Armadillo-IoT ゲートウェイ G3L
製品マニュアル

AGL3000-D10Z
AGL3000-C10Z
AGL3000-C11Z

Version 1.0.2
2016/12/20

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

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

Armadillo-IoT ゲートウェイ G3L 製品マニュアル

株式会社アットマークテクノ

Version 1.0.2
2016/12/20

目次

1. はじめに 13
1.1. 本書で扱うこと扱わないこと 13
1.1.1. 扱うこと 13
1.1.2. 扱わないこと 14
1.2. 本書で必要となる知識と想定する読者 14
1.3. ユーザー限定コンテンツ 14
1.4. 本書および関連ファイルのバージョンについて 14
1.5. 本書の構成 14
1.6. 表記について 15
1.6.1. フォント 15
1.6.2. コマンド入力例 15
1.6.3. アイコン 16
1.7. 謝辞 16
2. 注意事項 17
2.1. 安全に関する注意事項 17
2.2. 取扱い上の注意事項 18
2.3. ソフトウェア使用に関しての注意事項 19
2.4. 書込み禁止領域について 20
2.5. 電波障害について 20
2.6. 保証について 20
2.7. 輸出について 20
2.8. 商標について 21
2.9. 無線モジュールの安全規制について 21
3. 製品概要 23
3.1. 製品の特長 23
3.1.1. Armadillo とは 23
3.1.2. Armadillo-IoT ゲートウェイ G3L とは 23
3.2. 製品ラインアップ 24
3.2.1. Armadillo-IoT ゲートウェイ G3L 開発セット 24
3.2.2. Armadillo-IoT ゲートウェイ G3L 量産用 24
3.3. 仕様 24
3.4. Armadillo-IoT ゲートウェイ G3L の外観 26
3.5. ブロック図 27
3.6. ソフトウェア構成 28
4. Armadillo の電源を入れる前に 30
4.1. 準備するもの 30
4.2. 組立手順 30
4.2.1. 概要 30
4.2.2. 標準筐体の組立手順 30
4.2.3. 各種取付 33
4.3. 開発/動作確認環境の構築 33
4.3.1. ATDE6 セットアップ 34
4.3.1.1. VMware のインストール 34
4.3.1.2. ATDE6 アーカイブの取得 34
4.3.1.3. ATDE6 アーカイブの展開 35
4.3.1.4. ATDE6 の起動 38
4.3.2. 取り外し可能デバイスの使用 38
4.3.3. コマンドライン端末(GNOME 端末)の起動 38
4.3.4. シリアル通信ソフトウェア(minicom)の使用 39
4.4. インターフェースレイアウト 41

4.5. 接続方法 43
4.6. スライドスイッチの設定について 45
4.7. vi エディタの使用方法 45
4.7.1. vi の起動 45
4.7.2. 文字の入力 46
4.7.3. カーソルの移動 46
4.7.4. 文字の削除 47
4.7.5. 保存と終了 47
5. 起動と終了 48
5.1. 起動 48
5.2. ログイン 60
5.3. 時刻の設定 61
5.4. debian のユーザを管理する 62
5.5. 終了方法 63
6. ソフトウェアの更新と初期化 66
6.1. ソフトウェアを最新の状態に更新する 66
6.1.1. ブートローダーイメージの更新 66
6.1.2. Linux カーネルイメージの更新 67
6.1.3. DTB の更新 67
6.1.4. Debian GNU/Linux ルートファイルシステムの更新 67
6.1.4.1. Debian パッケージのアップデート 67
6.1.4.2. ルートファイルシステムの書き換え 68
6.2. ソフトウェアを初期化する 69
6.2.1. インストールディスクの作成 69
6.2.2. インストールの実行 69
7. 動作確認方法 71
7.1. 動作確認を行う前に 71
7.2. ネットワーク 71
7.2.1. 接続可能なネットワーク 71
7.2.2. ネットワークの設定方法 71
7.2.2.1. nmcli について 71
7.2.3. nmcli の基本的な使い方 72
7.2.3.1. コネクションの一覧 72
7.2.3.2. コネクションの有効化・無効化 72
7.2.3.3. コネクションの作成 73
7.2.3.4. コネクションの削除 73
7.2.3.5. コネクションを修正する 73
7.2.3.6. コネクションの修正を反映する 74
7.2.3.7. デバイスの一覧 75
7.2.3.8. デバイスの接続 75
7.2.3.9. デバイスの切断 75
7.2.4. 有線 LAN 76
7.2.4.1. 有線 LAN インターフェース(eth0)のコネクションの作成 76
7.2.4.2. 有線 LAN のネットワーク設定を変更する 76
7.2.4.3. 有線 LAN の接続を確認する 76
7.2.5. 無線 LAN 76
7.2.5.1. 無線 LAN アクセスポイントに接続する 77
7.2.5.2. 無線 LAN(wlan0)のコネクションの作成 77
7.2.5.3. 無線 LAN のネットワーク設定を変更する 77
7.2.5.4. 無線 LAN の接続を確認する 77
7.2.6. LTE 78
7.2.6.1. LTE データ通信設定を行う前に 78
7.2.6.2. LTE のコネクションを作成する 79

7.2.6.3. データ接続状態の確認 79
7.2.6.4. LTE で通信確認をする 80
7.2.6.5. LTE のデータ通信を終了する 80
7.2.6.6. LTE のデータ接続を再開する 80
7.2.6.7. LTE 再接続サービス 81
7.2.6.8. ModemManager について 81
7.2.7. NetworkManager による設定例 83
7.2.7.1. ネットワーク設定手順 84
7.2.8. ファイアーウォール 87
7.2.9. ネットワークアプリケーション 87
7.2.9.1. HTTP サーバー 87
7.2.10. Armadillo にファイルを転送する 88
7.2.10.1. scp コマンドを使って転送する 88
7.2.11. Wi-SUN 89
7.2.11.1. 設定情報を取得する 89
7.3. ストレージ 90
7.3.1. ストレージの使用方法 90
7.3.2. ストレージのパーティション変更とフォーマット 91
7.4. LED 92
7.4.1. 初期出荷状態の LED の動作 93
7.4.2. LED を点灯/消灯する 93
7.4.3. トリガを使用する 94
7.5. RTC 94
7.5.1. RTC に時刻を設定する 95
7.6. ユーザースイッチ 95
7.6.1. イベントを確認する 95
7.7. AD コンバーター 96
7.7.1. 電圧を取得する 96
7.8. RS422/RS485 97
7.8.1. RS422/RS485 の通信設定を変更する 98
8. Linux カーネル仕様 100
8.1. デフォルトコンフィギュレーション 100
8.2. デフォルト起動オプション 100
8.3. Linux ドライバー覧 100
8.3.1. Armadillo-IoT ゲートウェイ G3L 101
8.3.2. SPI フラッシュメモリ 101
8.3.3. UART 102
8.3.4. Ethernet 103
8.3.5. LTE 104
8.3.6. WLAN 104
8.3.7. BT 105
8.3.8. SD ホスト 106
8.3.9. USB ホスト 108
8.3.10. リアルタイムクロック 109
8.3.11. 温度センサ 110
8.3.12. AD コンバーター 111
8.3.13. LED 112
8.3.14. ユーザースイッチ 113
8.3.15. I2C 114
8.3.16. SPI 115
8.3.17. ウォッチドッグタイマー 115
8.3.18. CPU 周波数スケーリング 116
8.3.19. パワーマネジメント 118

9. Debian ユーザーランド仕様 120
9.1. Debian ユーザーランド 120
9.2. パッケージ管理 120
10. ブートローダー仕様 122
10.1. ブートローダー起動モード 122
10.2. ブートローダーの機能 122
10.2.1. Linux カーネルイメージと device tree blob の指定方法 122
10.2.2. ルートファイルシステムの指定方法 123
10.2.3. 環境変数の保存 124
10.2.4. Linux カーネルの起動オプション 124
10.2.4.1. 代表的な Linux カーネル起動オプション 124
10.2.4.2. Linux カーネル起動オプションの設定方法 125
11. ビルド手順 126
11.1. ブートローダーをビルドする 126
11.2. Linux カーネルをビルドする 127
11.3. Debian GNU/Linux ルートファイルシステムをビルドする 128
11.3.1. 出荷状態のルートファイルシステムアーカイブを構築する 128
11.3.2. カスタマイズされたルートファイルシステムアーカイブを構築する 128
11.3.2.1. ファイル/ディレクトリを追加する 129
11.3.2.2. パッケージを変更する 129
12. 開発の基本的な流れ 130
12.1. 軽量スクリプト言語によるセンサーデータの送信例(Ruby) 130
12.1.1. テスト用サーバーの実装 130
12.1.2. テスト用サーバーの動作確認 131
12.2. クライアントの実装 132
12.3. Armadillo-IoT G3L へのファイルの転送 132
12.4. クライアントの実行 133
12.5. C 言語による開発環境 133
12.5.1. 開発環境の準備 133
13. SMS を利用する 135
13.1. 初期設定 135
13.2. SMS を送信する 135
13.3. SMS リストを表示する 136
13.4. SMS の内容を表示する 136
13.5. SMS を削除する 136
13.6. SMS を他のストレージに移動する 137
14. i.MX7Dual の電源制御 138
14.1. ユーザースイッチ(SW2)の操作による制御 138
14.2. RTC による制御 138
15. SD ブートの活用 139
15.1. ブートディスクの作成 139
15.2. ルートファイルシステムの構築 143
15.2.1. Debian GNU/Linux のルートファイルシステムを構築する 143
15.3. Linux カーネルイメージと DTB の配置 144
15.4. SD ブートの実行 145
16. 電気的仕様 146
16.1. 絶対最大定格 146
16.2. 推奨動作条件 146
16.3. 入出力インターフェースの電気的仕様 146
17. インターフェース仕様 147
17.1. インターフェースレイアウト 147
17.2. メインユニット CON2 LAN インターフェース 149
17.3. メインユニット CON4 シリアルインターフェース 150

17.4. メインユニット CON5 デバッグシリアルインターフェース 150
17.5. メインユニット CON8 電源入力インターフェース 1 150
17.6. メインユニット CON10 電源入力インターフェース 2 151
17.7. メインユニット CON11 USB インターフェース 151
17.8. メインユニット CON12 microSD インターフェース 151
17.9. メインユニット CON13 アンテナインターフェース 3 152
17.10. メインユニット CON14 アンテナインターフェース 4 152
17.11. メインユニット CON15 アンテナインターフェース 1 152
17.12. メインユニット CON16 アンテナインターフェース 2 152
17.13. メインユニット SW2 ユーザースイッチ 153
17.14. メインユニット JP1 起動デバイス設定ジャンパ 153
17.15. サブユニット CON2 Wi-SUN モジュールインターフェース 153
17.16. サブユニット CON4 LTE アンテナインターフェース 1 154
17.17. サブユニット CON5 LTE アンテナインターフェース 2 154
17.18. サブユニット CON6 microSIM インターフェース 154
17.19. サブユニット CON8 WLAN アンテナインターフェース 1 155
17.20. サブユニット CON9 WLAN アンテナインターフェース 2 155
17.21. サブユニット LED3 ユーザー LED3 155
17.22. サブユニット LED4 ユーザー LED4 155
17.23. サブユニット LED5 ユーザー LED5 155
17.24. サブユニット SP1 Wi-SUN モジュールスタッド 156
17.25. サブユニット SP2 Wi-SUN モジュールスタッド 156
18. 筐体形状/寸法図 157
19. オプション品 158
19.1. USB シリアル変換アダプタ 158
19.2. Armadillo-IoT ゲートウェイ G3L LTE 用外付けアンテナセット 03 159
19.2.1. 概要 159
19.2.2. 組み立て 159
19.2.3. 形状図 159
19.3. Armadillo-IoT ゲートウェイ G3L 無線 LAN 用基板アンテナ 06 159
19.3.1. 概要 159
19.3.2. 組み立て 159
19.3.3. 形状図 160
20. Howto 161
20.1. イメージをカスタマイズする 161
20.2. ルートファイルシステムへの書き込みと電源断からの保護機能 163
20.2.1. 保護機能の使用方法 163
20.2.2. 保護機能を使用する上での注意事項 163
20.3. RS422/RS485 シリアルポートで通信を行なう 164
20.4. WL1837MOD モジュールを使って 2.4GHz 帯で通信する使用例 165
20.4.1. 「BVMCN1101AA」の信号を受信する 165
20.4.2. 「CC2650」を操作する 166
20.5. ssh で Armadillo-IoT G3L に接続する 167
21. ユーザー登録 168
21.1. 購入製品登録 168
21.1.1. 正規認証ファイルを取り出す手順 168

図目次

2.1. LTE モジュール: ELS31-J 認証マーク 21
2.2. WLAN+BT コンボモジュール: WL1837MOD 認証マーク 22
2.3. Wi-SUN モジュール: BP35A1 認証マーク 22
3.1. Armadillo-IoT ゲートウェイ G3L の外観 26
3.2. Armadillo-IoT ゲートウェイ G3L の各部名称 27
3.3. Armadillo-IoT ゲートウェイ G3L ブロック図 28
4.1. 筐体ケース内側の突起 31
4.2. ブラケットの爪 31
4.3. 噛み合わせの確認とネジ止め 32
4.4. 筐体ケースの嵌め込み 32
4.5. 各種取付 33
4.6. GNOME 端末の起動 39
4.7. GNOME 端末のウィンドウ 39
4.8. minicom 設定方法 40
4.9. minicom 起動方法 40
4.10. minicom 終了確認 40
4.11. Armadillo-IoT メインユニット インターフェースレイアウト(A 面) 41
4.12. Armadillo-IoT メインユニット インターフェースレイアウト(B 面) 41
4.13. Armadillo-IoT サブユニット インターフェースレイアウト(A 面) 42
4.14. Armadillo-IoT サブユニット インターフェースレイアウト(B 面) 42
4.15. Armadillo-IoT ゲートウェイ G3L の接続例 44
4.16. スライドスイッチの設定 45
4.17. vi の起動 45
4.18. 入力モードに移行するコマンドの説明 46
4.19. 文字を削除するコマンドの説明 47
5.1. 電源投入直後のログ 48
5.2. 起動ログ 48
5.3. 起動ログ 48
5.4. システムクロックを設定 61
5.5. 終了方法 63
6.1. Debian パッケージのアップデート 67
7.1. nmcli のコマンド書式 72
7.2. コネクションの一覧 72
7.3. コネクションの有効化 72
7.4. コネクションの有効化の例 72
7.5. コネクションの無効化 72
7.6. コネクションの作成 73
7.7. コネクションの削除 73
7.8. 固定 IP アドレス設定 74
7.9. DHCP 設定 74
7.10. DNS サーバーの指定 74
7.11. コネクションの修正の反映 74
7.12. デバイスの一覧 75
7.13. デバイスの接続 75
7.14. デバイスの接続例 75
7.15. デバイスの切断 76
7.16. 有線 LAN インターフェース(eth0)のコネクションを作成 76
7.17. 有線 LAN の PING 確認 76
7.18. 無線 LAN アクセスポイントに接続する 77
7.19. 無線 LAN(wlan0)のコネクションの作成 77

7.20. 無線 LAN の PING 確認 77
7.21. microSIM 78
7.22. microSIM の取り付け 79
7.23. LTE のコネクションの作成 79
7.24. nmcli device コマンドでの接続状態の確認 80
7.25. LTE の PING 確認 80
7.26. データ通信の終了 80
7.27. データ通信の開始 81
7.28. LTE 再接続サービスの停止 81
7.29. LTE 再接続サービスの開始 81
7.30. 認識されているモデムの一覧の取得 82
7.31. モデムの情報を取得する 82
7.32. microSIM の情報を取得する 83
7.33. 回線情報を取得する 83
7.34. ネットワーク構成図 84
7.35. iptables 87
7.36. Armadillo トップページ 88
7.37. mount コマンド書式 90
7.38. ストレージのマウント 91
7.39. ストレージのアンマウント 91
7.40. fdisk コマンドによるパーティション変更 91
7.41. EXT4 ファイルシステムの構築 92
7.42. ユーザー LED の位置 93
7.43. LED を点灯させる 93
7.44. LED を消灯させる 94
7.45. LED の状態を表示する 94
7.46. LED のトリガに timer を指定する 94
7.47. LED のトリガを表示する 94
7.48. ハードウェアクロックを設定 95
7.49. ユーザースイッチ: イベントの確認 95
7.50. AD コンバータへの入力電圧の計算式 96
7.51. AD コンバーターへの入力電圧を取得する 97
7.52. 電源電圧の計算式 97
10.1. U-Boot コマンドのヘルプを表示 122
10.2. eMMC のパーティション 1 に保存された Linux カーネルイメージから起動する 123
10.3. eMMC のパーティション 2 に保存されたルートファイルシステムを指定する 123
10.4. 全ての環境変数をデフォルト値に戻す 124
10.5. 利用可能なメモリ量を 384M にする 125
11.1. 出荷状態のルートファイルシステムアーカイブを構築する手順 128
11.2. 誤ったパッケージ名を指定した場合に起きるエラーメッセージ 129
12.1. ruby と sinatra のインストール 130
12.2. テスト用サーバー (server.rb) 131
12.3. IP アドレスの確認 (ip コマンド) 131
12.4. IP アドレスの確認 (ifconfig コマンド) 131
12.5. curl によるテストデータの送信 132
12.6. ATDE6 におけるテストデータの受信表示 132
12.7. 温度送信クライアント(client.rb) 132
12.8. Armadillo-IoT G3L への SSH サーバーのインストール 132
12.9. ATDE6 から Armadillo-IoT G3L への client.rb の転送 133
12.10. クライアントの実行方法 133
12.11. ATDE6 における温度データの受信表示 133
12.12. ツールチェーンのインストール 133
12.13. 開発用パッケージのインストールの例 (libssl の場合) 133

13.1. 言語設定 135
13.2. SMS の作成 135
13.3. SMS 番号の確認 135
13.4. SMS の送信 135
13.5. SMS の一覧の表示 136
13.6. SMS の内容を表示 136
13.7. SMS の削除 137
13.8. SIM のストレージに SMS を移動 137
13.9. LTE モジュールの内蔵ストレージに SMS を移動 137
14.1. アラーム割り込みの設定 138
15.1. 自動マウントされた microSD カードのアンマウント 139
15.2. SD ブート時の saveenv メッセージ 145
17.1. Armadillo-IoT メインユニット インターフェースレイアウト(A 面) 147
17.2. Armadillo-IoT メインユニット インターフェースレイアウト(B 面) 147
17.3. Armadillo-IoT サブユニット インターフェースレイアウト(A 面) 148
17.4. Armadillo-IoT サブユニット インターフェースレイアウト(B 面) 149
17.5. AC アダプタの極性マーク 151
18.1. 筐体形状図 157
19.1. USB シリアル変換アダプタの配線 158
19.2. LTE 用外付けアンテナ形状 159
19.3. 無線 LAN 用基板アンテナ形状 160

表目次

1.1. 使用しているフォント 15
1.2. 表示プロンプトと実行環境の関係 15
1.3. コマンド入力例での省略表記 16
2.1. LTE モジュール: ELS31-J 適合証明情報 21
2.2. WLAN+BT コンボモジュール: WL1837MOD 適合証明情報 21
2.3. Wi-SUN モジュール: BP35A1 適合証明情報 22
2.4. WL1837MOD 各国電波法規制への対応情報 22
3.1. Armadillo-IoT ゲートウェイ G3L ラインアップ 24
3.2. Armadillo-IoT ゲートウェイ G3L 開発セットのセット内容 24
3.3. 仕様 25
3.4. 各部名称と機能 27
3.5. Armadillo-IoT で利用可能なソフトウェア 28
3.6. QSPI フラッシュメモリ メモリマップ 29
3.7. eMMC メモリマップ 29
4.1. ユーザー名とパスワード 38
4.2. 動作確認に使用する取り外し可能デバイス 38
4.3. シリアル通信設定 40
4.4. Armadillo-IoT メインユニット インターフェース内容 41
4.5. Armadillo-IoT サブユニット インターフェース内容 43
4.6. 入力モードに移行するコマンド 46
4.7. カーソルの移動コマンド 46
4.8. 文字の削除コマンド 47
4.9. 保存・終了コマンド 47
5.1. シリアルコンソールログイン時のユーザ名とパスワード 60
5.2. 時刻フォーマットのフィールド 61
6.1. ソフトウェアと書き込み先の対応 66
6.2. インストールディスク作成に使用するファイル 69
6.3. インストールの進捗状況とユーザー LED の状態 70
7.1. ネットワークとネットワークデバイス 71
7.2. 固定 IP アドレス設定例 74
7.3. APN 情報設定例 79
7.4. nmcli device コマンドで取得可能な代表的な status 80
7.5. ネットワークのアドレス情報 84
7.6. デバイスの状態を disconnected にする方法 85
7.7. ストレージデバイス 90
7.8. LED クラスディレクトリと LED の対応 92
7.9. 初期出荷状態の LED の動作 93
7.10. trigger の種類 94
7.11. インプットデバイスファイルとイベントコード 95
7.12. 入力電圧の算出に必要なファイル 97
7.13. シリアルポートインターフェースと TTY デバイスファイル 97
7.14. RS485 設定と初期値 98
7.15. Linux カーネル起動オプションからの RS485 設定 98
8.1. Linux カーネル主要設定 100
8.2. Linux カーネルのデフォルト起動オプション 100
8.3. UART の接続先 102
8.4. SD ホストの接続先 106
8.5. USB インターフェースの接続先 108
8.6. キーコード 113
8.7. I2C デバイス 114

8.8. 対応する CPU 周波数と電源電圧 117
8.9. Governor の種類 117
8.10. 対応するパワーマネジメント状態 118
8.11. 起床要因として利用可能なデバイス 118
10.1. ブートローダー起動モード 122
10.2. 保守モード 有用なコマンド一覧 122
10.3. mmcdev の設定値と起動デバイス 123
10.4. ブートローダーの種類と mmcdev, mmcpart のデフォルト値 123
10.5. ブートローダーの種類と mmcroot のデフォルト値 123
10.6. Linux カーネルの起動オプションの一例 124
15.1. ブートディスクの作成に使用するファイル 140
15.2. ブートディスクの構成例 140
15.3. ルートファイルシステムの構築に使用するファイル 143
15.4. ブートディスクの作成に使用するファイル 144
15.5. ブートローダーが Linux カーネルを検出可能な条件 144
16.1. 絶対最大定格 146
16.2. 推奨動作条件 146
16.3. 入出力インターフェース電源の電気的仕様 146
17.1. Armadillo-IoT メインユニット インターフェース一覧 148
17.2. Armadillo-IoT サブユニット インターフェース一覧 149
17.3. メインユニット CON2 信号配列 149
17.4. LAN コネクタ LED 150
17.5. メインユニット CON4 信号配列 150
17.6. メインユニット CON5 信号配列 150
17.7. メインユニット CON8 信号配列 151
17.8. メインユニット CON10 信号配列 151
17.9. メインユニット CON11 信号配列 151
17.10. メインユニット CON12 信号配列 152
17.11. ユーザースイッチ SW2 の接続 153
17.12. メインユニット JP1 信号配列 153
17.13. ジャンパの機能 153
17.14. サブユニット CON2 信号配列 153
17.15. サブユニット CON6 信号配列 154
17.16. ユーザー LED3 の接続 155
17.17. ユーザー LED4 の接続 155
17.18. ユーザー LED5 の接続 155
19.1. Armadillo-IoT 関連のオプション品 158

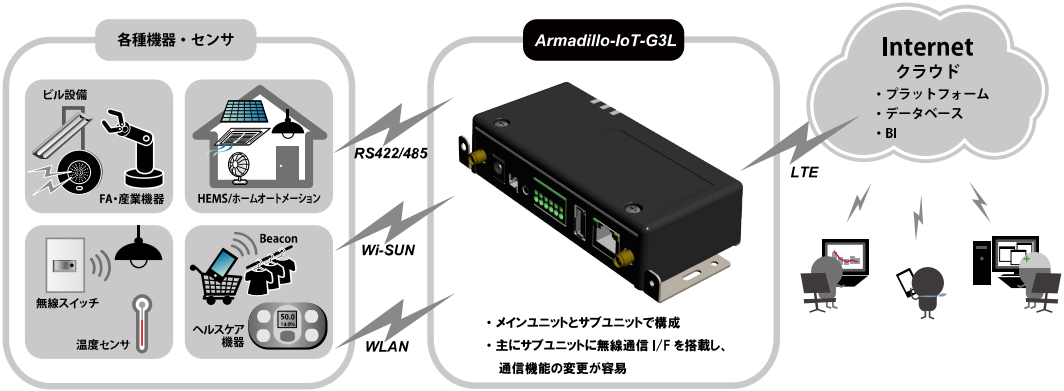
1. はじめに

このたびは Armadillo-IoT ゲートウェイ G3L 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 G3L(以下、Armadillo-IoT)は、Armadillo-IoT ゲートウェイ G3 をベースとして最低限必要となるインターフェースを実装した、各種センサとネットワークとの接続を中継する小型の IoT 向けゲートウェイです。

Armadillo-IoT は、センサ接続用インターフェースとして、RS422/RS485、LAN、無線 LAN(IEEE 802.11a/b/g/n)など一般的なセンサ接続に広く使われるインターフェースの他、Wi-SUN など新しい省電力無線通信規格にも対応しています。また、WAN(Wide Area Network)用インターフェースとして、モバイル通信(LTE)も利用可能です。

Armadillo-IoT は標準 OS として Linux がプリインストールされているため、オープンソースソフトウェアを含む多くのソフトウェア資産を活用し、自由にオリジナルのアプリケーションを開発することができます。開発言語としては、C/C++言語だけでなく、Java や Ruby などをサポートしています。さらに MQTT クライアントなど、クラウドサービスと親和性の高いソフトウェアスタックが用意され、ソフトウェア面でも開発の自由度と開発しやすさの両立を図っています。



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 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、Armadillo-IoT の機能をサポートする専用アプリケーションについても、その使い方を中心に説明しています。

Armadillo-IoT は一つの機器だけで完結するものではなく、接続するセンサや、クラウドシステムなどとの連携が不可欠です。そのため、参照すべきドキュメントも多岐に渡ります。本書では、アットマークテクノが運営する Armadillo サイトやユーザーズサイトを始め、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 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 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 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 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 ユーザーズ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。

限定コンテンツを取得するには、「21. 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 G3L ドキュメント・ダウンロード**
>
> http://armadillo.atmark-techno.com/armadillo-iot-g3l/downloads

1.5. 本書の構成

本書には、Armadillo-IoT 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や、動作を確認する方法を紹介します。**

「4. Armadillo の電源を入れる前に」、 「5. 起動と終了」、 「7. 動作確認方法」

◆ **工場出荷状態のソフトウェア仕様について紹介します。**

「8. Linux カーネル仕様」、 「9. Debian ユーザーランド仕様」、 「10. ブートローダー仕様」

◆ **システム開発に必要な情報を紹介します。**

「6. ソフトウェアの更新と初期化」、 「11. ビルド手順」、 「12. 開発の基本的な流れ」

◆ **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。**

「13. SMS を利用する」、 「14. i.MX7Dual の電源制御」、 「15. SD ブートの活用」、 「16. 電気的仕様」、 「17. インターフェース仕様」、 「18. 筐体形状/寸法図」、 「19. オプション品」

◆ **ソフトウェアのカスタマイズ方法を紹介します。**

「20. Howto」

◆ **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。**

「21. ユーザー登録」

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「~」で表わします。

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 上の root ユーザで実行
[PC /]$	作業用 PC 上の一般ユーザで実行
[armadillo /]#	Armadillo 上の root ユーザで実行
[armadillo /]$	Armadillo 上の一般ユーザで実行
=>	Armadillo 上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。適時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。

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。
- 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	基板間コネクタ、アンテナ端子、microSD スロット、microSIM スロット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、SD/SDIO、USB)以外へのコネクタやカードの着脱は、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や、ケーブルを接続するときには、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。
清掃	シンナー、ベンジン、アルコールなどの溶剤を含む化学薬品や洗浄剤を使用して清掃を行わないでください。
使用場所の制限	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。
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。 

[1] コネクタ非搭載箇所へのコネクタ等の増設は除く。
[2] 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。

この無線機(WL1837MOD)は 2.4GHz 帯を使用します。変調方式として DS-SS 及び OFDM 方式を採用しています。

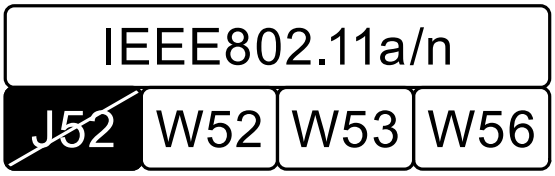
この無線機(WL1837MOD)は 2.4GHz 帯を使用します。変調方式として FH-SS 方式を採用しています。

電波に関する注意事項(5GHz 帯無線)

この無線機(WL1837MOD)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。



5.0GHz(W52,W53): Indoor Use Only

電波に関する注意事項(LTE)

この無線機(ELS31-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。

2.3. 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。

本製品の標準出荷状態でプリインストールされている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)

2.4. 書込み禁止領域について

i.MX7Dual 内蔵エレクトリカルヒューズ(e-Fuse)のデータは、本製品に含まれるソフトウェアで使用しています。正常に動作しなくなる可能性があるため、書込みを行わないでください。また、意図的に書込みを行った場合は保証対象外となります。

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、家庭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

この装置を、VCCI の技術基準に適合させるためには、DC ジャック(CON8)から AC アダプタで電源供給する必要があります。

2.6. 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 http://www.atmark-techno.com/support/warranty-policy

2.7. 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。

2.8. 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。

2.9. 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 ELS31-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 ELS31-J、WLAN+BT コンボモジュール WL1837MOD 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。

認証番号は次の通りです。

表 2.1 LTE モジュール: ELS31-J 適合証明情報

項目	内容
型式又は名称	ELS31-J
電波法に基づく工事設計認証における認証番号	003-150276
電気通信事業法に基づく設計認証における認証番号	D150192003

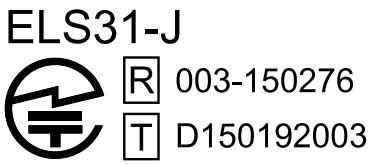


図 2.1 LTE モジュール: ELS31-J 認証マーク

表 2.2 WLAN+BT コンボモジュール: WL1837MOD 適合証明情報

項目	内容
型式又は名称	WL18MODGI
電波法に基づく工事設計認証における認証番号	201-140447

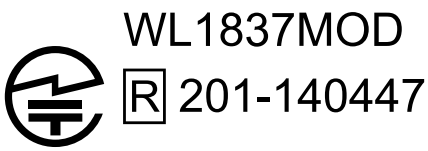


図 2.2 WLAN+BT コンボモジュール: WL1837MOD 認証マーク

表 2.3 Wi-SUN モジュール: BP35A1 適合証明情報

項目	内容
型式又は名称	BP35A1
電波法に基づく工事設計認証における認証番号	003-140032

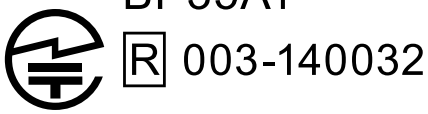


図 2.3 Wi-SUN モジュール: BP35A1 認証マーク

WL1837MOD の各国電波法規制への対応情報は以下の通りです。

- 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、または当社製品を組み込んだ最終製品を海外で販売する場合、日本国および関係する諸外国の関連法令・規制に従い、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。

表 2.4 WL1837MOD 各国電波法規制への対応情報

項目	内容
FCC ID	Z64-WL18DBMOD
IC	451I-WL18DBMOD

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 G3L とは

Armadillo-IoT ゲートウェイ G3L は、組み込みプラットフォームとして実績のある Armadillo をベースにした、IoT/M2M 向けのゲートウェイを簡単に、素早く開発するためのプラットフォームです。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。

モバイル通信(LTE)対応

モバイル通信用に、LTE 対応モジュールを搭載しています。Armadillo-IoT 専用回線プランも各社から提供されており、LTE 対応機能をすぐに導入できます。

Linux をベースとしたソフトウェアスタック

標準 OS として Linux をプリインストールしているため、オープンソースソフトウェアを中心とした、各種ソフトウェア資産を活用できます。また、Ruby や Oracle Java にも対応しているため、C/C++言語以外でのソフトウェア開発が可能です。

クラウド対応

MQTT クライアントなど、クラウドシステムと相性の良いソフトウェアスタックをプリインストール。また、各社のクラウドサービス対応エージェントが、Armadillo-IoT 向けにポーティング済みなので、クラウドと連携したシステムが開発しやすくなっています。

3.2. 製品ラインアップ

Armadillo-IoT ゲートウェイ G3L の製品ラインアップは次の通りです。

表 3.1 Armadillo-IoT ゲートウェイ G3L ラインアップ

名称	型番
Armadillo-IoT ゲートウェイ G3L D1 モデル開発セット	AGL3000-D10Z
Armadillo-IoT ゲートウェイ G3L D1 モデル量産用 (LTE 搭載、WLAN コンボ非搭載)[a]	AGL3000-C10Z
Armadillo-IoT ゲートウェイ G3L D1 モデル量産用 (LTE 搭載、WLAN コンボ搭載)[a]	AGL3000-C11Z

[a]発売予定

3.2.1. Armadillo-IoT ゲートウェイ G3L 開発セット

Armadillo-IoT ゲートウェイ G3L 開発セット(型番:AGL3000-D10Z)は、Armadillo-IoT 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。

表 3.2 Armadillo-IoT ゲートウェイ G3L 開発セットのセット内容

Armadillo-IoT ゲートウェイ G3L 本体 (LTE モジュール、WLAN+BT コンボモジュール、WLAN + BT 用基板アンテナ搭載)
3G/LTE 用 外付けアンテナ 03 (2 個)
開発用 USB シリアル変換アダプタ
USB2.0 ケーブル(A オス-miniB タイプ)
AC アダプタ(12V)
アンテナホルダー
ジャンパーソケット
ネジ類
開発用 DVD-ROM

3.2.2. Armadillo-IoT ゲートウェイ G3L 量産用

Armadillo-IoT ゲートウェイ G3L 量産用(型番:AGL3000-C10Z/AGL3000-C11Z)は、Armadillo-IoT を使った製品の量産用モデルです。

量産用モデルのうち、AGL3000-C10Z は WLAN コンボモジュール非搭載です。

量産用モデルでは、どちらのモデルも Wi-SUN モジュールはオプションでの搭載となります。

付属品など、量産時に必要なものを同時に発注することができる「BTO サービス」に対応しています。詳細についてはお問い合わせください。

3.3. 仕様

Armadillo-IoT ゲートウェイ G3L の主な仕様は次のとおりです。

表 3.3 仕様

項目	AGL3000-D10Z	AGL3000-C10Z	AGL3000-C11Z
プロセッサ	NXP Semiconductors i.MX7Dual		
	Main: ARM Cortex-A7 x 2 - 命令/データキャッシュ 32KByte/32KByte - L2 キャッシュ 512KByte - 内部 SRAM 256KByte - 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 Secondary: ARM Cortex-M4 - 命令/データキャッシュ 16KByte/16KByte		
システムクロック	CPU コアクロック(ARM Cortex-A7): 966MHz CPU コアクロック(ARM Cortex-M4): 240MHz DDR クロック: 533MHz 源発振クロック: 32.768kHz, 24MHz		
RAM	DDR3L:512MByte バス幅 32bit		
ROM	QSPI NOR 型フラッシュメモリ:8MByte eMMC(SLC Mode): 約 3.8GiB		
LAN(Ethernet)	RJ-45 x 1 100BASE-TX/10BASE-T, AUTO-MDIX 対応		
無線 LAN/BT	WLAN+BT コンボモジュール TI 製 WiLink WL1837MOD 搭載 IEEE 802.11a/b/g/n and BT	非搭載	WLAN+BT コンボモジュール TI 製 WiLink WL1837MOD 搭載 IEEE 802.11a/b/g/n and BT
モバイル通信	LTE Cat1 モジュール Gemalto 製 Cinterion ELS31-J 搭載 [a] [b] microSIM スロット x 1		
Wi-SUN	非搭載	Wi-SUN モジュール ROHM 製 BP35A1 オプション搭載可能	
シリアル	RS422 or RS485		
SD/MMC	microSD スロット x 1		
USB	USB 2.0 Host x 1 (High Speed)		
スイッチ	ユーザースイッチ x 1		
LED	ユーザー LED x 3		
電源電圧	DC 8.0V～26.4V		
消費電力	6W 以下 (突入時および USB 給電時を除く)		
使用温度範囲 [c]	-10～50℃		
外形サイズ	140 x 59.9 x 31.0mm (突起部、アンテナを除く)		
重量	約 240g[d]	約 220g	

