Armadillo-IoT ゲートウェイ A6
製品マニュアル

AG6110-U01D0
AG6110-U01D1
AG6110-U01D2
AG6110-C01D0
AG6110-C01D1
AG6110-C01D2
AG6110-C01Z
AG6110-C01Z

Version 1.2.1
2021/06/04
Debian GNU/Linux 10 (buster) 対応

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

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

Armadillo-IoT ゲートウェイ A6 製品マニュアル

株式会社アットマークテクノ

Version 1.2.1
2021/06/04

Armadillo-IoT ゲートウェイ A6 製品マニュアル

目次

図目次

Armadillo-IoT ゲートウェイ A6 製品マニュアル

表目次

1. はじめに

このたびは Armadillo-IoT ゲートウェイ A6 をご利用いただき、ありがとうございます。

Armadillo-IoT ゲートウェイシリーズ は、各種センサーとネットワークとの接続を中継する IoT 向けゲートウェイの開発プラットフォームです。ハードウェアやソフトウェアをカスタマイズして、オリジナルのゲートウェイを素早く、簡単に開発することができます。

Armadillo-IoT ゲートウェイ A6 は、Armadillo-IoT ゲートウェイシリーズの中でも、省電力や間欠動作機能に特化した小型な IoT ゲートウェイです。超低消費電力でクラウドと通信できるセルラー LPWA（LTE-M）モジュールを搭載。自立型のシステムを構築する際には、ソーラーパネルや蓄電池をより小さなものにでき、システム全体のコストを大幅に低減することができます。

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

Armadillo-IoT ゲートウェイ A6 では、Debian GNU/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、Oracle Java や Ruby なども利用することができるため、PC ライクな開発が可能です。

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

1.1. 本書で扱うこと扱わないこと

1.1.1. 扱うこと

本書では、Armadillo-IoT ゲートウェイ A6 の使い方、製品仕様(ソフトウェアおよびハードウェア)、オリジナルの製品を開発するために必要となる情報、その他注意事項について記載しています。Linux あるいは組み込み機器に不慣れな方でも読み進められるよう、コマンドの実行例なども記載しています。

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

1.1.2. 扱わないこと

本書では、一般的な Linux のプログラミング、デバッグ方法やツールの扱い方、各種モジュールの詳細仕様など、一般的な情報や、他に詳しい情報があるものは扱いません。また、(Armadillo-IoT ゲートウェイ A6 を使用した)最終製品あるいはサービスに固有な情報や知識も含まれていません。

1.2. 本書で必要となる知識と想定する読者

本書は、読者として Armadillo-IoT ゲートウェイ A6 を使ってオリジナルの機器を開発するエンジニアを想定して書かれています。また、「Armadillo-IoT ゲートウェイ A6 を使うと、どのようなことが実現可能なのか」を知りたいと考えている設計者・企画者も対象としています。Armadillo-IoT ゲートウェイ A6 は組込みプラットフォームとして実績のある Armadillo をベースとしているため、標準で有効になっている機能以外にも様々な機能を実現することができます。

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

1.3. ユーザー限定コンテンツ

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

- 各種信頼性試験データ・納入仕様書等製造関連情報

限定コンテンツを取得するには、「23. 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 A6 ドキュメントダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6/resources/documents

Armadillo サイト - Armadillo-IoT ゲートウェイ A6 ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6/resources/software

1.5. 本書の構成

本書には、Armadillo-IoT ゲートウェイ A6 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 A6 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」、「6. 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 (U-Boot) 仕様」

- **システム開発に必要な情報を紹介します。**

  「11. ビルド手順」、「12. イメージファイルの書き換え方法」、「13. 開発の基本的な流れ」、「14. i.MX6ULL の電源制御方法」、「15. USB ブートの活用」、「16. SD ブートの活用」

