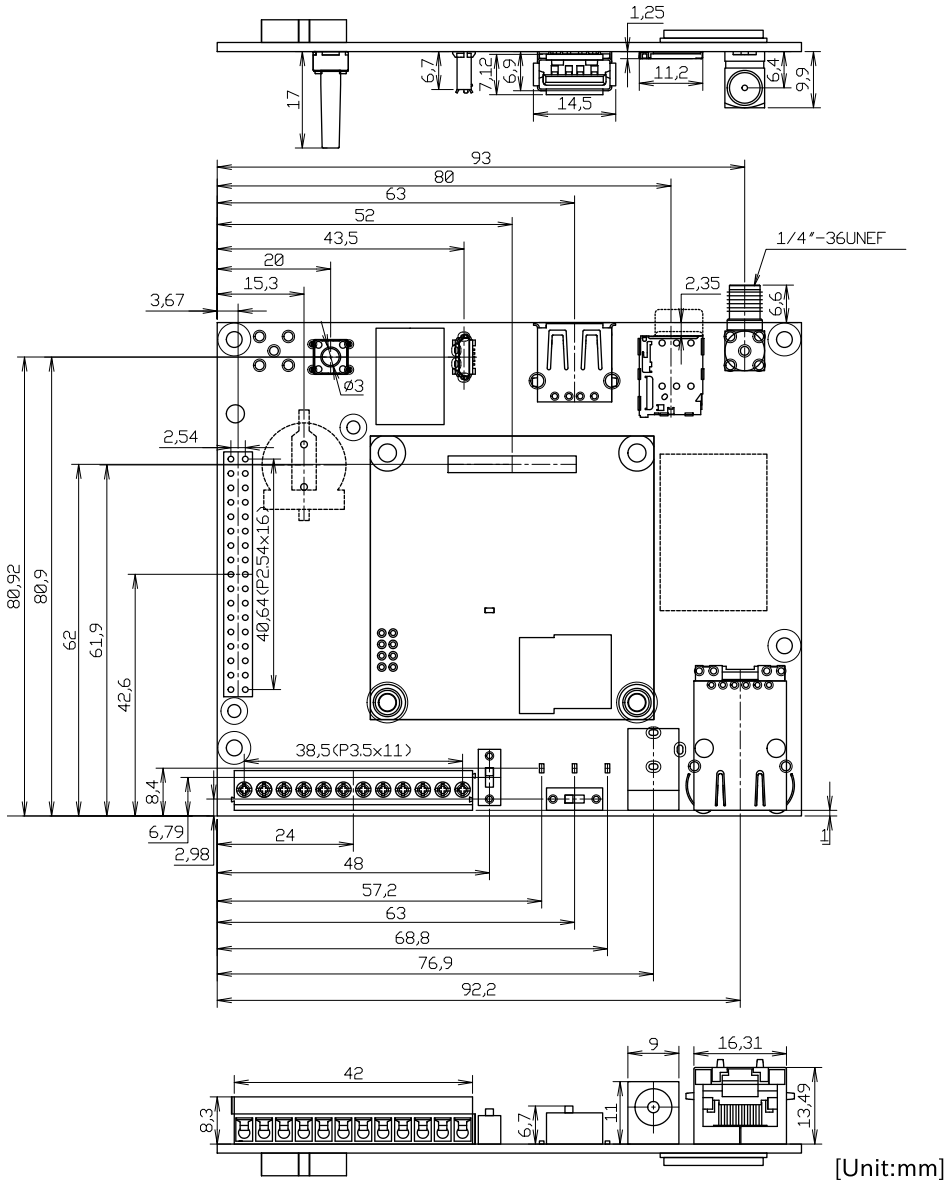


図 14.20 コネクタ、スイッチ、LED 位置