[a]外付けアンテナを接続する必要があります。

[b]LTE モバイル通信 microSIM カード(NTT ドコモの LTE エリア対応のもの)は別売です。

[c]結露無きこと。

[d]AC アダプター、USB シリアル変換アダプタ、USB2.0 ケーブル、開発用 DVD-ROM、ジャンパーソケットは含まず。

3.4. Armadillo-IoT ゲートウェイ G3L の外観

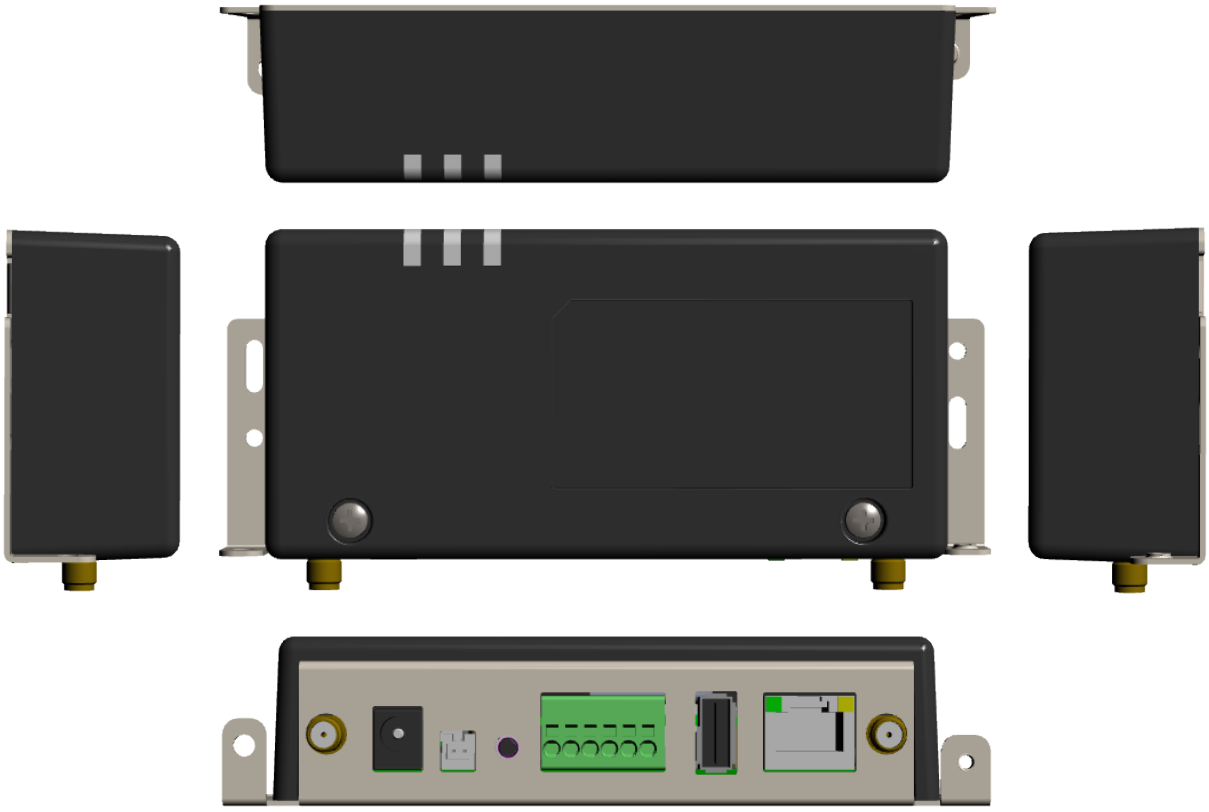


図 3.1 Armadillo-IoT ゲートウェイ G3L の外観

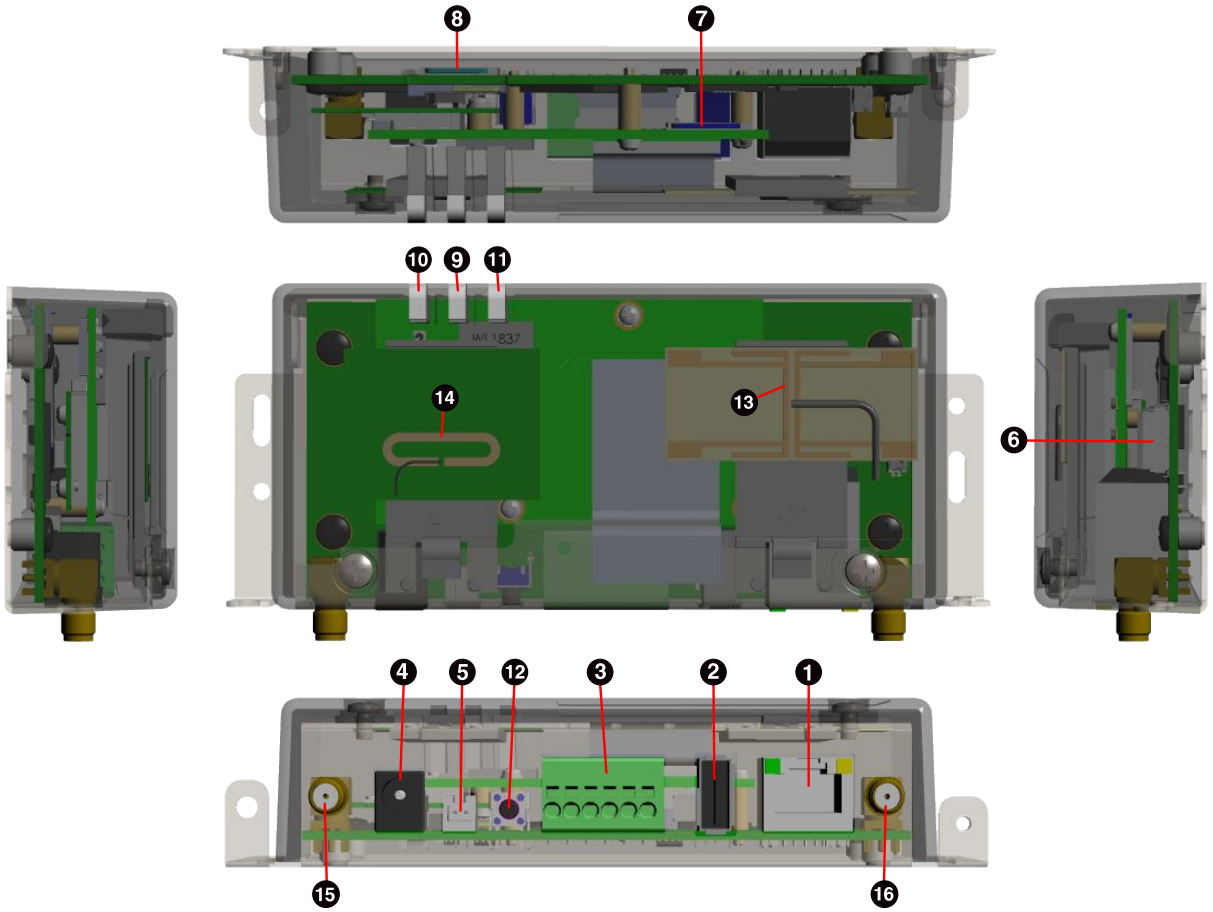


図 3.2 Armadillo-IoT ゲートウェイ G3L の各部名称

表 3.4 各部名称と機能

番号	名称	説明
1	LAN コネクタ	LAN ケーブルを接続します。
2	USB コネクタ	USB メモリ等を接続します。
3	シリアルポート	RS422/RS485 のシリアルケーブルを接続します。
4	電源コネクタ 1	付属の AC アダプタを接続します。
5	電源コネクタ 2	付属の AC アダプタ以外の電源ケーブルを接続します。
6	デバッグシリアルコネクタ	付属の USB シリアル変換アダプタを接続します。
7	microSIM スロット	microSIM カードを接続します。
8	microSD スロット	microSD カードを接続します。
9	ユーザー LED3	ユーザーで自由に機能を設定できる緑色 LED です。
10	ユーザー LED4	
11	ユーザー LED5	
12	ユーザースイッチ	ユーザーで自由に機能を設定できるタクトスイッチです。
13	無線 LAN 用基板アンテナ	無線 LAN 用の内蔵アンテナです。
14	920MHz 帯基板アンテナ	Wi-SUN 用の内蔵アンテナです。(オプション)
15	アンテナコネクタ 1	付属の LTE 用外付けアンテナを取り付けます。
16	アンテナコネクタ 2	

3.5. ブロック図

Armadillo-IoT ゲートウェイ G3L は、メインユニットとサブユニットで構成されています。ブロック図は次のとおりです。

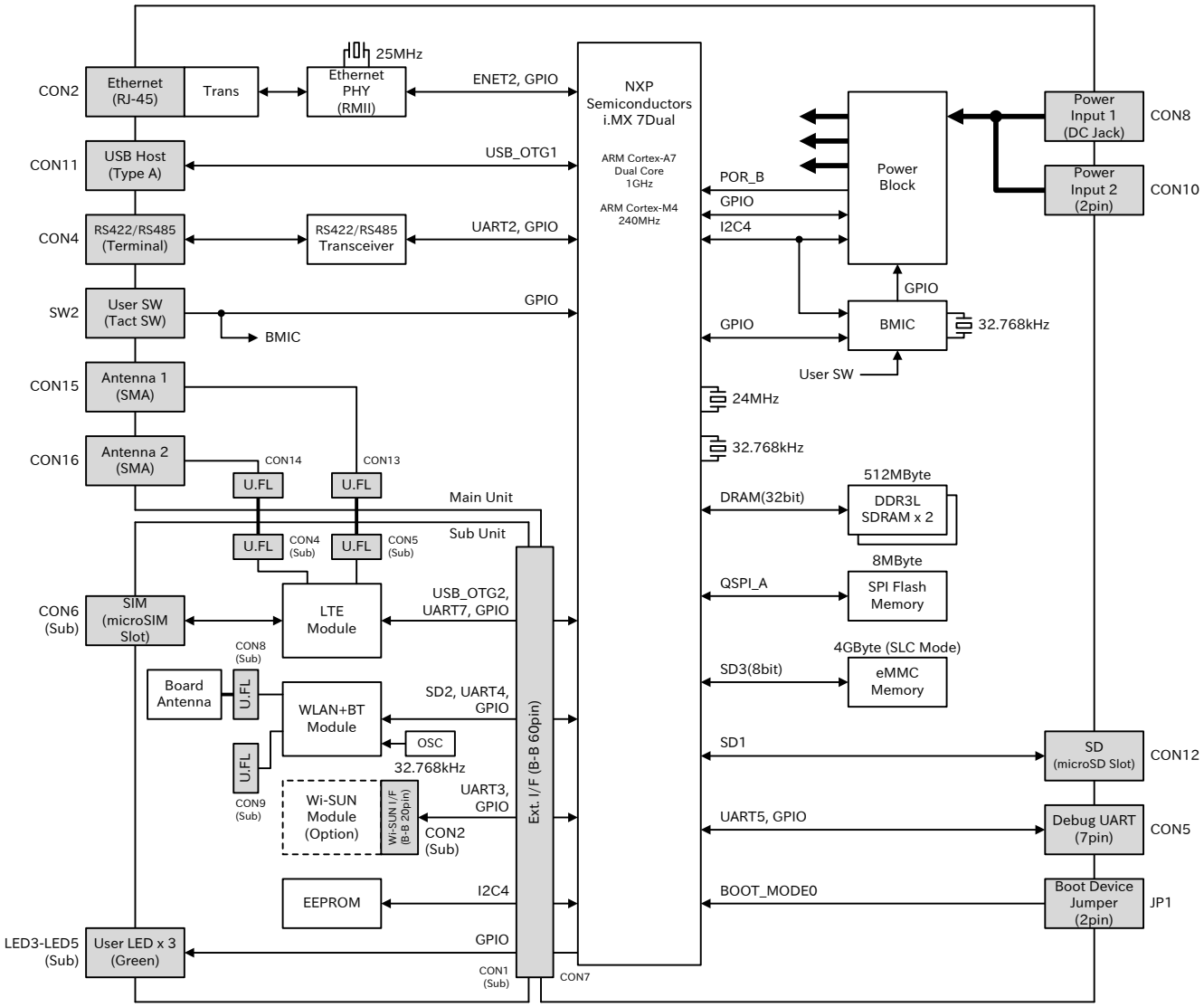


図 3.3 Armadillo-IoT ゲートウェイ G3L ブロック図

3.6. ソフトウェア構成

Armadillo-IoT で動作するソフトウェアの構成について説明します。

Armadillo-IoT で利用可能なソフトウェアを「表 3.5. Armadillo-IoT で利用可能なソフトウェア」に示します。

表 3.5 Armadillo-IoT で利用可能なソフトウェア

ソフトウェア	説明
U-Boot	ブートローダーです。工場出荷状態ではブートローダーイメージは QSPI フラッシュメモリに配置されています。
Linux カーネル	バージョン 3.14 の Linux カーネルです。工場出荷状態では Linux カーネルイメージは eMMC に配置されていますが、ブートローダーの機能により microSD カードに配置することもできます。
Debian GNU/Linux	Debian Project によって作成された Linux ディストリビューションです。パッケージ管理システムを備えているため、Debian Project が提供する豊富なソフトウェアパッケージを簡単に追加することができます。工場出荷状態では Debian GNU/Linux のルートファイルシステムは eMMC に配置されていますが、Linux カーネルがサポートしている microSD カードなどのストレージデバイスに配置することもできます。

Armadillo-IoT の QSPI フラッシュメモリのメモリマップを「表 3.6. QSPI フラッシュメモリ メモリマップ」に示します。

表 3.6 QSPI フラッシュメモリ メモリマップ

物理アドレス	サイズ	説明
0x00000000 ｜ 0x001003FF	約 1 MByte	U-Boot ブートローダーイメージ
0x00100400 ｜ 0x001403FF	256 KBytes	ライセンス情報
0x00140400 ｜ 0x007FFFFF	約 6.7 MBytes	予約領域

Armadillo-IoT の eMMC のメモリマップを「表 3.7. eMMC メモリマップ」に示します。

表 3.7 eMMC メモリマップ

パーティション	サイズ	説明
1	32 MBytes	Linux カーネルイメージ/Device Tree Blob
2	約 3.4 GBytes	Debian GNU/Linux
3	128 MBytes	リカバリイメージ

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。「4.3. 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。
microSD カード	SD スロットの動作を確認する 場合などに利用します。
USB メモリ	USB の動作を確認する 場合などに利用します。
microSIM(UIM カード)と APN 情報	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 microSIM(UIM カード)が必要です。
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[1]で展開できます。Windows では、7-Zip や Lhaz などが対応しています。7-Zip は、開発用 DVD に収録されています。

4.2. 組立手順

4.2.1. 概要

Armadillo-IoT ゲートウェイ G3L の標準筐体は、大きく分けて、基板等が実装されているブラケットと、実装されている基板等を覆う筐体ケースの 2 つで構成されています。

4.2.2. 標準筐体の組立手順

基板等が実装されているブラケットに対して、筐体ケースを取り付けます。

筐体ケースとブラケットとは、筐体ケースの突起にブラケットの爪を引っ掛けて外れないようにした後、付属のネジでネジ止めをして固定します。

筐体ケースをブラケットの爪に引っ掛けて固定するための突起は「図 4.1. 筐体ケース内側の突起」に示すように、筐体ケース内側下端左右の角 2 箇所に有ります。

[1]tar.xz 形式のファイルを展開するには Jxf オプションを指定します。

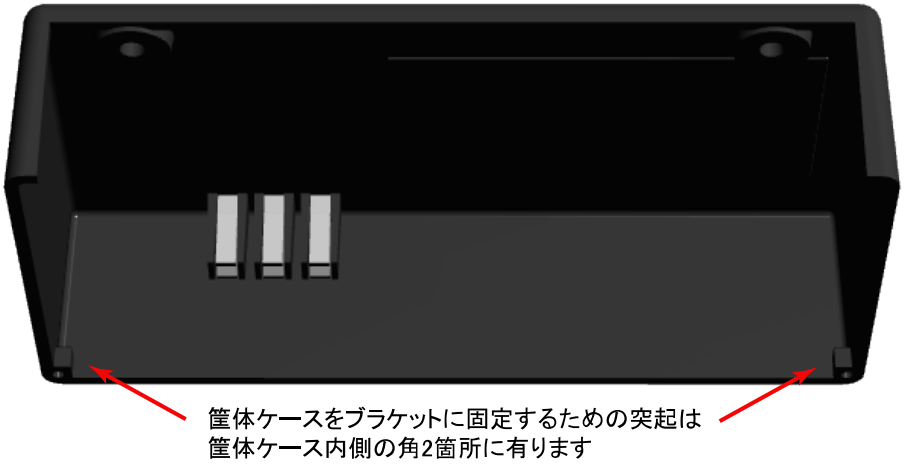


図 4.1 筐体ケース内側の突起

筐体ケースの突起を引っ掛けて固定するためのブラケットの爪は「図 4.2. ブラケットの爪」に示すような位置の 2 箇所に有ります。

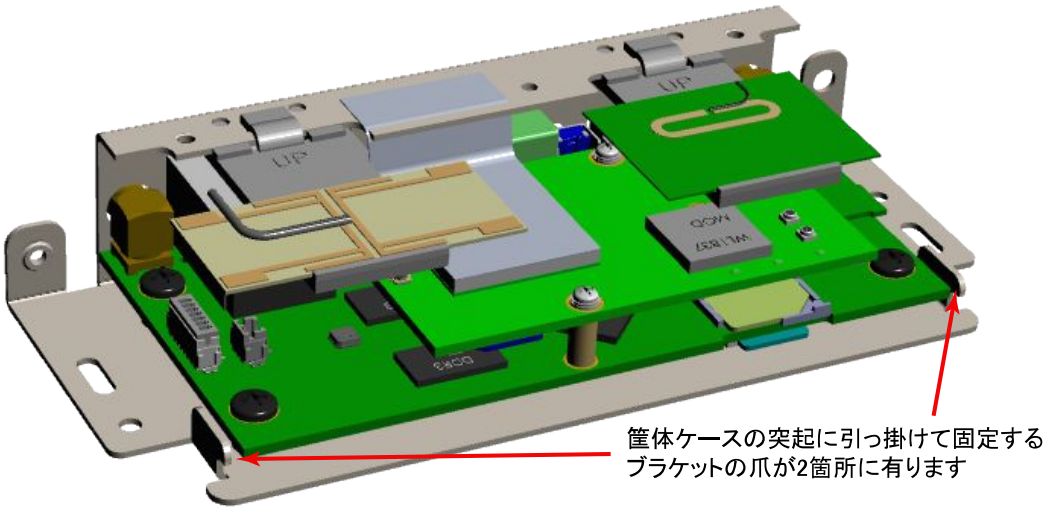


図 4.2 ブラケットの爪

標準筐体の組立手順を次に示します。

手順 4.1 標準筐体の組立手順

1. まず「図 4.3. 噛み合わせの確認とネジ止め」に示すように、ブラケットの各種端子実装面から少し離した位置で、筐体ケースをブラケットの上に載せます。

2. 次にブラケットに載せた筐体ケースの下端とブラケットとの間に隙間ができないように注意して、筐体ケースをブラケットの各種端子実装面の側へとスライドさせます。 筐体ケースの突起がブラケットの爪に引っ掛って止るところまでスライドさせて嵌め込みます。

3. 「図 4.4. 筐体ケースの嵌め込み」に示すように、筐体ケースとブラケットの爪が正しく噛み合って、筐体ケースとブラケットとの間に隙間ができていないことを確認します。

4.　さらに筐体ケース上面に 2 箇所あいている穴とブラケットのネジ穴の位置がずれずに一致していることを確認します。 確認ができたら付属のネジ 2 本でネジ止めをして、筐体ケースとブラケットを完全に固定します。

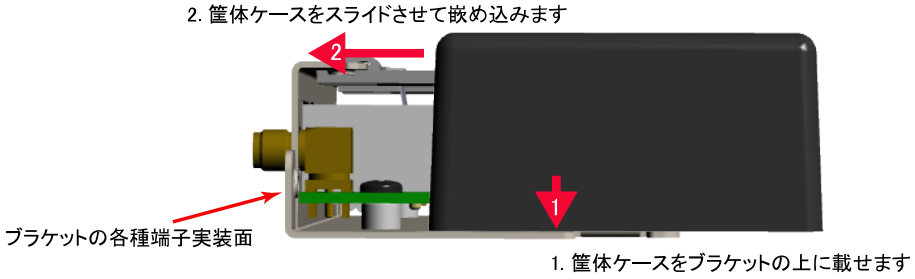
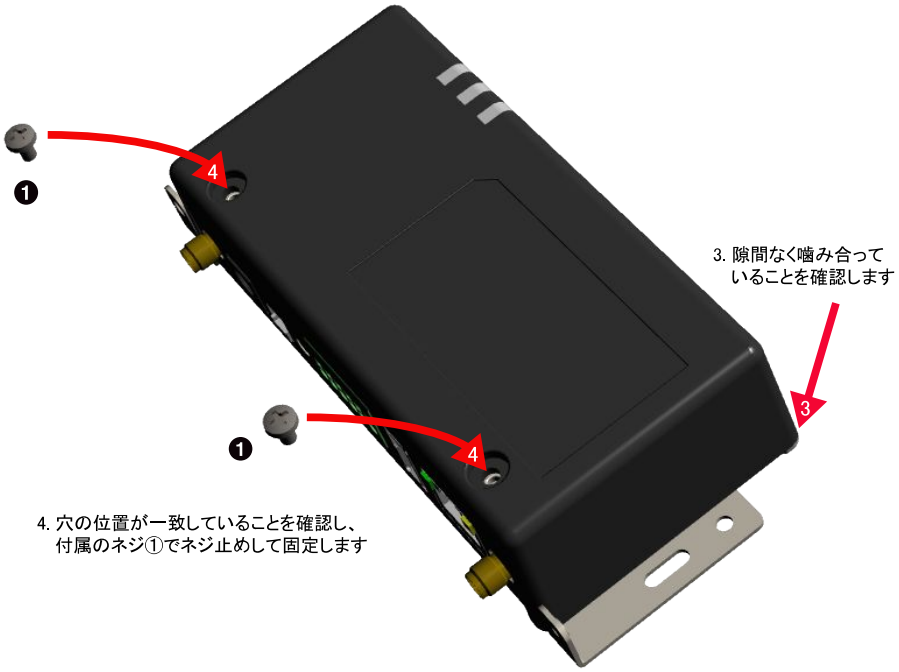


図 4.3 噛み合わせの確認とネジ止め



❶　なべ小ねじ(M3、L=4mm)x2

図 4.4 筐体ケースの嵌め込み

ネジを締める前に、筐体ケースとブラケットの爪が正しく噛み合って、筐体ケースとブラケットとの間に隙間ができていないことを必ず確認してください。隙間が有るままでネジを締めるとケースが破損する恐れが有ります。

また、ネジをきつく締め過ぎると、ケースが破損する恐れがありますので、十分にご注意ください。

4.2.3. 各種取付

Armadillo-IoT ゲートウェイ G3L 標準筐体への外付けアンテナの取付や、ブラケットに開けられた各種取付用の穴について、「図 4.5. 各種取付」に示します。各種取付用の穴の位置については「図 18.1. 筐体形状図」を参照してください。

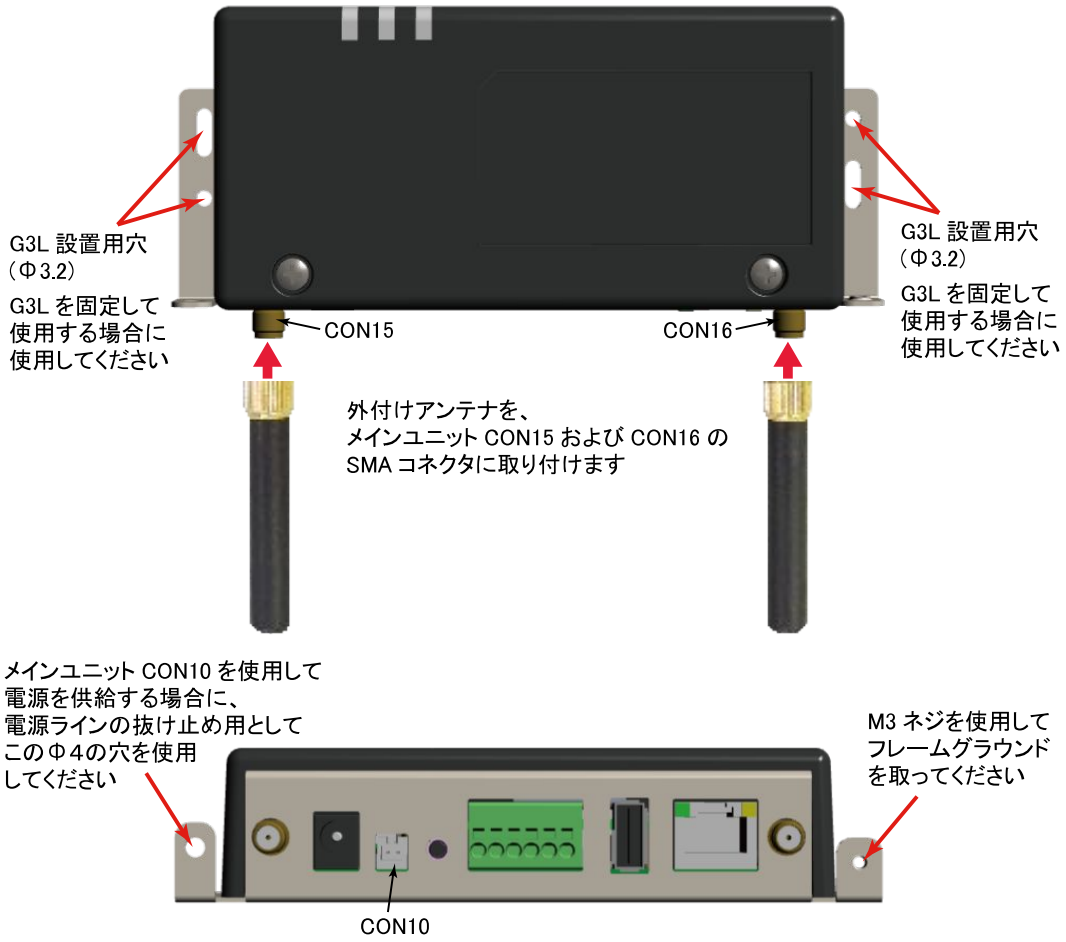


図 4.5 各種取付

> 外付けアンテナを取り付ける際、無理な力を加えると破損の原因となりますので、十分に注意してください。

4.3. 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[2]
- 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、ATDE6 の v20161118 以降です。

ATDE6 は Debian GNU/Linux 8(コードネーム jessie)をベースに、Armadillo-IoT ゲートウェイ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。

4.3.1. ATDE6 セットアップ

4.3.1.1. VMware のインストール

ATDE6 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、ATDE6 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 ATDE6 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。

4.3.1.2. ATDE6 アーカイブの取得

ATDE6 のアーカイブは Armadillo サイト(http://armadillo.atmark-techno.com)または、開発セット付属の DVD から取得可能です。

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

本製品に対応している ATDE6 のバージョンは v20161118 以降です。

作業用 PC の動作環境(ハードウェア、VMware、ATDE6 の対応アーキテクチャなど)により、ATDE6 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。

4.3.1.3. ATDE6 アーカイブの展開

ATDE6 のアーカイブを展開します。ATDE6 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「手順 4.2. Windows で ATDE6 のアーカイブ展開する」に、Linux での展開方法を「手順 4.3. Linux で tar.xz 形式のファイルを展開する」に示します。

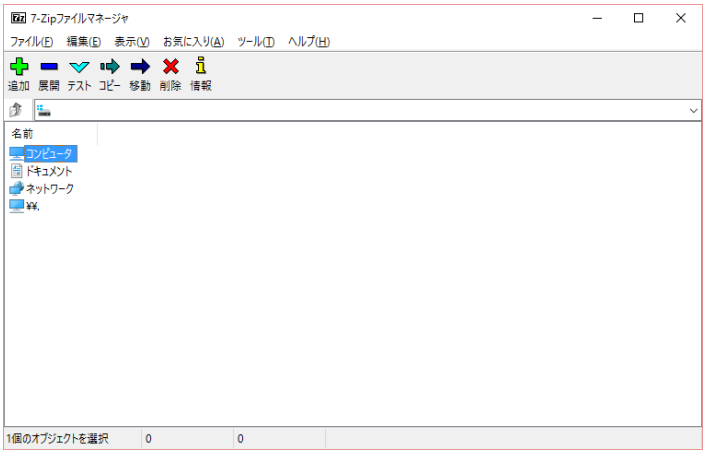
手順 4.2 Windows で ATDE6 のアーカイブ展開する

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(http://sevenzip.sourceforge.jp)または、開発セット付属の DVD から取得可能です。

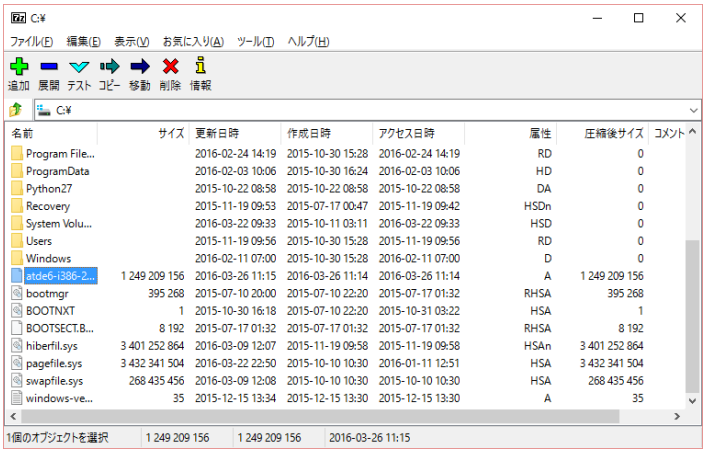
2. **7-Zip の起動**

   7-Zip を起動します。



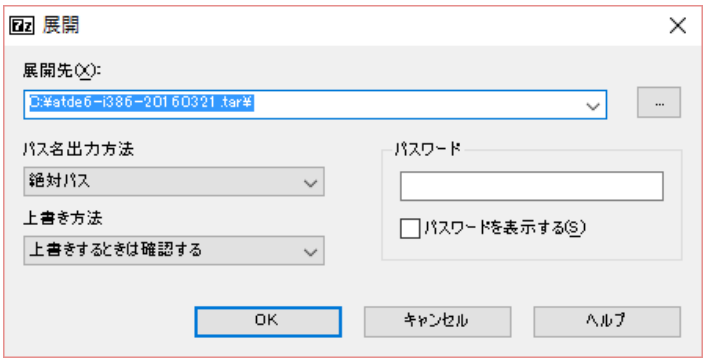
3. **xz 圧縮ファイルの選択**

   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。



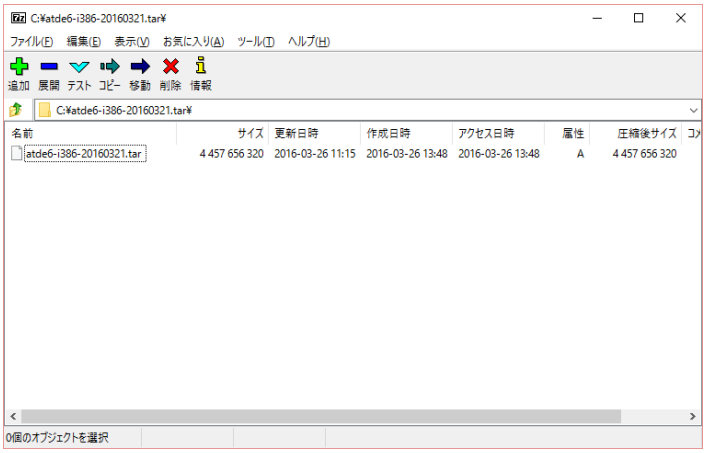
5. **xz 圧縮ファイルの展開**

展開が始まります。

6. **tar アーカイブファイルの選択**

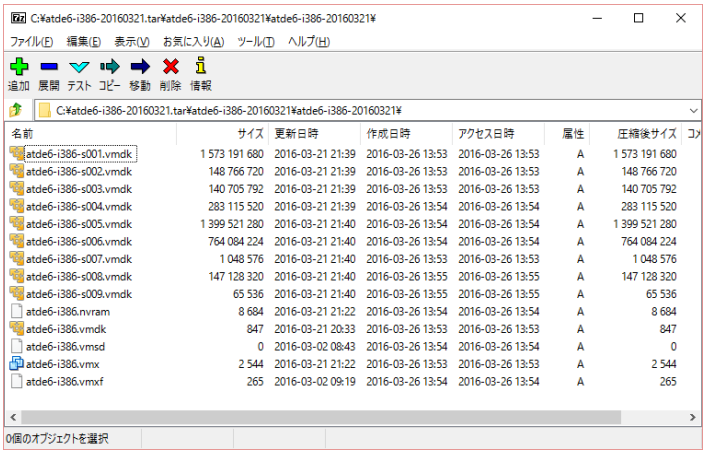
xz 圧縮ファイルの展開が終了すると、tar 形式のファイルが出力されます。

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 ATDE6 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、ATDE6 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 ATDE6 のデータイメージが出力されています。



手順 4.3 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 Jxf オプション使用して tar.xz 圧縮ファイルを展開します。

```
[PC ~]$ tar Jxf atde6-i386-[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、ATDE6 アーカイブの展開は完了です。atde6-i386-*[version]*ディレクトリに ATDE6 のデータイメージが出力されています。

```
[PC ~]$ ls atde6-i386-[version]/
atde6-i386.nvram      atde6-i386-s005.vmdk  atde6-i386.vmdk
atde6-i386-s001.vmdk  atde6-i386-s006.vmdk  atde6-i386.vmsd
atde6-i386-s002.vmdk  atde6-i386-s007.vmdk  atde6-i386.vmx
```

```
atde6-i386-s003.vmdk  atde6-i386-s008.vmdk  atde6-i386.vmxf
atde6-i386-s004.vmdk  atde6-i386-s009.vmdk
```

4.3.1.4. ATDE6 の起動

ATDE6 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、ATDE6 を起動することができます。ATDE6 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[3]。

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 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。

4.3.2. 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 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 シリアル変換アダプタ	Future Devices FT232R USB UART
作業用 PC の物理シリアルポート	シリアルポート

4.3.3. コマンドライン端末(GNOME 端末)の起動

ATDE6 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。ATDE6 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

[3]特権ユーザーで GUI ログインを行うことはできません。

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 4.6. GNOME 端末の起動」のようにデスクトップ左上のメニューから「端末」を選択してください。

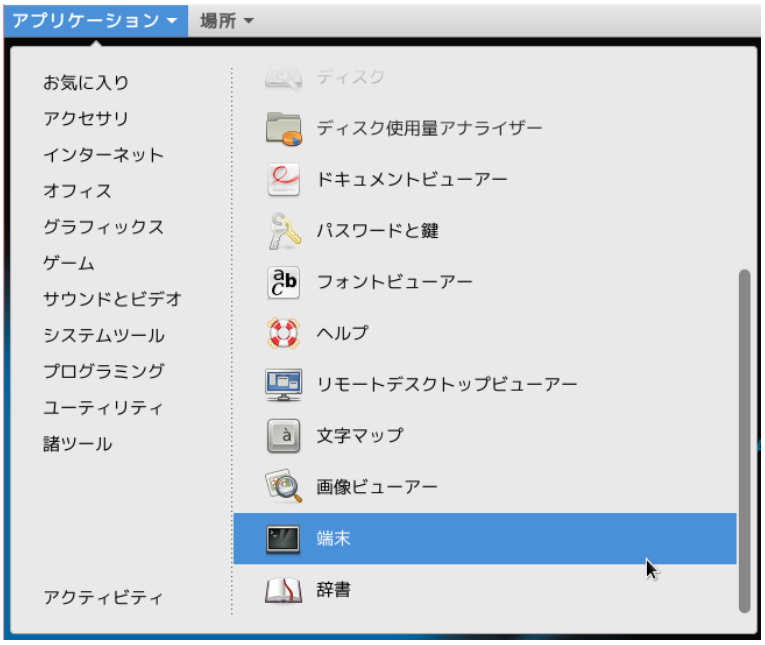


図 4.6 GNOME 端末の起動

「図 4.7. GNOME 端末のウィンドウ」のようにウィンドウが開きます。

図 4.7 GNOME 端末のウィンドウ

4.3.4. 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

minicom の設定を開始するには、「図 4.8. minicom 設定方法」のようにしてください。設定完了後、デフォルト設定(dfl)に保存して終了します。

```
[PC ~]$ LANG=C minicom --setup
```

図 4.8 minicom 設定方法

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。

```
[PC ~]$ LANG=C minicom --noinit --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 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。

4.4. インターフェースレイアウト

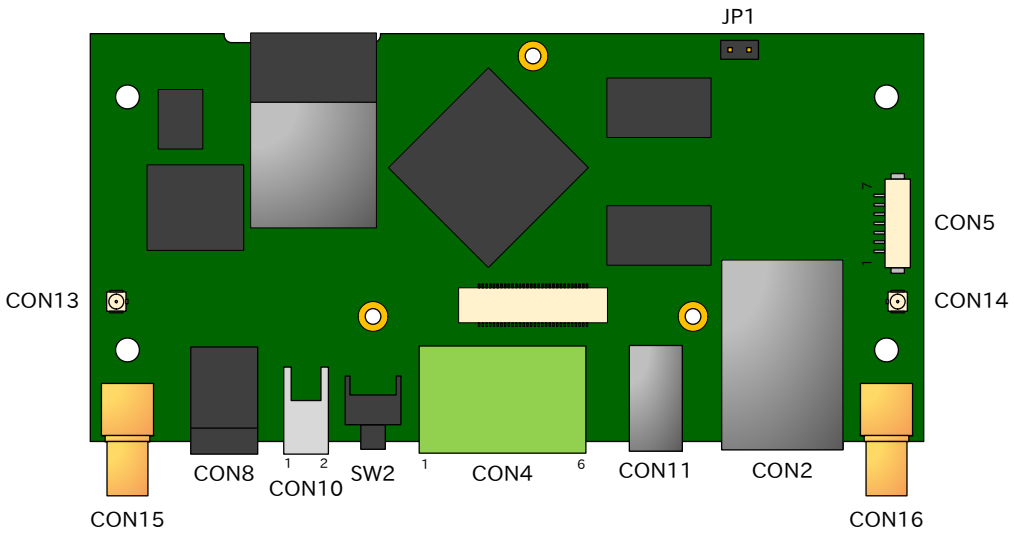


図 4.11 Armadillo-IoT メインユニット インターフェースレイアウト(A 面)

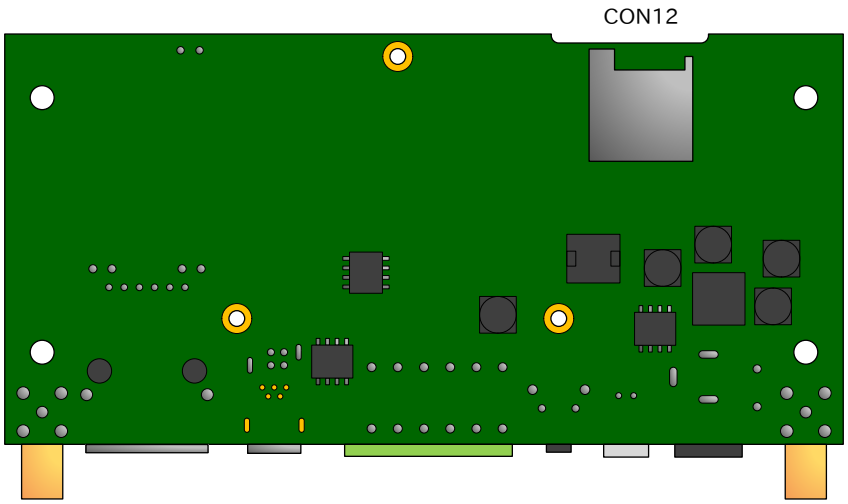


図 4.12 Armadillo-IoT メインユニット インターフェースレイアウト(B 面)

表 4.4 Armadillo-IoT メインユニット インターフェース内容

部品番号	インターフェース名	形状	備考
CON2	LAN インターフェース	RJ-45 コネクタ	
CON4	シリアルインターフェース	端子台 6 ピン(3.5mm ピッチ)	
CON5	デバッグシリアルインターフェース	ピンヘッダ 7 ピン(1.25mm ピッチ)	挿抜寿命:40 回 [a]

部品番号	インターフェース名	形状	備考
CON8	電源入力インターフェース 1	DC ジャック	対応プラグ:内径 2.1mm 外径:5.5mm
CON10	電源入力インターフェース 2	ピンヘッダ 2 ピン(2mm ピッチ)	
CON11	USB インターフェース	Type A コネクタ	
CON12	microSD インターフェース	microSD スロット	
CON13	アンテナインターフェース 3	小型同軸コネクタ	挿抜寿命:20 回 [a]
CON14	アンテナインターフェース 4	小型同軸コネクタ	挿抜寿命:20 回 [a]
CON15	アンテナインターフェース 1	同軸コネクタ	
CON16	アンテナインターフェース 2	同軸コネクタ	
SW2	ユーザースイッチ	タクトスイッチ	
JP1	起動バイス設定ジャンパ	ピンヘッダ 2 ピン(2.54mm ピッチ)	