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

  「17. 電気的仕様」、「18. インターフェース仕様」、「19. 寸法図」、「20. オプション品」、「21. 設計情報」

- **ソフトウェアのカスタマイズ方法を紹介します。**

  「22. 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。**

  「23. 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「 ~ 」で表します。

表 1.2 表示プロンプトと実行環境の関係

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

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

表 1.3 コマンド入力例での省略表記

表記	説明
[version]	ファイルのバージョン番号

1.6.3. アイコン

本書では以下のようにアイコンを使用しています。

注意事項を記載します。

役に立つ情報を記載します。

用語の説明や補足的な説明を記載します。

1.7. 謝辞

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

2. 注意事項

2.1. 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。

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

内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

2.2. 取扱い上の注意事項

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

破損しやすい箇所	microSD コネクタおよびそのカバーやフラットケーブルコネクタは、破損しやすい部品になっています。無理に力を加えて破損することのないよう十分注意してください。
本製品の改造	本製品に改造[1]を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設[2]を行う場合は、作業前に必ず動作確認を行ってください。
電源投入時のコネクタ着脱	本製品や周辺回路に電源が入っている状態で、活線挿抜対応インターフェース(LAN, USB) [3]以外へのコネクタ着脱は、絶対に行わないでください。
静電気	本製品には CMOS デバイスを使用しており、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。
ラッチアップ	電源および入出力からの過大なノイズやサージ、電源電圧の急激な変動等により、使用している CMOS デバイスがラッチアップを起こす可能性があります。いったんラッチアップ状態となると、電源を切断しないかぎりこの状態が維持されるため、デバイスの破損につながることがあります。ノイズの影響を受けやすい入出力ラインには、保護回路を入れることや、ノイズ源となる装置と共通の電源を使用しない等の対策をとることをお勧めします。
衝撃	落下や衝撃などの強い振動を与えないでください。
使用場所の制限	無線機能を搭載した製品は、テレビ・ラジオに近接する場所で使用すると、受信障害を招く恐れがあります。
振動	振動が発生する環境では、Armadillo が動かないよう固定して使用してください。
電波に関する注意事項(LTE)	この無線機(EMS31-J)は LTE 通信を行います。LTE 通信機能は、 心臓ペースメーカーや除細動器等の植込み型医療機器の近く(15cm 程度以内)で使用しないでください。
電気通信事業法に関する注意事項について	本製品の有線 LAN を、 電気通信事業者の通信回線(インターネットサービスプロバイダーが提供している通信網サービス等)に直接接続す

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

ることはできません。 接続する場合は、 必ず電気通信事業法の認定を受けた端末設備(ルーター等)を経由して接続してください。

2.3. ソフトウェア使用に関しての注意事項

本製品に含まれるソフトウェアについて

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

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

以下のソフトウェアは、オープンソースソフトウェアではありません。

ボード情報取得ツール(get-board-info)

2.4. 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

2.5. 無線モジュールの安全規制について

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

これらの無線モジュールを国内で使用するときに無線局の免許は必要ありません。

以下の事項を行うと法律により罰せられることがあります。

・ 無線モジュールやアンテナを分解/改造すること。

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

認証番号は次のとおりです。

表 2.1 LTE モジュール:EMS31-J 適合証明情報

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

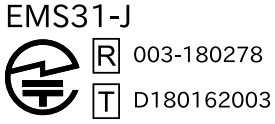


図 2.1 LTE モジュール:EMS31-J 認証マーク

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 https://armadillo.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。

3. 製品概要

3.1. 製品の特長

3.1.1. Armadillo とは

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

- ARM プロセッサ搭載・省電力設計

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

- 小型・手のひらサイズ

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

- 標準 OS として Linux をプリインストール

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

- 開発環境

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

3.1.2. Armadillo-IoT ゲートウェイ A6 とは

Armadillo-IoT ゲートウェイ A6 は、組み込みプラットフォームとして実績のある Armadillo-640 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。

超低消費電力でクラウドと通信できるセルラー LPWA（LTE-M）モジュールを搭載、従来モデル以上に省電力で動作し、自立型のシステムを構築する際には、ソーラーパネルや蓄電池をより小さなものにでき、システム全体のコストを大幅に低減することができます。

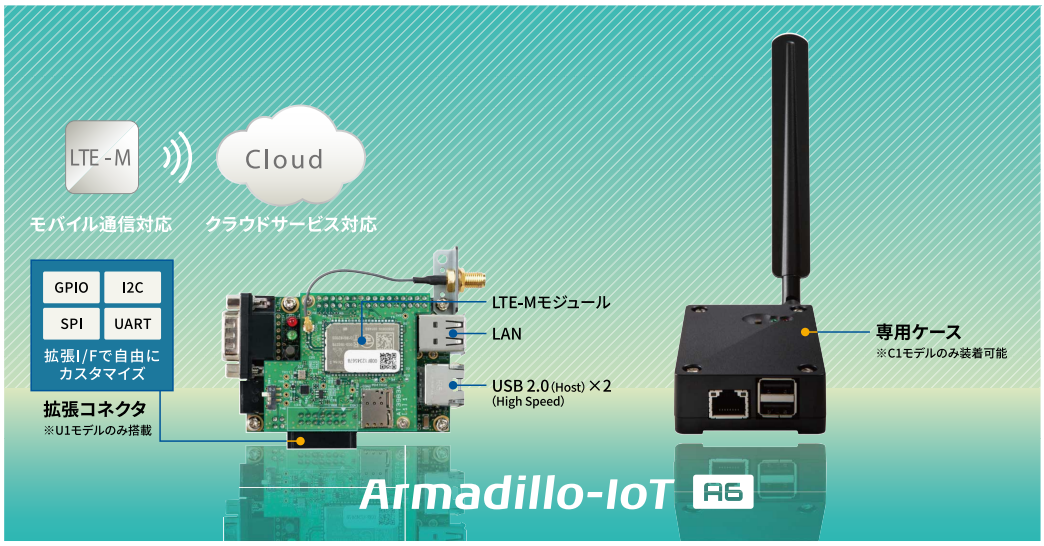


図 3.1 Armadillo-IoT ゲートウェイ A6 とは

- 省電力モード搭載・バッテリー駆動の機器に最適

  省電力モードを搭載し、「アプリケーションから Armadillo-IoT ゲートウェイ A6 本体の電源を OFF にする」「RTC(リアルタイムクロック)のアラームで決まった時間に本体の電源を ON にする」「省電力モードで動作させ、SMS の受信で復帰する」といった細かな電源制御、間欠動作が可能です。必要な時だけ本体を起動するといった間欠動作運用が可能なので、バッテリーで稼動させるような機器に適しています。