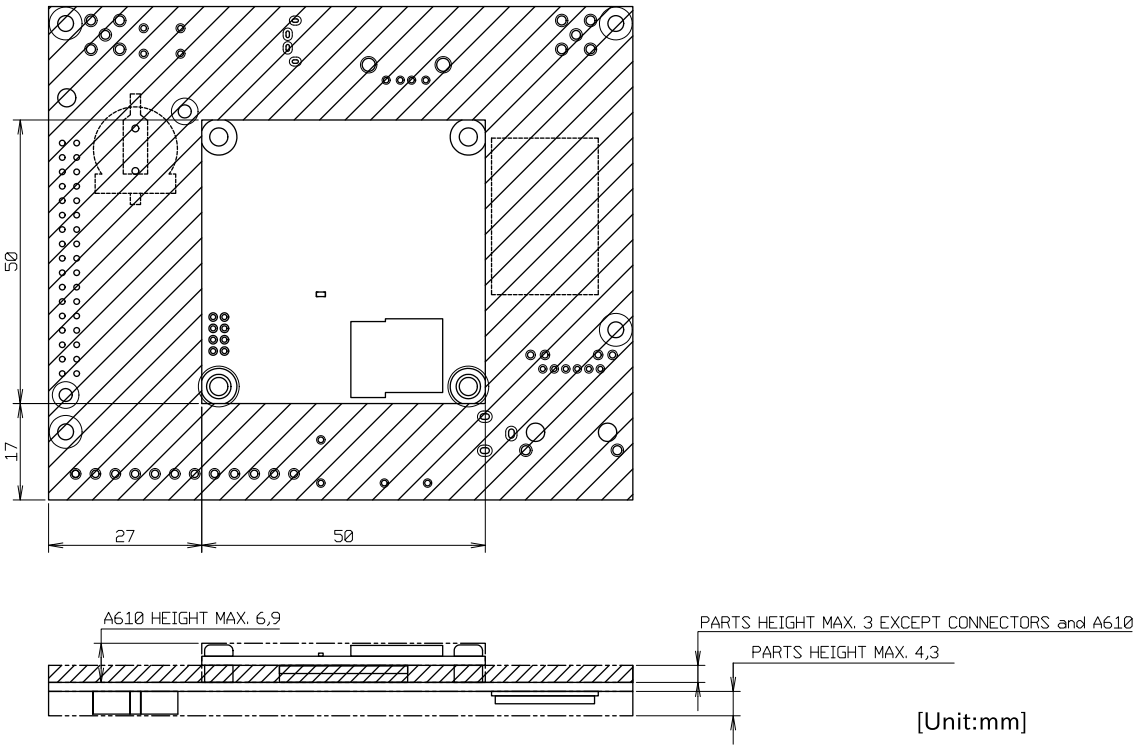
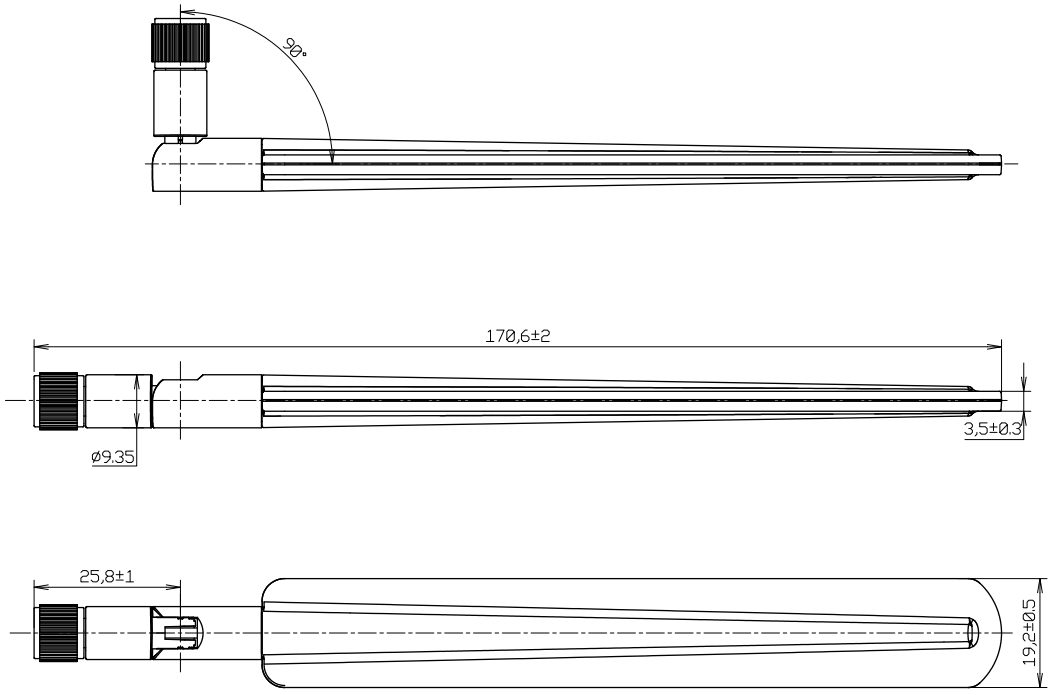