[a]挿抜寿命は製品出荷時における目安であり、実際の挿抜可能な回数を保証するものではありません。

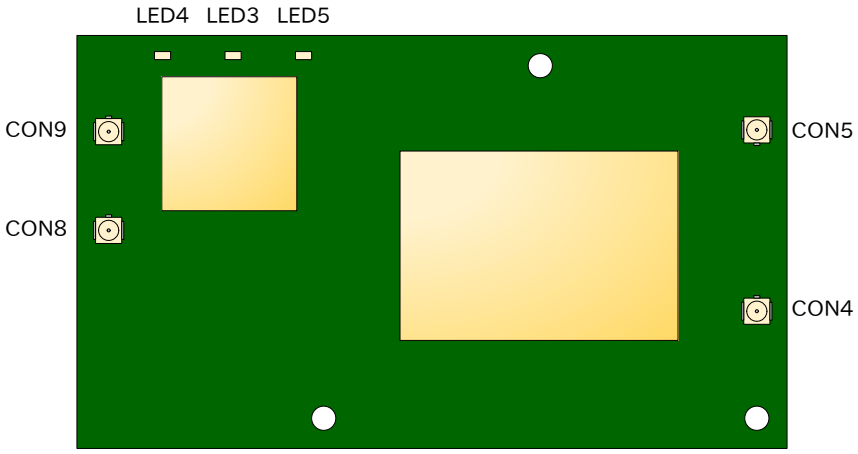


図 4.13 Armadillo-IoT サブユニット インターフェースレイアウト(A 面)

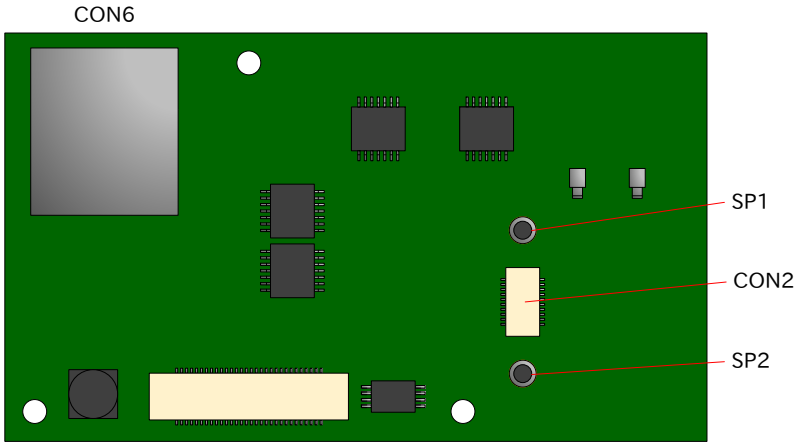


図 4.14 Armadillo-IoT サブユニット インターフェースレイアウト(B 面)

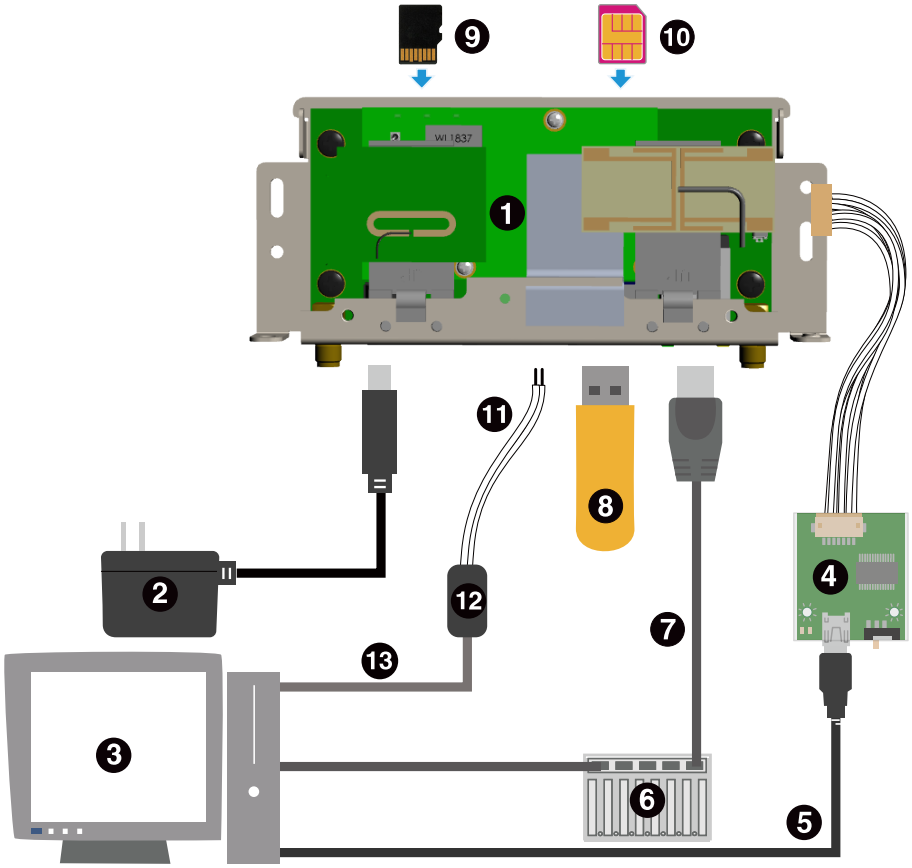
表 4.5 Armadillo-IoT サブユニット インターフェース内容

部品番号	インターフェース名	形状	備考
CON2	Wi-SUN モジュールインターフェース	基板間コネクタ 20 ピン(0.5mm ピッチ)	挿抜寿命:40 回 [a]
CON4	LTE アンテナインターフェース 1	小型同軸コネクタ	挿抜寿命:20 回 [a]
CON5	LTE アンテナインターフェース 2	小型同軸コネクタ	挿抜寿命:20 回 [a]
CON6	microSIM インターフェース	microSIM スロット	
CON8	WLAN アンテナインターフェース 1	小型同軸コネクタ	挿抜寿命:20 回 [a]
CON9	WLAN アンテナインターフェース 2	小型同軸コネクタ	挿抜寿命:20 回 [a]
LED3	ユーザー LED3	LED(緑色、面実装、導光板有り)	
LED4	ユーザー LED4	LED(緑色、面実装、導光板有り)	
LED5	ユーザー LED5	LED(緑色、面実装、導光板有り)	
SP1	Wi-SUN モジュールスタッド	スペーサー(M2, L=3mm)	
SP2	Wi-SUN モジュールスタッド	スペーサー(M2, L=3mm)	

[a]挿抜寿命は製品出荷時における目安であり、実際の挿抜可能な回数を保証するものではありません。

4.5. 接続方法

Armadillo-IoT ゲートウェイ G3L と周辺装置の接続例を次に示します。



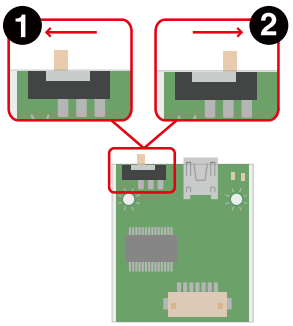
❶ Armadillo-IoT ゲートウェイ
❷ AC アダプタ(12V) [4]
❸ 作業用 PC
❹ USB シリアル変換アダプタ [4]
❺ USB2.0 ケーブル(A-miniB タイプ) [4]
❻ LAN HUB
❼ LAN ケーブル
❽ USB メモリ
❾ microSD カード
❿ microSIM カード
⓫ RS422/RS485 ケーブル
⓬ RS422/RS485-RS232C 変換アダプタ
⓭ RS232C ケーブル

図 4.15 Armadillo-IoT ゲートウェイ G3L の接続例

[4]Armadillo-IoT ゲートウェイ開発セット付属品

4.6. スライドスイッチの設定について

USB シリアル変換アダプタのスライドスイッチを操作することで、ブートローダーの起動モードを変更することができます。



❶　ブートローダーは保守モード[5]になります。

❷　ブートローダーはオートブートモード[6]になります。

図 4.16 スライドスイッチの設定

4.7. 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。

4.7.1. 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。

```
[PC ~]# vi [file]
```

図 4.17 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。

[5]ブートローダーのコマンドプロンプトが起動します。
[6]OS を自動起動します。

4.7.2. 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 4.6. 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 4.6 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 4.18. 入力モードに移行するコマンドの説明」に示します。

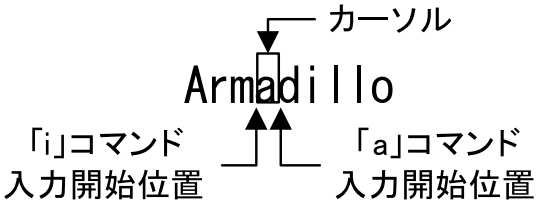


図 4.18 入力モードに移行するコマンドの説明

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、「4.7.4. 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。

4.7.3. 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 4.7. 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 4.7 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

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

4.7.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.8. 文字の削除コマンド」に示すコマンドを入力します。

表 4.8 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 4.19. 文字を削除するコマンドの説明」に示します。

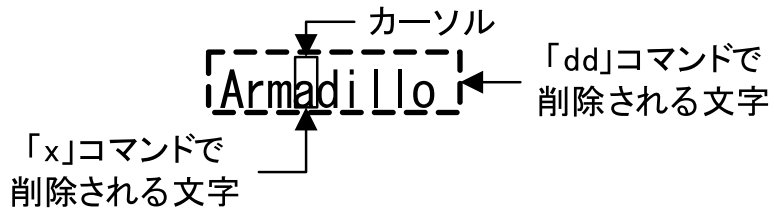


図 4.19 文字を削除するコマンドの説明

4.7.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.9. 保存・終了コマンド」に示します。

表 4.9 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]*	ファイル名を *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

Armadillo-IoT G3L に電源を接続するとき USB シリアル変換アダプタのスライドスイッチによって起動モードが変わります。 詳しくは「4.6. スライドスイッチの設定について」を参照してください。 本節では、保守モードに設定しているときの例を示します。 オートブートモードを選択した場合は、途中でコマンドを入力することなく起動が完了します。 USB シリアル変換アダプタを接続せずに Armadillo-IoT G3L に電源を接続した場合、オートブートモードで起動します。

```
U-Boot 2016.07-at4 (Nov 17 2016 - 15:52:10 +0900)

CPU:   Freescale i.MX7D rev1.2 996 MHz (running at 792 MHz)
CPU:   Extended Commercial temperature grade (-20C to 105C) at 37C
Reset cause: POR
       Watchdog enabled
I2C:   ready
DRAM:  512 MiB
Board Type: Armadillo-IoT G3L(0a200000)
Revison: 0002
S/N: 100
DRAM: 00001d05
XTAL: 00
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
SF: Detected N25Q64 with page size 256 Bytes, erase size 64 KiB, total 8 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Found PFUZE300! deviceid 0x30, revid 0x11
Net:   FEC0
=>
```

図 5.1 電源投入直後のログ

Linux システムを起動するには、次のように "boot"コマンドを実行してください。コマンドを実行するとブートローダーが Linux システムを起動させます。シリアル通信ソフトウェアには Linux の起動ログが表示されます。

図 5.2 起動ログ

```
=> boot
switch to partitions #0, OK
mmc1(part 0) is current device
switch to partitions #0, OK
mmc1(part 0) is current device
reading boot.scr
```

```
** Unable to read file boot.scr **
reading boot.scr
** Unable to read file boot.scr **
reading uImage
9835800 bytes read in 243 ms (38.6 MiB/s)
Booting from mmc ...
reading armadillo_iotg_g3l.dtb
54774 bytes read in 18 ms (2.9 MiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-3.14.79-at6
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    9835736 Bytes = 9.4 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 84800000
   Booting using the fdt blob at 0x84800000
   Loading Kernel Image ... OK
   Using Device Tree in place at 84800000, end 848105f5

Starting kernel ...

Booting Linux on physical CPU 0x0
Linux version 3.14.79-at6 (atmark@atde6) (gcc version 4.9.2 ( 4.9.2-10) ) #1627 SMP PREEMPT Thu Nov
17 20:02:16 JST 2016
CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c53c7d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine model: Atmark-Techno Armadillo-X1L Board
cma: CMA: reserved 64 MiB at 9c000000
Memory policy: Data cache writealloc
PERCPU: Embedded 8 pages/cpu @9bbb4000 s8256 r8192 d16320 u32768
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
Kernel command line: console=ttymxc4,115200 root=/dev/mmcblk2p2 rootwait rw
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 433860K/524288K available (8742K kernel code, 511K rwdata, 7784K rodata, 2548K init, 441K
bss, 90428K reserved, 0K highmem)
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
    vmalloc : 0xa0800000 - 0xff000000   (1512 MB)
    lowmem  : 0x80000000 - 0xa0000000   ( 512 MB)
    pkmap   : 0x7fe00000 - 0x80000000   (   2 MB)
    modules : 0x7f000000 - 0x7fe00000   (  14 MB)
      .text : 0x80008000 - 0x8102bd7c   (16528 kB)
      .init : 0x8102c000 - 0x812a9040   (2549 kB)
      .data : 0x812aa000 - 0x81329dc0   ( 512 kB)
       .bss : 0x81329dcc - 0x8139822c   ( 442 kB)
SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=2, Nodes=1
Preemptible hierarchical RCU implementation.
    RCU restricting CPUs from NR_CPUS=4 to nr_cpu_ids=2.
RCU: Adjusting geometry for rcu_fanout_leaf=16, nr_cpu_ids=2
NR_IRQS:16 nr_irqs:16 16
Switching to timer-based delay loop
sched_clock: 32 bits at 3000kHz, resolution 333ns, wraps every 1431655765682ns
Architected cp15 timer(s) running at 8.00MHz (phys).
sched_clock: 56 bits at 8MHz, resolution 125ns, wraps every 2147483648000ns
```

```
Ignoring duplicate/late registration of read_current_timer delay
Console: colour dummy device 80x30
Calibrating delay loop (skipped), value calculated using timer frequency.. 6.00 BogoMIPS (lpj=30000)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
CPU: Testing write buffer coherency: ok
/cpus/cpu@0 missing clock-frequency property
/cpus/cpu@1 missing clock-frequency property
CPU0: thread -1, cpu 0, socket 0, mpidr 80000000
Setting up static identity map for 0x808515f0 - 0x80851648
CPU1: Booted secondary processor
CPU1: thread -1, cpu 1, socket 0, mpidr 80000001
Brought up 2 CPUs
SMP: Total of 2 processors activated (12.00 BogoMIPS).
CPU: All CPU(s) started in SVC mode.
devtmpfs: initialized
VFP support v0.3: implementor 41 architecture 2 part 30 variant 7 rev 5
pinctrl core: initialized pinctrl subsystem
regulator-dummy: no parameters
NET: Registered protocol family 16
DMA: preallocated 256 KiB pool for atomic coherent allocations
cpuidle: using governor ladder
cpuidle: using governor menu
Use WDOG1 as reset source
GPIO line 13 (MCU_INTB) hogged as input
GPIO line 1 (LTE_GPIO12_3) hogged as output/high
GPIO line 43 (SVEN) hogged as output/high
GPIO line 36 (AOM_PWREN) hogged as output/high
GPIO line 85 (WLAN_PWR_EN) hogged as output/high
GPIO line 70 (LTE_POWER_EN) hogged as input
GPIO line 98 (BP35A1_NMIX) hogged as output/high
GPIO line 109 (LTE_VUSB) hogged as output/high
GPIO line 113 (LTE_SYS_REST_N) hogged as input
syscon 30340000.iomuxc-gpr: regmap [mem 0x30340000-0x3034ffff] registered
syscon 30350000.ocotp-ctrl: regmap [mem 0x30350000-0x3035ffff] registered
syscon 30360000.anatop: regmap [mem 0x30360000-0x3036ffff] registered
vdd1p0d: 800 <--> 1200 mV at 1000 mV
vdd1p2: 1100 <--> 1300 mV
syscon 30390000.src: regmap [mem 0x30390000-0x3039ffff] registered
DDR type is DDR3!
prom_parse: Bad cell count for /regulators0/regulator@0
hw-breakpoint: found 5 (+1 reserved) breakpoint and 4 watchpoint registers.
hw-breakpoint: maximum watchpoint size is 8 bytes.
imx7d-pinctrl 302c0000.iomuxc-lpsr: initialized IMX pinctrl driver
imx7d-pinctrl 30330000.iomuxc: initialized IMX pinctrl driver
MU is ready for cross core communication!
bio: create slab <bio-0> at 0
mxs-dma 33000000.dma-apbh: initialized
USB_OTG1_VBUS: 5000 mV
VDD_SD1: 3300 mV
WLAN_EN: 1800 mV
vgaarb: loaded
i2c-core: driver [max17135] using legacy suspend method
i2c-core: driver [max17135] using legacy resume method
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
```

```
usbcore: registered new device driver usb
30800000.aips-bus:usbphy_nop1 supply vcc not found, using dummy regulator
30800000.aips-bus:usbphy_nop2 supply vcc not found, using dummy regulator
gpio_bmic 3-0014: version: 2.0
bmic_regulator 3-0016: version: 1.0
BMIC_VREF: 1710 <--> 3600 mV at 3297 mV
i2c i2c-3: IMX I2C adapter registered
Linux video capture interface: v2.00
pps_core: LinuxPPS API ver. 1 registered
pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti <giometti@linux.it>
PTP clock support registered
MIPI CSI2 driver module loaded
Advanced Linux Sound Architecture Driver Initialized.
Bluetooth: Core ver 2.18
NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
Bluetooth: HCI socket layer initialized
Bluetooth: L2CAP socket layer initialized
Bluetooth: SCO socket layer initialized
cfg80211: Calling CRDA to update world regulatory domain
Switched to clocksource arch_sys_counter
NET: Registered protocol family 2
TCP established hash table entries: 4096 (order: 2, 16384 bytes)
TCP bind hash table entries: 4096 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 4096 bind 4096)
TCP: reno registered
UDP hash table entries: 256 (order: 1, 8192 bytes)
UDP-Lite hash table entries: 256 (order: 1, 8192 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
imx rpmsg driver is registered.
Bus freq driver module loaded
futex hash table entries: 512 (order: 3, 32768 bytes)
VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
squashfs: version 4.0 (2009/01/31) Phillip Lougher
NFS: Registering the id_resolver key type
Key type id_resolver registered
Key type id_legacy registered
jffs2: version 2.2. (NAND) © 2001-2006 Red Hat, Inc.
fuse init (API version 7.22)
msgmni has been set to 975
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
imx-sdma 30bd0000.sdma: no event needs to be remapped
imx-sdma 30bd0000.sdma: loaded firmware 4.1
imx-sdma 30bd0000.sdma: initialized
pfuze100-regulator 3-0009: Full layer: 1, Metal layer: 1
pfuze100-regulator 3-0009: FAB: 0, FIN: 0
pfuze100-regulator 3-0009: pfuze3000 found.
SW1A: 700 <--> 1475 mV at 1100 mV
SW1B: 700 <--> 1475 mV at 1000 mV
SW2: 1500 <--> 1850 mV at 1800 mV
SW3: 900 <--> 1650 mV at 1350 mV
```

```
SWBST: 5000 <--> 5150 mV at 5000 mV
VSNVS: 1000 <--> 3000 mV at 3000 mV
VREFDDR: 750 mV
VLDO1: 1800 <--> 3300 mV at 1800 mV
VLDO2: 800 <--> 1550 mV at 1500 mV
VCCSD: 2850 <--> 3300 mV at 3300 mV
V33: 2850 <--> 3300 mV at 3300 mV
VLDO3: 1800 <--> 3300 mV at 3300 mV
VLDO4: 1800 <--> 3300 mV at 3300 mV
30890000.serial: ttymxc1 at MMIO 0x30890000 (irq = 59, base_baud = 5000000) is a IMX
30880000.serial: ttymxc2 at MMIO 0x30880000 (irq = 60, base_baud = 1500000) is a IMX
30a60000.serial: ttymxc3 at MMIO 0x30a60000 (irq = 61, base_baud = 5000000) is a IMX
30a70000.serial: ttymxc4 at MMIO 0x30a70000 (irq = 62, base_baud = 1500000) is a IMX
console [ttymxc4] enabled
30a90000.serial: ttymxc6 at MMIO 0x30a90000 (irq = 158, base_baud = 1500000) is a IMX
serial: Freescale lpuart driver
imx sema4 driver is registered.
[drm] Initialized drm 1.1.0 20060810
[drm] Initialized vivante 1.0.0 20120216 on minor 0
brd: module loaded
loop: module loaded
(stk) :sysfs entries created
(stk) : debugfs entries created
(hci_tty): inside hci_tty_init
(hci_tty): allocated 248, 0
fsl-quadspi 30bb0000.qspi: Unsupported cmd 0x65
fsl-quadspi 30bb0000.qspi: Unsupported cmd 0x61
fsl-quadspi 30bb0000.qspi: Unsupported cmd 0x65
fsl-quadspi 30bb0000.qspi: n25q064 (8192 Kbytes)
3 ofpart partitions found on MTD device 30bb0000.qspi
Creating 3 MTD partitions on "30bb0000.qspi":
0x000000000000-0x000000100000 : "bootloader"
0x000000100000-0x000000140000 : "license"
0x000000140000-0x000000800000 : "reserved"
fsl-quadspi 30bb0000.qspi: QuadSPI SPI NOR flash driver
CAN device driver interface
30bf0000.ethernet supply phy not found, using dummy regulator
pps pps0: new PPS source ptp0
libphy: fec_enet_mii_bus: probed
fec 30bf0000.ethernet eth0: registered PHC device 0
PPP generic driver version 2.4.2
usbcore: registered new interface driver cdc_ether
usbcore: registered new interface driver cdc_eem
usbcore: registered new interface driver sierra_net
usbcore: registered new interface driver cdc_ncm
usbcore: registered new interface driver qmi_wwan
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci-pci: EHCI PCI platform driver
ehci-mxc: Freescale On-Chip EHCI Host driver
usbcore: registered new interface driver cdc_acm
cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
usbcore: registered new interface driver cdc_wdm
usbcore: registered new interface driver usb-storage
usbcore: registered new interface driver usbserial
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial support registered for generic
usbcore: registered new interface driver ftdi_sio
usbserial: USB Serial support registered for FTDI USB Serial Device
```

```
usbcore: registered new interface driver option
usbserial: USB Serial support registered for GSM modem (1-port)
usbcore: registered new interface driver sierra
usbserial: USB Serial support registered for Sierra USB modem
usbcore: registered new interface driver usb_serial_simple
usbserial: USB Serial support registered for zio
usbserial: USB Serial support registered for funsoft
usbserial: USB Serial support registered for flashloader
usbserial: USB Serial support registered for google
usbserial: USB Serial support registered for vivopay
usbserial: USB Serial support registered for moto_modem
usbserial: USB Serial support registered for hp4x
usbserial: USB Serial support registered for suunto
usbserial: USB Serial support registered for siemens_mpi
30b10200.usbmisc supply vbus-wakeup not found, using dummy regulator
30b20200.usbmisc supply vbus-wakeup not found, using dummy regulator
30b30200.usbmisc supply vbus-wakeup not found, using dummy regulator
30b20000.usb supply vbus not found, using dummy regulator
ci_hdrc ci_hdrc.0: EHCI Host Controller
ci_hdrc ci_hdrc.0: new USB bus registered, assigned bus number 1
ci_hdrc ci_hdrc.1: EHCI Host Controller
ci_hdrc ci_hdrc.1: new USB bus registered, assigned bus number 2
ci_hdrc ci_hdrc.0: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
mousedev: PS/2 mouse device common for all mice
input: 30370000.snvs-pwrkey as /devices/soc/30000000.aips-bus/30370000.snvs-pwrkey/input/input0
snvs_pwrkey 30370000.snvs-pwrkey: i.MX snvs powerkey probed
i2c-core: driver [isl29023] using legacy suspend method
i2c-core: driver [isl29023] using legacy resume method
bmic_rtc 3-0011: version: 1.1
bmic_rtc 3-0011: rtc core: registered bmic_rtc as rtc0
snvs_rtc 30370034.snvs-rtc-lp: rtc core: registered 30370034.snvs-rtc-l as rtc1
i2c /dev entries driver
IR NEC protocol handler initialized
IR RC5(x) protocol handler initialized
IR RC6 protocol handler initialized
IR JVC protocol handler initialized
IR Sony protocol handler initialized
IR RC5 (streamzap) protocol handler initialized
IR SANYO protocol handler initialized
IR MCE Keyboard/mouse protocol handler initialized
usbcore: registered new interface driver uvcvideo
USB Video Class driver (1.1.1)
i2c-core: driver [mag3110] using legacy suspend method
i2c-core: driver [mag3110] using legacy resume method
bmic_thermal 3-0013: version: 1.0
imx2-wdt 30280000.wdog: timeout 10 sec (nowayout=0)
Bluetooth: HCI UART driver ver 2.2
Bluetooth: HCI H4 protocol initialized
Bluetooth: HCI BCSP protocol initialized
Bluetooth: HCILL protocol initialized
Bluetooth: HCIATH3K protocol initialized
usbcore: registered new interface driver bcm203x
usbcore: registered new interface driver btusb
```

```
usbcore: registered new interface driver ath3k
(stc):  chnl_id list empty :4 sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
sdhci-esdhc-imx 30b40000.usdhc: Got CD GPIO #128.
sdhci-esdhc-imx 30b40000.usdhc: Got WP GPIO #129.

(stk) : st_kim_startmmc0: no vqmmc regulator found
mmc0: SDHCI controller on 30b40000.usdhc [30b40000.usdhc] using ADMA
mmc1: no vqmmc regulator found
(stk) :ldisc_install = 1
mmc1: SDHCI controller on 30b50000.usdhc [30b50000.usdhc] using DMA
sdhci-esdhc-imx 30b60000.usdhc: allocated mmc-pwrseq
sdhci-esdhc-imx 30b60000.usdhc: could not get ultra high speed state, work on normal mode
sdhci-esdhc-imx 30b50000.usdhc: card claims to support voltages below defined range
mmc2: no vqmmc regulator found
mmc2: no vmmc regulator found
mmc1: queuing unknown CIS tuple 0x91 (3 bytes)
mmc1: new high speed SDIO card at address 0001
wl18xx_driver wl18xx.0.auto: Direct firmware load failed with error -2
wl18xx_driver wl18xx.0.auto: Falling back to user helper
mmc2: SDHCI controller on 30b60000.usdhc [30b60000.usdhc] using ADMA
caam 30900000.caam: Instantiated RNG4 SH0
caam 30900000.caam: Instantiated RNG4 SH1
caam 30900000.caam: device ID = 0x0a160300 (Era 8)
mmc2: BKOPS_EN bit is not set
caam 30900000.caam: job rings = 3, qi = 0
mmc2: new high speed DDR MMC card at address 0001
mmcblk2: mmc2:0001 Q2J55L 3.56 GiB
mmcblk2boot0: mmc2:0001 Q2J55L partition 1 2.00 MiB
mmcblk2boot1: mmc2:0001 Q2J55L partition 2 2.00 MiB
mmcblk2rpmb: mmc2:0001 Q2J55L partition 3 4.00 MiB
 mmcblk2: p1 p2 p3
 mmcblk2boot1: unknown partition table
 mmcblk2boot0: unknown partition table
caam algorithms registered in /proc/crypto
hmac-sha1-caam alg registration failed
sha1-caam alg registration failed
hmac-sha224-caam alg registration failed
sha224-caam alg registration failed
hmac-sha256-caam alg registration failed
sha256-caam alg registration failed
hmac-md5-caam alg registration failed
md5-caam alg registration failed
xcbc-aes-caam alg registration failed
xcbc-aes-caam alg registration failed
caam_jr 30901000.jr0: registering rng-caam
snvs-secvio 30370000.caam-snvs: violation handlers armed - non-secure state
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
usbcore: registered new interface driver r8712u
bmic_adc 3-0012: version: 1.0
coresight-tmc 30086000.etr: TMC initialized
coresight-tmc 30084000.etf: TMC initialized
coresight-tpiu 30087000.tpiu: TPIU initialized
coresight-funnel 30083000.funnel: FUNNEL initialized
coresight-funnel 30041000.funnel: FUNNEL initialized
coresight-replicator replicator: REPLICATOR initialized
```

```
coresight-etm3x 3007c000.etm: ETM initialized
coresight-etm3x 3007d000.etm: ETM initialized
usbcore: registered new interface driver snd-usb-audio
NET: Registered protocol family 26
nf_conntrack version 0.5.0 (7803 buckets, 31212 max)
ipip: IPv4 over IPv4 tunneling driver
gre: GRE over IPv4 demultiplexor driver
ip_gre: GRE over IPv4 tunneling driver
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP: cubic registered
Initializing XFRM netlink socket
NET: Registered protocol family 10
mip6: Mobile IPv6
ip6_tables: (C) 2000-2006 Netfilter Core Team
sit: IPv6 over IPv4 tunneling driver
ip6_gre: GRE over IPv6 tunneling driver
NET: Registered protocol family 17
NET: Registered protocol family 15
can: controller area network core (rev 20120528 abi 9)
NET: Registered protocol family 29
can: raw protocol (rev 20120528)
can: broadcast manager protocol (rev 20120528 t)
can: netlink gateway (rev 20130117) max_hops=1
Bluetooth: RFCOMM TTY layer initialized
Bluetooth: RFCOMM socket layer initialized
Bluetooth: RFCOMM ver 1.11
Bluetooth: BNEP (Ethernet Emulation) ver 1.3
Bluetooth: BNEP filters: protocol multicast
Bluetooth: BNEP socket layer initialized
Bluetooth: HIDP (Human Interface Emulation) ver 1.2
Bluetooth: HIDP socket layer initialized
8021q: 802.1Q VLAN Support v1.8
Key type dns_resolver registered
cpu cpu0: dev_pm_opp_get_opp_count: device OPP not found (-19)
ThumbEE CPU extension supported.
VDD_SD1: disabling
regulator-dummy: disabling
imx mcc test is registered.
using random self ethernet address
using random host ethernet address
usb0: HOST MAC 46:e1:a2:ca:c1:d2
usb0: MAC 66:97:ef:7a:41:1e
g_cdc gadget: CDC Composite Gadget, version: King Kamehameha Day 2008
g_cdc gadget: g_cdc ready
input: gpio-keys as /devices/gpio-keys/input/input1
bmic_rtc 3-0011: setting system clock to 2016-11-23 07:17:12 UTC (1479885432)
ALSA device list:
  No soundcards found.
Warning: unable to open an initial console.
Freeing unused kernel memory: 2548K (8102c000 - 812a9000)
systemd-udevd[155]: starting version 215
random: systemd-udevd: uninitialized urandom read (16 bytes read, 28 bits of entropy available)
(stk) :ldisc installation timeout
(stk) :ldisc_install = 0wlcore: wl18xx HW: 183x or 180x, PG 2.2 (ROM 0x11)
wlcore: loaded
EXT4-fs (mmcblk2p2): mounted filesystem with ordered data mode. Opts: (null)
random: systemd: uninitialized urandom read (16 bytes read, 75 bits of entropy available)
systemd[1]: systemd 215 running in system mode. (+PAM +AUDIT +SELINUX +IMA +SYSVINIT +LIBCRYPTSETUP
```

```
+GCRYPT +ACL +XZ -SECCOMP -APPARMOR)
systemd[1]: Detected architecture 'arm'.

Welcome to Debian GNU/Linux 8 (jessie)!

systemd[1]: Set hostname to <armadillo>.
random: systemd-sysv-ge: uninitialized urandom read (16 bytes read, 78 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 81 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 81 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 81 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 82 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 82 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 82 bits of entropy available)
random: systemd: uninitialized urandom read (16 bytes read, 84 bits of entropy available)
systemd[1]: Cannot add dependency job for unit display-manager.service, ignoring: Unit display-
manager.service failed to load: No such file or directory.
systemd[1]: Starting Forward Password Requests to Wall Directory Watch.
systemd[1]: Started Forward Password Requests to Wall Directory Watch.
systemd[1]: Expecting device dev-ttymxc4.device...
         Expecting device dev-ttymxc4.device...
systemd[1]: Starting Remote File Systems (Pre).
[  OK  ] Reached target Remote File Systems (Pre).
systemd[1]: Reached target Remote File Systems (Pre).
systemd[1]: Starting Arbitrary Executable File Formats File System Automount Point.
[  OK  ] Set up automount Arbitrary Executable File Formats F...utomount Point.
systemd[1]: Set up automount Arbitrary Executable File Formats File System Automount Point.
systemd[1]: Starting Encrypted Volumes.
[  OK  ] Reached target Encrypted Volumes.
systemd[1]: Reached target Encrypted Volumes.
systemd[1]: Starting Dispatch Password Requests to Console Directory Watch.
systemd[1]: Started Dispatch Password Requests to Console Directory Watch.
systemd[1]: Starting Paths.
[  OK  ] Reached target Paths.
systemd[1]: Reached target Paths.
systemd[1]: Starting Swap.
[  OK  ] Reached target Swap.
systemd[1]: Reached target Swap.
systemd[1]: Starting Root Slice.
[  OK  ] Created slice Root Slice.
systemd[1]: Created slice Root Slice.
systemd[1]: Starting User and Session Slice.
[  OK  ] Created slice User and Session Slice.
systemd[1]: Created slice User and Session Slice.
systemd[1]: Starting /dev/initctl Compatibility Named Pipe.
[  OK  ] Listening on /dev/initctl Compatibility Named Pipe.
(stk) : timed out waiting for ldisc to be un-installed
systemd[1]: Listening on /dev/initctl Compatibility Named Pipe.
systemd[1]: Starting Delayed Shutdown Socket.
[  OK  ] Listening on Delayed Shutdown Socket.
systemd[1]: Listening on Delayed Shutdown Socket.
systemd[1]: Starting Journal Socket (/dev/log).
[  OK  ] Listening on Journal Socket (/dev/log).
systemd[1]: Listening on Journal Socket (/dev/log).
systemd[1]: Starting udev Kernel Socket.
[  OK  ] Listening on udev Kernel Socket.
systemd[1]: Listening on udev Kernel Socket.
systemd[1]: Starting udev Control Socket.
[  OK  ] Listening on udev Control Socket.
```

```
(stk) :ldisc_install = 1
systemd[1]: Listening on udev Control Socket.
systemd[1]: Starting Journal Socket.
[  OK  ] Listening on Journal Socket.
systemd[1]: Listening on Journal Socket.
systemd[1]: Starting System Slice.
[  OK  ] Created slice System Slice.
systemd[1]: Created slice System Slice.
systemd[1]: Starting system-getty.slice.
[  OK  ] Created slice system-getty.slice.
systemd[1]: Created slice system-getty.slice.
systemd[1]: Starting system-serial\x2dgetty.slice.
[  OK  ] Created slice system-serial\x2dgetty.slice.
systemd[1]: Created slice system-serial\x2dgetty.slice.
systemd[1]: Starting Increase datagram queue length...
         Starting Increase datagram queue length...
systemd[1]: Mounting Debug File System...
         Mounting Debug File System...
systemd[1]: Starting udev Coldplug all Devices...
         Starting udev Coldplug all Devices...
systemd[1]: Mounted POSIX Message Queue File System.
systemd[1]: Started Set Up Additional Binary Formats.
systemd[1]: Started Create list of required static device nodes for the current kernel.
systemd[1]: Starting Create Static Device Nodes in /dev...
         Starting Create Static Device Nodes in /dev...
systemd[1]: Starting Load Kernel Modules...
         Starting Load Kernel Modules...
systemd[1]: Mounted Huge Pages File System.
systemd[1]: Starting LSB: Start User Mode Init manager daemons...
         Starting LSB: Start User Mode Init manager daemons...
systemd[1]: Starting Slices.
[  OK  ] Reached target Slices.
systemd[1]: Reached target Slices.
systemd[1]: Starting Remount Root and Kernel File Systems...
         Starting Remount Root and Kernel File Systems...
systemd[1]: Expecting device dev-mtdblock1.device...
         Expecting device dev-mtdblock1.device...
[  OK  ] Mounted Debug File System.
systemd[1]: Mounted Debug File System.
[  OK  ] Started Increase datagram queue length.
systemd[1]: Started Increase datagram queue length.
[  OK  ] Started Create Static Device Nodes in /dev.
systemd[1]: Started Create Static Device Nodes in /dev.
[  OK  ] Started Load Kernel Modules.
systemd[1]: Started Load Kernel Modules.
[  OK  ] Started Remount Root and Kernel File Systems.
systemd[1]: Started Remount Root and Kernel File Systems.
[  OK  ] Started udev Coldplug all Devices.
systemd[1]: Started udev Coldplug all Devices.
[  OK  ] Started LSB: Start User Mode Init manager daemons.
systemd[1]: Started LSB: Start User Mode Init manager daemons.
(stc): st_tty_open
(stk) :line discipline installed(stk) :ti-connectivity/TIInit_11.8.32.bts
(stk) :change remote baud rate command in firmware(stk) :skipping the wait event of change remote
baudsystemd[1]: Started Various fixups to make systemd work better on Debian.
systemd[1]: Starting Load/Save Random Seed...

         Starting Load/Save Random Seed...
```

```
systemd[1]: Mounting FUSE Control File System...
         Mounting FUSE Control File System...
systemd[1]: Starting Apply Kernel Variables...
         Starting Apply Kernel Variables...
systemd[1]: Mounting Configuration File System...
         Mounting Configuration File System...
systemd[1]: Starting udev Kernel Device Manager...
         Starting udev Kernel Device Manager...
systemd[1]: Starting Local File Systems (Pre).
[  OK  ] Reached target Local File Systems (Pre).
systemd-udevd[257]: starting version 215
systemd[1]: Reached target Local File Systems (Pre).
systemd[1]: Starting Local File Systems.
[  OK  ] Reached target Local File Systems.
systemd[1]: Reached target Local File Systems.
systemd[1]: Starting Create Volatile Files and Directories...
         Starting Create Volatile Files and Directories...
systemd[1]: Starting Remote File Systems.
[  OK  ] Reached target Remote File Systems.
systemd[1]: Reached target Remote File Systems.
systemd[1]: Starting Syslog Socket.
[  OK  ] Listening on Syslog Socket.
systemd[1]: Listening on Syslog Socket.
systemd[1]: Starting Journal Service...
         Starting Journal Service...
[  OK  ] Started Journal Service.
systemd[1]: Started Journal Service.
         Starting Trigger Flushing of Journal to Persistent Storage...
[  OK  ] Mounted Configuration File System.
[  OK  ] Mounted FUSE Control File System.
[  OK  ] Started udev Kernel Device Manager.
[  OK  ] Started Load/Save Random Seed.
[  OK  ] Started Apply Kernel Variables.
[  OK  ] Started Create Volatile Files and Directories.
random: nonblocking pool is initialized
[  OK  ] Started Trigger Flushing of Journal to Persistent Storage.
(stc): add_channel_to_table: id 4
(stc): add_channel_to_table: id 2
(stc): add_channel_to_table: id 3
[  OK  ] Found device /dev/ttymxc4.
systemd-journald[266]: Received request to flush runtime journal from PID 1
[  OK  ] Found device /dev/mtdblock1.
         Starting Trigger Flushing of Journal to Persistent Storage...
[  OK  ] Created slice system-systemd\x2drfkill.slice.
         Starting Load/Save RF Kill Switch Status of rfkill0...
         Starting Load/Save RF Kill Switch Status of rfkill1...
         Mounting /opt/license...
         Starting Update UTMP about System Boot/Shutdown...
         Starting LSB: Raise network interfaces....
         Starting Copy rules generated while the root was ro...
[  OK  ] Mounted /opt/license.
[  OK  ] Started Load/Save RF Kill Switch Status of rfkill0.
[  OK  ] Started Load/Save RF Kill Switch Status of rfkill1.
[  OK  ] Started Copy rules generated while the root was ro.
systemd-journald[266]: Received request to flush runtime journal from PID 1
[  OK  ] Started Trigger Flushing of Journal to Persistent Storage.
[  OK  ] Started Update UTMP about System Boot/Shutdown.
[  OK  ] Started LSB: Raise network interfaces..
```

```
[  OK  ] Reached target Network.
[  OK  ] Reached target Network is Online.
[  OK  ] Reached target System Initialization.
[  OK  ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Sockets.
[  OK  ] Reached target Timers.
[  OK  ] Reached target Basic System.
         Starting Bluetooth service...
         Starting Regular background program processing daemon...
[  OK  ] Started Regular background program processing daemon.
         Starting Modem Manager...
         Starting Network Manager...
         Starting Connection Recover...
         Starting Lighttpd Daemon...
         Starting Restore /etc/resolv.conf if the system cras...s shut down....
         Starting /etc/rc.local Compatibility...
         Starting Login Service...
         Starting LSB: Load kernel modules needed to enable cpufreq scaling...
         Starting LSB: exim Mail Transport Agent...
         Starting Avahi mDNS/DNS-SD Stack...
         Starting D-Bus System Message Bus...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Avahi mDNS/DNS-SD Stack.
[  OK  ] Started Bluetooth service.
[  OK  ] Reached target Bluetooth.
         Starting System Logging Service...
         Starting Permit User Sessions...
[  OK  ] Started Connection Recover.
[  OK  ] Started Restore /etc/resolv.conf if the system crash...was shut down..
[  OK  ] Started LSB: Load kernel modules needed to enable cpufreq scaling.
[  OK  ] Started Permit User Sessions.
[  OK  ] Started System Logging Service.
[  OK  ] Started Login Service.
         Starting Hostname Service...
         Starting Authenticate and Authorize Users to Run Privileged Tasks...
         Starting LSB: set CPUFreq kernel parameters...
[  OK  ] Started LSB: set CPUFreq kernel parameters.
[  OK  ] Started Hostname Service.
[  OK  ] Started Authenticate and Authorize Users to Run Privileged Tasks.
[  OK  ] Started Modem Manager.
[  OK  ] Started Network Manager.
[  OK  ] Started Lighttpd Daemon.
fec 30bf0000.ethernet eth0: Freescale FEC PHY driver [SMSC LAN8710/LAN8720]                ⏎
(mii_bus:phy_addr=30bf0000.etherne:00, irq=-1)
IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
usb 2-1: new high-speed USB device number 2 using ci_hdrc
IPv6: ADDRCONF(NETDEV_UP): usb0: link is not ready
[  OK  ] Started LSB: exim Mail Transport Agent.
wlcore: PHY firmware version: Rev 8.2.0.0.195
wlcore: firmware booted (Rev 8.8.0.0.13)
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
cdc_ether 2-1:1.0 usb1: register 'cdc_ether' at usb-ci_hdrc.1-1, CDC Ethernet Device,    ⏎
02:80:79:98:77:20
cdc_acm 2-1:1.2: This device cannot do calls on its own. It is not a modem.
         Starting WPA supplicant...
cdc_acm 2-1:1.2: ttyACM0: USB ACM device
[  OK  ] Started WPA supplicant.
```

```
[  OK  ] Started /etc/rc.local Compatibility.
         Starting change status LED...
         Starting Getty on tty1...
[  OK  ] Started Getty on tty1.
         Starting Serial Getty on ttymxc4...
[  OK  ] Started Serial Getty on ttymxc4.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started change status LED.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Debian GNU/Linux 8 armadillo ttymxc4

armadillo login:
```

図 5.3 起動ログ

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。「表 5.1. シリアルコンソールログイン時のユーザ名とパスワード」に示すユーザでログインすることができます。

表 5.1 シリアルコンソールログイン時のユーザ名とパスワード

ユーザ名	パスワード	権限
root	root	root ユーザ
atmark	atmark	一般ユーザ

初めて機器を接続したときは、必ず以下の手順に従って、初期パスワードを変更してください。

1. root でログイン

   初期パスワードを変更します。

```
[armadillo ~]# passwd
Enter new UNIX password: # 新しいパスワードを入力
Retype new UNIX password: # 再入力
```

2. atmark でログイン

   初期パスワードを変更します。

```
[armadillo ~]$ passwd
Enter new UNIX password: # 新しいパスワードを入力
Retype new UNIX password: # 再入力
[armadillo ~]$
```

Armadillo-IoT G3L はネットワークに接続されることを前提としている機器ですので、 初期パスワードのままご利用になるとセキュリティリスクが

非常に高まります。 セキュリティ強度の高いパスワードに変更し、その後も適切なパスワード運用を行うことを強くお勧めします。

5.3. 時刻の設定

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。システムクロックが正しく設定されていない場合、SSL 通信が行えないなどの問題が発生します。

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[MMDDhhmmYYYY.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 5.2 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
YYYY	年(省略可)
ss	秒(省略可)

2015 年 6 月 2 日 12 時 34 分 56 秒に設定する例を次に示します。

```
[armadillo ~]# date ❶
Sat Jan  1 09:00:00 JST 2000
[armadillo ~]# date 060212342015.56 ❷
Tue Jun  2 12:34:56 JST 2015
[armadillo ~]# date ❸
Tue Jun  2 12:34:57 JST 2015
```

❶ 現在のシステムクロックを表示します。

❷ システムクロックを設定します。

❸ システムクロックが正しく設定されていることを確認します。

図 5.4 システムクロックを設定

Armadillo-IoT が接続しているネットワーク内にタイムサーバーがある場合は、NTP(Network Time Protocol)クライアントを利用してシステムクロックを設定することができます。

```
[armadillo ~]# ntpdate [NTP SERVER]
2 Jun 12:34:56 ntpdate[742]: adjust time server x.x.x.x offset 0.004883
 sec
```

```
[armadillo ~]# date
Tue Jun  2 12:34:57 JST 2015
```

5.4. debian のユーザを管理する

1. ユーザを作成

   例として guest というユーザを作成します。

```
[armadillo ~]# adduser guest
Adding user `[user_name]' ...
Adding new group `guest' (1001) ...
Adding new user `guest' (1001) with group `guest' ...
Creating home directory `/home/guest' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: # パスワードを入力
Retype new UNIX password: # 再入力
passwd: password updated successfully
Changing the user information for guest
Enter the new value, or press ENTER for the default
        Full Name []: # Enter を入力
        Room Number []: # Enter を入力
        Work Phone []: # Enter を入力
        Home Phone []: # Enter を入力
        Other []: # Enter を入力
Is the information correct? [Y/n] # Enter を入力
```

2. パスワードの変更

   例として guest ユーザのパスワードを変更します。

```
[armadillo ~]# passwd guest
Enter new UNIX password: # 新しいパスワードを入力
Retype new UNIX password: # 再入力
```

3. sudo を許可する

   例として guest ユーザに sudo を許可します。vi の使い方については、「4.7. vi エディタの使用方法」を参照にしてください。

```
[armadillo ~]# visudo
...
# User privilege specification
root    ALL=(ALL:ALL) ALL
guest   ALL=(ALL:ALL) ALL # この行を追加します
...
```

4. ユーザを削除

   例として guest ユーザを削除します。

```
[armadillo ~]# userdel guest
```

ホームディレクトリも合わせて消したいときは、"r"オプションをつけます。

```
[armadillo ~]# userdel -r guest
```

5.5. 終了方法

安全に終了させる場合は、次のようにコマンドを実行し、「Power down」と表示され、ユーザー LED3、ユーザー LED4、ユーザー LED5 全てが消灯したことを確認してから電源を切断します。ユーザー LED の位置については、「図 7.42. ユーザー LED の位置」を参照してください

また、SW2 を 10 秒間長押しすることで、終了することも可能です。

```
[armadillo ~]# poweroff
         Starting Synchronise Hardware Clock to System Clock...
[  OK  ] Stopped target Bluetooth.
         Stopping User Manager for UID 0...
         Stopping WPA supplicant...
         Stopping Hostname Service...
         Stopping Authenticate and Authorize Users to Run Privileged Tasks...
         Stopping Bluetooth service...
(stc): remove_channel_from_table: id 3
(stc): remove_channel_from_table: id 2
(stc): remove_channel_from_table: id 4
(stc):  all chnl_ids unregistered [
  OK  (stk) :ldisc_install = 0] Stopped target Graphical Interfa(stc): st_tty_close ce.
[  OK  ] Stopped target Multi-User System.
         Stopping Connection Recover...
         Stopping Modem Manager...
         Stopping Network Manager...
         Stopping Lighttpd Daemon...
         Stopping Regular background program processing daemon...

IPv6: ADDRCONF(NETDEV_UP): usb0: link is not ready
         Stopping input event poweroff daemon...
wlcore: down
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[  OK  ] Stopped target Login Prompts.
         Stopping Getty on tty1...
         Stopping Serial Getty on ttymxc4...
         Stopping LSB: exim Mail Transport Agent...
         Stopping LSB: set CPUFreq kernel parameters...
         Stopping Avahi mDNS/DNS-SD Stack...
         Stopping D-Bus System Message Bus...
         Stopping System Logging Service...
[  OK  ] Stopped Bluetooth service.
[  OK  ] Stopped Modem Manager.
[  OK  ] Stopped Network Manager.
```

```
[  OK  ] Stopped Regular background program processing daemon.
[  OK  ] Stopped Avahi mDNS/DNS-SD Stack.
[  OK  ] Stopped D-Bus System Message Bus.
[  OK  ] Stopped System Logging Service.
[  OK  ] Stopped Hostname Service.
[  OK  ] Stopped Authenticate and Authorize Users to Run Privileged Tasks.
[  OK  ] Stopped Lighttpd Daemon.
[  OK  ] Stopped Getty on tty1.
[  OK  ] Stopped Serial Getty on ttymxc4.
[  OK  ] Stopped WPA supplicant.
[  OK  ] Stopped User Manager for UID 0.
[  OK  ] Stopped Connection Recover.
[  OK  ] Stopped input event poweroff daemon.
[  OK  ] Stopped LSB: exim Mail Transport Agent.
[  OK  ] Stopped LSB: set CPUFreq kernel parameters.
         Stopping LSB: Load kernel modules needed to enable cpufreq scaling...
[  OK  ] Stopped target Network is Online.
         Stopping Login Service...
[  OK  ] Removed slice user-0.slice.
[  OK  ] Removed slice system-serial\x2dgetty.slice.
[  OK  ] Removed slice system-getty.slice.
         Stopping /etc/rc.local Compatibility...
[  OK  ] Stopped /etc/rc.local Compatibility.
         Stopping Permit User Sessions...
[  OK  ] Stopped Login Service.
[  OK  ] Stopped LSB: Load kernel modules needed to enable cpufreq scaling.
[  OK  ] Stopped Permit User Sessions.
[  OK  ] Stopped target Network.
[  OK  ] Stopped target Remote File Systems.
[  OK  ] Stopped target Remote File Systems (Pre).
[  OK  ] Stopped target Basic System.
[  OK  ] Stopped target Slices.
[  OK  ] Stopped target Paths.
[  OK  ] Stopped target Timers.
[  OK  ] Stopped target Sockets.
[  OK  ] Closed Avahi mDNS/DNS-SD Stack Activation Socket.
[  OK  ] Closed Syslog Socket.
[  OK  ] Closed D-Bus System Message Bus Socket.
[  OK  ] Stopped target System Initialization.
         Stopping Load/Save RF Kill Switch Status of rfkill0...
         Stopping Load/Save RF Kill Switch Status of rfkill1...
[  OK  ] Stopped target Encrypted Volumes.
         Stopping Update UTMP about System Boot/Shutdown...
         Stopping Apply Kernel Variables...
[  OK  ] Stopped Apply Kernel Variables.
         Stopping Load Kernel Modules...
[  OK  ] Stopped Load Kernel Modules.
         Stopping LSB: Raise network interfaces....
         Stopping LSB: Start User Mode Init manager daemons...
[  OK  ] Stopped target Swap.
[  OK  ] Removed slice User and Session Slice.
[  OK  ] Stopped Load/Save RF Kill Switch Status of rfkill0.
[  OK  ] Stopped Load/Save RF Kill Switch Status of rfkill1.
[  OK  ] Stopped Update UTMP about System Boot/Shutdown.
[  OK  ] Stopped LSB: Raise network interfaces..
[  OK  ] Stopped LSB: Start User Mode Init manager daemons.
         Stopping Load/Save Random Seed...
         Stopping Create Volatile Files and Directories...
```

```
[  OK  ] Stopped Create Volatile Files and Directories.
[  OK  ] Stopped target Local File Systems.
         Unmounting /run/user/0...
[  OK  ] Removed slice system-systemd\x2drfkill.slice.
[  OK  ] Stopped Load/Save Random Seed.
[  OK  ] Unmounted /run/user/0.
[  OK  ] Reached target Unmount All Filesystems.
[  OK  ] Stopped target Local File Systems (Pre).
         Stopping Create Static Device Nodes in /dev...
[  OK  ] Stopped Create Static Device Nodes in /dev.
         Stopping Remount Root and Kernel File Systems...
[  OK  ] Stopped Remount Root and Kernel File Systems.
[  OK  ] Started Synchronise Hardware Clock to System Clock.
[  OK  ] Reached target Shutdown.
systemd-shutdown[1]: Sending SIGTERM to remaining processes...
systemd-journald[270]: Received SIGTERM from PID 1 (systemd-shutdow).
systemd-shutdown[1]: Sending SIGKILL to remaining processes...
systemd-shutdown[1]: Unmounting file systems.
systemd-shutdown[1]: Unmounting /sys/kernel/config.
systemd-shutdown[1]: Unmounting /sys/fs/fuse/connections.
systemd-shutdown[1]: Unmounting /sys/kernel/debug.
EXT4-fs (mmcblk2p2): re-mounted. Opts: (null)
EXT4-fs (mmcblk2p2): re-mounted. Opts: (null)
EXT4-fs (mmcblk2p2): re-mounted. Opts: (null)
systemd-shutdown[1]: All filesystems unmounted.
systemd-shutdown[1]: Deactivating swaps.
systemd-shutdown[1]: All swaps deactivated.
systemd-shutdown[1]: Detaching loop devices.
systemd-shutdown[1]: All loop devices detached.
systemd-shutdown[1]: Detaching DM devices.
systemd-shutdown[1]: All DM devices detached.
systemd-shutdown[1]: Powering off.
imx2-wdt 30280000.wdog: Device shutdown: Expect reboot!
reboot: Power down
```

図 5.5 終了方法

ストレージにデータを書き込んでいる途中に電源を切断した場合、ファイルシステム、及び、データが破損する恐れがあります。ストレージをアンマウントしてから電源を切断するようにご注意ください。

6. ソフトウェアの更新と初期化

Armadillo-IoT G3L は工場出荷状態で、動作確認に必要な全てのソフトウェアが書き込まれています。 しかし、ソフトウェアはセキュリティアップデートや新機能の追加によって随時アップデートを実施しているため、 開発、評価を行う際は最新バージョンのソフトウェアを利用いただくことを推奨します。

本章では Armadillo のソフトウェアを最新の状態に更新する方法と、ソフトウェアを初期化する方法を説明します。

本章で使用するブートローダーイメージや Linux カーネルイメージなどは、開発セット付属の DVD に収録されています。 最新版のファイルは"Armadillo サイト"からダウンロードすることができます。

> **Armadillo サイト - Armadillo-IoT G3L ドキュメント・ダウンロード**
>
> http://armadillo.atmark-techno.com/armadillo-iot-g3l/downloads

6.1. ソフトウェアを最新の状態に更新する

工場出荷状態の Armadillo に書き込まれているイメージファイルは、最新版ではない可能性があります。本章では Armadillo のソフトウェアを最新の状態に更新する方法を説明します。

ソフトウェアと書き込み先の対応を次に示します。

表 6.1 ソフトウェアと書き込み先の対応

名称	ファイル名	ストレージ	デバイスファイル
ブートローダーイメージ	u-boot-x1-*[version]*.bin	QSPI フラッシュメモリ	/dev/mtdblock0
Linux カーネルイメージ	uImage-x1-*[version]*	eMMC	/dev/mmcblk2p1
Device Tree Blob	armadillo_iotg_g3l-*[version]*.dtb		/dev/mmcblk2p1
Debian GNU/Linux ルートファイルシステム	debian-jessie-armhf_aiotg3l_*[version]*.tar.gz		/dev/mmcblk2p2

6.1.1. ブートローダーイメージの更新

ブートローダーイメージの更新方法を次に示します。MTD のブロックデバイスに直接イメージファイルを書き込むことで行います。

```
[armadillo ~]# dd if=u-boot-x1-[version].bin of=/dev/mtdblock0 ❶
282+1 records in
282+1 records out
288816 bytes (289 kB) copied, 5.4582 s, 52.9 kB/s
[armadillo ~]$ sync
```

❶　MTD のブロックデバイスの先頭からブートローダーイメージを書き込みます。

6.1.2. Linux カーネルイメージの更新

Linux カーネルイメージの更新方法を次に示します。

```
[armadillo ~]# mount -t vfat /dev/mmcblk2p1 /mnt ❶
[armadillo ~]# cp uImage-x1-[version] /mnt/uImage ❷
[armadillo ~]# umount /mnt ❸
```

❶ eMMC の第 1 パーティションを/mnt/ディレクトリにマウントします。

❷ Linux カーネルイメージを/mnt/ディレクトリにコピーします。

❸ /mnt/ディレクトリにマウントした eMMC の第 1 パーティションをアンマウントします。

6.1.3. DTB の更新

DTB の更新方法を次に示します。

```
[armadillo ~]# mount -t vfat /dev/mmcblk2p1 /mnt ❶
[armadillo ~]# cp armadillo_iotg_g3l-[version].dtb /mnt/armadillo_iotg_g3l.dtb ❷
[armadillo ~]# umount /mnt ❸
```

❶ eMMC の第 1 パーティションを/mnt/ディレクトリにマウントします。

❷ DTB を/mnt/ディレクトリにコピーします。

❸ /mnt/ディレクトリにマウントした eMMC の第 1 パーティションをアンマウントします。

6.1.4. Debian GNU/Linux ルートファイルシステムの更新

Debian GNU/Linux ルートファイルシステムを更新する方法は 2 つあります。Debian パッケージをアップデートする方法と、ルートファイルシステムをすべて書き換える方法です。

Debian パッケージのセキュリティアップデートやバグフィックスを Armadillo に適用したい場合は「6.1.4.1. Debian パッケージのアップデート」を行ってください。 Armadillo 上のルートファイルシステムを工場出荷時相当に書き換えたい場合は 「6.1.4.2. ルートファイルシステムの書き換え」 を行ってください。

6.1.4.1. Debian パッケージのアップデート

apt-get コマンドを使用して Debian パッケージのアップデートを行います。apt-get update でローカルに持っているパッケージのリストを最新のものに更新し、apt-get upgrade で更新可能なパッケージをすべてインストールします。

apt-get コマンドを使用するには、ネットワーク（インターネット）に接続されている必要があります。ネットワークの設定方法については「7.2. ネットワーク」を参照してください。

```
[armadillo ~]# apt-get update
[armadillo ~]# apt-get upgrade
```

図 6.1 Debian パッケージのアップデート

6.1.4.2. ルートファイルシステムの書き換え

eMMC 上のルートファイルシステムを書き換える手順を次に示します。

手順 6.1 eMMC 上のルートファイルシステムを書き換える

1. eMMC 上のルートファイルシステムを書き換えるには、SD ブートを行う必要があります。必ず SD ブートで起動した事を確認の上、以降の手順を実施してください。ブートディスクの作成方法や SD ブートの実行方法ついては「15. SD ブートの活用」を参照してください。

2. Debian GNU/Linux ルートファイルシステムアーカイブを準備しておきます。

```
[armadillo ~]# ls
debian-jessie-armhf_aiotg-g3l_[version].tar.gz
```

3. ルートファイルシステムを eMMC の第 2 パーティションに再構築します。

```
[armadillo ~]# mkfs.ext4 /dev/mmcblk2p2  ❶
mke2fs 1.42.12 (29-Aug-2014)
/dev/mmcblk2p2 contains a ext4 file system
    last mounted on /root on Thu Jan  1 09:00:07 1970
Proceed anyway? (y,n) y  ❷
...[省略]...

[armadillo ~]# mount -t ext4 /dev/mmcblk2p2 /mnt  ❸
[armadillo ~]# tar zxf debian-jessie-armhf_aiotg-g3l_[version].tar.gz -C /mnt  ❹
[armadillo ~]# umount /mnt  ❺
```

❶ eMMC の第 2 パーティションのファイルシステムを再構築します。

❷ y に続き ENTER を入力します。

❸ eMMC の第 2 パーティションを/mnt/ディレクトリにマウントします。

❹ ルートファイルシステムアーカイブを/mnt/ディレクトリに展開します。

❺ /mnt/ディレクトリにマウントした eMMC の第 2 パーティションをアンマウントします。

4. Armadillo-IoT G3L の内蔵ストレージ(eMMC 及び QSPI フラッシュメモリ)から起動するために、次のようにコマンドを実行し Armadillo を終了させます。「System halted.」と表示されたのを確認してから電源を切断します。

```
[armadillo ~]# halt
```

5. JP1 をオープンにしてから、電源を投入してください。

6. 次のように"boot"コマンドを実行すると、書き換えたルートファイルシステムで起動します。必要に応じて「6.1.4.1. Debian パッケージのアップデート」を行ってください。

```
[armadillo ~]# boot
```

6.2. ソフトウェアを初期化する

インストールディスクを使用すると、内蔵ストレージ上のすべてのソフトウェアをまとめて書き換えることができます。 すべてのソフトウェアを初期化したい場合や、ソフトウェアの問題や作業の誤りによって Armadillo が起動しなくなった場合の復旧方法などにご使用頂けます。

内蔵ストレージに保存されている、すべてのイメージファイルが上書きされるため、既に保存されているデータやアプリケーションなどは削除されます。

特定のイメージのみ書き換えたい場合には 「6.1. ソフトウェアを最新の状態に更新する」 を参照してください。

表 6.2 インストールディスク作成に使用するファイル

ファイル	ファイル名
インストールディスクイメージ	install_disk_sd_*[version]*.img

6.2.1. 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. ATDE に microSD カードを接続します。詳しくは「4.3.2. 取り外し可能デバイスの使用」を参照してください。
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。

```
[PC ~]$ mount
(省略)
/dev/sdb1 on /media/atmark/B18A-3218 type vfat
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=437,iocharse
t=utf8,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)
[PC ~]$ sudo umount /dev/sdb1
```

4. microSD カードにインストールディスクイメージを書き込みます。

```
[PC ~]$ sudo dd if=install_disk_sd_[version].img of=/dev/sdb bs=4M
94+1 レコード入力
94+1 レコード出力
397410304 バイト (397 MB) コピーされました、 45.8441 秒、 8.7 MB/秒
[PC ~]$ sync
```

6.2.2. インストールの実行

1. 電源が切断されていることを確認します。接続されていた場合は、電源を切断してください。

2. USB シリアル変換アダプタを Armadillo-IoT から切断します。USB シリアル変換アダプタを接続した状態でインストールを実行したい場合は、スライドスイッチを「図 4.16. スライドスイッチの設定」の 2 側（オートブートモード）に設定してください。

3. インストールディスクを使用して SD ブートを行います。microSD スロット(メインユニット CON12)にインストールディスクを接続し、JP1 をショートに設定してください。

4. Armadillo に電源を投入します。Armadillo が起動するとインストールが始まり、自動的に eMMC と QSPI が書き換えられます。インストールの実行中は電源を切断しないでください。

5. インストールの進捗状況はユーザー LED で確認することができます。インストールの進捗状況とユーザー LED の状態を次に示します。ユーザー LED の位置については「図 7.42. ユーザー LED の位置」を参照してください。

表 6.3 インストールの進捗状況とユーザー LED の状態

インストールの進捗	ユーザー LED4	ユーザー LED3	ユーザー LED5
起動中	点灯	消灯	消灯
実行中	点灯	間欠点灯	消灯
正常終了	点灯	点灯	点灯
異常終了	間欠点灯	間欠点灯	間欠点灯

6. インストールが終了したら、電源を切断してください。

7. 動作確認方法

7.1. 動作確認を行う前に

工場出荷状態でフラッシュメモリに書き込まれているイメージファイルは、最新版ではない可能性があります。最新版のイメージファイルは、Armadillo サイトからダウンロード可能です。最新版のイメージファイルに書き換えてからのご使用を推奨します。

イメージファイルの書き換え方法は、「6. ソフトウェアの更新と初期化」を参照してください。

7.2. ネットワーク

ここでは、ネットワークの設定方法やネットワークを利用するアプリケーションについて説明します。

7.2.1. 接続可能なネットワーク

Armadillo-IoT は、複数の種類のネットワークに接続することができます。接続可能なネットワークと Linux から使用するネットワークデバイスの対応を次に示します。

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	備考
有線 LAN	eth0	
無線 LAN	wlan0	TI 製 WL1837MOD 搭載
LTE	ttyACM0	Gemalto 製 ELS31-J 搭載

7.2.2. ネットワークの設定方法

Armadillo-IoT ゲートウェイ G3L 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 はデフォルトで eth0(LAN のネットワークインターフェース)が自動で up し、DHCP でネットワーク設定を取得するようになっています。

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。

7.2.2.1. 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

各オブジェクトについての詳しい情報は **man nmcli** を参照してください。

Armadillo-IoT には nmcli の他ユーザーフレンドリーな nmtui もインストールされていますが本書では取り扱いません。

7.2.3. 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。

7.2.3.1. コネクションの一覧

登録されているコネクションの一覧を確認するには、次のようにコマンドを実行します。[1]

```
[armadillo ~]# nmcli connection
NAME                UUID                                  TYPE            DEVICE
Wired connection 1  64e2e184-ede4-4cc6-ab70-0713d7cb0f0b  802-3-ethernet  eth0
```

図 7.2 コネクションの一覧

7.2.3.2. 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、次のようにコマンドを実行します。***[ID]*** には「7.2.3.1. コネクションの一覧」で確認した NAME に表示される値を入力します。

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

図 7.3 コネクションの有効化

```
[armadillo ~]# nmcli connection up "Wired connection 1"
```

図 7.4 コネクションの有効化の例

コネクションを無効化するには、次のようにコマンドを実行します。

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

図 7.5 コネクションの無効化

[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。

7.2.3.3. コネクションの作成

コネクションを作成するには、次のようにコマンドを実行します。

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

図 7.6 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、*[interface name]* にはインターフェース名(デバイス)を入力します。具体的なコネクションの作成方法はそれぞれのデバイスの章で説明します。

/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。これを 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。

7.2.3.4. コネクションの削除

コネクションを削除するには、次のように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

/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。

7.2.3.5. コネクションを修正する

具体的なコネクションの修正方法を紹介します。

wifi の設定情報を nmcli connection modify コマンドで設定情報を編集すると、パスフレーズ情報がリセットされます。 編集する際は、都度パスフレーズも同時に設定してください。 パスフレーズの設定方法は、「7.2.5. 無線 LAN」を参照してください。

ネットワーク接続に関する不明な点については、ネットワークの管理者へ相談してください。

7.2.3.5.1. 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 7.8. 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 delete [ID]
[armadillo ~]# nmcli connection add con-name [ID] type [type] ifname [interface name]
[armadillo ~]# nmcli connection modify [ID] \
ipv4.method manual ipv4.addresses "192.0.2.10/24 192.0.2.1"
```

図 7.8 固定 IP アドレス設定

7.2.3.5.2. DHCP に設定する

DHCP に設定する例を、「図 7.9. DHCP 設定」に示します。

```
[armadillo ~]# nmcli connection delete [ID]
[armadillo ~]# nmcli connection add con-name [ID] type [type] ifname [interface name]
[armadillo ~]# nmcli connection modify [ID] ipv4.method auto
```

図 7.9 DHCP 設定

7.2.3.5.3. DNS サーバーを指定する

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

```
[armadillo ~]# nmcli connection delete [ID]
[armadillo ~]# nmcli connection add con-name [ID] type [type] ifname [interface name]
[armadillo ~]# nmcli connection modify [ID] ipv4.dns 192.0.2.10
```

図 7.10 DNS サーバーの指定

7.2.3.6. 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 7.11 コネクションの修正の反映

7.2.3.7. デバイスの一覧

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、次のようにコマンドを実行します。[2]

```
[armadillo ~]# nmcli device
DEVICE    TYPE      STATE         CONNECTION
eth0      ethernet  connected     Wired connection 1
ttyACM0   gsm       disconnected  --
wlan0     wifi      disconnected  --
gre0      gre       unmanaged     --
gretap0   gretap    unmanaged     --
ip6gre0   ip6gre    unmanaged     --
ip6tnl0   ip6tnl    unmanaged     --
tunl0     ipip      unmanaged     --
lo        loopback  unmanaged     --
sit0      sit       unmanaged     --
ip6_vti0  vti6      unmanaged     --
```

図 7.12 デバイスの一覧

7.2.3.8. デバイスの接続

デバイスを接続するには、次のようにコマンドを実行します。*[ifname]* には「図 7.12. デバイスの一覧」で確認した DEVICE に表示される値を入力します。

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

図 7.13 デバイスの接続

```
[armadillo ~]# nmcli device connect eth0
```

図 7.14 デバイスの接続例

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あるかを確認してください。

7.2.3.9. デバイスの切断

デバイスを切断するには、次のようにコマンドを実行します。

[2] **nmcli device** と **nmcli device status** は等価です。

また、**nmcli device show** から、より詳細な情報を表示することができます。

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

図 7.15 デバイスの切断

7.2.4. 有線 LAN

ここでは有線 LAN の使用方法について説明します。

7.2.4.1. 有線 LAN インターフェース(eth0)のコネクションの作成

有線 LAN インターフェース用のコネクションを作成するには、次のようにコマンドを実行します。

```
[armadillo ~]# nmcli connection add con-name ethernet-eth0 type ethernet ifname eth0
Connection 'ethernet-eth0' (ac491d33-9647-4096-8b91-5c7abcf5850d) successfully added.
```

図 7.16 有線 LAN インターフェース(eth0)のコネクションを作成

7.2.4.2. 有線 LAN のネットワーク設定を変更する

ネットワークの設定方法は「7.2.3.5. コネクションを修正する」を参照してください。コネクションを修正を行った後にはかならず「7.2.3.6. コネクションの修正を反映する」を参考に、修正の反映を行ってください。

7.2.4.3. 有線 LAN の接続を確認する

有線 LAN で正常に通信が可能か確認します。設定を変更した場合、かなら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 4 192.0.2.20
```

図 7.17 有線 LAN の PING 確認

有線 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 の接続確認をする場合は、事前に有線 LAN 以外のコネクションを無効化してください。

7.2.5. 無線 LAN

ここでは、Armadillo-IoT 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。

7.2.5.1. 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ます。

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

図 7.18 無線 LAN アクセスポイントに接続する

7.2.5.2. 無線 LAN(wlan0)のコネクションの作成

「7.2.5.1. 無線 LAN アクセスポイントに接続する」方法は簡単ですが、この方法は DHCP にしか対応していません。そのため、無線 LAN を固定 IP にする場合や、細かなネットワーク設定を行うためにはコネクションから作成する必要があります。

無線 LAN(wlan0)のコネクションの作成するためには、次のようにコマンドを実行します。

```
[armadillo ~]# nmcli connection add con-name wifi-wlan0 type wifi ifname wlan0 ssid [essid] ❶
Connection 'wifi-wlan0' (d3cbb49d-b843-4dbf-94d5-7e7275449e8a) successfully added.
[armadillo ~]# nmcli connection modify wifi-wlan0 \ ❷
802-11-wireless-security.key-mgmt wpa-psk \ ❸
802-11-wireless-security.psk [passphrase]
```

図 7.19 無線 LAN(wlan0)のコネクションの作成

❶ type を wifi に設定し、無線 LAN のコネクションを作成します。

❷ 暗号化キー管理方式を wpa-psk に設定します。

❸ パスフレーズを設定します。

7.2.5.3. 無線 LAN のネットワーク設定を変更する

ネットワークの設定方法は「7.2.3.5. コネクションを修正する」を参照してください。コネクションの修正を行う際は「図 7.19. 無線 LAN(wlan0)のコネクションの作成」を参考にパスフレーズも合わせて設定してください。 また、コネクションを修正を行った後にはかならず「7.2.3.6. コネクションの修正を反映する」を参考に、修正の反映を行ってください。

7.2.5.4. 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 -c 4 192.0.2.20
```

図 7.20 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。

7.2.6. LTE

ここでは、Armadillo-IoT に搭載されている LTE モジュール「Gemalto 製 LTE 通信モジュール ELS31-J」の使用方法について説明します。

LTE モジュール「Gemalto 製 LTE 通信モジュール ELS31-J」はドコモ相互接続性試験を完了しています。

LTE モジュールの制約により工場出荷状態では、LTE 通信網側から Armadillo-IoT に対して、直接 SSH でログインすることができません。LTE 通信網側から直接 SSH でログインを行いたい場合は、別途ご相談ください。

7.2.6.1. 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 microSIM(UIM カード)と APN 情報を準備します。

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

microSIM(UIM カード)の切り欠きを挿入方向と反対側に向け、端子面を上にして挿入してください。

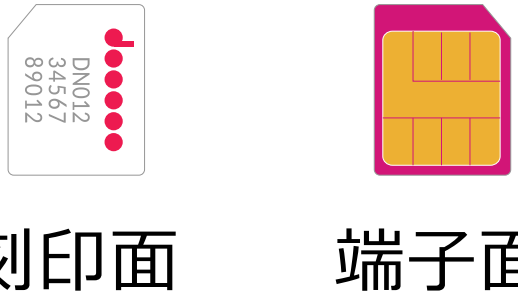


図 7.21 microSIM

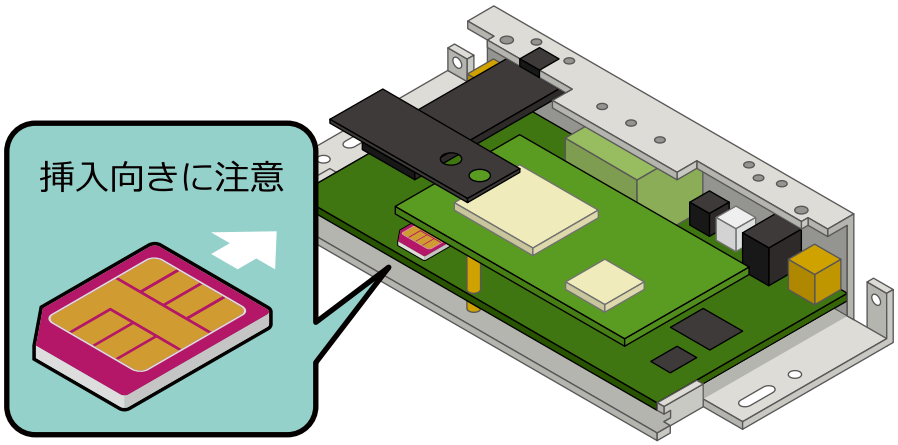


図 7.22 microSIM の取り付け

APN の設定を行うには、次に示す情報が必要です。

- APN
- ユーザー名
- パスワード
- 認証方式(PAP または CHAP)
- PDP Type(IP のみサポート)

7.2.6.2. LTE のコネクションを作成する

「表 7.3. APN 情報設定例」の内容に設定する例を、「図 7.23. LTE のコネクションの作成」に示します。

コネクションの作成後は、自動的にデータ接続が行われます。また、一度コネクションを作成すると起動時は自動的にデータ接続が行われます。

表 7.3 APN 情報設定例

項目	設定
APN	*[apn]*
ユーザー名	*[user]*
パスワード	*[password]*

```
[armadillo ~]# nmcli connection add con-name gsm-ttyACM0 type gsm \
ifname ttyACM0 apn [apn] user [user] password [password]
Connection 'gsm-ttyACM0' (a9e51a2d-bbee-443f-80ba-07b65c3097e8) successfully added.
```

図 7.23 LTE のコネクションの作成

7.2.6.3. データ接続状態の確認

コネクションの作成後、データ接続が完了する前には 40 秒程度かかります。この間、nmcli device コマンドを実行することで接続状態の確認を行うことができます。

```
[armadillo ~]# nmcli device
DEVICE    TYPE      STATE         CONNECTION
ttyACM0   gsm       [status]      --
...(省略)...
```

図 7.24 nmcli device コマンドでの接続状態の確認

代表的な[status]を次に示します。

表 7.4 nmcli device コマンドで取得可能な代表的な status

status
disconnected
connecting
connected

7.2.6.4. LTE で通信確認をする

LTE で正常に通信が可能か確認します。

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 -c 4 www.atmark-techno.com
```

図 7.25 LTE の PING 確認

LTE 以外のコネクションが有効化されている場合、LTE 経由での PING が実行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、適切なルーティング設定を行うか、事前に LTE 以外のコネクションを無効化してください。

7.2.6.5. LTE のデータ通信を終了する

nmcli コマンドでデータ通信の終了を行う前に、LTE の再接続スクリプトを停止します。再接続スクリプトを停止せずにデータ通信の終了を実行した場合、同機能によって再度データ接続が開始されます。

LTE の再接続スクリプトを停止したのち、データ通信を終了します。

```
[armadillo ~]# systemctl stop connection-recover.service
[armadillo ~]# nmcli connection down gsm-ttyACM0
```

図 7.26 データ通信の終了

7.2.6.6. LTE のデータ接続を再開する

データ通信を開始します。

```
[armadillo ~]# nmcli connection up gsm-ttyACM0
[armadillo ~]# systemctl start connection-recover.service
```

図 7.27 データ通信の開始

7.2.6.7. 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。

7.2.6.7.1. サービスの仕様

microSIM が接続されており、NetworkManager の有効な LTE のコネクション設定がされいるとき、定期的にコネクションの状態を監視します。

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。

7.2.6.7.2. 工場出荷状態の設定

工場出荷状態で有効化されており、システム起動時にサービスが自動的に開始されます。300 秒に 1 度コネクションの状態を監視します。PING を実行する宛先は、デフォルトで"8.8.8.8"です。ご利用の環境に合わせて設定ファイル(/etc/connection-recover/gsm-ttyACM0_connection-recover.conf)を適宜変更してください。