50
17
27
50
A610 HEIGHT MAX. 6,9
PARTS HEIGHT MAX. 3 EXCEPT CONNECTORS and A610
PARTS HEIGHT MAX. 4,3
[Unit:mm]

図 14.21 部品高さ

14.17.3. アンテナ形状図



[Unit:mm]

図 14.22 アンテナ形状

基板改版や部品変更により、基板上の部品位置、高さは変更になることがあります。ケースや拡張基板を設計する場合、ある程度の余裕をもった寸法での設計をお願いいたします。

DXF 形式の形状図を「アットマークテクノ Armadillo サイト」から「購入者向けの限定公開データ」としてダウンロード可能です。

14.18. 組み立てと分解

本製品はねじを使用しないスナップフィット方式を採用しており、容易に組み立てと分解が可能です。分解する際には手のけがやパーツの破損を防止するためマイナスドライバーなどの工具を使用してください。

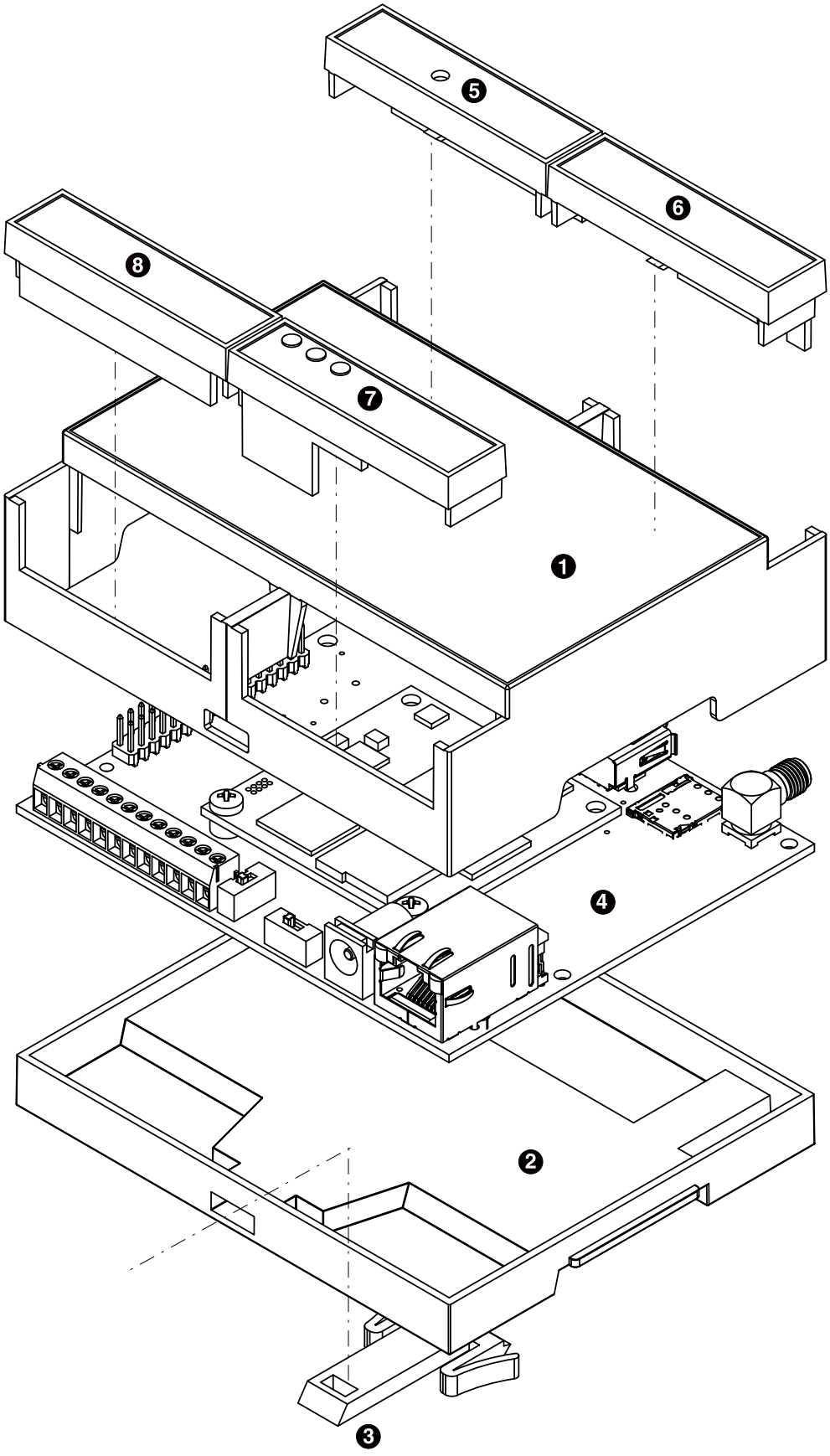


図 14.23 ケースモデル展開図

表 14.22 ケースモデル展開図パーツ一覧

番号	名称	説明
1	ケーストップ	ケース上側のパーツです。 ケースボトムとは 4 か所のツメで固定されます。 ケースを分解する際は、マイナスドライバーを使用してツメを破損させないよう慎重に取り外してください。
2	ケースボトム	ケース下側のパーツです。
3	フック	ケースを DIN レールに固定するためのパーツです。
4	基板	
5	カバーパーツ A	ケース開口部のカバーです。 ケーストップとは 1 か所のツメで固定されます。
6	カバーパーツ B	ケース開口部のカバーです。
7	カバーパーツ C	ケース開口部のカバーです。 LED のライトパイプ 3 本が装着されています。 強い衝撃を加えた場合、ライトパイプが外れる場合があります。 ライトパイプが外れた場合は、カバー丸穴に差し込みなおしてください。
8	カバーパーツ D	ケース開口部のカバーです。