7.2.6.7.3. 停止・開始

LTE 再接続サービスを停止するには、次に示すコマンドを実行してください。

```
[armadillo ~]# systemctl stop connection-recover.service
```

図 7.28 LTE 再接続サービスの停止

LTE 再接続スクリプトを開始するには、次に示すコマンドを実行してください。

```
[armadillo ~]# systemctl start connection-recover.service
```

図 7.29 LTE 再接続サービスの開始

7.2.6.8. ModemManager について

ここでは ModemMamager と、mmcli について説明します。

Armadillo-IoT 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。

7.2.6.8.1. 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、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -L

Found 1 modems:
    /org/freedesktop/ModemManager1/Modem/0 [Cinterion] ELS31-J
```

図 7.30 認識されているモデムの一覧の取得

7.2.6.8.2. モデムの情報を取得する

モデムの状態を取得するには、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -m 0

/org/freedesktop/ModemManager1/Modem/0 (device id '256a474d480bc596eeaa3b3a2bed156fac53d99d')
  -------------------------
  Hardware |   manufacturer: 'Cinterion'
           |          model: 'ELS31-J'
           |       revision: 'REVISION 4.3.2.1b'
           |      supported: 'gsm-umts, lte'
           |        current: 'gsm-umts, lte'
           |   equipment id: '356778079984978'
  -------------------------
  System   |         device: '/sys/devices/soc/30800000.aips-bus/30b20000.usb/ci_hdrc.1/usb1/1-1'
           |        drivers: 'cdc_acm, cdc_ether'
           |         plugin: 'Cinterion ELS'
           |   primary port: 'ttyACM0'
           |          ports: 'ttyACM0 (at), usb1 (net)'
  -------------------------

(省略)
```

図 7.31 モデムの情報を取得する

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

7.2.6.8.3. microSIM の情報を取得する

microSIM の情報を取得するには、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -m 0
(省略)
-------------------------
SIM     |         path: '/org/freedesktop/ModemManager1/SIM/[number] # [number]を次のコマンドで指定
(省略)
[armadillo ~]# mmcli -i [number]
SIM '/org/freedesktop/ModemManager1/SIM/0'
  ------------------------
  Properties |          imsi : 'XXXXXXXXXXXXXXX'
             |            id : 'XXXXXXXXXXXXXXXXXX'
             |   operator id : 'XXXXX'
             | operator name : 'XXXXXXXXXX'
```

図 7.32 microSIM の情報を取得する

7.2.6.8.4. 回線情報を取得する

回線情報を取得するには、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -m 0
(省略)
-------------------------
Bearers  |          paths: '/org/freedesktop/ModemManager1/Bearer/[number] # [number]を次のコマン ⏎
ドで指定
[armadillo ~]# mmcli -b [number]
Bearer '/org/freedesktop/ModemManager1/Bearer/0'
  ------------------------
  Status             |   connected: 'yes'
                     |   suspended: 'no'
                     |   interface: 'ttyACM0'
                     |  IP timeout: '20'
  -------------------------
  Properties         |         apn: 'XXXXXXXXXX'
                     |     roaming: 'allowed'
                     |     IP type: 'none'
                     |        user: 'XXXX'
                     |    password: 'XXXX'
                     |      number: '*99#'
                     | Rm protocol: 'unknown'
  -------------------------
  IPv4 configuration |   method: 'ppp'
                     |  address: 'unknown'
                     |   prefix: '0'
                     |  gateway: 'unknown'
                     |      DNS: none
  -------------------------
  IPv6 configuration |   method: 'unknown'
```

図 7.33 回線情報を取得する

7.2.7. NetworkManager による設定例

ここでは、「図 7.34. ネットワーク構成図」を例にとって NetworkManager を使った設定例を紹介します。

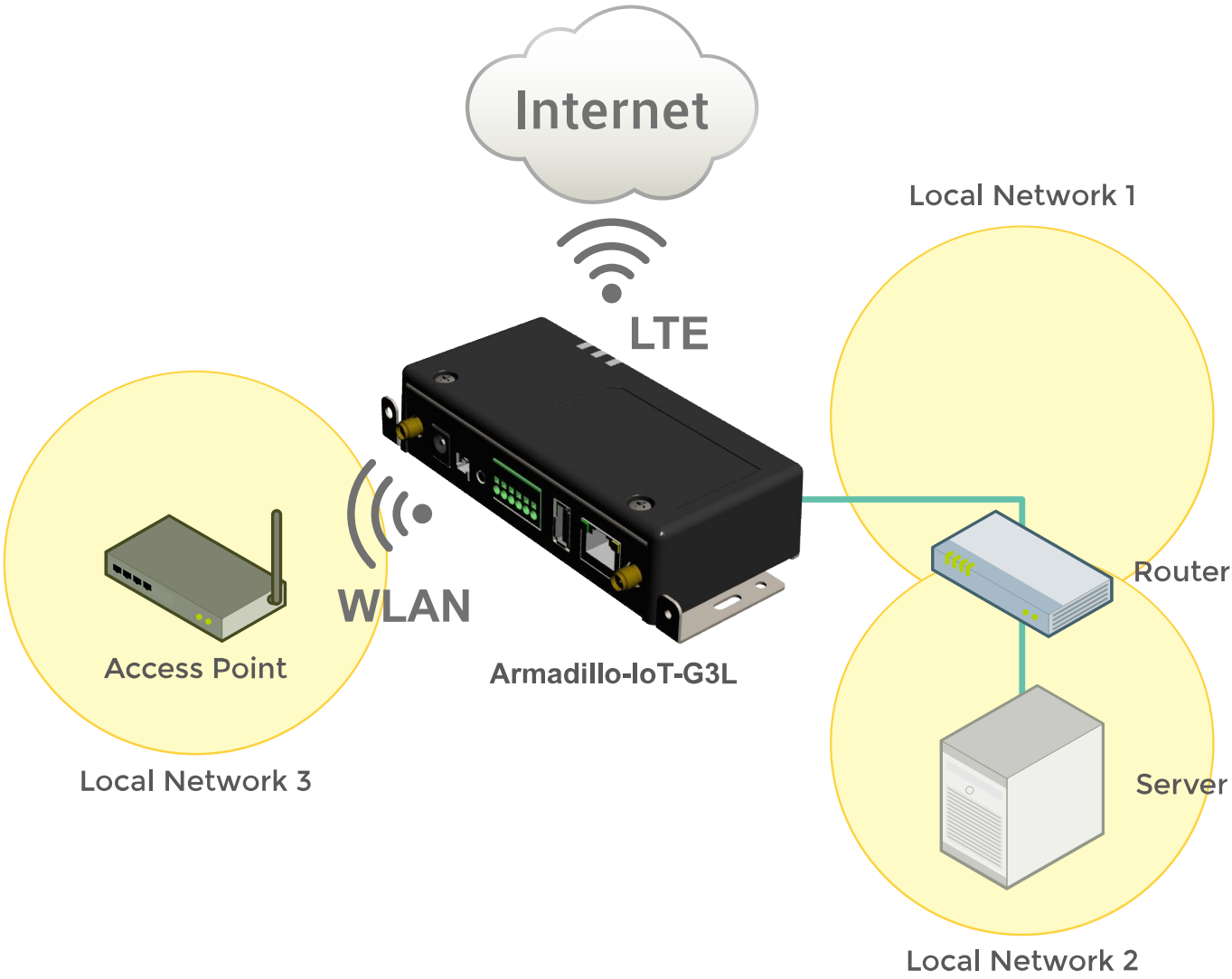


図 7.34 ネットワーク構成図

表 7.5 ネットワークのアドレス情報

ノード名	ネットワークデバイス	IP アドレス	ネットワークアドレス
Armadillo	eth0	192.168.0.2	192.168.0.0/24
	wlan0	172.16.xxx.xxx[a]	172.16.0.0/16[a]
	ttyACM0[b]	xxx.xxx.xxx.xxx[a]	xxx.xxx.xxx.xxx[a]
Router	eth0	192.168.0.1	192.168.0.0/24
	eth1	192.168.10.1	192.168.10.0/24
Server	eth0	192.168.10.2	192.168.10.0/24
Access Point	eth0	172.16.0.1	172.16.0.0/16

[a]DHCP による自動取得
[b]LTE モジュールのネットワークデバイス

7.2.7.1. ネットワーク設定手順

「図 7.34. ネットワーク構成図」に示すネットワークを構築する際のネットワーク設定手順は以下のようになります。なお、作成したコネクションは/etc/NetworkManager/system-connections/に保存され、Armadillo の再起動後も有効化されます。

手順 7.1 ネットワーク設定手順

1. eth0, ttyACM0, wlan0 の状態が disconnected になっていることを確認します。

```
[armadillo ~]# nmcli device
DEVICE    TYPE      STATE         CONNECTION
eth0      ethernet  disconnected  --
ttyACM0   gsm       disconnected  --
wlan0     wifi      disconnected  --
gre0      gre       unmanaged     --
gretap0   gretap    unmanaged     --
ip6gre0   ip6gre    unmanaged     --
ip6tnl0   ip6tnl    unmanaged     --
tunl0     ipip      unmanaged     --
lo        loopback  unmanaged     --
sit0      sit       unmanaged     --
ip6_vti0  vti6      unmanaged     --
```

disconnected 以外の状態になっていた場合、「表 7.6. デバイスの状態を disconnected にする方法」に従い、各デバイスの状態を disconnected にしてください。

表 7.6 デバイスの状態を disconnected にする方法

デバイスの状態	対処方法
unmanaged	/etc/network/interfaces にデバイスの設定が記述されていないか確認してください。記述されていた場合は、その記述を削除してください。
unavailable	LAN ケーブルが抜けていないか確認してください。抜けていた場合、LAN ケーブルを接続してください。
connecting	デバイスを使用したコネクションを有効化する処理が実行されています。「図 7.5. コネクションの無効化」を参照してコネクションを無効化してください。
connected	デバイスを使用してコネクションが有効化されています。「図 7.5. コネクションの無効化」を参照してコネクションを無効化してください。

2. 無線 LAN(wlan0)の設定を行います。

```
[armadillo ~]# nmcli connection add type wifi ifname wlan0 ssid [essid] ❶
[armadillo ~]# nmcli connection modify wifi-wlan0 ipv4.never-default yes ❷
[armadillo ~]# nmcli connection modify wifi-wlan0 \
802-11-wireless-security.key-mgmt wpa-psk \
802-11-wireless-security.psk [passphrase] ❸
[armadillo ~]# nmcli connection down wifi-wlan0 ❹
[armadillo ~]# nmcli connection up wifi-wlan0 ❺
```

❶ 無線 LAN(wlan0)のコネクションを作成します。

❷ 無線 LAN(wlan0)のコネクションのデフォルトゲートウェイを無効化します。

❸ 暗号化キー管理方式を wpa-psk に設定し、パスフレーズを設定します。

❹ 修正を反映させるため、一旦、無線 LAN(wlan0)のコネクションを無効化します。

❺ 無線 LAN(wlan0)のコネクションを有効化します。

3. 有線 LAN インターフェース(eth0)の設定を行います。

```
[armadillo ~]# nmcli connection add type ethernet ifname eth0 ❶
[armadillo ~]# nmcli connection modify ethernet-eth0 ipv4.method manual \
ipv4.addresses "192.168.0.2/24" ❷
[armadillo ~]# nmcli connection modify ethernet-eth0 \
ipv4.routes "192.168.10.0/24 192.168.0.1" ❸

[armadillo ~]# nmcli connection modify ethernet-eth0 ipv4.never-default yes ❹
[armadillo ~]# nmcli connection down ethernet-eth0 ❺
[armadillo ~]# nmcli connection up ethernet-eth0 ❻
```

❶ 有線 LAN インターフェース(eth0)のコネクションを作成します。

❷ 有線 LAN インターフェース(eth0)のコネクションに固定 IP アドレスを設定します。

❸ 有線 LAN インターフェース(eth0)のコネクションに経路情報を追加します。

❹ 有線 LAN インターフェース(eth0)のコネクションのデフォルトゲートウェイを無効化します。

❺ 修正を反映させるため、一旦、有線 LAN インターフェース(eth0)のコネクションを無効化します。

❻ 有線 LAN インターフェース(eth0)のコネクションを有効化します。

4. LTE(ttyACM0)の設定を行います。

```
[armadillo ~]# nmcli connection add type gsm ifname ttyACM0 apn [apn] user [user] ⏎
password [password] ❶
```

❶ LTE(ttyACM0)のコネクションを作成します。

5. eth0, ttyACM0, wlan0 の状態が connected になっていることを確認します。

```
[armadillo ~]# nmcli device
DEVICE    TYPE      STATE      CONNECTION
eth0      ethernet  connected  ethernet-eth0
ttyACM0   gsm       connected  gsm-ttyACM0
wlan0     wifi      connected  wifi-wlan0
gre0      gre       unmanaged  --
gretap0   gretap    unmanaged  --
ip6gre0   ip6gre    unmanaged  --
ip6tnl0   ip6tnl    unmanaged  --
tunl0     ipip      unmanaged  --
lo        loopback  unmanaged  --
sit0      sit       unmanaged  --
ip6_vti0  vti6      unmanaged  --
```

6. ルーティングテーブルを確認します。

```
[armadillo ~]# route
Kernel IP routing table
```

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         xxx.xxx.xxx.xxx 0.0.0.0         UG    1024   0        0 ppp0
link-local      *               255.255.0.0     U     1000   0        0 wlan0
172.16.0.0      *               255.255.0.0     U     0      0        0 wlan0
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
192.168.10.0    192.168.0.1     255.255.255.0   UG    1      0        0 eth0
```

7.2.8. ファイアーウォール

Armadillo では、簡易ファイアーウォールが動作しています。設定されている内容を参照するには、「図 7.35. iptables」のようにコマンドを実行してください。

```
[armadillo ~]# iptables --list
```

図 7.35 iptables

7.2.9. ネットワークアプリケーション

工場出荷イメージで利用することができるネットワークアプリケーションについて説明します。

ATDE と Armadillo のネットワーク設定がデフォルト状態であることを想定して記述しています。ネットワーク設定を変更している場合は適宜読み換えてください。

7.2.9.1. HTTP サーバー

Armadillo では、HTTP サーバーが動作しています。ATDE などの PC の Web ブラウザから Armadillo の URL (http://[Armadillo の IP アドレス]/ にアクセスすると、lighttpd のトップページ(index.html)が表示されます。

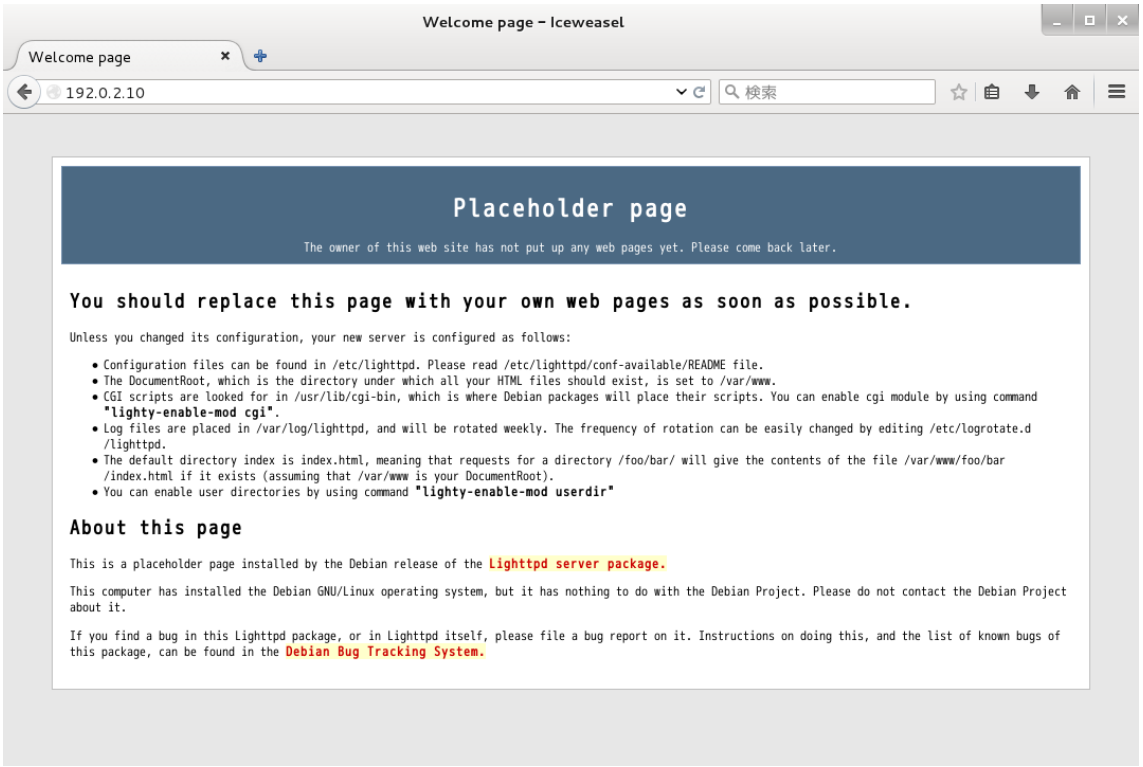


図 7.36 Armadillo トップページ

7.2.10. Armadillo にファイルを転送する

ATDE 上のテキストファイル(hello.txt)を Armadillo へ転送することを例に、ATDE から Armadillo へファイルを転送する方法を紹介します。

7.2.10.1. scp コマンドを使って転送する

1. ATDE から Armadillo へ scp コマンドを使ってファイルを転送するには、Armadillo に SSH Server がインストールされている必要があります。 apt-get コマンドで openssh-server をインストールしてください。 [3]

```
[armadillo ~]# apt-get install openssh-server
```

2. Armadillo の IP アドレスを ip コマンドで確認します。ここで確認した IP アドレスはファイルを転送する時に使用します。

```
[armadillo ~]# ip addr show eth0
     2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
group default qlen 1000
    link/ether 00:0c:29:30:b0:e0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.10/16 brd 192.168.0.255 scope global dynamic eth0
       valid_lft 65913sec preferred_lft 65913sec
```

[3] Armadillo サイトからダウンロード出来る、ルートファイルシステムおよびインストールディスクには SSH Server がインストールされていません。

```
    inet6 fe80::20c:29ff:fe30:b0e0/64 scope link
       valid_lft forever preferred_lft forever
```

3. ATDE 上で転送するテキストファイル(hello.txt)を作成します。

```
[PC ~]$ echo "Hello world." > hello.txt
```

4. 作成したテキストファイルを scp コマンドで転送します。IP アドレスは先ほど ip コマンドで確認したものを入力します。

   scp コマンドを実行するとパスワードが要求されますので、atmark ユーザーのパスワードを入力してください。

```
[PC /]$ scp hello.txt atmark@192.168.0.10:~/
atmark@192.168.0.10's password:
hello.txt                                     100%   13     0.0KB/s   00:00
```

5. Armadillo にファイルが転送されたことを確認します。

```
[armadillo ~]# ls /home/atmark/
hello.txt
[armadillo ~]# cat /home/atmark/hello.txt
Hello world.
```

7.2.11. Wi-SUN

Armadillo-IoT G3L は、「17.15. サブユニット CON2 Wi-SUN モジュールインターフェース」 に ROHM 製 Wi-SUN モジュール「BP35A1」を搭載させることができます。

Wi-SUN モジュールはシリアル接続されているため、TTY デバイスファイル（/dev/ttymxc2）経由でデーター通信を行うことができます。

7.2.11.1. 設定情報を取得する

BP35A1 を制御する例として、設定情報の取得を行います。

手順 7.2 設定情報の取得

1. cu コマンドを実行して/dev/ttymxc2 に接続します。ボーレートは 115200bps です。

```
[armadillo ~]$ cu -l /dev/ttymxc2 -s 115200
Connected.
```

2. SKINFO コマンドを実行すると、BP35A1 の設定情報が表示されます。

```
SKINFO
EINFO FE80:0000:0000:0000:021D:1290:0003:8273 001D129000038273 21 FFFF FFFE
OK
```

3. cu を終了するには、"~."(チルダ「~」に続いてドット「.」)を入力します。

```
Disconnected.
[armadillo ~]$
```

その他の ASCII コマンドや、BP35A1 の詳細な情報については ROHM 製ドキュメントを参照してください。

「ROHM Sub-GHz シリーズ」サポートページ ドキュメントダウンロード | 半導体のローム ROHM

http://micro.rohm.com/jp/download_support/wi-sun

7.3. ストレージ

Armadillo-IoT 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 7.7 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
microSD/microSDHC/ microSDXC カード	/dev/mmcblk* [a]	/dev/mmcblk*p1	microSD インターフェース（メインユニット CON12）
USB フラッシュメモリ	/dev/sd* [b]	/dev/sd*1	USB ホストインターフェース（メインユニット CON11）

[a]microSD/microSDHC/microSDXC カードを接続した場合は、認識された順に mmcblk0 mmcblk1 となります。
[b]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。

7.3.1. ストレージの使用方法

ここでは、microSDHC 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、microSD/microSDHC/microSDXC カードを microSD カードと表記します。

microSDXC カードを使用する場合は、事前に「7.3.2. 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 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 7.37 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[4]。FAT32 ファイルシステムの場合は vfat[5]、EXT4 ファイルシステムの場合は ext4 を指定します。

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/mmcblk0p1、パーティション 2 の場合は/dev/mmcblk0p2 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 スロットに microSDHC カードを挿入した状態で「図 7.38. ストレージのマウント」に示すコマンドを実行すると、/mnt ディレクトリに microSDHC カードのファイルシステムをマウントします。microSD カード内のファイルは、/mnt ディレクトリ以下に見えるようになります。

```
[armadillo ~]# mount -t vfat /dev/mmcblk0p1 /mnt
```

図 7.38 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、アンマウントする必要があります。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 /mnt
```

図 7.39 ストレージのアンマウント

7.3.2. 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 microSDHC 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 7.40. 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/mmcblk0p1、二つめは/dev/mmcblk0p2 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/mmcblk0

The number of cylinders for this disk is set to 62528.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): d
Selected partition 1
```

[4]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。
[5]通常、購入したばかりの microSDHC カードは FAT32 ファイルシステムでフォーマットされています。

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-62528, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-62528, default 62528): +100M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (3054-62528, default 3054):
Using default value 3054
Last cylinder or +size or +sizeM or +sizeK (3054-62528, default 62528):
Using default value 62528

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
 mmcblk0: p1 p2
 mmcblk0: p1 p2
Syncing disks.
```

図 7.40 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、EXT4 ファイルシステムでフォーマットするには、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/mmcblk0p1
```

図 7.41 EXT4 ファイルシステムの構築

7.4. LED

Armadillo-IoT の LED は、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 7.8 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/led3/	ユーザー LED3	none
/sys/class/leds/led4/	ユーザー LED4	default-on

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/led5/	ユーザー LED5	none

Armadillo-IoT の外観から見たユーザー LED の位置を次に示します。

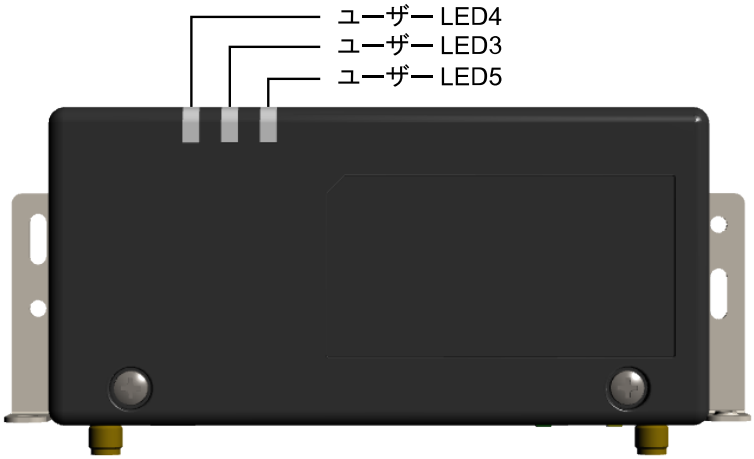


図 7.42 ユーザー LED の位置

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。

7.4.1. 初期出荷状態の LED の動作

初期出荷状態の LED の動作を次に示します。

表 7.9 初期出荷状態の LED の動作

インターフェース	用途	詳細
ユーザー LED3	LTE LED	LTE のデータ接続中に点灯、切断で消灯します
ユーザー LED4	電源 LED	本体の電源が入っていると点灯、電源の切断で消灯します
ユーザー LED5	未使用	常時消灯しています

7.4.2. 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 7.43 LED を点灯させる

Armadillo-IoT の LED には輝度制御の機能が無いため、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 7.44 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
0
```

図 7.45 LED の状態を表示する

7.4.3. トリガを使用する

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 に書き込む有効な値を次に示します。

表 7.10 trigger の種類

設定	説明
none	トリガを設定しません。
mmc0	microSD インターフェース(メインユニット CON12)のアクセスランプにします。
mmc2	eMMC のアクセスランプにします。
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。

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 7.46 LED のトリガに timer を指定する

trigger を読み出すと LED のトリガが取得できます。"[ ]"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 nand-disk mmc0 mmc2 timer oneshot heartbeat backlight gpio de
fault-on rfkill0 phy0rx phy0tx phy0assoc phy0radio phy0tpt rfkill1
```

図 7.47 LED のトリガを表示する

7.5. RTC

Armadillo-IoT は、Board Management IC の RTC 機能を利用しています。

外付バッテリーに対応していないため、電源が切断された場合には時刻を保持することはできません。

7.5.1. 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。

システムクロックの設定方法は 「5.3. 時刻の設定」 を参照してください。

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
Sat Jan  1 00:00:00 2000  0.000000 seconds
[armadillo ~]# hwclock --utc --systohc ❷
[armadillo ~]# hwclock --utc ❸
Tue Jun  2 12:35:08 2015  -0.897934 seconds
```

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。

図 7.48 ハードウェアクロックを設定

7.6. ユーザースイッチ

Armadillo-IoT 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 7.11 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
SW2	/dev/input/event1	code 356 (KEY_POWER2)

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。

7.6.1. 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/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 356 (KEY_POWER2)
Properties:
Testing ... (interrupt to exit)
Event: time 1481116235.679594, type 1 (EV_KEY), code 356 (KEY_POWER2), value 1 ❶
Event: time 1481116235.679594, -------------- EV_SYN ------------
Event: time 1481116235.869587, type 1 (EV_KEY), code 356 (KEY_POWER2), value 0 ❷
Event: time 1481116235.869587, -------------- EV_SYN ------------
^C
[armadillo ~]#
```

❶ SW2 のボタンプッシュイベントを検出したときの表示。

❷ SW2 のボタンリリースイベントを検出したときの表示。

図 7.49 ユーザースイッチ: イベントの確認

7.7. AD コンバーター

Armadillo-IoT は、BMIC(Board Management IC)の AD コンバーター機能により、電源電圧を取得することができます。

7.7.1. 電圧を取得する

電源電圧は、分圧されて AD コンバーターへ入力されています。電源電圧を取得するためには、まず AD コンバーターへの入力電圧を取得する必要があります。

AD コンバーターは IIO(Industrial I/O) デバイスとして実装しています。 /sys/bus/iio/devices/iio:device0/ディレクトリ以下のファイルから入力電圧を算出することができます。

IIO デバイスは、デバイスを認識した順番で iio:deviceN (N は'0'からの連番)となります。IIO デバイスは、IIO デバイス名から特定することができます。BMIC の AD コンバーターの IIO デバイス名は "3-0012"です。

```
[armadillo ~]# cat /sys/bus/iio/devices/iio:device0/name
3-0012
```

AD コンバータへの入力電圧は、AD 変換値と最小入力電圧変動から算出する事ができます。

```
[AD コンバータへの入力電圧 (mV)] = [in_voltage_raw] × [in_voltage_scale]
```

図 7.50 AD コンバータへの入力電圧の計算式

/sys/bus/iio/devices/iio:device0/ディレクトリ以下にある、入力電圧の算出に必要なファイルを次に示します。

表 7.12 入力電圧の算出に必要なファイル

ファイル	説明
in_voltage0_raw	シングルエンド入力 CH0(電源電圧)の AD 変換値
in_voltage_scale	シングルエンド入力の最小入力電圧変動

例として、電源電圧の取得方法について記載します。

```
[armadillo ~]# cat /sys/bus/iio/devices/iio:device0/in_voltage0_raw
1766
[armadillo ~]# cat /sys/bus/iio/devices/iio:device0/in_voltage_scale
0.714111328
```

図 7.51 AD コンバーターへの入力電圧を取得する

「図 7.51. AD コンバーターへの入力電圧を取得する」の例では、AD コンバーターの入力電圧は、約 1.261V (1766 × 0.714111328 [mV])である事がわかります。

AD コンバーターへの入力電圧から、電源電圧を求める計算式を次に示します。

```
[電源電圧 (mV)] = [AD コンバーターへの入力電圧] × (200 + 24) ÷ 24
```

図 7.52 電源電圧の計算式

「図 7.51. AD コンバーターへの入力電圧を取得する」を例にとると、AD コンバーターの入力電圧 1.261V から、電源電圧は約 11.770V であることを求めることができます。

awk コマンドを利用して、次のように電源電圧を表示することができます。

```
[armadillo ~]# adin_raw=`cat /sys/bus/iio/devices/iio:device0/
in_voltage0_raw`
[armadillo ~]# adin_scale=`cat /sys/bus/iio/devices/iio:device0/
in_voltage_scale`
[armadillo ~]# echo $adin_raw $adin_scale | awk '{printf ("%d\n",$1*
$2*(200+24)/24)}'
11770
```

7.8. RS422/RS485

Armadillo-IoT には、RS422/RS485 のシリアルポートが 1 ポート登載されています。シリアルポートのデバイスドライバは、TTY デバイスとして実装されているため TTY デバイスファイルから制御を行うことができます。

シリアルポートインターフェースと、TTY デバイスファイルの対応を次に示します。

表 7.13 シリアルポートインターフェースと TTY デバイスファイル

シリアルポートインターフェース	TTY デバイスファイル
CON4	/dev/ttymxc1

7.8.1. RS422/RS485 の通信設定を変更する

変更が可能な RS485 設定とその初期値を「表 7.14. RS485 設定と初期値」に示します。flags は各ビットごとの論理和を示します。

表 7.14 RS485 設定と初期値

設定		説明	初期値
flags	ENABLED(bit0)	0: RS485 無効 1: RS485 有効	1
	RTS_ON_SEND(bit1)	0: データ送信時の RTS(Driver Enable)が Low 1: データ送信時の RTS(Driver Enable)が High	1
	RTS_AFTER_SEND(bit2)	0: データ非送信時の RTS(Driver Enable)が Low 1: データ非送信時の RTS(Driver Enable)が High	0
	RX_DURING_TX(bit4)	0: 半二重通信 1: 全二重通信	0
delay_rts_before_send		送信前遅延時間(ミリ秒)	0
delay_rts_after_send		送信後遅延時間(ミリ秒)	0

flags の RTS_ON_SEND と RTS_AFTER_SEND は初期値を変更しないでください。変更した場合はデータ送信を行うことができなくなります。

RS485 をコンソールとして利用することはできません。

RS485 設定は、アプリケーションプログラムまたは、Linux カーネル起動オプションで変更することができます。

アプリケーションプログラムの作成方法については、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/serial/serial-rs485.txt)を参照してください。

Linux カーネル起動オプションでは、次のオプション指定子で RS485 設定を行います。

表 7.15 Linux カーネル起動オプションからの RS485 設定

オプション指定子	説明
imx.rs485_uart2	CON4 の UART2(ttymxc1)の RS485 設定を指定します。

RS485 設定のフォーマットは次の通りです。

```
<flags>,<delay_rts_before_send>,<delay_rts_after_send>
```

例として、RS485 設定を全二重通信にする場合は、保守モードで起動してから次のようにコマンドを実行してください。

```
=> setenv optargs imx.rs485_uart2=0x13,0,0
=> saveenv
```

8. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-IoT の Linux カーネル仕様について説明します。

8.1. デフォルトコンフィギュレーション

工場出荷状態のフラッシュメモリに書き込まれている Linux カーネルイメージには、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。 Armadillo-IoT 用の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-3.14-x1-at[VERSION].tar.gz)に含まれる `arch/arm/configs/x1_defconfig` です。

`x1_defconfig` で有効になっている主要な設定を「表 8.1. Linux カーネル主要設定」に示します。

表 8.1 Linux カーネル主要設定

コンフィグ	説明
SMP	Symmetric Multi-Processing
SMP_ON_UP	Allow booting SMP kernel on uniprocessor systems
ARM_CPU_TOPOLOGY	Support cpu topology definition
HAVE_ARM_ARCH_TIMER	Architected timer support
VMSPLIT_2G	Memory split (2G/2G user/kernel split)
NO_HZ	Tickless System (Dynamic Ticks)
HIGH_RES_TIMERS	High Resolution Timer Support
PREEMPT	Preemptible Kernel
AEABI	Use the ARM EABI to compile the kernel
COMPACTION	Allow for memory compaction
BINFMT_ELF	Kernel support for ELF binaries

8.2. デフォルト起動オプション

工場出荷状態の Armadillo-IoT の Linux カーネルの起動オプションについて説明します。デフォルト状態では、次のように設定されています。

表 8.2 Linux カーネルのデフォルト起動オプション

起動オプション	説明
console=ttymxc4,115200	起動ログなどが出力されるイニシャルコンソールに ttymxc4(メインユニット CON5)を、ボーレートに 115200bps を指定します。
root=/dev/mmcblk2p2	ルートファイルシステムに eMMC を指定します。
rootwait	"root="で指定したデバイスが利用可能になるまでルートファイルシステムのマウントを遅らせます。
rw	ルートファイルシステムを読み書き可能としてマウントします。

8.3. Linux ドライバー覧

Armadillo-IoT 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内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。

8.3.1. Armadillo-IoT ゲートウェイ G3L

Armadillo-IoT ゲートウェイ G3L のハードウェアの構成情報やピンマルチプレクスの情報、i.MX7Dual 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_g3l.dts
arch/arm/boot/dts/armadillo_x1l.dts
arch/arm/boot/dts/armadillo_x1l-extboard01.dtsi
arch/arm/boot/dts/imx7d.dtsi

カーネルコンフィギュレーション

```
System Type  --->
    [*] Freescale i.MX family                                   <CONFIG_ARCH_MXC>
          Freescale i.MX support  --->
              [*] i.MX7 Dual support                            <CONFIG_SOC_IMX7D>
```

8.3.2. SPI フラッシュメモリ

Armadillo-IoT では、フラッシュメモリを制御するソフトウェアとして MTD(Memory Technology Device) を利用しています。MTD のキャラクタデバイスまたはブロックデバイスを経由して、ユーザーランドからアクセスすることができます。

関連するソースコード

drivers/mtd/mtdcore.c
drivers/mtd/mtdsuper.c
drivers/mtd/mtdconcat.c
drivers/mtd/mtdpart.c
drivers/mtd/mtdchar.c
drivers/mtd/cmdlinepart.c
drivers/mtd/ofpart.c
drivers/mtd/mtdblock.c
drivers/mtd/spi-nor/spi-nor.c
drivers/mtd/spi-nor/fsl-quadspi.c

デバイスファイル

デバイスファイル	デバイスタイプ	対応するパーティション名
/dev/mtd0	キャラクタ	bootloader
/dev/mtd0ro		
/dev/mtdblock0	ブロック	
/dev/mtd1	キャラクタ	license
/dev/mtd1ro		
/dev/mtdblock1	ブロック	
/dev/mtd2	キャラクタ	reserved
/dev/mtd2ro		
/dev/mtdblock2	ブロック	

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> Memory Technology Device (MTD) support  --->                    <CONFIG_MTD>
        <*>   Command line partition table parsing            <CONFIG_MTD_CMDLINE_PARTS>
        <*>   OpenFirmware partitioning information support       <CONFIG_MTD_OF_PARTS>
        <*>   Caching block device access to MTD devices             <CONFIG_MTD_BLOCK>
        <*>   SPI-NOR device support  --->                          <CONFIG_MTD_SPI_NOR>
            <*>   Freescale Quad SPI controller                  <CONFIG_SPI_FSL_QUADSPI>
```

8.3.3. UART

Armadillo-IoT のシリアルは、i.MX7Dual の UART(Universal Asynchronous Receiver/Transmitter) を利用しています。

Armadillo-IoT で利用している シリアルインターフェースと、接続先を次に示します。

表 8.3 UART の接続先

シリアルインターフェース	接続先
UART2	メインユニット CON4
UART3	サブユニット CON2
UART4	サブユニット WL1837MOD モジュール
UART5	メインユニット CON5
UART7	サブユニット ELS31-J

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: 1.5Mbps(UART3/5/7), 4.0Mbps(UART2/4)

関連するソースコード

drivers/tty/n_tty.c
drivers/tty/tty_buffer.c
drivers/tty/tty_io.c
drivers/tty/tty_ioctl.c
drivers/tty/tty_ldisc.c
drivers/tty/tty_ldsem.c
drivers/tty/tty_mutex.c
drivers/tty/tty_port.c
drivers/tty/serial/serial_core.c
drivers/tty/serial/imx.c

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
UART3	/dev/ttymxc2
UART4	/dev/ttymxc3
UART5	/dev/ttymxc4
UART7	/dev/ttymxc6

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_CONSOL>
                <*> Freescale lpuart serial port support            <SERIAL_FSL_LPUART>
                [*]   Console on Freescale lpuart serial port <SERIAL_FSL_LPUART_CONSOLE>
```

8.3.4. Ethernet

Armadillo-IoT の Ethernet(LAN)は、i.MX7Dual の ENET(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/Space.c
drivers/net/loopback.c
drivers/net/mii.c
drivers/net/ethernet/freescale/fec_main.c
drivers/net/ethernet/freescale/fec_ptp.c
drivers/net/phy/mdio_bus.c
drivers/net/phy/phy.c
drivers/net/phy/phy_device.c
drivers/net/phy/smsc.c

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>
            <*>   Drivers for SMSC PHYs                                  <SMSC_PHY>
```

8.3.5. LTE

Armadillo-IoT には、Gemalto 製 ELS31-J が搭載されています。

ELS31-J は、「8.3.9. USB ホスト」に示す OTG2 と、「8.3.3. UART」に示す UART7 に接続されています。

機能

リンク検出サポート

ネットワークデバイス

usb1[1]

デバイスファイル

/dev/ttyACM0[2]
/dev/ttymxc6

関連するソースコード

drivers/net/usb/usbnet.c
drivers/net/usb/cdc_ether.c
drivers/usb/class/cdc-acm.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] Network device support  --->                                    <NETDEVICES>
        USB Network Adapters  --->
            <*> Multi-purpose USB Networking Framework                   <USB_USBNET>
            <*>   CDC Ethernet support (smart devices such as cable modems)
                                                                  <USB_NET_CDCETHER>
    [*] USB support  --->                                                <USB_SUPPORT>
        <*>     USB Modem (CDC ACM) support                                 <USB_ACM>
```

8.3.6. WLAN

Armadillo-IoT には、TEXAS INSTRUMENTS 社製 WL1837MOD が搭載されています。

[1]USB Ethernet などを接続している場合は、番号が異なる可能性があります。
[2]USB シリアルなどを接続している場合は、番号が異なる可能性があります。

WL1837MOD の WLAN 機能は、「8.3.8. SD ホスト」に示す uSDHC2 に接続されています。

機能

IEEE 802.11 a/b/g/n 準拠
最大リンク速度: 150Mbps
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

ネットワークデバイス

wlan0

関連するソースコード

drivers/net/wireless/ti/wl18xx/
drivers/net/wireless/ti/wlcore/

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] Network device support  --->                                    <NETDEVICES>
        [*]   Wireless LAN  --->                                            <WLAN>
        [*]   TI Wireless LAN support  --->                              <WL_TI>
            <*>   TI wl18xx support                                      <WL18XX>
            -*-   TI wlcore support                                      <WLCORE>
            <*>     TI wlcore SDIO support                             <WLCORE_SDIO>
```

WL1837MOD のファームウェアは、ATDE にインストールされている firmware-ti-connectivity 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-ti-connectivity/copyright を参照してください。

8.3.7. BT

Armadillo-IoT には、TEXAS INSTRUMENTS 社製 WL1837MOD が搭載されています。

WL1837MOD の BT 機能は、「8.3.3. UART」に示す UART4 に接続されています。

WL1837MOD は、Bluetooth(R) version 4.1 に対応しており、BLE(Bluetooth Low Energy)、EDR(Enhanced Data Rate)が利用できます。

デバイス

hci0

関連するソースコード

drivers/misc/ti-st/
drivers/bluetooth/btwilink.c
drivers/bluetooth/hci_ldisc.c
drivers/bluetooth/hci_ll.c

カーネルコンフィギュレーション

```
[*] Networking support  --->                                                <NET>
    <*>   Bluetooth subsystem support  --->                                  <BT>
        <*>   RFCOMM protocol support                                  <BT_RFCOMM>
        [*]     RFCOMM TTY support                                 <BT_RFCOMM_TTY>
        <*>   BNEP protocol support                                      <BT_BNEP>
        [*]     Multicast filter support                        <BT_BNEP_MC_FILTER>
        [*]     Protocol filter support                      <BT_BNEP_PROTO_FILTER>
        <*>   HIDP protocol support                                      <BT_HIDP>
              Bluetooth device drivers  --->
                  <*> HCI UART driver                                  <BT_HCIUART>
                  [*]   HCILL protocol support                      <BT_HCIUART_LL>
                  <*> Texas Instruments WiLink7 driver                  <BT_WILINK>
    Device Drivers  --->
        Misc devices  --->
            Texas Instruments shared transport line discipline  --->
                <*> Shared transport core driver                            <TI_ST>
                <*>   HCI TTY emulation driver for Bluetooth               <ST_HCI>
```

WL1837MOD のファームウェアは、ATDE にインストールされている firmware-ti-connectivity 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-ti-connectivity/copyright を参照してください。

8.3.8. SD ホスト

Armadillo-IoT の SD ホストは、i.MX7Dual の uSDHC(Ultra Secured Digital Host Controller)を利用しています。

Armadillo-IoT で利用している SD ホストと、接続先を次に示します。

表 8.4 SD ホストの接続先

SD ホスト	接続先
uSDHC1	メインユニット CON12
uSDHC2	サブユニット WL1837MOD モジュール

機能(メインユニット CON12)

カードタイプ: microSD/microSDHC/microSDXC/microSDIO
バス幅: 1bit or 4bit

スピードモード: Default Speed(24MHz), High Speed(48MHz), UHS-I(196.36MHz)
カードディテクトサポート
ライトプロテクトサポート

デバイスファイル

メモリカードの場合は、カードを認識した順番で/dev/mmcblkN (N は'0'または'1')となります。
I/O カードの場合は、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/mmc/card/block.c
drivers/mmc/card/queue.c
drivers/mmc/core/
drivers/mmc/host/sdhci-esdhc-imx.c
drivers/mmc/host/sdhci-pltfm.c
drivers/mmc/host/sdhci.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> MMC/SD/SDIO card support  --->                                  <MMC>
        [*]   Additional delay after SDIO reset          <MMC_DELAY_AFTER_SDIO_RESET>
              *** MMC/SD/SDIO Card Drivers ***
        <*>   MMC block device driver                                  <MMC_BLOCK>
        (8)     Number of minors per block device                <MMC_BLOCK_MINORS>
        [*]     Use bounce buffer for simple hosts               <MMC_BLOCK_BOUNCE>
              *** MMC/SD/SDIO Host Controller Drivers ***
        <*>   Secure Digital Host Controller Interface support         <MMC_SDHCI>
        <*>   SDHCI platform and OF driver helper                <MMC_SDHCI_PLTFM>
        <*>     SDHCI support for the Freescale eSDHC/uSDHC i.MX controller
                                                                <MMC_SDHCI_OF_ESDHC>
```

SDIO カードを利用する場合は、arch/arm/boot/dts/armadillo_x1l.dts の"usdhc1"ノードに"use-sdio"プロパティを追加してください。

```
&usdhc1 {
    pinctrl-names = ”default”, ”state_100mhz”, ”state_200mhz”,
                    ”state_power_off”;
    pinctrl-0 = <&pinctrl_usdhc1>;
    pinctrl-1 = <&pinctrl_usdhc1_100mhz>;
    pinctrl-2 = <&pinctrl_usdhc1_200mhz>;
    pinctrl-3 = <&pinctrl_usdhc1_power_off>;
    cd-gpios = <&gpio5 0 GPIO_ACTIVE_LOW>;
    wp-gpios = <&gpio5 1 GPIO_ACTIVE_HIGH>;
    pinctrl-assert-gpios = <&gpio5 4 GPIO_ACTIVE_LOW>, /* SD1_CMD */
                           <&gpio5 5 GPIO_ACTIVE_LOW>, /* SD1_DATA0 */
                           <&gpio5 6 GPIO_ACTIVE_LOW>, /* SD1_DATA1 */
                           <&gpio5 7 GPIO_ACTIVE_LOW>, /* SD1_DATA2 */
                           <&gpio5 8 GPIO_ACTIVE_LOW>; /* SD1_DATA3 */
    tuning-step = <2>;
    vmmc-supply = <&reg_sd1_vmmc>;
    enable-sdio-wakeup;
```

```
    keep-power-in-suspend;
    use-sdio;
    status = "okay";
};
```

"use-sdio"プロパティを追加しない場合、Advanced DMA エラーが発生する場合があります。

8.3.9. USB ホスト

Armadillo-IoT の USB ホストは、i.MX7Dual 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 で利用している USB インターフェースと、接続先を次に示します。

表 8.5 USB インターフェースの接続先

USB インターフェース	接続先
OTG1	メインユニット CON3
OTG2	サブユニット ELS31-J

機能(メインユニット CON3)

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/ci_hdrc_imx.c
drivers/usb/chipidea/ci_hdrc_msm.c
drivers/usb/chipidea/ci_hdrc_zevio.c
drivers/usb/chipidea/core.c
drivers/usb/chipidea/host.c
drivers/usb/chipidea/otg.c
drivers/usb/chipidea/usbmisc_imx.c
drivers/usb/host/ehci-hcd.c
drivers/usb/host/ehci-hub.c
drivers/usb/phy/phy-generic.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] USB support  --->                                          <USB_SUPPORT>
        <*>   Support for Host-side USB                                   <USB>
                *** USB Host Controller Drivers ***
        <*>     EHCI HCD (USB 2.0) support                      <USB_EHCI_HCD>
        <*>   ChipIdea Highspeed Dual Role Controller           <USB_CHIPIDEA>
        [*]     ChipIdea host controller                    <USB_CHIPIDEA_HOST>
              USB Physical Layer drivers  --->
            <*>     NOP USB Transceiver Driver                  <NOP_USB_XCEIV>
```

8.3.10. リアルタイムクロック

Armadillo-IoT のリアルタイムクロックは、Board Management IC の RTC 機能を利用しています。Board Management IC の RTC 機能は、I2C4 (I2C ノード: 3-0011) に接続されています。

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

関連するソースコード

drivers/rtc/class.c
drivers/rtc/hctosys.c
drivers/rtc/interface.c
drivers/rtc/rtc-dev.c
drivers/rtc/rtc-lib.c
drivers/rtc/rtc-proc.c
drivers/rtc/rtc-sysfs.c
drivers/rtc/systohc.c
drivers/rtc/rtc-bmic.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> Real Time Clock  --->
        [*]   Set system time from RTC on startup and resume        <RTC_HCTOSYS>
        [*]   Set the RTC time based on NTP synchronization         <RTC_SYSTOHC>
        (rtc0)  RTC used to set the system time                 <RTC_HCTOSYS_DEVICE>
              *** RTC interfaces ***
        [*]   /sys/class/rtc/rtcN (sysfs)                          <RTC_INTF_SYSFS>
        [*]   /proc/driver/rtc (procfs for rtcN)                   <RTC_INTF_PROC>
        [*]   /dev/rtcN (character devices)                          <RTC_INTF_DEV>
        [*]     RTC UIE emulation on dev interface              <RTC_INTF_DEV_UIE_EMUL>
              *** I2C RTC drivers ***
        <*>   Atmark Techno BMIC RTC                                <RTC_DRV_BMIC>
```

アラーム割り込みは、sysfs RTC クラスディレクトリ以下のファイルから利用できます。

wakealarm ファイルに UNIX エポックからの経過秒数、または先頭に+を付けて現在時刻からの経過秒数を書き込むと、アラーム割り込み発生時刻を指定できます。アラーム割り込み発生時刻を変更するには wakealarm ファイルに"+0"を書き込み、アラーム割り込みのキャンセル後に再設定する必要があります。アラーム割り込みの利用例を次に示します。

```
[armadillo ~]# cat /proc/interrupts | grep bmic_rtc_irq ❶
173:          0          0  gpio-mxc  13  bmic_rtc_irq
[armadillo ~]# echo +10 > /sys/class/rtc/rtc0/wakealarm ❷
[armadillo ~]# cat /sys/class/rtc/rtc0/wakealarm ❸
1458781144
[armadillo ~]# cat /sys/class/rtc/rtc0/since_epoch ❹
1458781145
[armadillo ~]# cat /proc/interrupts | grep bmic_rtc_irq ❺
173:          1          0  gpio-mxc  13  bmic_rtc_irq
```

❶ アラーム割り込みの発生回数を確認します。この例では 0 回です。

❷ アラーム割り込みの発生時刻を 10 秒後に設定します。

❸ アラーム割り込みの発生時刻(UNIX エポックからの経過秒数)を確認します。この例では 1458781144 秒です。

❹ 現在時刻(UNIX エポックからの経過秒数)を確認します。アラーム割り込みの発生時刻を超えるまで待ちます。

❺ 再度アラーム割り込みの発生回数を確認します。1 増えているのでアラーム割り込みが発生したことを確認できます。

デバイスファイル(/dev/rtc0)経由でもアラーム割り込みを利用することができます。サンプルプログラムなどのより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/rtc.txt)を参照してください。

date コマンドを利用して、UNIX エポックからの経過秒数を日時に変換することができます。

```
[armadillo ~]# date --date=@`cat /sys/class/rtc/rtc0/since_epoch`
Thu Mar 24 10:02:56 JST 2016
```

8.3.11. 温度センサ

Armadillo-IoT の温度センサーは、i.MX7Dual の TEMPMON(Temperature Monitor)を利用しています。

起動直後の設定では、i.MX7Dual の測定温度が 105℃以上になった場合、Linux カーネルが/sbin/poweroff コマンドを実行し、システムを停止します。

sysfs ディレクトリ

/sys/class/thermal/thermal_zone1/

関連するソースコード

drivers/thermal/imx_thermal.c
drivers/thermal/thermal_core.c
drivers/thermal/cpu_cooling.c
drivers/thermal/device_cooling.c
drivers/thermal/of-thermal.c
drivers/thermal/step_wise.c
drivers/thermal/thermal_hwmon.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> Generic Thermal sysfs driver  --->                                  <THERMAL>
        [*]   Expose thermal sensors as hwmon device                      <THERMAL_HWMON>
        [*]   APIs to parse thermal data out of device tree                  <THERMAL_OF>
        -*-   Step_wise thermal governor                               <THERMAL_GOV_STEP_WISE>
        [*]   generic cpu cooling support                                  <CPU_THERMAL>
        <*>   Temperature sensor driver for Freescale i.MX SoCs            <IMX_THERMAL>
```

8.3.12. AD コンバーター

Armadillo-IoT に搭載された Board Management IC の AD コンバーター機能を利用することができます。

Board Management IC の AD コンバーター機能は、I2C4(I2C ノード: 3-0012)に接続されています。Armadillo-IoT の電源電圧を測定することができます。

機能(Board Management IC)

分解能: 12bit
測定範囲: 0V ～ 3.3V(Board Management IC の電源電圧)

sysfs ディレクトリ

デバイスを認識した順番で /sys/bus/iio/devices/iio:deviceN (N は'0'からの連番)となります。

デバイスファイル

デバイスを認識した順番で /dev/iio:deviceN (N は'0'からの連番)となります。

関連するソースコード

drivers/iio/industrialio-core.c
drivers/iio/industrialio-buffer.c
drivers/iio/industrialio-event.c
drivers/iio/industrialio-trigger.c
drivers/iio/industrialio-triggered-buffer.c
drivers/iio/inkern.c
drivers/iio/kfifo_buf.c

drivers/iio/trigger/iio-trig-sysfs.c
drivers/iio/adc/bmic_adc.c
drivers/iio/adc/mcp320x.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> Industrial I/O support  --->                                          <IIO>
        [*]   Enable buffer support within IIO                              <IIO_BUFFER>
        -*-     Industrial I/O buffering based on kfifo                   <IIO_KFIFO_BUF>
        -*-   Enable triggered sampling support                             <IIO_TRIGGER>
        (2)     Maximum number of consumers per trigger       <IIO_CONSUMERS_PER_TRIGGER>
              Analog to digital converters  --->
                  <*> Atmark Techno BMIC ADC                                <BMIC_ADC>
                  <*> Microchip Technology MCP3x01/02/04/08                  <MCP320X>
```

8.3.13. LED

Armadillo-IoT 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/led3
/sys/class/leds/led4
/sys/class/leds/led5

関連するソースコード

drivers/leds/led-class.c
drivers/leds/led-core.c
drivers/leds/led-triggers.c
drivers/leds/leds-gpio-register.c
drivers/leds/leds-gpio.c
drivers/leds/trigger/ledtrig-timer.c
drivers/leds/trigger/ledtrig-oneshot.c
drivers/leds/trigger/ledtrig-heartbeat.c
drivers/leds/trigger/ledtrig-backlight.c
drivers/leds/trigger/ledtrig-gpio.c
drivers/leds/trigger/ledtrig-default-on.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] LED Support  --->                                          <NEW_LEDS>
        <*>   LED Class Support                                    <LEDS_CLASS>
              *** LED drivers ***
        <*>   LED Support for GPIO connected LEDs                  <LEDS_GPIO>
              *** LED Triggers ***
        [*]   LED Trigger support  --->                          <LEDS_TRIGGERS>
            <*>   LED Timer Trigger                            <LEDS_TRIGGER_TIMER>
            <*>   LED One-shot Trigger                       <LEDS_TRIGGER_ONESHOT>
            <*>   LED Heartbeat Trigger                     <LEDS_TRIGGER_HEARTBEAT>
            <*>   LED backlight Trigger                     <LEDS_TRIGGER_BACKLIGHT>
            <*>   LED GPIO Trigger                               <LEDS_TRIGGER_GPIO>
            <*>   LED Default ON Trigger                   <LEDS_TRIGGER_DEFAULT_ON>
```

8.3.14. ユーザースイッチ

Armadillo-IoT 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。GPIO が接続され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。

ユーザースイッチには、次に示すキーコードが割り当てられています。

表 8.6 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
SW2	/dev/input/event1	code 2 (KEY_1)

デバイスファイル

/dev/input/event1[3]

関連するソースコード

drivers/input/evdev.c
drivers/input/ff-core.c
drivers/input/input-compat.c
drivers/input/input-mt.c
drivers/input/input-polldev.c
drivers/input/input.c
drivers/input/keyboard/gpio_keys.c

[3]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、番号が異なる可能性があります

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>
        -*-   Polled input device skeleton                                  <INPUT_POLLDEV>
              *** Userland interfaces ***
        <*>   Event interface                                                <INPUT_EVDEV>
              *** Input Device Drivers ***
        [*]   Keyboards  --->                                               <INPUT_KEYBOARD>
            <*>   GPIO Buttons                                               <KEYBOARD_GPIO>
```

8.3.15. I2C

Armadillo-IoT の I2C インターフェースは、i.MX 7Dual の I2C(I2C Controller) を利用します。また、GPIO を利用した I2C バスドライバ(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 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 8.7 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
3(I2C4)	0x09	PF3000 パワーマネジメント IC
	0x10～0x17	Board Management IC
	0x50	M24C01-W EEPROM

Armadillo-IoT 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-3 (I2C4)

関連するソースコード

drivers/i2c/i2c-boardinfo.c
drivers/i2c/i2c-core.c
drivers/i2c/i2c-dev.c
drivers/i2c/algos/i2c-algo-bit.c
drivers/i2c/busses/i2c-gpio.c
drivers/i2c/busses/i2c-imx.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    <*> I2C support  --->                                                <I2C>
        <*>   I2C device interface                                  <I2C_CHARDEV>
        [*]   Autoselect pertinent helper modules                <I2C_HELPER_AUTO>
              I2C Hardware Bus support  --->
            <*> GPIO-based bitbanging I2C                             <I2C_GPIO>
            <*> IMX I2C interface                                      <I2C_MXC>
```

8.3.16. SPI

Armadillo-IoT の SPI インターフェースは、i.MX7Dual の ECSPI(Enhanced Configurable SPI)を利用します。

標準状態では無効になっている CONFIG_SPI_SPIDEV を有効化すると、ユーザードライバで SPI デバイスを制御することができます。

関連するソースコード

drivers/spi/spi-bitbang.c
drivers/spi/spi-imx.c
drivers/spi/spi.c
drivers/spi/spidev.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] SPI support  --->                                                <SPI>
              *** SPI Master Controller Drivers ***
        -*-   Utilities for Bitbanging SPI masters                  <SPI_BITBANG>
        <*>   Freescale i.MX SPI controllers                           <SPI_MXC>
              *** SPI Protocol Masters ***
        < >   User mode SPI device driver support                   <SPI_SPIDEV>
```

8.3.17. ウォッチドッグタイマー

Armadillo-IoT のウォッチドッグタイマーは、i.MX7Dual の WDOG(Watchdog Timer) を利用しています。

ウォッチドッグタイマーは、U-Boot によって有効化されます。標準状態でタイムアウト時間は 10 秒に設定されます。Linux カーネルは、ウォッチドッグタイマードライバの初期化時にタイムアウト時間を 10 秒に再設定します。

何らかの要因でウォッチドッグタイマーのキックができなくなりタイムアウトすると、システムリセットが発生します。

関連するソースコード

drivers/watchdog/imx2_wdt.c

カーネルコンフィギュレーション

```
Device Drivers  --->
    [*] Watchdog Timer Support  --->                                    <WATCHDOG>
        <*>   IMX2+ Watchdog                                            <IMX2_WDT>
```

i.MX7Dual の WDOG は、一度有効化すると無効化することができません。そのため、halt コマンドなどを実行して Linux カーネルを停止した場合は、ウォッチドッグタイマーのキックができなくなるためシステムリセットが発生します。

WDOG ドライバーの終了処理では、タイムアウト時間を WDOG の最大値である 128 秒に設定します。

8.3.18. CPU 周波数スケーリング

Armadillo-IoT の CPU 周波数スケーリング機能は、Linux の CPUFreq を利用しています。適切に CPU 周波数を調整することにより、CPU の消費電力を抑えることができます。

sysfs ディレクトリ

/sys/devices/system/cpu/cpu0/cpufreq/
/sys/devices/system/cpu/cpu1/cpufreq/

関連するソースコード

drivers/cpufreq/cpufreq.c
drivers/cpufreq/cpufreq_conservative.c
drivers/cpufreq/cpufreq_governor.c
drivers/cpufreq/cpufreq_interactive.c
drivers/cpufreq/cpufreq_ondemand.c
drivers/cpufreq/cpufreq_performance.c
drivers/cpufreq/cpufreq_stats.c
drivers/cpufreq/cpufreq_userspace.c
drivers/cpufreq/freq_table.c
drivers/cpufreq/imx7-cpufreq.c

カーネルコンフィギュレーション

```
CPU Power Management  --->
    CPU Frequency scaling  --->
        [*] CPU Frequency scaling                                       <CPU_FREQ>
        <*>   CPU frequency translation statistics                    <CPU_FREQ_STAT>
              Default CPUFreq governor (interactive)  --->
                  (X) interactive                      <CPU_FREQ_DEFAULT_GOV_INTERACTIVE>
              <*>   'performance' governor                      <CPU_FREQ_GOV_PERFORMANCE>
              <*>   'powersave' governor                          <CPU_FREQ_GOV_POWERSAVE>
              <*>   'userspace' governor for userspace frequency scaling
                                                                  <CPU_FREQ_GOV_USERSPACE>
              <*>   'ondemand' cpufreq policy governor             <CPU_FREQ_GOV_ONDEMAND>
              -*-   'interactive' cpufreq policy governor       <CPU_FREQ_GOV_INTERACTIVE>
              <*>   'conservative' cpufreq governor           <CPU_FREQ_GOV_CONSERVATIVE>
                    ARM CPU frequency scaling drivers  --->
                        <*> Freescale i.MX7 cpufreq support           <ARM_IMX7D_CPUFREQ>
```

Armadillo-IoT が対応する CPU 周波数と、CPU(ARM Cortex-A7)の電源電圧の対応を次に示します。

表 8.8 対応する CPU 周波数と電源電圧

CPU 周波数	電源電圧
996MHz	1.075V
792MHz	0.975V

CPU 周波数を決定する機構を、"Governor"と呼びます。工場出荷状態の Armadillo-IoT の Linux カーネルで利用可能な Governor を次に示します。

表 8.9 Governor の種類

Governor	説明
performance	負荷に関わらず、常に最大クロックで動作する。
powersave	負荷に関わらず、常に最小クロックで動作する。
userspace	SYSFS ファイルからユーザーがクロックを指定する。
ondemand	負荷がかかるとクロックを上げ、負荷が下がるとクロックも下げる。負荷がかかるとクロックが急激に上がる。
conservative	負荷がかかるとクロックを上げ、負荷が下がるとクロックも下げる。ondemand よりも、負荷に対するクロックの変化が段階的に行われる。
interactive	負荷がかかるとクロックを上げ、負荷が下がるとクロックも下げる。ondemand よりも、負荷に対するクロックの変化が速やかに行われる。

Governor の変更は sysfs ディレクトリ以下の scaling_governor から行うことができます。CPU0 の Governor 変更例を次に示します。

```
[armadillo ~]# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors ❶
conservative ondemand userspace powersave interactive performance
[armadillo ~]# echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor ❷
[armadillo ~]# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor ❸
performance
```

❶ 利用可能な Governor を表示します。

❷ Governor を performance に設定します。

❸ Governor を確認します。performance に設定されていることが確認できます。

8.3.19. パワーマネジメント

Armadillo-IoT 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 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>
    [*] Run-time PM core functionality                           <PM_RUNTIME>
```

Armadillo-IoT 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 8.10 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
Power-On Suspend	standby	Suspend-to-RAM よりも短時間で復帰することができる。
Suspend-to-RAM	mem	Power-On Suspend よりも消費電力を抑えることができる。

起床要因として利用可能なデバイスは次の通りです。

表 8.11 起床要因として利用可能なデバイス

デバイス	起床要因の有効化	起床要因
UART2(メインユニット CON4)	[armadillo ~]# **echo enabled > /sys/bus/platform/drivers/imx-uart/30890000.serial/tty/ttymxc1/power/wakeup**	データ受信
UART5(メインユニット CON5)	[armadillo ~]# **echo enabled > /sys/bus/platform/drivers/imx-uart/30a70000.serial/tty/ttymxc4/power/wakeup**	データ受信
Ethernet(メインユニット CON2)	[armadillo ~]# **apt-get install ethtool** [armadillo ~]# **ethtool -s eth0 wol g**	Wake-on-LAN のマジックパケットを受信

デバイス	起床要因の有効化	起床要因
USB ホスト(メインユニット CON11)	[armadillo ~]# **echo enabled > /sys/bus/platform/devices/30b10000.usb/power/wakeup** ⏎ [armadillo ~]# **echo enabled > /sys/bus/platform/drivers/ci_hdrc/ci_hdrc.0/power/wakeup** ⏎ [armadillo ~]# **echo enabled > /sys/bus/platform/drivers/ci_hdrc/ci_hdrc.0/usb1/power/wakeup** ⏎	USB デバイスの挿抜

Ethernet から起床要因である Wake-on-LAN のマジックパケットを、ATDE から送信する例を次に示します。

```
[PC ~]$ sudo apt-get install wakeonlan
[PC ~]$ wakeonlan [MAC Address] ❶
```

❶　Armadillo-IoT の有線 LAN の MAC アドレスを指定します。

9. Debian ユーザーランド仕様

本章では、工場出荷状態の Armadillo-IoT G3L の Deabin ユーザーランドの基本的な仕様について説明します。

9.1. Debian ユーザーランド

Armadillo-IoT G3L の標準ルートファイルシステムは、32-bit hard-float ARMv7 (「armhf」)アーキテクチャ用の Debian GNU/Linux 8 (コードネーム 「jessie」)です。出荷状態、または標準イメージを展開した直後のユーザーランド内には、Armadillo の動作に必要な最小限のパッケージや設定が含まれています。

Armadillo-IoT G3L にインストールされた Debian GNU/Linux 8 は、eMMC または microSD カード上で動作します。Linux カーネルが動作している状態で Armadillo の電源を切断する場合は、必ず「halt」コマンドによる終了を行い、RAM 上にキャッシュされている eMMC または microSD カードへの書き込み処理を完了するようにしてください。再起動を行う場合も同様に、reboot コマンドによる再起動を行ってください。

9.2. パッケージ管理

パッケージ管理システム APT(Advanced Packaging Tool)を使用して、パッケージを管理する方法について記載します。工場出荷状態の Debian には動作に必要な最低限のパッケージしかインストールされていませんが、APT を使用することで、簡単にパッケージを追加することができます。

工場出荷状態では、APT はインターネット上の Debian サイト(HTTP サーバー)から利用可能なパッケージのインデックスを取得します[1]。そのため、APT を使用するためにはネットワークを有効化し、インターネットに接続できる状態にしておく必要があります。

ネットワークを有効化する方法については、「7.2. ネットワーク」を参照してください。

システムクロックが大幅にずれた状態で、APT を利用すると警告メッセージが出力される場合があります。事前に「7.5. RTC」を参照してシステムクロックを合わせてください。

apt-get update

パッケージインデックスファイルを最新の状態にアップデートします。

引数　無し

使用例

```
[armadillo ~]# apt-get update
```

[1] /etc/apt/sources.list で設定しています。記述ルールなどについては、sources.list のマニュアルページを参照してください。

apt-get upgrade

現在インストールされている全てのパッケージを最新バージョンにアップグレードします。

引数　無し

使用例

```
[armadillo ~]# apt-get upgrade
```

apt-get install [パッケージ名]

引数に指定したパッケージをインストールします。すでにインストール済みの場合はアップグレードします。

引数　パッケージ名(複数指定可能)

使用例

```
[armadillo ~]# apt-get install gcc
```

apt-get remove [パッケージ名]

引数に指定したパッケージをアンインストールします。インストールされていない場合は何もしません。

引数　パッケージ名(複数指定可能)

使用例

```
[armadillo ~]# apt-get remove apache2
```

apt-cache search [キーワード]

引数に指定したキーワードをパッケージ名または説明文に含むパッケージを検索します。

引数　キーワード(正規表現が使用可能)

使用例

```
[armadillo ~]# apt-cache search "Bourne Again SHell"
bash-doc - Documentation and examples for the The GNU Bourne Again SHell
bash-static - The GNU Bourne Again SHell (static version)
bash - The GNU Bourne Again SHell
```

10. ブートローダー仕様

本章では、ブートローダーの起動モードや利用することができる機能について説明します。

10.1. ブートローダー起動モード

ブートローダーが起動すると、USB シリアル変換アダプタのスライドスイッチの状態により、2 つのモードのどちらかに遷移します。USB シリアル変換アダプタのスライドスイッチの詳細については、「4.6. スライドスイッチの設定について」を参照してください。

表 10.1 ブートローダー起動モード

起動モードの種別	スライドスイッチ	説明
保守モード	外側	各種設定が可能な U-Boot コマンドプロンプトが起動します。
オートブートモード	内側	電源投入後、自動的に Linux カーネルを起動させます。

USB シリアル変換アダプタが未接続の場合オートブートモードとなり、Linux カーネルが起動します。

10.2. ブートローダーの機能

U-Boot の保守モードでは、Linux カーネルの起動オプションの設定などを行うことできます。

保守モードで利用できる有用なコマンドは、「表 10.2. 保守モード 有用なコマンド一覧」に示します。

表 10.2 保守モード 有用なコマンド一覧

コマンド	説明
boot	OS を起動する場合に使用します
bdinfo	ハードウェアの情報を表示します
md mm nm mw cp cmp	簡易的にメモリアクセスする場合に使用します
printenv setenv saveenv	環境変数の設定をする場合に使用します、環境変数にて OS の起動設定等をおこなうことができます
crc32	メモリ空間のチェックサムを表示する場合に使用します
version	ブートローダーのバージョンを表示します

各コマンドのヘルプを表示するには「図 10.1. U-Boot コマンドのヘルプを表示」のようにします。

```
=> help [コマンド]
```

図 10.1 U-Boot コマンドのヘルプを表示

10.2.1. Linux カーネルイメージと device tree blob の指定方法

ブートローダーが OS を起動させる場合、eMMC または、microSD カード内に保存されている Linux カーネルイメージと device tree blob を使用することができます。

ファイルを保存しているデバイスを指定するには、環境変数 "mmcdev" を、パーティション番号を指定するには 環境変数 "mmcpart" を使用します。

Linux カーネルイメージはファイル名 "uImage" で保存されたものを使用します。device tree blob はファイル名 "armadillo_iotg_g3l.dtb" で保存されたものを使用します。

"mmcdev" で設定可能な値と、起動デバイスの関係を 「表 10.3. mmcdev の設定値と起動デバイス」に示します。

表 10.3 mmcdev の設定値と起動デバイス

設定値	起動デバイス
0	microSD カード(メインユニット CON12 に接続)
1	eMMC

eMMC のパーティション 1 を指定する場合、「図 10.2. eMMC のパーティション 1 に保存された Linux カーネルイメージから起動する」のようにします。

```
=> setenv mmcdev 1
=> setenv mmcpart 1
```

図 10.2 eMMC のパーティション 1 に保存された Linux カーネルイメージから起動する

QSPI 用のブートローダーと、SD 用のブートローダーにて、"mmcdev"と"mmcpart"のデフォルト値が異なります。 ブートローダーの種類と、デフォルト値の関係を「表 10.4. ブートローダーの種類と mmcdev, mmcpart のデフォルト値」に示します。

表 10.4 ブートローダーの種類と mmcdev, mmcpart のデフォルト値

ブートローダーの種類	ブートローダーファイル名	mmcdev デフォルト値	mmcpart デフォルト値
QSPI 用	u-boot-x1-at*.bin	1(eMMC)	1
SD 用	u-boot-x1-sd-at*.bin	0(SD)	1

10.2.2. ルートファイルシステムの指定方法

ルートファイルシステムが構築されているデバイスは、環境変数 "mmcroot" で指定することができます。

eMMC のパーティション 2 を指定する場合、「図 10.3. eMMC のパーティション 2 に保存されたルートファイルシステムを指定する」のようにします。

```
=> setenv mmcroot /dev/mmcblk2p2
```

図 10.3 eMMC のパーティション 2 に保存されたルートファイルシステムを指定する

QSPI 用のブートローダーと、SD 用のブートローダーにて、"mmcroot"のデフォルト値が異なります。ブートローダーの種類と、デフォルト値の関係を「表 10.5. ブートローダーの種類と mmcroot のデフォルト値」に示します。

表 10.5 ブートローダーの種類と mmcroot のデフォルト値

ブートローダーの種類	ブートローダーファイル名	mmcroot デフォルト値
QSPI 用	u-boot-x1-v*.*.*.bin	/dev/mmcblk2p2(eMMC パーティション 2)

ブートローダーの種類	ブートローダーファイル名	mmcroot デフォルト値
SD 用	u-boot-x1-sd-v*.*.*.bin	/dev/mmcblk0p2(SD パーティション 2)

10.2.3. 環境変数の保存

環境変数は"saveenv"コマンドにて保存することができます。保存を行わずに、Armadillo-IoT の電源を切ると setenv で設定した環境変数は消えてしまいます。

QSPI 用のブートローダーを使用した場合、環境変数は QSPI 内に保存されます。SD 用のブートローダーを使用した場合、環境変数は SD 内に保存されます。

全ての環境変数をデフォルト値に戻すには、「図 10.4. 全ての環境変数をデフォルト値に戻す」のようにします。

```
=> env default -a
=> saveenv
```

図 10.4 全ての環境変数をデフォルト値に戻す

10.2.4. Linux カーネル起動オプション

10.2.4.1. 代表的な Linux カーネル起動オプション

Linux カーネルには様々な起動オプションがあります。詳しくは、Linux の解説書や、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/kernel-parameters.txt)を参照してください。

ここでは Armadillo-IoT で使用することができる、代表的な起動オプションを「表 10.6. Linux カーネルの起動オプションの一例」に紹介します。

表 10.6 Linux カーネルの起動オプションの一例

オプション指定子	説明
console=	起動ログなどが出力されるイニシャルコンソールを指定します。 次の例では、コンソールに ttymxc1 を、ボーレートに 115200 を指定しています。 `console=ttymxc1,115200`

root=	ルートファイルシステムが構築されているデバイスを指定します。
	デバイスには Linux カーネルが認識した場合のデバイスを指定します。
	initrd をルートファイルシステムとする場合には、以下の例のように設定します。
	`root=/dev/ram0`
	microSD カードにルートファイルシステムを配置する場合には、microSD カードのデバイスファイルを指定します。次の例では、デバイスに microSD カードの第 2 パーティションを指定しています。
	`root=/dev/mmcblk0p2`

rootwait	"root="で指定したデバイスが利用可能になるまでルートファイルシステムのマウントを遅らせます。

オプション指定子	説明
mem	Linux カーネルが利用可能なメモリの量を指定します。RAM の一部を専用メモリとして利用したい場合などに設定します。

10.2.4.2. Linux カーネル起動オプションの設定方法

Linux カーネル起動オプションは環境変数 "mmcargs" で指定することができます。

"mmcargs" のデフォルト値は次に示す値に設定されています。

```
setenv mmcargs setenv bootargs console=${console},${baudrate} root=${mmcroot} ${optargs}
```

デフォルトでは、コンソールには環境変数 "console"が、コンソールのボーレートには環境変数 "baudrate"が、ルートファイルシステムには、環境変数 "mmcroot"が設定されています。

Linux カーネル起動オプションの追加をしたい場合、環境変数 "optargs"を使用すると便利です。

次に、例として、Linux カーネルが利用可能なメモリの量を 384M に設定する方法を「図 10.5. 利用可能なメモリ量を 384M にする」に示します。

```
=> setenv optargs mem=384M
=> saveenv
=> printenv optargs
mem=384M
```

図 10.5 利用可能なメモリ量を 384M にする

11. ビルド手順

本章では、工場出荷イメージと同じイメージを作成する手順について説明します。

使用するソースコードは、開発セット付属の DVD に収録されています。最新版のソースコードは、Armadillo サイトからダウンロードすることができます。新機能の追加や不具合の修正などが行われているため、DVD に収録されているものよりも新しいバージョンがリリースされているかを確認して、最新バージョンのソースコードを利用することを推奨します。

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

http://armadillo.atmark-techno.com/armadillo-iot-g3l/downloads

開発作業では、基本ライブラリ・アプリケーションやシステム設定ファイルの作成・配置を行います。各ファイルは作業ディレクトリ配下で作成・配置作業を行いますが、作業ミスにより誤って作業用 PC 自体の OS を破壊しないために、すべての作業は root ユーザーではなく**一般ユーザー**で行ってください。

11.1. ブートローダーをビルドする

ここでは、ブートローダーである「U-Boot」のソースコードからイメージファイルを作成する手順を説明します。

手順 11.1 ブートローダーをビルド

1. **ソースコードの準備**

   U-Boot のソースコードアーカイブを準備し展開します。

```
[PC ~]$ ls
uboot_2016.07-at[version].tar.gz
[PC ~]$ tar xf uboot_2016.07-at[version].tar.gz
[PC ~]$ ls
uboot_2016.07-at[version]  uboot_2016.07-at[version].tar.gz
```

2. **デフォルトコンフィギュレーションの適用**

   U-Boot ディレクトリに入り、Armadillo-IoT ゲートウェイ G3L 用のデフォルトコンフィギュレーションを適用します。ここでは例としてフラッシュメモリ起動用イメージを作成します。デフォルトコンフィグには x1_config を指定します。SD 起動用イメージを作成する場合は、x1_sd_config を指定してください。

```
[PC ~]$ cd uboot_2016.07-at[version]
[PC ~/uboot_2016.07-at[version]]$ make ARCH=arm x1_config
```

3. **ビルド**

   ビルドには **make** コマンドを利用します。

```
[PC ~/uboot_2016.07-at[version]]$ make CROSS_COMPILE=arm-linux-gnueabihf-
```

4. **イメージファイルの生成確認**

   ビルドが終了すると、U-Boot ディレクトリにイメージファイルが作成されています。

```
[PC ~/uboot_2016.07-at[version]]$ ls u-boot-x1.bin
u-boot-x1.bin
```

11.2. Linux カーネルをビルドする

ここでは、Linux カーネルのソースコードと initramfs アーカイブから、イメージファイルを作成する手順を説明します。

手順 11.2 Linux カーネルをビルド

1. **アーカイブの展開**

   Linux カーネルのソースコードアーカイブを展開します。

```
[PC ~]$ ls
initramfs_x1-[version].cpio.gz linux-3.14-x1-at[version].tar.gz
[PC ~]$ tar xf linux-3.14-x1-at[version].tar.gz
[PC ~]$ ls
initramfs_x1-[version].cpio.gz linux-3.14-x1-at[version]  linux-3.14-x1-
at[version].tar.gz
```