フックは以下の図を参考に取り付けてください。

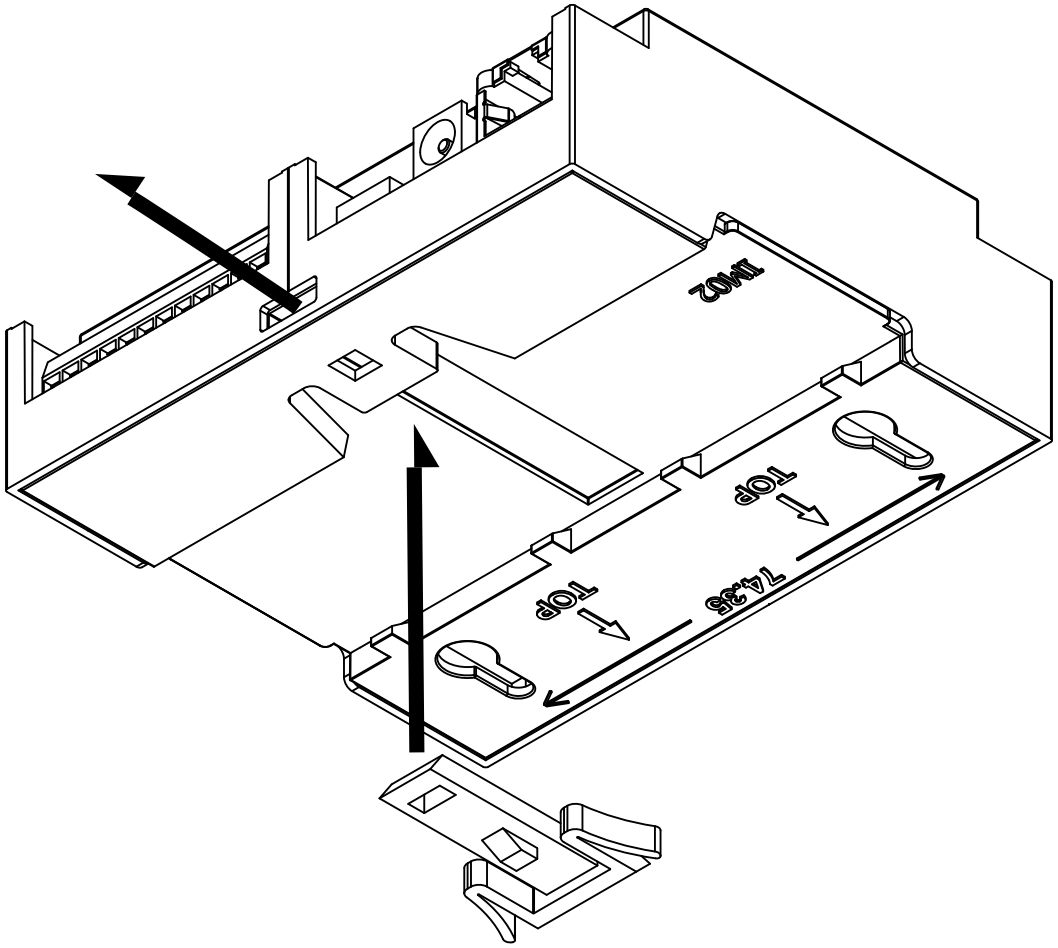


図 14.24 フック取り付け 1

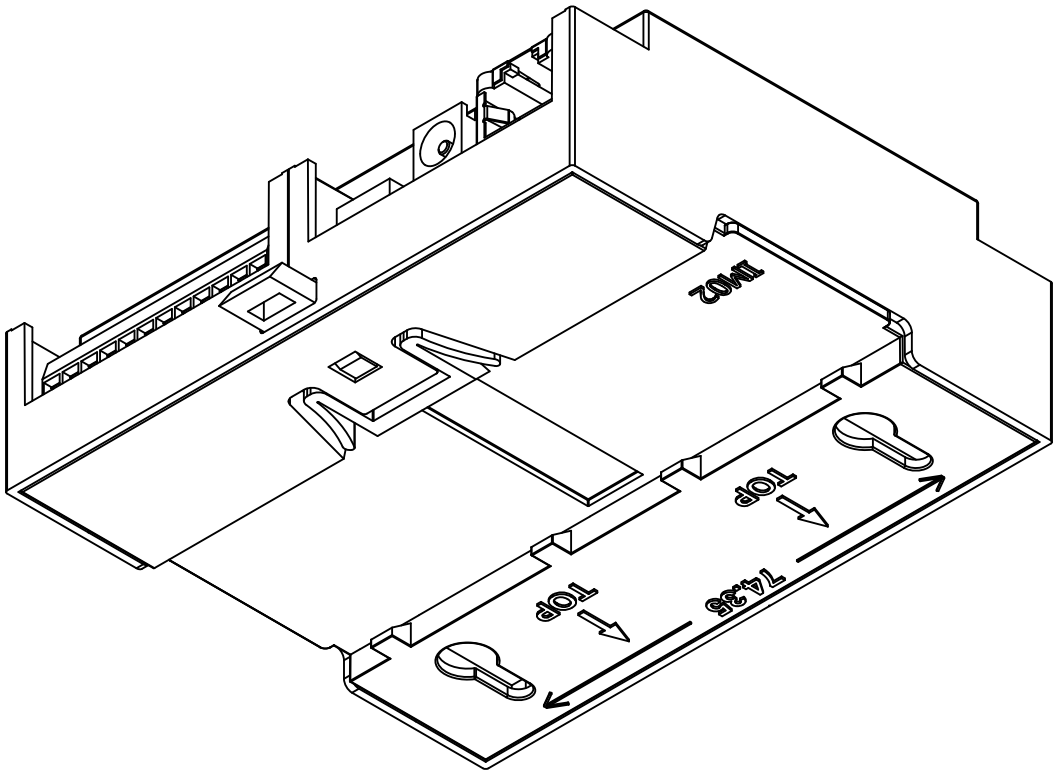


図 14.25 フック取り付け 2

14.18.1. ケースの組み立て手順

microSD カードの取り付けは、ケースの組み立て前に行う必要があります。取り付け手順については、を参照してください。

以下の手順に従い、ケースを組み立ててください。

1. 基板をケーストップに入れる
2. ケースボトムをケーストップにはめ込み、基板を固定する
3. フックをケースボトムにはめ込む
4. カバーパーツをケーストップにはめ込む

14.18.2. ケースの分解

ツメに強い力を加えますと破損する恐れがありますので、十分ご注意ください。

マイナスドライバーなどの工具を用意してください。以下の手順に従い、ケースを分解してください。

1. フックをケースボトムから取り外す
2. ケースボトムを取り外す
3. 基板を取り外す
4. カバーパーツを取り外す

フックはツメで固定されていますので、「図 14.26. フックのツメ」を参考にツメを押しながらフックを引き出してください。

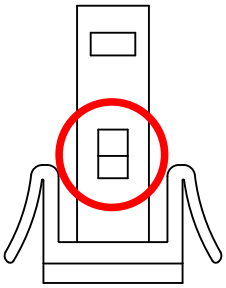


図 14.26 フックのツメ

ケースボトムはツメ 4 か所で固定されていますので、「図 14.27. ケースボトムのツメ」を参考にマイナスドライバーをケースの隙間に差し込み順に外してください。

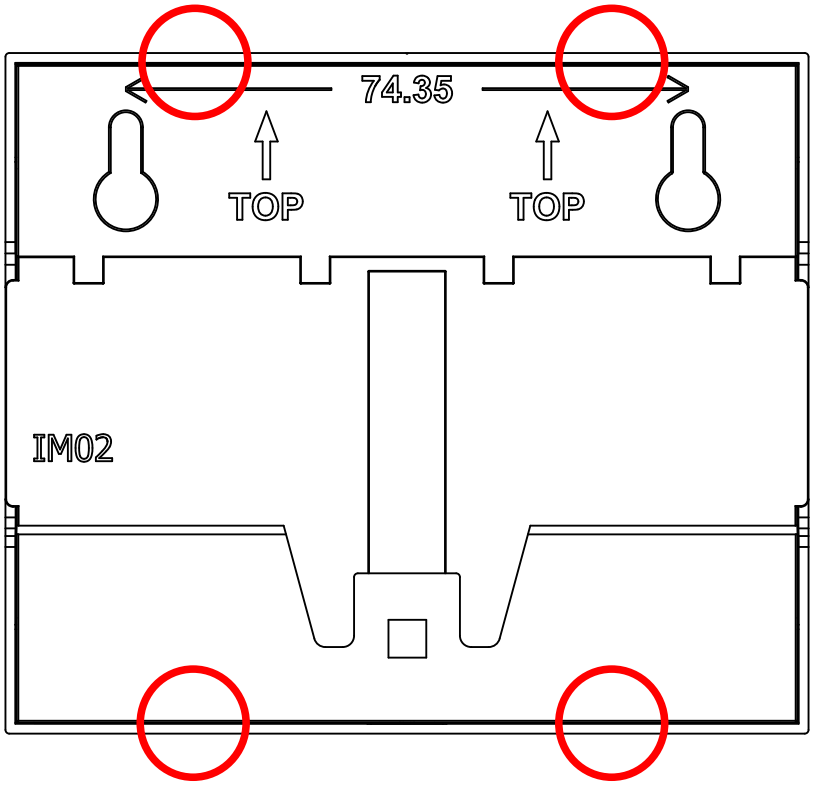


図 14.27 ケースボトムのツメ

基板はアンテナコネクタがケース開口部より飛び出しているため、反対側の LAN コネクタ側から先にケーストップから出すようにしてください。基板を取り外す際、LAN コネクタの突起部がケーストップに当たらないよう、ケースを広げながら基板を取り外すようにしてください。