2. **initramfs アーカイブへのシンボリックリンク作成**

   Linux カーネルディレクトリに移動して、initramfs アーカイブへのシンボリックリンク作成します。

```
[PC ~]$ cd linux-3.14-x1-at[version]
[PC ~/linux-3.14-x1-at[version]]$ ln -s ../initramfs_x1-[version].cpio.gz
initramfs_x1.cpio.gz
```

3. **コンフィギュレーション**

   コンフィギュレーションをします。

```
[PC ~/linux-3.14-x1-at[version]]$ make ARCH=arm x1_defconfig
```

4. **ビルド**

   ビルドするには、次のようにコマンドを実行します。

```
[PC ~/linux-3.14-x1-at[version]]$ make CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm
[PC ~/linux-3.14-x1-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- ⏎
LOADADDR=0x80008000 uImage
```

5. **イメージファイルの生成確認**

ビルドが終了すると、arch/arm/boot/ディレクトリと、arch/arm/boot/dts/以下にイメージファイル(Linux カーネルと DTB)が作成されています。

```
[PC ~/linux-3.14-x1-at[version]]$ ls arch/arm/boot/uImage
uImage
[PC ~/linux-3.14-x1-at[version]]$ ls arch/arm/boot/dts/armadillo_iotg_g3l.dtb
armadillo_iotg_g3l.dtb
```

11.3. Debian GNU/Linux ルートファイルシステムをビルドする

ここでは、x1-debian-builder を使って、Debian GNU/Linux ルートファイルシステムを構築する方法を示します。

x1-debuan-builder は ATDE6 等の PC で動作している Linux 上で Armadillo-IoT G3L 用の armhf アーキテクチャに対応した Debian GNU/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。

Armadillo-IoT G3L を一度起動した後のルートファイルシステム上には、使い方によっては ssh の秘密鍵や、動作ログ、シェルのコマンド履歴、ハードウェアの UUID に紐付く設定ファイル等が生成されています。そのまま、他の Armadillo-IoT G3L にルートファイルシステムをコピーした場合は、鍵の流出や UUID の不一致による動作の相違が起きる可能性があります。そのため、量産等に使用するルートファイルシステムは新規に x1-debuan-builder を使って構築することをお勧めします。

11.3.1. 出荷状態のルートファイルシステムアーカイブを構築する

出荷状態のルートファイルシステムアーカイブを構築する手順を次に示します。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、40 分程度かかります。

```
[armadillo ~]$ sudo apt-get update && sudo apt-get install qemu-user-static
[armadillo ~]$ tar xf x1-debian-builder-[VERSION].tar.gz
[armadillo ~]$ cd x1-debian-builder-[VERSION]
[armadillo ~]$ sudo ./build.sh
```

図 11.1 出荷状態のルートファイルシステムアーカイブを構築する手順

11.3.2. カスタマイズされたルートファイルシステムアーカイブを構築する

x1-debian-builder-*[VERSION]*/aiotg3l_resources 内のファイルを変更し、build.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。

11.3.2.1. ファイル/ディレクトリを追加する

aiotg3l_resources/ 以下に配置したファイルやディレクトリは resources ディレクトリを除いて、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。ファイルの UID と GID は共に root になります。

11.3.2.2. パッケージを変更する

aiotg3l_resources/resources/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-IoT G3L 上で "apt-get install" 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、ビルドログに以下のようなエラーメッセージが表示されて当該のパッケージが含まれないアーカイブが生成されます。

```
E: Unable to locate package XXXXX
```

図 11.2 誤ったパッケージ名を指定した場合に起きるエラーメッセージ

パッケージに依存する他のパッケージは明記しなくても、apt によって自動的にインストールされます。また、apt や dpkg 等の Debian GNU/Linux の根幹となるパッケージも自動的にインストールされます。

packages には lua と ruby のインタプリタや、Web サーバー(lighttpd)が含まれていますが、これらが不要な場合は、それぞれの行を削除してください。

openssh-server のような「パッケージのインストールの際に、自動的に秘密鍵を生成する」パッケージは、基本的に packages には追加せず、Armadillo を起動した後に "apt-get install" を使って個別にインストールしてください。

openssh-server を packages に追加した場合、構築したルートファイルシステムアーカイブを書き込んだ全ての Armadillo に、単一の公開鍵を使ってログインすることができてしまいます。もし、意図的に、複数の Armadillo で同一の秘密鍵を利用したい場合、脆弱性となり得ることを理解して適切な対策をとった上で利用してください。

12. 開発の基本的な流れ

この章では Armadillo-IoT G3L を使ったアプリケーションソフトウェアの 開発方法について説明します。

Armadillo-IoT G3L を使ったアプリケーションソフトウェア開発には、 Ruby 等の軽量スクリプト言語を使うことができます。

新たにパケット通信、各種アドオンボードを利用したセンサーからのデータ読み出し を実装したアプリケーションプログラムを実装するときは、Ruby 等の軽量スクリプト言語を 使った開発をお勧めします。

Armadillo-IoT G3L の出荷用のユーザーランドには、最初から Ruby インタプリタ がインストールされているので、PC と同じように開発を進めることができます。

もちろん、Ruby に限らず、Debian の提供する豊富なパッケージ群から Python や Go、 Haskel といったスクリプト言語を自由にインストールして使うことも可能です。

12.1. 軽量スクリプト言語によるセンサーデータの送信例(Ruby)

ここでは、サンプルとして Armadillo-IoT G3L に搭載された温度センサーの値を 定期的に HTTP POST でパラメータ名 "temp" に格納した値として送信する例を示します。

温度センサーからの値の取得は sysfs から可能です。

ここで作成するアプリケーションは Armadillo-IoT G3L で動作するクライアントと ATDE で動作するテスト用のサーバーの 2 つです。 ATDE で動作させるためのテスト用のサーバーは、典型的な HTTP プロトコルで アクセスできる Web API を持ったサービスを模擬しています。テスト用サーバーは 単に入力された POST リクエストの内容を変数に格納してコンソールに出力し、 クライアントには"Thanks!"という文字列を返します。

12.1.1. テスト用サーバーの実装

最初に ATDE6 にテスト用サーバーの動作に必要なパッケージをインストールします。

```
[PC ~]$ sudo apt-get install ruby
[PC ~]$ sudo gem install sinatra
```

図 12.1 ruby と sinatra のインストール

次にエディタで次のコードを入力して、server.rb として保存してください。

```
require 'sinatra'

post '/' do
  puts "Temperature is #{params[:temp]}"
  "Thanks!\n"
end
```

図 12.2 テスト用サーバー (server.rb)

12.1.2. テスト用サーバーの動作確認

Armadillo-IoT G3L でクライアントアプリケーションを動かす前に、テスト用サーバーの動作確認を行います。 動作確認は Armadillo-IoT G3L から cURL コマンドを使って、クライアントアプリケーションと同等の リクエストを送ってみます。

まず、ATDE6 の IP アドレス を確認しておきます。下記の例では、ip または ifconfig コマンドで確認すると ATDE6 の IP アドレス が 172.16.2.117 であることがわかります。

```
[PC ~]$ ip addr show eth0
     2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group ⏎
default qlen 1000
    link/ether 00:0c:29:30:b0:e0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.117/16 brd 172.16.255.255 scope global dynamic eth0
       valid_lft 65913sec preferred_lft 65913sec
    inet6 fe80::20c:29ff:fe30:b0e0/64 scope link
       valid_lft forever preferred_lft forever
```

図 12.3 IP アドレスの確認 (ip コマンド)

```
[PC ~]$ sudo ifconfig
eth0      Link encap:イーサネット  ハードウェアアドレス 00:0c:29:30:b0:e0
          inet アドレス:172.16.2.117 ブロードキャスト:172.16.255.255  マスク:255.255.0.0
          inet6 アドレス: fe80::20c:29ff:fe30:b0e0/64 範囲:リンク
          UP BROADCAST RUNNING MULTICAST  MTU:1500  メトリック:1
          RX パケット:132712 エラー:59329 損失:0 オーバラン:0 フレーム:0
          TX パケット:13632 エラー:0 損失:0 オーバラン:0 キャリア:0
      衝突(Collisions):0 TX キュー長:1000
          RX バイト:124163876 (118.4 MiB)  TX バイト:970956 (948.1 KiB)
          割り込み:19 ベースアドレス:0x2024
```

図 12.4 IP アドレスの確認 (ifconfig コマンド)

次の例のように server.rb を実行すると、全ての IP アドレスからのリクエストを 8081 番ポートで Web サーバーとして待ち受けます。

```
[PC ~]$ ruby server.rb -p 8081 -o 0.0.0.0
[2016-03-28 16:02:15] INFO  WEBrick 1.3.1
[2016-03-28 16:02:15] INFO  ruby 2.1.5 (2014-11-13) [i386-linux-gnu]
== Sinatra (v1.4.7) has taken the stage on 4567 for development with backup from WEBrick
[2016-03-28 16:02:15] INFO  WEBrick::HTTPServer#start: pid=10849 port=8081
```

ここで、Armadillo-IoT G3L から cURL を使ってテストデータを送ってみましょう。正しく通信できた場合は、"Thanks!" の文字列が表示されます。もし、"Connection refused" 等が表示された場合は、一旦 ATDE6 の IP アドレスに ping を送信してネットワークの設定に問題が無いか確認してください。

```
[Armadillo ~]$   curl -d "temp=30" 172.16.2.117:8081
Thanks!
```

図 12.5 curl によるテストデータの送信

正しく受信できた場合は、ATDE6 で起動しているテスト用サーバーが起動しているコンソールに下記の文字列が出力されます。

```
Temperature is 30
```

図 12.6 ATDE6 におけるテストデータの受信表示

12.2. クライアントの実装

Armadillo-IoT G3L で動作するクライアントを実装します。下記のコードをエディタで入力して、client.rb として保存してください。ファイルは、ATDE6 上で作成しても Armadillo-IoT G3L 上で、vi 等を使って作成しても構いません。ATDE6 で作成した場合は次の手順で、Armadillo-IoT G3L へ転送します。

```
require 'net/http'

uri = URI.parse(ARGV[0])
thermal_sys = "/sys/class/thermal/thermal_zone0/temp"
File.open(thermal_sys, "r") do |f|
  @temp=(f.read.to_f/1000).round(2)
end
response = Net::HTTP.post_form(uri, {"temp" => @temp})

puts response.body
```

図 12.7 温度送信クライアント(client.rb)

12.3. Armadillo-IoT G3L へのファイルの転送

ATDE6 上で作成したソースコードを Armadillo-IoT G3L に配置する方法の一例として、 ここでは、SSH を使った転送方法を説明します。

```
[armadillo ~]#  apt-get install openssh-server
```

図 12.8 Armadillo-IoT G3L への SSH サーバーのインストール

```
[ATDE ~]$ scp client.rb atmark@[armadillo の IP アドレス]:~/
```

図 12.9 ATDE6 から Armadillo-IoT G3L への client.rb の転送

12.4. クライアントの実行

作成した温度送信クライアントを実行します。第一引数にはテスト用サーバーが動いている ATDE6 の IP アドレスとポートを HTTP スキーマの URI で記述してください。

```
[armadillo ~]#  ruby client.rb  http://172.16.2.117:8081
Thanks!
```

図 12.10 クライアントの実行方法

正しくクライアントとの通信ができた場合、ATDE6 で動作しているサーバーのコンソールには小数点以下 2 ケタの温度が表示されます。

```
Temperature is 33.02
```

図 12.11 ATDE6 における温度データの受信表示

12.5. C 言語による開発環境

C/C++等の資産がある場合は、Armadillo 上で gcc/g++を使ってアプリケーションを コンパイルする事もできます。

12.5.1. 開発環境の準備

アプリケーションをコンパイルするために、Armadillo に gcc 等を含むツールチェーンを インストールします。 Armadillo のコンソールで次のコマンドを実行してください。

```
[armadillo ~]#  apt-get install build-essential
```

図 12.12 ツールチェーンのインストール

これで、gcc, make, gdb 等が使えるようになりました。 次に、アプリケーションのビルドに必要なライブラリとヘッダーファイルを インストールします。例えば libssl であれば次のコマンドでインストールする ことができます。

```
[armadillo ~]#  apt-get install libssl-dev
```

図 12.13 開発用パッケージのインストールの例 (libssl の場合)

例に示すように、コンパイルに必要なヘッダーファイルを含むパッケージは、 普通 -dev という名前が付いています。

必要なヘッダファイルの名前や、共有ライブラリのファイル名がわかっている 場合は、Debian プロジェクトサイトの「パッケージの内容を検索」からファイルの 含まれるパッケージの名前を探す事ができます。

Debian -- パッケージ パッケージの内容を検索 https://www.debian.org/distrib/packages#search_contents

また、パッケージの部分的な名前が分っている場合は「9.2. パッケージ管理」で紹介した、 apt-cache search コマンドを使って必要なパッケージを探す事もできます。

13. SMS を利用する

Armadillo-IoT G3L 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。

13.1. 初期設定

SMS が利用可能な契約の microSIM を挿入して LTE のデータ接続を行うと、ModemManager が自動的に必要な初期設定を行い SMS が利用可能になります。

LTE のデータ接続方法については、「7.2.6. LTE」を参照してください。

以下のようにコマンドを実行し、言語設定を行います。

```
[armadillo ~]# export LANG="ja_JP.UTF-8"
```

図 13.1 言語設定

LTE のデータ接続後、SMS の受信は自動的に行われます。

13.2. SMS を送信する

SMS を作成するには、次のように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 13.2 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 13.3 SMS 番号の確認

以下のようにコマンドを実行し、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 13.4 SMS の送信

13.3. SMS リストを表示する

SMS リストを表示するには、次のようにコマンドを実行します。

```
[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 13.5 SMS の一覧の表示

13.4. SMS の内容を表示する

SMS の内容を表示するには、次のように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 13.6 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 G3L に標準搭載されている、ELS31-J では、最大 10 件まで SMS を保存することが可能です。

ELS31-J の内蔵ストレージに 10 件 SMS を保存した状態で Armadillo-IoT G3L に SMS を送信しても、Armadillo-IoT G3L は受信を行いません。

受信を行うには、ELS31-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。

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 のストレージに SMS が保存されています。SIM のストレージに保存できる SMS の件数は SIM 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 G3L の電源を切断してもデータが保持されます。

13.5. SMS を削除する

SMS を削除するには、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -m 0 --messaging-delete-sms [SMS 番号]
```

図 13.7 SMS の削除

13.6. SMS を他のストレージに移動する

SIM のストレージに SMS を移動するには、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号] --store-in-storage="sm"
```

図 13.8 SIM 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、次のようにコマンドを実行します。

```
[armadillo ~]# mmcli -s [SMS 番号] --store-in-storage="me"
```

図 13.9 LTE モジュールの内蔵ストレージに SMS を移動

14. i.MX7Dual の電源制御

本章では、パワーマネジメント IC による i.MX7Dual の電力供給を制御する方法について説明します。

i.MX7Dual の電源は、パワーマネジメント IC によって制御されています。パワーマネジメント IC の電圧出力を停止・開始することで、i.MX7Dual の電源を ON または OFF にすることができます。

14.1. ユーザースイッチ(SW2)の操作による制御

ユーザースイッチ(SW2)の操作によって、i.MX7Dual の電源を ON、または OFF にすることができます。

Linux Kernel が起動した状態でユーザースイッチ(SW2)を 10 秒間長押しすることで、poweroff が実行され、i.MX7Dual の電源が OFF されます。保守モードで起動した場合は、ユーザースイッチ(SW2)による電源 OFF を行うことはできません。

その後、ユーザースイッチ(SW2)を押すことで、i.MX7Dual の電源が ON になり起動することができます。

ユーザースイッチ(SW2)の位置については、「3.4. Armadillo-IoT ゲートウェイ G3L の外観」を参照してください。

14.2. RTC による制御

RTC のアラーム割り込みによって、i.MX7Dual の電源を ON にすることができます。

アラーム割り込みは、sysfs RTC クラスディレクトリ以下の wakealarm ファイルから利用できます。

wakealarm ファイルに UNIX エポックからの経過秒数、または先頭に+を付けて現在時刻からの経過秒数を書き込むと、アラーム割り込み発生時刻を指定できます。

3600 秒後、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 ~]# echo +3600 > /sys/class/rtc/rtc0/wakealarm
```

図 14.1 アラーム割り込みの設定

コマンド実行後、 「14.1. ユーザースイッチ(SW2)の操作による制御」を参照し、i.MX7Dual の電源を OFF にします。

3600 秒後、アラーム割り込みによって i.MX7Dual の電源が ON になります。

15. 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 2GByte 以上の microSD カードを必要とします。以下では、例として Debian GNU/Linux 8(コードネーム jessie)を SD ブートする手順を示しますが、他の OS を SD ブートすることも可能です。

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。

microSD カードに対する作業は、ATDE で行います。そのため、ATDE に microSD カードを接続する必要があります。詳しくは「4.3.2. 取り外し可能デバイスの使用」を参照してください。

ATDE に microSD カードを接続すると、自動的に/media/ディレクトリにマウントされます。本章に記載されている手順を実行するためには、次のように microSD カードをアンマウントしておく必要があります。

```
[PC ~]$ 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,sh
ortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
[PC ~]$ sudo umount /dev/sdb1
```

図 15.1 自動マウントされた microSD カードのアンマウント

本章で使用するブートローダーイメージファイルなどは、評価セット付属の DVD に収録されています。最新版のファイルは、"Armadillo サイト"でダウンロードすることができます。新機能の追加や不具合の修正などが行われているため、DVD に収録されているものよりも新しいバージョンがリリースされているかを確認して、最新バージョンを利用することを推奨します。

Armadillo サイト - Armadillo-IoT G3L ドキュメント・ダウンロード

http://armadillo.atmark-techno.com/armadillo-iot-g3l/downloads

15.1. ブートディスクの作成

ATDE でブートディスクを作成します。ブートディスクの作成に使用するファイルを次に示します。

表 15.1 ブートディスクの作成に使用するファイル

ファイル	ファイル名
SD ブート用ブートローダーイメージ	u-boot-x1-sd-*[version]*.bin

「表 15.2. ブートディスクの構成例」に示すブートディスクを作成する手順を、「手順 15.1. ブートディスクの作成例」に示します。

表 15.2 ブートディスクの構成例

パーティション番号	パーティションサイズ	ファイルシステム	説明
1	128MByte	FAT32	SD ブート用のブートローダーイメージを配置します。
2	残り全て	ext4	ルートファイルシステムを構築するために ext4 ファイルシステムを構築しておきます。

手順 15.1 ブートディスクの作成例

1. SD ブート用のブートローダーイメージファイルを取得します。

```
[PC ~]$ ls
u-boot-x1-sd-[version].bin
```

ブートローダーイメージファイルには以下 2 種類があります。

格納場所	イメージファイル
microSD カード	u-boot-x1-sd-[version].bin
フラッシュメモリ	u-boot-x1-[version].bin

2. microSD カードに 2 つのプライマリパーティションを作成します。

```
[PC ~]$ sudo fdisk /dev/sdb ❶

Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): o ❷
Created a new DOS disklabel with disk identifier 0x2b685734.

Command (m for help): n ❸
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): ❹

Using default response p.
Partition number (1-4, default 1): ❺
First sector (2048-7761919, default 2048): ❻
Last sector, +sectors or +size{K,M,G,T,P} (2048-7761919, default 7761919): +128M ❼
```

```
Created a new partition 1 of type 'Linux' and of size 128 MiB.

Command (m for help): n ❽
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p):  ❾

Using default response p.
Partition number (2-4, default 2):  ❿
First sector (264192-7761919, default 264192):  ⓫
Last sector, +sectors or +size{K,M,G,T,P} (264192-7761919, default 7761919):  ⓬

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

Command (m for help): t ⓭
Partition number (1,2, default 2): 1 ⓮
Hex code (type L to list all codes): b ⓯

If you have created or modified any DOS 6.x partitions, please see the fdisk
documentation for additional information.
Changed type of partition 'Linux' to 'W95 FAT32'.

Command (m for help): w ⓰
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[PC ~]$
```

❶ microSD カードのパーティションテーブル操作を開始します。USB メモリなどを接続している場合は、SD カードのデバイスファイルが sdc や sdd など本実行例と異なる場合があります。

❷ 新しく空の DOS パーティションテーブルを作成します。

❸ 新しくパーティションを追加します。

❹ パーティション種別にはデフォルト値(p: プライマリ)を指定するので、そのまま改行を入力してください。

❺ パーティション番号にはデフォルト値(1)を指定するので、そのまま改行を入力してください。

❻ 開始セクタにはデフォルト値(使用可能なセクタの先頭)を使用するので、そのまま改行を入力してください。

❼ 最終シリンダは、128MByte 分を指定します。

❽ 新しくパーティションを追加します。

❾ パーティション種別にはデフォルト値(p: プライマリ)を指定するので、そのまま改行を入力してください。

❿ パーティション番号にはデフォルト値(2)を指定するので、そのまま改行を入力してください。

⓫ 開始セクタにはデフォルト値(第 1 パーティションの最終セクタの次のセクタ)を使用するので、そのまま改行を入力してください。

⓬ 最終セクタにはデフォルト値(末尾セクタ)を使用するので、そのまま改行を入力してください。

⓭ パーティションのシステムタイプを変更します。

⓮ 第 1 パーティションを指定します。

⓯ パーティションのシステムタイプに 0xb(Win95 FAT32)を指定します。

⓰ 変更を microSD カードに書き込みます。

3. パーティションリストを表示し、2 つのパーティションが作成されていることを確認してください。

```
[PC ~]$ sudo fdisk -l /dev/sdb

Disk /dev/sdb: 3.7 GiB, 3974103040 bytes, 7761920 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x2b685734

Device     Boot  Start     End Sectors  Size Id Type
/dev/sdb1         2048  264191  262144  128M  b W95 FAT32
/dev/sdb2       264192 7761919 7497728  3.6G 83 Linux
```

4. それぞれのパーティションにファイルシステムを構築します。

```
[PC ~]$ sudo mkfs.vfat -F 32 /dev/sdb1  ❶
mkfs.fat 3.0.27 (2014-11-12)
[PC ~]$ sudo mkfs.ext4 -L rootfs /dev/sdb2  ❷
Creating filesystem with 937216 4k blocks and 234320 inodes
Filesystem UUID: AAAAAAAA-BBBB-CCCC-DDDD-EEEEEEEEEEEE
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[PC ~]$
```

❶ 第 1 パーティションに FAT32 ファイルシステムを構築します。

❷ 第 2 パーティションに ext4 ファイルシステムを構築します。ボリュームラベルには "rootfs"を設定します。

5. SD ブート用のブートローダーイメージファイルを microSD カードに書き込みます。

```
[PC ~]$ ls
u-boot-x1-sd-[version].bin
[PC ~]$ sudo dd if=u-boot-x1-sd-[version].bin of=/dev/sdb bs=1k seek=1
```

15.2. ルートファイルシステムの構築

「15.1. ブートディスクの作成」で作成したブートディスクにルートファイルシステムを構築します。

Debian GNU/Linux のルートファイルシステムを構築することができます。ルートファイルシステムの構築に使用するファイルを次に示します。

表 15.3 ルートファイルシステムの構築に使用するファイル

Linux ディストリビューション	ファイル名	ファイルの説明
Debian GNU/Linux	debian-jessie-armhf_aiotg-g3l_*[version]*.tar.gz	ARM(armhf)アーキテクチャ用 Debian GNU/Linux 8(コードネーム jessie)のルートファイルシステムアーカイブ

15.2.1. Debian GNU/Linux のルートファイルシステムを構築する

Debian GNU/Linux ルートファイルシステムアーカイブから、ルートファイルシステムを構築する手順を次に示します。

手順 15.2 Debian GNU/Linux ルートファイルシステムアーカイブからルートファイルシステムを構築する

1. Debian GNU/Linux ルートファイルシステムアーカイブを準備しておきます。

```
[PC ~]$ ls
debian-jessie-armhf_aiotg-g3l_[version].tar.gz
```

2. ルートファイルシステムをブートディスクの第 2 パーティションに構築します。

```
[PC ~]$ mkdir sd ❶
[PC ~]$ sudo mount -t ext4 /dev/sdb2 sd ❷
[PC ~]$ sudo tar zxf debian-jessie-armhf_aiotg-g3l_[version].tar.gz -C sd ❸
[PC ~]$ sudo umount sd ❹
[PC ~]$ rmdir sd ❺
```

❶ microSD カードをマウントするための sd/ディレクトリを作成します。

❷ 第 2 パーティションを sd/ディレクトリにマウントします。

❸ ルートファイルシステムアーカイブを sd/ディレクトリに展開します。

❹ sd/ディレクトリにマウントしたブートディスクの第 2 パーティションをアンマウントします。

❺ sd/ディレクトリを削除します。

アンマウントが完了する前に microSD カードを作業用 PC から取り外すと、microSD カードのデータが破損する場合があります。

15.3. Linux カーネルイメージと DTB の配置

「15.1. ブートディスクの作成」で作成したブートディスクに Linux カーネルイメージおよび DTB(Device Tree Blob)を配置します。使用するファイルを次に示します。以降、DTB(Device Tree Blob)を DTB と表記します。

表 15.4 ブートディスクの作成に使用するファイル

ファイル	ファイル名
Linux カーネルイメージ	ulmage-x1-*[version]*
DTB	armadillo_iotg_g3l-*[version]*.dtb

microSD カードに Linux カーネルイメージおよび DTB を配置する際は、次の条件を満たすようにしてください。この条件から外れた場合、ブートローダーが Linux カーネルイメージまたは DTB を検出することができなくなる場合があります。

表 15.5 ブートローダーが Linux カーネルを検出可能な条件

項目	条件
ファイルシステム	FAT32
圧縮形式	非圧縮
Linux カーネルイメージファイル名	uImage
DTB ファイル名	armadillo_iotg_g3l.dtb

Linux カーネルイメージおよび DTB をブートディスクに配置する手順を次に示します。

手順 15.3 Linux カーネルイメージおよび DTB の配置

1. Linux カーネルイメージおよび DTB を準備しておきます。

```
[PC ~]$ ls
uImage-x1-[version]  armadillo_iotg_g3l-[version].dtb
```

2. Linux カーネルイメージをブートディスクの第 1 パーティションに配置します。

```
[PC ~]$ mkdir sd ❶
[PC ~]$ sudo mount -t vfat /dev/sdb1 sd ❷
[PC ~]$ sudo cp uImage-x1-[version] sd/uImage ❸
[PC ~]$ sudo cp armadillo_iotg_g3l-[version].dtb sd/armadillo_iotg_g3l.dtb ❹
[PC ~]$ sudo umount sd ❺
[PC ~]$ rmdir sd ❻
```

❶ microSD カードをマウントするための sd/ディレクトリを作成します。

❷ 第 2 パーティションを sd/ディレクトリにマウントします。

❸ Linux カーネルイメージを sd/ディレクトリにコピーします。

❹ DTB を sd/ディレクトリにコピーします。

❺ sd/ディレクトリにマウントしたブートディスクの第 2 パーティションをアンマウントします。

❻ sd/ディレクトリを削除します。

アンマウントが完了する前に microSD カードを作業用 PC から取り外すと、microSD カードのデータが破損する場合があります。

15.4. SD ブートの実行

「15.1. 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。

Armadillo に電源を投入する前に次の準備を行います。

1. microSD スロット(メインユニット CON12)にブートディスクを接続します。
2. JP1 をショートに設定します。

準備が完了後、電源を投入すると SD ブートさせることができます。SD ブートに成功した場合は、saveenv を実行すると「図 15.2. SD ブート時の saveenv メッセージ」のようにメッセージが表示されます。環境変数の保存先が"MMC"になっていることを確認してください。

```
=> saveenv
Saving Environment to MMC...
Writing to MMC(0)... done
=>
```

図 15.2 SD ブート時の saveenv メッセージ

16. 電気的仕様

16.1. 絶対最大定格

表 16.1 絶対最大定格

項目	記号	Min.	Max.	単位	備考
電源電圧	VIN	-0.3	28.0	V	
入力電圧(USB 信号)	VI_USB	-0.3	3.63	V	USBx_DP,USBx_DM
入出力電圧(RS422/RS485 信号)	VI,VO	-8	12.5	V	TX+,TX-,RX+,RX-,DATA+,DATA-
動作温度範囲	Topr	-10	50	℃	ただし結露なきこと

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。

16.2. 推奨動作条件

表 16.2 推奨動作条件

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	VIN	10.0	12.0	26.4	V	
使用周囲温度	Ta	-10	25	50	℃	ただし結露なきこと

16.3. 入出力インターフェースの電気的仕様

表 16.3 入出力インターフェース電源の電気的仕様

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	USB1_VBUS	4.75	5	5.25	V	最大供給電流:500mA

17. インターフェース仕様

Armadillo-IoT のインターフェース仕様について説明します。

17.1. インターフェースレイアウト

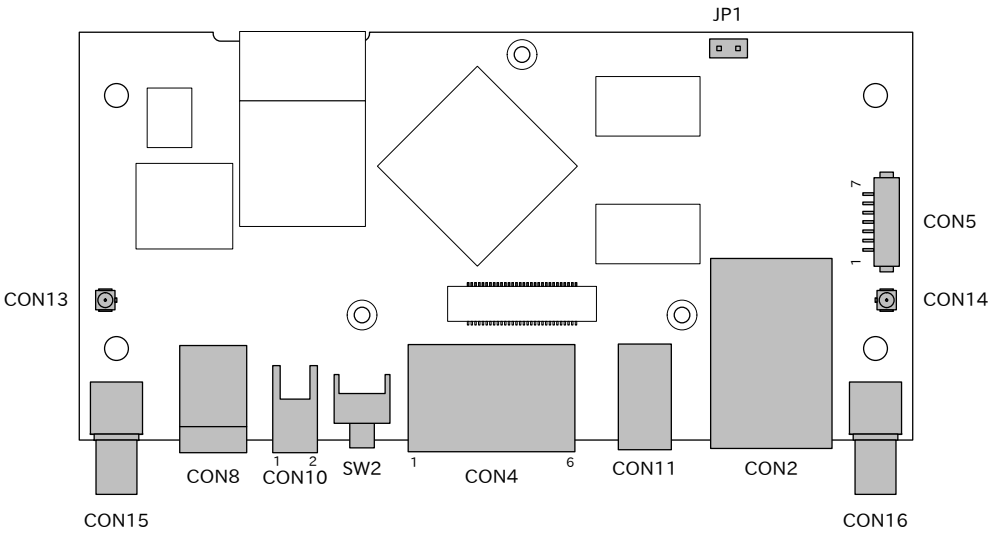


図 17.1 Armadillo-IoT メインユニット インターフェースレイアウト(A 面)

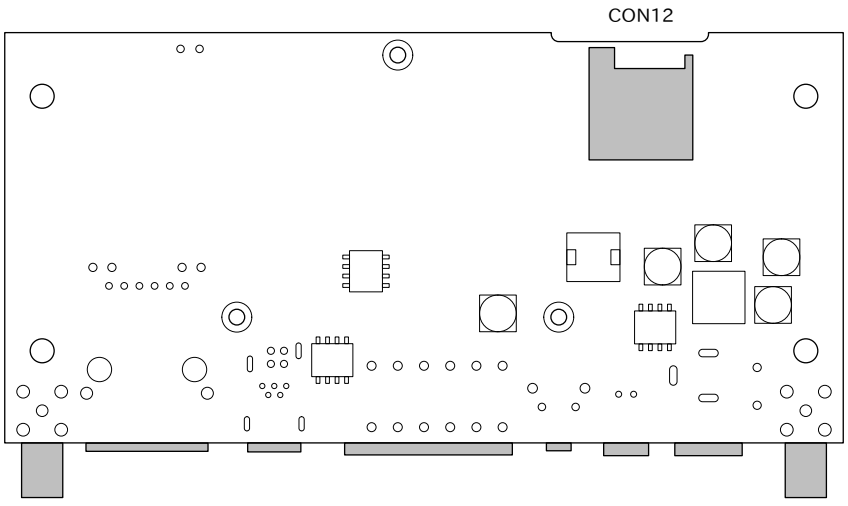


図 17.2 Armadillo-IoT メインユニット インターフェースレイアウト(B 面)

表 17.1 Armadillo-IoT メインユニット 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
CON2	LAN インターフェース	08B0-1X1T-36-F	Bel Fuse
CON4	シリアルインターフェース	1990779	Phoenix Contact
CON5	デバッグシリアルインターフェース	DF13C-7P-1.25V(51)	HIROSE ELECTRIC
CON8	電源入力インターフェース 1	PJ-102AH	CUI
CON10	電源入力インターフェース 2	S02B-PASK-2(LF)(SN)	J.S.T. Mfg.
CON11	USB インターフェース	UBAL-4R-D14-4S(LF)(SN)	J.S.T. Mfg.
CON12	microSD インターフェース	SDHL-8BNS-K-363-A0-ETB(HF)	HIROSE ELECTRIC
CON13	アンテナインターフェース 3	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
CON14	アンテナインターフェース 4	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
CON15	アンテナインターフェース 1	S-037	COSMTEC RESOURCES
CON16	アンテナインターフェース 2	S-037	COSMTEC RESOURCES
SW2	ユーザースイッチ	SKHHLUA010	ALPS ELECTRIC
JP1	起動デバイス設定ジャンパ	2A-2PA-2.54DSA(71)	HIROSE ELECTRIC

「表 17.1. Armadillo-IoT メインユニット インターフェース一覧」に記載した部品型番は、必ずしも搭載されていることを保証しているわけではありません。お手元の製品の搭載部品は、アットマークテクノ ユーザーズサイトからダウンロード可能な、納入仕様書および変更履歴表にてご確認ください。

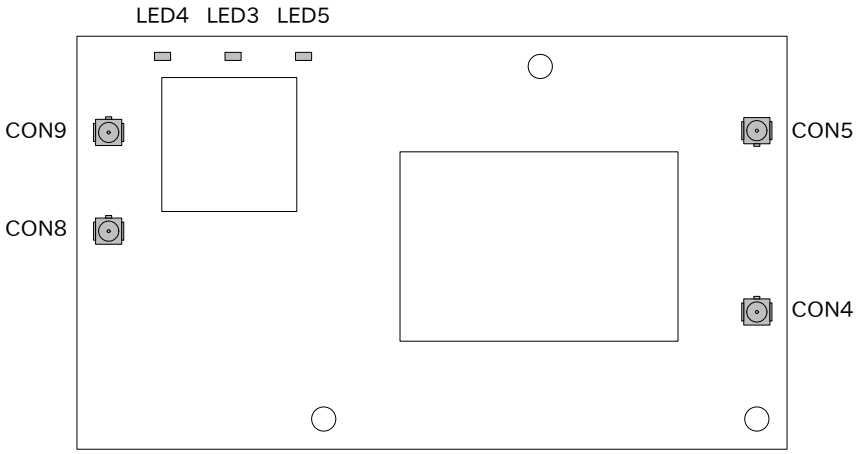


図 17.3 Armadillo-IoT サブユニット インターフェースレイアウト(A 面)

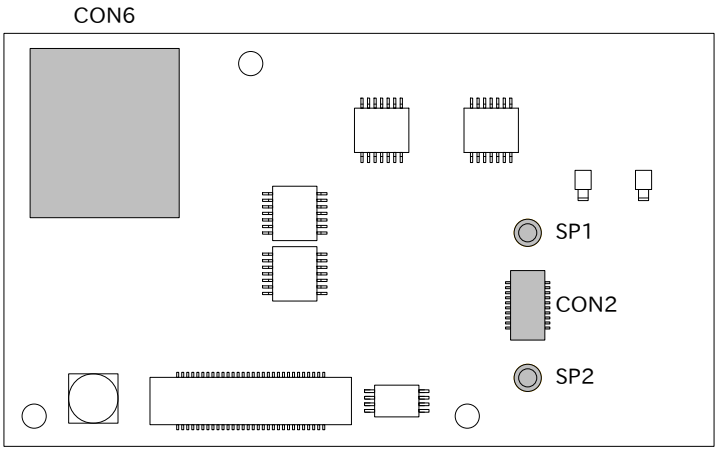


図 17.4 Armadillo-IoT サブユニット インターフェースレイアウト(B 面)

表 17.2 Armadillo-IoT サブユニット 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
CON2	Wi-SUN モジュールインターフェース	20P3.0-JMCS-G-B-TF(N)	J.S.T. Mfg.
CON4	LTE アンテナインターフェース 1	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
CON5	LTE アンテナインターフェース 2	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
CON6	microSIM インターフェース	CIM-J78	MITSUMI
CON8	WLAN アンテナインターフェース 1	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
CON9	WLAN アンテナインターフェース 2	U.FL-R-SMT-1(10)	HIROSE ELECTRIC
LED3	ユーザー LED3	SML-D13M8WT86	ROHM
LED4	ユーザー LED4	SML-D13M8WT86	ROHM
LED5	ユーザー LED5	SML-D13M8WT86	ROHM
SP1	Wi-SUN モジュールスタッド	TH-1.6-3.0-M2	Mac-Eight
SP2	Wi-SUN モジュールスタッド	TH-1.6-3.0-M2	Mac-Eight

「表 17.2. Armadillo-IoT サブユニット インターフェース一覧」に記載した部品型番は、必ずしも搭載されていることを保証しているわけではありません。お手元の製品の搭載部品は、アットマークテクノ ユーザーズサイトからダウンロード可能な、納入仕様書および変更履歴表にてご確認ください。

17.2. メインユニット CON2 LAN インターフェース

メインユニット CON2 は 10BASE-T/100BASE-TX に対応した LAN インターフェースです。信号線は Ethernet PHY を経由して、i.MX7Dual の Ethernet MAC(ENET2)に接続されています。

表 17.3 メインユニット CON2 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+

ピン番号	ピン名	I/O	説明
4	-	-	
5	-	-	
6	RX-	In/Out	受信データ-
7	-	-	
8	-	-	

表 17.4 LAN コネクタ LED

名称	状態	説明
LINK_ACTIVITY_LED	消灯	リンクが確立されていない
	点灯(黄色)	リンクが確立されている
	点滅(黄色)	リンクが確立されており、データを送受信している
SPEED_LED	消灯	10Mbps で接続されている
	点灯(緑色)	100Mbps で接続されている

17.3. メインユニット CON4 シリアルインターフェース

メインユニット CON4 は RS422/RS485 のシリアルインターフェースで、端子台を使用しています。信号線は i.MX7Dual の UART2 インターフェースに接続されています。

表 17.5 メインユニット CON4 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	RS422_TX-/ RS485_DATA-	In/Out	RS422 の送信データ(マイナス信号)、RS485 の送受信データ(マイナス信号)
3	RS422_TX+/ RS485_DATA+	In/Out	RS422 の送信データ(プラス信号)、RS485 の送受信データ(プラス信号)
4	GND	Power	電源(GND)
5	RS422_RX-	In	RS422 の受信データ(マイナス信号)
6	RS422_RX+	In	RS422 の受信データ(プラス信号)

17.4. メインユニット CON5 デバッグシリアルインターフェース

メインユニット CON5 はデバッグ用のシリアルインターフェースです。信号線は i.MX7Dual の UART_5 インターフェースに接続されています。

表 17.6 メインユニット CON5 信号配列

ピン番号	ピン名	I/O	説明
1	DEBUG_UART_RXD	In	受信データ、i.MX7Dual の GPIO1_IO06 ピンに接続
2	GND	Power	電源(GND)
3	DEBUG_UART_TXD	Out	送信データ、i.MX7Dual の GPIO1_IO07 ピンに接続
4	VCC_3.3V	Power	電源(VCC_3.3V)
5	DEBUG_UART_CTS	In	送信可能、i.MX7Dual の GPIO1_IO05 ピンに接続
6	BOOTLOADER_EN_B	In	起動モード設定、i.MX7Dual の GPIO1_IO09 ピンに接続 (Low:保守モード、High:OS 自動起動モード)
7	DEBUG_UART_RTS	Out	送信要求、i.MX7Dual の GPIO1_IO04 ピンに接続

17.5. メインユニット CON8 電源入力インターフェース 1

メインユニット CON8 は電源供給用のインターフェースです。「図 17.5. 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。

表 17.7 メインユニット CON8 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源入力(VIN)
2	GND	Power	電源(GND)

図 17.5 AC アダプタの極性マーク

メインユニット CON8 を使用する場合、同時にメインユニット CON10 から電源供給しないでください。故障の原因となる可能性があります。

17.6. メインユニット CON10 電源入力インターフェース 2

メインユニット CON10 は電源供給用のインターフェースです。

表 17.8 メインユニット CON10 信号配列

ピン番号	ピン名	I/O	説明
1	VIN	Power	電源入力(VIN)
2	GND	Power	電源(GND)

メインユニット CON10 を使用する場合、同時にメインユニット CON8 から電源供給しないでください。故障の原因となる可能性があります。

17.7. メインユニット CON11 USB インターフェース

メインユニット CON11 は TypeA の USB2.0 ホストインターフェースです。信号線は i.MX7Dual の USB1 インターフェースに接続されています。

表 17.9 メインユニット CON11 信号配列

ピン番号	ピン名	I/O	説明
1	USB1_VBUS	Power	USB 電源出力(USB1_VBUS)
2	USB1_DM	In/Out	USB マイナス側信号 i.MX7Dual の USB_OTG1_DN に接続
3	USB1_DP	In/Out	USB プラス側信号 i.MX7Dual の USB_OTG1_DP に接続
4	GND	Power	電源(GND)

17.8. メインユニット CON12 microSD インターフェース

メインユニット CON12 は microSD ソケットインターフェースです。信号線は i.MX7Dual の SDIO1 インターフェースに接続されています。

表 17.10 メインユニット CON12 信号配列

ピン番号	ピン名	I/O	説明
1	SD_DAT2	In/Out	SD データバス(bit2) i.MX7Dual の SD1_DATA2 に接続
2	SD_DAT3	In/Out	SD データバス(bit3) i.MX7Dual の SD1_DATA3 に接続
3	SD_CMD	In/Out	SD コマンド/レスポンス i.MX7Dual の SD1_CMD に接続
4	SD_VDD	Power	SD 電源出力(SD_VDD)
5	CLK	Out	SD クロック i.MX7Dual の SD1_CLK に接続
6	GND	Power	電源(GND)
7	SD_DAT0	In/Out	SD データバス(bit0) i.MX7Dual の SD1_DATA0 に接続
8	SD_DAT1	In/Out	SD データバス(bit1) i.MX7Dual の SD1_DATA1 に接続

17.9. メインユニット CON13 アンテナインターフェース 3

メインユニット CON13 は U.FL アンテナインターフェースです。RF ケーブルを使用してサブユニットのアンテナインターフェースと接続します。メインユニット CON13 はメインユニット上でメインユニット CON15 に接続されています。

アンテナ端子にアンテナケーブルを接続する際、無理な力を加えると破損の原因となりますので、 十分にご注意ください。

17.10. メインユニット CON14 アンテナインターフェース 4

メインユニット CON14 は U.FL アンテナインターフェースです。RF ケーブルを使用してサブユニットのアンテナインターフェースと接続します。メインユニット CON14 はメインユニット上でメインユニット CON16 に接続されています。

アンテナ端子にアンテナケーブルを接続する際、無理な力を加えると破損の原因となりますので、 十分にご注意ください。

17.11. メインユニット CON15 アンテナインターフェース 1

メインユニット CON15 は SMA アンテナインターフェースです。外付アンテナの取り付けに使用します。メインユニット CON15 はメインユニット上でメインユニット CON13 に接続されています。

アンテナ端子に外付アンテナを取り付ける際、無理な力を加えると破損の原因となりますので、 十分にご注意ください。

17.12. メインユニット CON16 アンテナインターフェース 2

メインユニット CON16 は SMA アンテナインターフェースです。外付アンテナの取り付けに使用します。メインユニット CON16 はメインユニット上でメインユニット CON14 に接続されています。

アンテナ端子に外付アンテナを取り付ける際、無理な力を加えると破損の原因となりますので、 十分にご注意ください。

17.13. メインユニット SW2 ユーザースイッチ

メインユニット SW2 はユーザー側で自由に使用できるタクトスイッチです。

表 17.11 ユーザースイッチ SW2 の接続

部品番号	説明
SW2	i.MX7Dual の GPIO1_IO02 に接続 (ON:Low、OFF:High)

17.14. メインユニット JP1 起動デバイス設定ジャンパ

メインユニット JP1 は起動デバイスを設定するジャンパです。

表 17.12 メインユニット JP1 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_3.3V	Power	電源(VCC_3.3V)
2	SDBOOT_EN	In	起動デバイス設定、i.MX7Dual の BOOT_MODE0 に接続 (Low:SPI フラッシュメモリブート、High:SD ブート)

表 17.13 ジャンパの機能

部品番号	機能	動作
メインユニット JP1	起動デバイス設定	オープン:SPI フラッシュメモリのブートローダーを起動 ショート:CON12 に挿入された microSD カードのブートローダーを起動

17.15. サブユニット CON2 Wi-SUN モジュールインターフェース

サブユニット CON2 は ROHM の Wi-SUN モジュールを接続するためのインターフェースです。信号線は i.MX7Dual の UART3 インターフェースに接続されています。

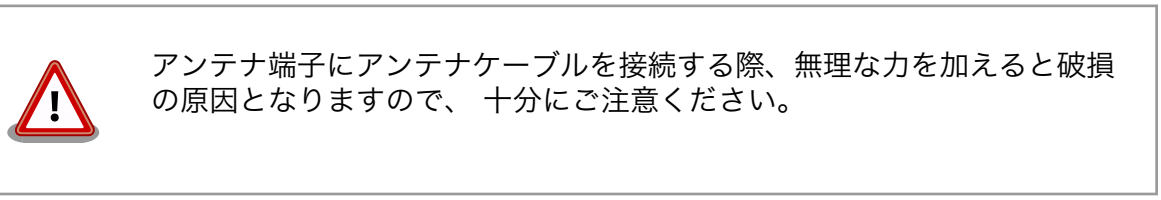
表 17.14 サブユニット CON2 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_3.3V	Power	電源(+3.3V)
2	GND	Power	電源(GND)
3	Wi_TXD	In	Wi-SUN シリアル送信 i.MX7Dual の UART3_RXD に接続
4	Wi_RXD	Out	Wi-SUN シリアル受信 i.MX7Dual の UART3_TXD に接続
5	Wi_NMIX	Out	Wi-SUN スリープ信号 i.MX7Dual の GPIO4_IO02 に接続
6	Wi_RESET	Out	Wi-SUN リセット信号 i.MX7Dual の GPIO4_IO03 に接続
7	GND	Power	電源(GND)
8	GND	Power	電源(GND)
9	GND	Power	電源(GND)
10	GND	Power	電源(GND)
11	VCC_3.3V	Power	電源(+3.3V)
12	GND	Power	電源(GND)

ピン番号	ピン名	I/O	説明
13	GND	Power	電源(GND)
14	Wi_CTS	Out	Wi-SUN 送信許可 i.MX7Dual の UART3_CTS_B に接続
15	Wi_RTS	In	Wi-SUN 送信要求 i.MX7Dual の UART3_RTS_B に接続
16	NC	-	未接続
17	NC	-	未接続
18	NC	-	未接続
19	GND	Power	電源(GND)
20	Wi_MOD	Out	Wi-SUN デバッグ切替 i.MX7Dual の GPIO3_IO27 に接続

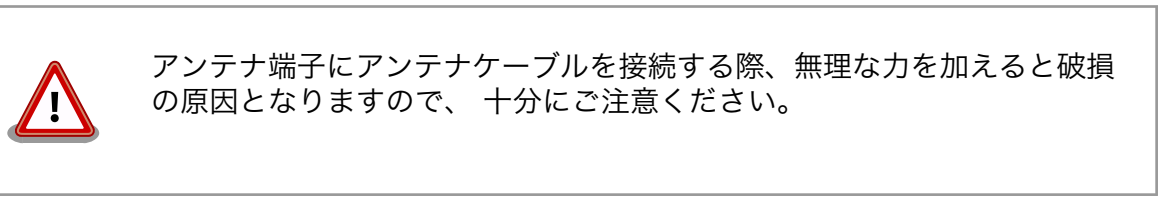
17.16. サブユニット CON4 LTE アンテナインターフェース 1

サブユニット CON4 は LTE モジュール(ELS31-J)用の U.FL アンテナインターフェースです。RF ケーブルを使用してメインユニットのアンテナインターフェース 4 と接続します。



17.17. サブユニット CON5 LTE アンテナインターフェース 2

サブユニット CON5 は LTE モジュール(ELS31-J)用の U.FL アンテナインターフェースです。RF ケーブルを使用してメインユニットのアンテナインターフェース 3 と接続します。

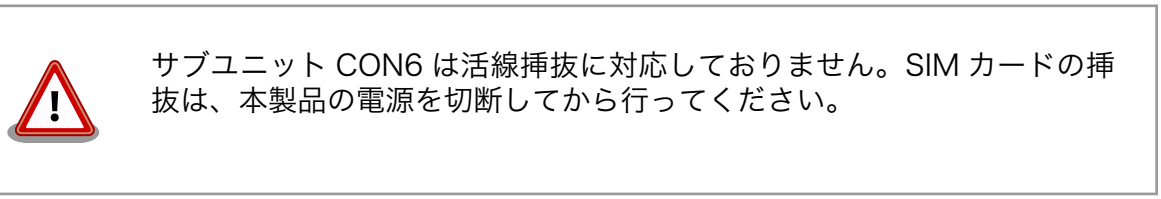


17.18. サブユニット CON6 microSIM インターフェース

サブユニット CON6 は LTE モジュール(ELS31-J)用の microSIM インターフェースです。

表 17.15 サブユニット CON6 信号配列

ピン番号	ピン名	I/O	説明
1	GND	Power	電源(GND)
2	SIM_VCC	Power	SIM 電源、LTE モジュールの CCVCC に接続
3	SIM_RST	Out	SIM リセット、LTE モジュールの CCRST に接続
4	SIM_CLK	Out	SIM クロック、LTE モジュールの CCCLK に接続
5	NC	-	未接続
6	SIM_I/O	In	SIM データ、LTE モジュールの CCIO に接続



17.19. サブユニット CON8 WLAN アンテナインターフェース 1

サブユニット CON8 は WLAN+BT コンボモジュール(WL1837MOD)用のアンテナインターフェースです。

アンテナ端子にアンテナケーブルを接続する際、無理な力を加えると破損の原因となりますので、 十分にご注意ください。

17.20. サブユニット CON9 WLAN アンテナインターフェース 2

サブユニット CON9 は WLAN+BT コンボモジュール(WL1837MOD)用のアンテナインターフェースです。

アンテナ端子にアンテナケーブルを接続する際、無理な力を加えると破損の原因となりますので、 十分にご注意ください。

17.21. サブユニット LED3 ユーザー LED3

ユーザーが自由に機能を設定できる緑色 LED です。

表 17.16 ユーザー LED3 の接続

部品番号	説明
LED3	i.MX7Dual の GPIO3_IO10 に接続 (Low:消灯、High:点灯)

17.22. サブユニット LED4 ユーザー LED4

ユーザーが自由に機能を設定できる緑色 LED です。

表 17.17 ユーザー LED4 の接続

部品番号	説明
LED4	i.MX7Dual の GPIO3_IO11 に接続 (Low:消灯、High:点灯)

17.23. サブユニット LED5 ユーザー LED5

ユーザーが自由に機能を設定できる緑色 LED です。

表 17.18 ユーザー LED5 の接続

部品番号	説明
LED5	i.MX7Dual の GPIO3_IO12 に接続 (Low:消灯、High:点灯)

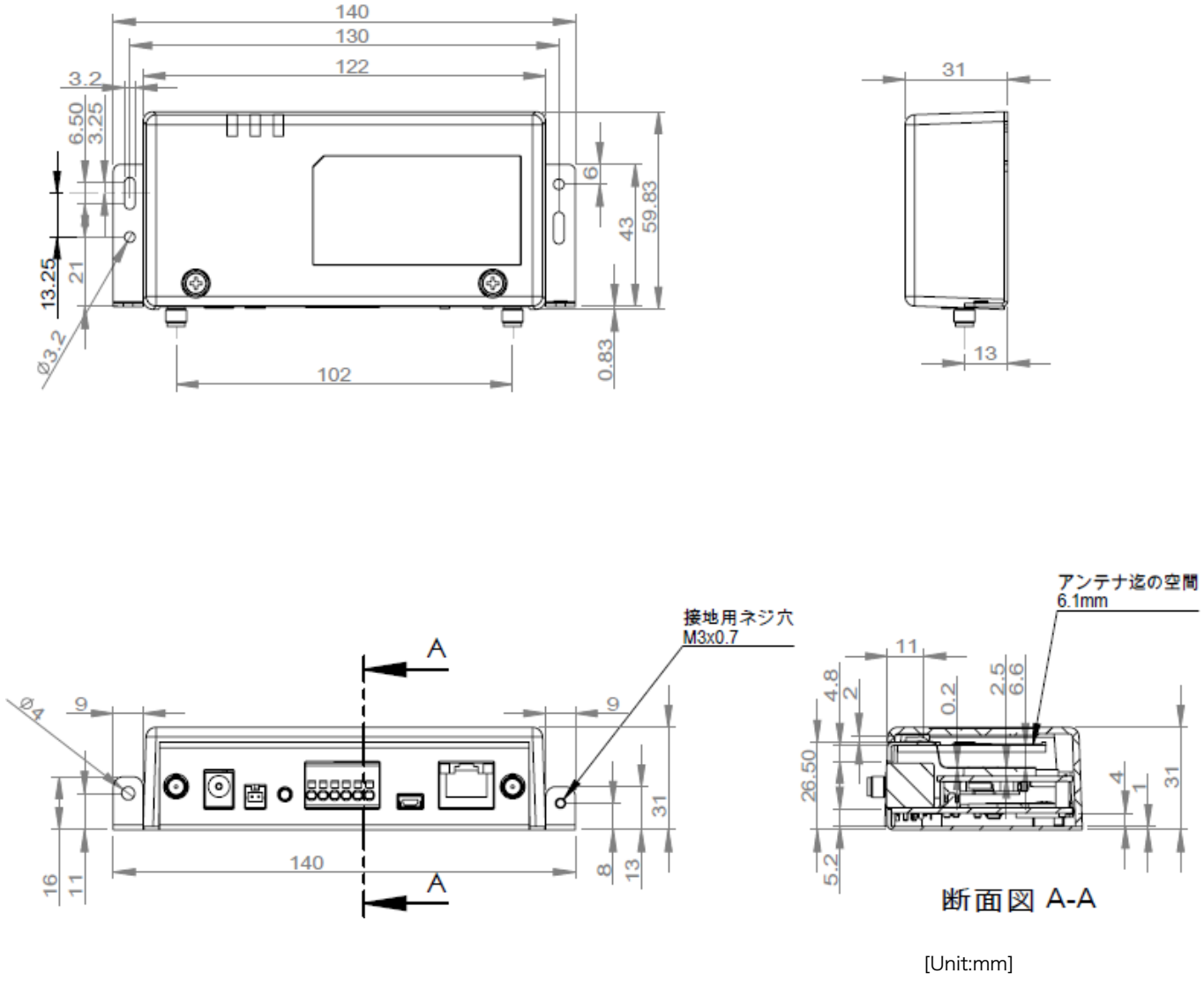
17.24. サブユニット SP1 Wi-SUN モジュールスタッド

Wi-SUN モジュールを取り付けるためのスルーホールタップタイプのスペーサーです。

17.25. サブユニット SP2 Wi-SUN モジュールスタッド

Wi-SUN モジュールを取り付けるためのスルーホールタップタイプのスペーサーです。

18. 筐体形状/寸法図



[Unit:mm]

図 18.1 筐体形状図

19. オプション品

本章では、Armadillo-IoT 関連のオプション品について説明します。

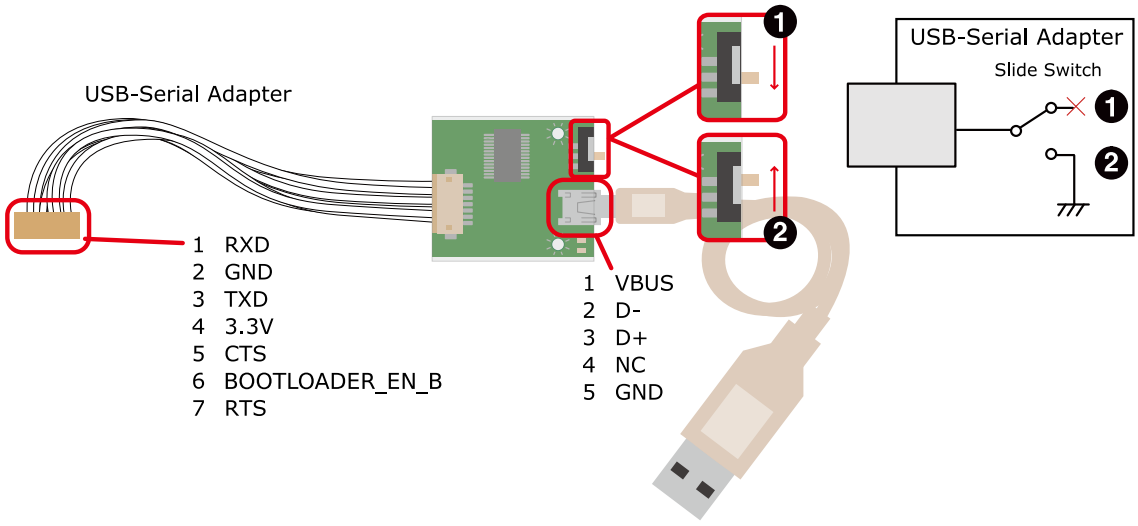
表 19.1 Armadillo-IoT 関連のオプション品

名称	型番
USB シリアル変換アダプタ	SA-SCUSB-00
Armadillo-IoT ゲートウェイ G3L LTE 用外付けアンテナセット 03	W1910
Armadillo-IoT ゲートウェイ G3L 無線 LAN 用基板アンテナ 06	CAF94505
AC アダプタ（12V/2.0A φ2.1mm ）温度拡張品	OP-AC12V3-00

USB シリアル変換アダプタは、試作・開発用の製品です。外観や仕様を予告なく変更する場合があります。

19.1. USB シリアル変換アダプタ

USB シリアル変換アダプタは、FT232RL を搭載した USB-シリアル変換アダプタです。シリアルの信号レベルは 3.3V CMOS です。デバッグシリアルインターフェース(メインユニット CON5)に接続して使用することが可能です。スライドスイッチが実装されており、信号線の接続先を切替することができます。



❶　OS 自動起動モード

❷　保守モード

図 19.1 USB シリアル変換アダプタの配線

19.2. Armadillo-IoT ゲートウェイ G3L LTE 用外付けアンテナセット 03

19.2.1. 概要

Armadillo-IoT ゲートウェイ G3L LTE 用外付けアンテナセット 03 は、LTE モジュール(ELS31-J/Gemalto)対応のアンテナセットです。

19.2.2. 組み立て

LTE 用アンテナは、LTE アンテナインターフェース 3(メインユニット CON15)または LTE アンテナインターフェース 4(メインユニット CON16)に取り付けます。

アンテナ端子に外付けアンテナを取り付ける際、無理な力を加えると破損の原因となりますので十分に注意してください。

19.2.3. 形状図

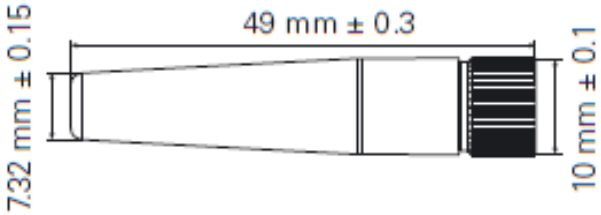


図 19.2 LTE 用外付けアンテナ形状

19.3. Armadillo-IoT ゲートウェイ G3L 無線 LAN 用基板アンテナ 06

19.3.1. 概要

Armadillo-IoT ゲートウェイ G3L 無線 LAN 用基板アンテナ 06 は、WLAN+BT コンボモジュール(WL1837MOD/TI)対応のアンテナセットです。

19.3.2. 組み立て

WLAN 用アンテナは、基板アンテナ本体に取り付けられている長さ約 100mm のアンテナケーブル(先端に IPEX MHF 端子付き)を使用して、WLAN アンテナインターフェース 1(サブユニット CON8)または WLAN アンテナインターフェース 2(サブユニット CON9)に取り付けます。

アンテナ端子に基板アンテナのケーブルを接続する際、無理な力を加えると破損の原因となりますので十分に注意してください。

19.3.3. 形状図

図 19.3 無線 LAN 用基板アンテナ形状

20. Howto

本章では、Armadillo-IoT のソフトウェアをカスタマイズをする方法などについて説明します。

20.1. イメージをカスタマイズする

コンフィギュレーションを変更して Linux カーネルイメージをカスタマイズする方法を説明します。

手順 20.1 イメージをカスタマイズ

1. **Linux カーネルアーカイブの展開**

   Linux カーネルのソースコードアーカイブと、initramfs アーカイブを準備し、Linux カーネルのソースコードアーカイブを展開します。

```
[PC ~]$ ls
initramfs_x1-[version].cpio.gz linux-3.14-x1-at[version].tar.gz
[PC ~]$ tar xf linux-3.14-x1-at[version].tar.gz
[PC ~]$ ls
initramfs_x1-[version].cpio.gz linux-3.14-x1-at[version]  linux-3.14-x1-
at[version].tar.gz
```

2. **initramfs アーカイブへのシンボリックリンク作成**

   Linux カーネルディレクトリに移動して、initramfs アーカイブへのシンボリックリンク作成します。

```
[PC ~]$ cd linux-3.14-x1-at[version]
[PC ~/linux-3.14-x1-at[version]]$ ln -s ../initramfs_x1-[version].cpio.gz
initramfs_x1.cpio.gz
```

3. **コンフィギュレーション**

   コンフィギュレーションをします。

```
[PC ~/linux-3.14-x1-at[version]]$ make ARCH=arm x1_defconfig
[PC ~/linux-3.14-x1-at[version]]$ make ARCH=arm menuconfig
```

4. **カーネルコンフィギュレーションの変更**

   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 ? <ESC><ESC> to continue.」で"Yes"とし、カーネルコンフィギュレーションを確定します。

```
.config - Linux/arm 3.14.79-at6 Kernel Configuration
------------------------------------------------------------------------------
             Linux/arm 3.14.79-at6 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  [ ]
------------------------------------------------------------------------------

    -*- Patch physical to virtual translations at runtime
        General setup  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
------------------------------------------------------------------------------
      <Select>    < Exit >    < Help >    < Save >    < Load >
```

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。

5. ビルド

ビルドするには、次のようにコマンドを実行します。

```
[PC ~/linux-3.14-x1-at[version]]$ make CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm
[PC ~/linux-3.14-x1-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x80008000 uImage
```

6. イメージファイルの生成確認

ビルドが終了すると、arch/arm/boot/ディレクトリと arch/arm/boot/dts/以下に、イメージファイル(Linux カーネルと DTB)が作成されています。

```
[PC ~/linux-3.14-x1-at[version]]$ ls arch/arm/boot/uImage
uImage
[PC ~/linux-3.14-x1-at[version]]$ ls arch/arm/boot/dts/armadillo_iotg_g3l.dtb
armadillo_iotg_g3l.dtb
```

20.2. ルートファイルシステムへの書き込みと電源断からの保護機能

Armadillo-IoT G3L のルートファイルシステムは、標準で eMMC に配置されます。 Linux が稼動している間は、ログや、設定ファイル、各種アプリケーション によるファイルへの書き込みが発生します。もし、停電等で終了処理を実行できずに 電源を遮断した場合は RAM 上に残ったキャッシュが eMMC に書き込まれずに、 ファイルシステムの破綻やファイルの内容が古いままになる状況が発生します。

また、eMMC 内部の NAND Flash Memory には消去回数に上限があるため、書き込み 回数を制限することを検討する必要がある場合もあります。

そこで、Armadillo-IoT G3L では、overlayfs を利用して、eMMC への書き込み保護を行う機能を提供しています。

20.2.1. 保護機能の使用方法

eMMC への書き込み保護を使うには、kernel の起動オプションに"overlay=50%" ("=50%" は省略可、"overlay"のみ書くと RAM の 50%を使用)というパラメタを追加するだけです。

パラメタを追加すると、debian の起動前に initramfs によってルートファイルシステムが upper=RAM ディスク(tmpfs)、 lower=eMMC(ext4)とした overlayfs に切り替えられて、Debian が起動します。

overlayfs の機能によって、起動後のルートファイルシステムに対する差分は、 全て RAM ディスク(/overlay/ramdisk にマウント) に記録されるようになりま す。そのため、起動後の情報は保存されませんが、電源を遮断した場合でも、 eMMC は起動前と変らない状態のまま維持されています。

kernel の起動オプションの指定を行うには Armadillo-IoT G3L を保守モードで起動し、次のようにコマンドを実行してください。

```
=> setenv optargs overlay
=> saveenv
```

20.2.2. 保護機能を使用する上での注意事項

overlayfs は差分を ファイル単位で管理するため、予想以上に RAM ディスクを消費する場合があります。 単に、新しいファイルやディレクトリを作れば、その分 RAM ディスクが消費されるのは想像に難くないと思います。

しかし、「lower=eMMC に既に存在していたファイルの書き換え」をする場合は、upper=RAM ディスク に対象のファイル全体をコピーして書き換え」ます。

具体的に、問題になりそうな例を紹介します。 例えば、sqlite は DB 毎に 1 つのファイルでデータ格納します。ここで、1GB の DB を作って eMMC に保存した後、 overlayfs による保護を有効にして起動した後に、たった 10 バイトのレコードを 追加しただけで RAM ディスクは 1GB + 10 バイト消費されます。実際には、 Armadillo に 1GB も RAM は無いので、追記を開始した時点で RAM ディスクが不足します。

overlayfs による、eMMC への書き込み保護を行う場合、 必ず実際の運用状態でのテストを行い、RAM ディスクが不足しないか確認してください。 動作中に書き込むファイルを必要最小限に留めると共に、追記を行う大きなファイルを作らない実装の検討を行ってください。

Armadillo-IoT G3L の eMMC の記録方式は出荷時に SLC に設定しており、MLC 方式の eMMC よりも消去回数の上限が高くなっています。 そのため、開発するシステムの構成によっては eMMC への書き込み保護機能を必要としない可能性があります。

eMMC への書き込み保護機能を有効にすると、eMMC を安全に使用できるというメリットがありますが、その分、使用できる RAM サイズが減る、システム構成が複雑になる、デメリットもあります。 開発・運用したいシステムの構成、eMMC への書き込み保護機能のメリット・デメリットを十分に考慮・評価したうえで、保護機能を使用する、しないの判断を行ってください。

20.3. RS422/RS485 シリアルポートで通信を行なう

RS422/RS485 シリアルポートを使って、他の機器と通信する手順について説明します。

手順 20.2 RS485 シリアルポートを用いた通信

1. **対向機器と接続**

   「表 17.5. メインユニット CON4 信号配列」を確認し、Armadillo-IoT G3L の CON4 と対向機器を接続します。

2. **RS485 の通信設定**

   保守モードで起動し、Linux カーネル起動オプションで RS485 設定を行います。 例として、RS485 設定を全二重通信にします。

```
=> setenv optargs imx.rs485_uart2=0x13,0,0
=> saveenv
```

3. **TTY デバイスの設定**

   Linux 起動後、接続した対向機器に合わせて TTY デバイスファイルを設定します。 例として、stty コマンドでボーレートを 115200bps に設定します。

```
[armadillo ~]# stty -F /dev/ttymxc1 115200
```

4. **データの送受信**

TTY デバイスファイルに任意のデータを書き込むことで、データを送信することができます。例として、"hoge"という文字列を送信します。

```
[armadillo ~]# echo ”hoge” > /dev/ttymxc1
```

対向機器からデータを受信するには、TTY デバイスファイルを cat コマンド等でオープンしてください。 例として、受信したデータを recvdata というファイルに出力します。

```
[armadillo ~]# cat /dev/ttymxc1 > recvdata
```

20.4. WL1837MOD モジュールを使って 2.4GHz 帯で通信する使用例

2 つの機器をサンプルに WL1837MOD モジュールを使って 2.4GHz 帯で通信するときの使い方について説明します。

20.4.1. 「BVMCN1101AA」の信号を受信する

Braveridge 社製のビーコン「BVMCN1101AA」を例にビーコン信号を受信する方法を説明します。

「BVMCN1101AA」のアドバタイジング・パケットを受信するためには、bluetoothctl コマンドを使います。 [bluetooth]のプロンプトが表示されたら、scan on で信号を受信できます。 ご利用の環境によっては、ほかの機器からの信号も受信されます。

```
[armadillo ~]# bluetoothctl
[NEW] Controller [AA:AA:AA:AA:AA:AA] armadillo-iotg [default]
[bluetooth]# scan on
Discovery started
[CHG] Controller [AA:AA:AA:AA:AA:AA] Discovering: yes
[NEW] Device [BB:BB:BB:BB:BB:BB] BBAAEdit
[CHG] Device [BB:BB:BB:BB:BB:BB] RSSI: -67
[CHG] Device [BB:BB:BB:BB:BB:BB] RSSI: -72
```

スキャンを中止するには、scan off を実行します。

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

bluetoothctl を終了するには、exit を実行します。

```
[bluetooth]# exit
```

20.4.2. 「CC2650」を操作する

TEXAS INSTRUMENTS 社製のセンサータグ「CC2650」を例にセンサータグを gatttool で操作する方法を説明します。

手順 20.3 「CC2650」の操作手順

1. hcitool lescan を実行して、「CC2650」の MAC アドレスを確認します。 ご利用の環境によっては、他の機器も検出されます。

```
[armadillo ~]# hcitool lescan
LE Scan ...
[CC:CC:CC:CC:CC:CC] (unknown)
[CC:CC:CC:CC:CC:CC] CC2650 SensorTag # <- この MAC アドレスを確認します。
```

2. 「CC2650」に接続します。

```
[armadillo ~]# gatttool -b [CC:CC:CC:CC:CC:CC] -I
[[CC:CC:CC:CC:CC:CC]][LE]> connect
Attempting to connect to [CC:CC:CC:CC:CC:CC]
Connection successful
```

3. プライマリサービスを確認するには、primary を実行します。

```
[[CC:CC:CC:CC:CC:CC]][LE]> primary
attr handle: 0x0001, end grp handle: 0x0007 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0008, end grp handle: 0x000b uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000c, end grp handle: 0x001e uuid: 0000180a-0000-1000-8000-00805f9b34fb
attr handle: 0x001f, end grp handle: 0x0026 uuid: f000aa00-0451-4000-b000-000000000000
attr handle: 0x0027, end grp handle: 0x002e uuid: f000aa20-0451-4000-b000-000000000000
attr handle: 0x002f, end grp handle: 0x0036 uuid: f000aa40-0451-4000-b000-000000000000
attr handle: 0x0037, end grp handle: 0x003e uuid: f000aa80-0451-4000-b000-000000000000
attr handle: 0x003f, end grp handle: 0x0046 uuid: f000aa70-0451-4000-b000-000000000000
attr handle: 0x0047, end grp handle: 0x004b uuid: 0000ffe0-0000-1000-8000-00805f9b34fb
attr handle: 0x004c, end grp handle: 0x0050 uuid: f000aa64-0451-4000-b000-000000000000 ⏎
# <- ここを詳しく調べます
attr handle: 0x0051, end grp handle: 0x0058 uuid: f000ac00-0451-4000-b000-000000000000
attr handle: 0x0059, end grp handle: 0x0060 uuid: f000ccc0-0451-4000-b000-000000000000
attr handle: 0x0061, end grp handle: 0xffff uuid: f000ffc0-0451-4000-b000-000000000000
```

4. 0x004c～0x0050 でハンドルされているプロファイルの UUID を確認するには、char-desc を実行します。

```
[[CC:CC:CC:CC:CC:CC]][LE]> char-desc 4c 50
handle: 0x004c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x004d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x004e, uuid: f000aa65-0451-4000-b000-000000000000
handle: 0x004f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0050, uuid: f000aa66-0451-4000-b000-000000000000
```

5. プロファイルの情報を読み取るには、char-read-hnd を実行します。

```
[[CC:CC:CC:CC:CC:CC]][LE]> char-read-hnd 50
Characteristic value/descriptor: 00
```

6. プロファイルの情報を設定するには、char-write-cmd を実行します。

```
[[CC:CC:CC:CC:CC:CC]][LE]> char-write-cmd 50 01 # <- 「CC2650」のブザーが鳴り、LED が光 ⏎
ります
[[CC:CC:CC:CC:CC:CC]][LE]> char-write-cmd 50 01 # <- 「CC2650」のブザーが止まり、LED が ⏎
消えます
```

7. 操作を終了するには、exit を実行します。

```
[[CC:CC:CC:CC:CC:CC]][LE]> exit
```

20.5. ssh で Armadillo-IoT G3L に接続する

1. ssh-server をインストールする

```
[armadillo ~]# apt-get update
[armadillo ~]# apt-get install -y ssh
```

2. ssh で root のログインを禁止する

   /etc/ssh/sshd_config 内の PermitRootLogin を no に設定します。

```
[armadillo ~]# vi /etc/ssh/sshd_config
...
# Authentication:
LoginGraceTime 120
PermitRootLogin without-password  # -> no に変更
StrictModes yes
...
```

21. 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 ユーザーズサイト」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 ユーザーズサイト」をご覧ください。

アットマークテクノ ユーザーズサイト

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

21.1. 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[1]」をダウンロードすることができるようになります。

Armadillo-IoT 購入製品登録

https://users.atmark-techno.com/armadillo-iot-g3l/register

Armadillo-IoT の購入製品登録を行うには、ユーザーズサイトで「正規認証ファイル」のアップロードを行う必要があります

Armadillo-IoT から正規認証ファイル(board-info.txt)を取り出す手順を「21.1.1. 正規認証ファイルを取り出す手順」に示します。

21.1.1. 正規認証ファイルを取り出す手順

Armadillo にログインし、コマンドを実行すると正規認証ファイルが生成されます。そのファイルをお使いの Web ブラウザを使ってダウンロードしてください。

1. ATDE で minicom を立ち上げて、Armadillo-IoT に root ユーザーでログインします。デバイスファイル名(/dev/ttyUSB0)は、ご使用の環境により ttyUSB1 や ttyS0、ttyS1 などになる場合があります。Armadillo に接続されているシリアルポートのデバイスファイルを指定してください。

[1]アドオンモジュールの回路図データなど

```
atmark@atde6:~$ LANG=C minicom --noinit --wrap --device /dev/ttyUSB0

armadillo-iotg login: root
Password:
[root@armadillo-iotg (ttymxc1) ~]#
```

2. "get-board-info"コマンドを wget で取得します。

```
[root@armadillo-iotg (ttymxc1) ~]#  wget http://download.atmark-techno.com/misc/product-registration/armadillo-iot-g3l/get-board-info
[root@armadillo-iotg (ttymxc1) ~]#  chmod +x get-board-info
```

3. "get-board-info"コマンドを実行して正規認証ファイル(board-info.txt)を作成します。

```
[root@armadillo-iotg (ttymxc1) ~]# ./get-board-info
[root@armadillo-iotg (ttymxc1) ~]# ls
board-info.txt
[root@armadillo-iotg (ttymxc1) ~]#
```

4. Armadillo 上で動いている WEB サーバーがアクセスできる場所に、正規認証ファイルを移動し、アクセス権限を変更します。

```
[root@armadillo-iotg (ttymxc1) ~]# mv board-info.txt /var/www/html/
```

5. minicom を終了させ、お使いの Web ブラウザから、Armadillo の URL にアクセスしてください。下記どちらかの指定方法でアクセス可能です。

   - http://armadillo-iotg.local/board-info.txt
   - http://[Armadillo の IP アドレス]/board-info.txt [2]

   取り出した正規認証ファイルを「Armadillo-IoT G3L 購入製品登録」ページの「正規認証ファイル」欄に指定し、アップロードしてください。

[2] Armadillo の IP アドレスが 192.0.2.10 の場合、http://192.0.2.10/board-info.txt となります。

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2016/11/24	・初版発行
1.0.1	2016/12/08	・ユーザースイッチ(SW2)での終了機能追加に伴い、「5. 起動と終了」、「7.6. ユーザースイッチ」、「14. i.MX7Dual の電源制御」を変更 ・「2.2. 取扱い上の注意事項」の「破損しやすい箇所」に「アンテナ端子」、「microSD スロット」、「microSIM スロット」を追加 ・「図 4.11. Armadillo-IoT メインユニット インターフェースレイアウト(A 面)」、「図 4.12. Armadillo-IoT メインユニット インターフェースレイアウト(B 面)」、「図 4.13. Armadillo-IoT サブユニット インターフェースレイアウト(A 面)」、「図 4.13. Armadillo-IoT サブユニット インターフェースレイアウト(A 面)」を見やすいものに変更 ・「図 17.1. Armadillo-IoT メインユニット インターフェースレイアウト(A 面)」、「図 17.2. Armadillo-IoT メインユニット インターフェースレイアウト(B 面)」、「図 17.3. Armadillo-IoT サブユニット インターフェースレイアウト(A 面)」、「図 17.4. Armadillo-IoT サブユニット インターフェースレイアウト(B 面)」を見やすいものに変更 ・LED でのインストール状況の進捗表示対応に伴い、「6.2.2. インストールの実行」を変更 ・VCCI Class B 登録完了に伴い、「2.5. 電波障害について」の記載を修正 ・誤記、わかりにくい記載を修正
1.0.2	2016/12/20	・誤記修正

Armadillo-IoT ゲートウェイ G3L 製品マニュアル
Version 1.0.2
2016/12/20