カバーはツメ 1 か所でケーストップに固定されています。「図 14.28. カバーのツメ」を参考にマイナスドライバーをケースの隙間に差し込み外してください。

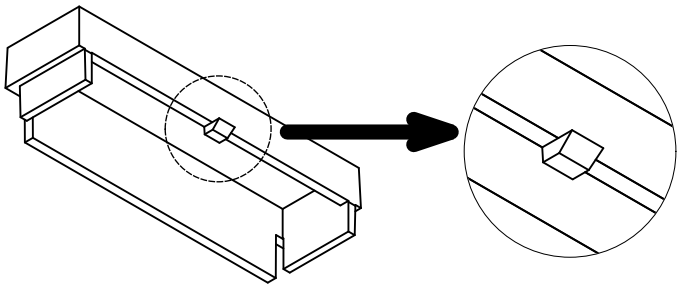


図 14.28 カバーのツメ

14.19. 設計情報

本章では、Armadillo-IoT ゲートウェイ A6E の機能拡張や信頼性向上のための設計情報について説明します。

14.19.1. 信頼性試験データについて

Armadillo-IoT ゲートウェイ A6E の各種信頼性試験データを、「アットマークテクノ Armadillo サイト」から「購入者向けの限定公開データ」としてダウンロード可能ですのでご確認ください。

14.19.2. 放射ノイズ

CON8(拡張インターフェース)を使用して、Armadillo-IoT ゲートウェイ A6E と拡張基板を接続すると、放射ノイズが問題になる場合があります。放射ノイズを減らすために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6E の GND(固定穴等)と拡張基板の GND を太い導線や金属スペーサ等で接続する
- 未使用の拡張ピンは Low レベル出力とする
- 使用する拡張ピンはコンデンサ(1000pF 程度)を介して GND と接続する
- ハーネスケーブル等で拡張する場合は、最短で接続する。
- シールド付きのケーブルを使用する
  - 長さが余る場合は、ケーブルを折りたたむ
  - シールドは拡張基板の GND に接続する

14.19.3. ESD/雷サージ

Armadillo-IoT ゲートウェイ A6E の ESD 耐性を向上させるために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6E を金属筐体に組み込み、GND(固定穴等)を金属ねじ等で接続する
- 金属筐体を接地する

また、Armadillo-IoT ゲートウェイ A6E に接続されたケーブルが屋外に露出するような設置環境では、ケーブルに侵入した雷サージ等のストレスによりインターフェース回路が破壊される場合があります。ストレスへの耐性を向上させるために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6E と通信対向機の GND 接続を強化する
- シールド付きのケーブルを使用する

14.19.4. 拡張ボードの設計

14.19.4.1. 基板形状

Armadillo-IoT ゲートウェイ A6E の拡張ボードを設計する際の推奨形状は「図 14.29. 拡張ボード例」のとおりです。拡張ボード側にピンソケットを実装して Armadillo-IoT ゲートウェイ A6E と接続します。

一般的なピンソケットを実装した場合、嵌合高さは約 11mm となります。

拡張ボード固定用に、φ2.3mm の穴を 2 箇所用意しており、M2 のスペーサーとねじを接続可能です。

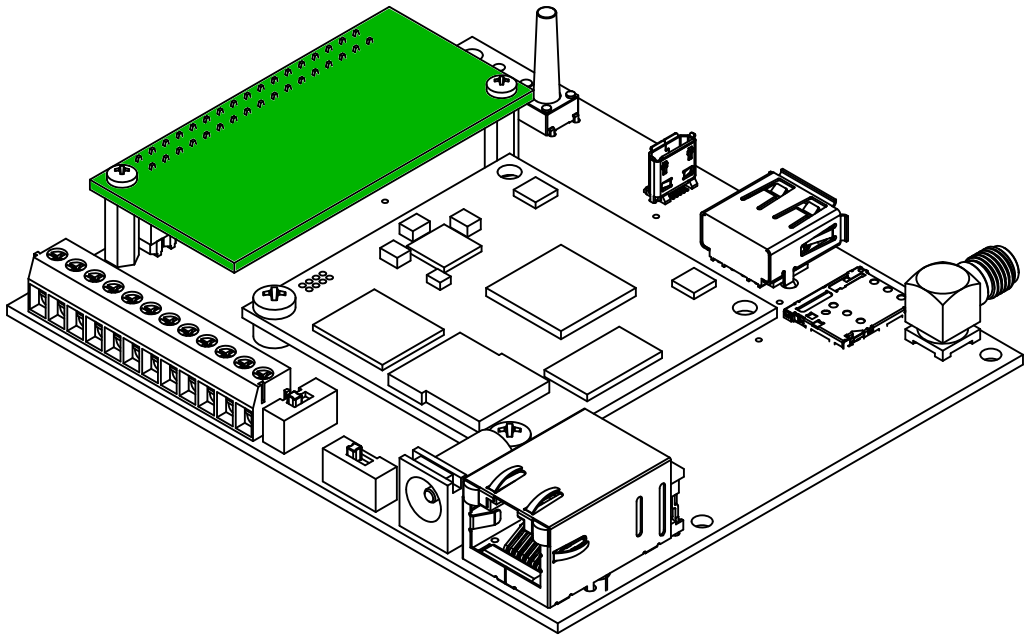


図 14.29 拡張ボード例

14.20. オプション品

本章では、Armadillo-IoT ゲートウェイ A6E のオプション品について説明します。

.Armadillo-IoT ゲートウェイ A6E 関連のオプション品

名称	型番
AC アダプタ（12V/2.0A φ2.1mm）温度拡張品 効率レベル VI 品	OP-AC12V4-00

Armadillo-IoT ゲートウェイ A6E 製品マニュアル

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2022/10/28	・初版発行
1.0.1	2022/11/04	・「10.2. ゲートウェイコンテナを動かす」及び「10.5. コンテナの運用」において、コンテナイメージのアーキテクチャ指定が誤っていたのを修正 ・その他、誤記・分かりにくい記載の修正
1.1.0	2022/11/28	・「4.1. 準備するもの」「7.1.5. LTE (Cat.M1 モデルのみ)」「3G/LTE」の記述を「LTE」に修正 ・「7.1.5.4. MCC/MNC を指定した LTE のコネクションを作成する」 追加 ・「7.1.5.10. LTE 再接続サービス」 設定ファイルのパスを修正 ・「9.4.1.3. ゲートウェイコンテナイメージを改変する」 に改変した場合の注意事項を追加 ・「表 10.1. [DEFAULT] 設定可能パラメータ」 データ送信間隔とコンテナ終了までの待ち時間の設定可能値を修正 ・「表 10.4. [DI1,DI2] 設定可能パラメータ」 DI 動作種別の設定値に none を追加 ・「10.2.5.2. 接続先の クラウド 情報の設定」 AWS IoT Core へデバイス登録完了後にクリアされるパラメーターを明記 ・「10.2.6.1. Armadillo からクラウドに送信するデータ」 に「表 10.12. CPU 温度データー覧」 追加 ・「10.2.9. ログ内容確認」 ログフォーマットの説明追加 ・「10.3. ゲートウェイコンテナを拡張する」 実行コードのファイルパスを修正 ・「10.3. ゲートウェイコンテナを拡張する」 参考サイトの URL を追加 ・「10.7.1. ブートディスクの作成」「10.8.1.1. 初期化インストールディスクの作成」 「10.9.5. mkswu の desc ファイル」 ブートローダーの名称を修正 ・「10.11. Device Tree をカスタマイズする」 内容を記載 ・その他、誤記・分かりにくい記載の修正
1.2.0	2022/12/26	・誤記修正 ・「8.5. 状態遷移トリガにコンテナ終了通知を利用する」に、コンテナ終了契機を記載 ・「10.7.3. ゲートウェイコンテナのインストール」を追加 ・ゲートウェイコンテナ Azure に関する記述を追加 ・「10.6.2. Linux カーネルをビルドする」の、説明内容を修正 ・「10.2. ゲートウェイコンテナを動かす」の、AWS クラウド構築時に使用するファイルの名称を修正
1.3.0	2023/01/30	・「10.1. ゲートウェイアプリケーションを開発する」を追加 ・「10.2.5. 設定ファイルの編集」の、章タイトルを変更 ・「10.10.4. ボタンやキーを扱う」を追加 ・「10.10.7. Network Time Protocol (NTP, ネットワーク・タイム・プロトコル)」を追加 ・「10.14. 動作中の Armadillo の温度を測定する」を追加 ・誤記修正

Armadillo-IoT ゲートウェイ A6E 製品マニュアル
Version 1.3.0
2023/01/27