- ボードモデル・ケースモデルをラインアップ

  拡張を行いたい・防水筐体を使用したい場合などはボードモデル（U1 モデル）、そのまますぐに設置したい場合はケースモデル（C1 モデル）と、用途や設置環境に合わせてモデルを選択することができます。ボードモデルには各種 I/F を拡張できるコネクタを搭載しているので、自由度の高いハードウェア設計が可能です。

- Debian GNU/Linux に標準対応

  Armadillo-IoT ゲートウェイ A6 は、標準ルートファイルシステムに Debian GNU/Linux を採用し、PC ライクな開発が可能です。カーネルやデバイスドライバなどの基本アプリケーションは Web サイトで無償公開されているので、Linux の豊富な開発資産も利用できます。

3.2. 製品ラインアップ

Armadillo-IoT ゲートウェイ A6 の製品ラインアップは次のとおりです。

表 3.1 Armadillo-IoT ゲートウェイ A6 ラインアップ

名称	型番
Armadillo-IoT ゲートウェイ A6 C1 モデル 開発セット	AG6110-C01D0
Armadillo-IoT ゲートウェイ A6 C1 モデル 開発セット(おかわりキャンペーン 1 杯目)	AG6110-C01D1
Armadillo-IoT ゲートウェイ A6 C1 モデル 開発セット(おかわりキャンペーン 2 杯目)	AG6110-C01D2
Armadillo-IoT ゲートウェイ A6 C1 モデル 量産用（LTE アンテナ付属）	AG6110-C01Z
Armadillo-IoT ゲートウェイ A6 U1 モデル 開発セット	AG6110-U01D0
Armadillo-IoT ゲートウェイ A6 U1 モデル 開発セット(おかわりキャンペーン 1 杯目)	AG6110-U01D1
Armadillo-IoT ゲートウェイ A6 U1 モデル 開発セット(おかわりキャンペーン 2 杯目)	AG6110-U01D2

名称	型番
Armadillo-IoT ゲートウェイ A6 U1 モデル 量産用（ケース無、LTE アンテナ付属）	AG6110-U01Z

3.2.1. Armadillo-IoT ゲートウェイ A6 C1 モデル 開発セット

Armadillo-IoT ゲートウェイ A6 C1 モデル 開発セット(型番: AG6110-C01D0, AG6110-C01D1, AG6110-C01D2)は、Armadillo-IoT ゲートウェイ A6 C1 モデル を使った開発がすぐに開始できるように、開発に必要なものを一式含んだセットです。

- Armadillo-IoT ゲートウェイ A6 C1 モデル 本体（拡張コネクタ非実装）
- Armadillo-600 シリーズオプションケース(樹脂製)
- ケース用ネジ
- LTE 用外付けアンテナ
- シリアルクロスケーブル
- D-Sub9/10 ピンシリアル変換ケーブル
- USB-RS232C 変換ケーブル
- AC アダプタ(5V/2.0A, EIAJ#2 準拠)
- ゴム足

3.2.2. Armadillo-IoT ゲートウェイ A6 U1 モデル 開発セット

Armadillo-IoT ゲートウェイ A6 U1 モデル 開発セット(型番: AG6110-U01D0, AG6110-U01D1, AG6110-U01D2)は、Armadillo-IoT ゲートウェイ A6 U1 モデル を使った開発がすぐに開始できるように、開発に必要なものを一式含んだセットです。

- Armadillo-IoT ゲートウェイ A6 U1 モデル 本体（拡張コネクタ実装済、スペーサ組立済）
- LTE 用外付けアンテナ
- シリアルクロスケーブル
- USB-RS232C 変換ケーブル
- AC アダプタ(5V/2.0A, EIAJ#2 準拠)

3.2.3. Armadillo-IoT ゲートウェイ A6 量産用

Armadillo-IoT ゲートウェイ A6 量産用は、Armadillo-IoT ゲートウェイ A6 開発セットのセット内容を必要最小限に絞った量産向けのラインアップです。C1 モデル(AG6110-C01Z)、U1 モデル(AG6110-U01Z)の 2 種類あります。

3.3. 仕様

Armadillo-IoT ゲートウェイ A6 の主な仕様は次のとおりです。

表 3.2 仕様

型番	AG6110-U01D0 AG6110-U01D1 AG6110-U01D2 AG6110-U01Z	AG6110-C01D0 AG6110-C01D1 AG6110-C01D2 AG6110-C01Z
プロセッサ	NXP セミコンダクターズ製 i.MX6ULL ARM Cortex-A7 x 1 ・命令/データキャッシュ 32KByte/32KByte ・L2 キャッシュ 128KByte ・内部 SRAM 128KByte ・メディアプロセッシングエンジン(NEON)搭載 ・Thumb code(16bit 命令セット)サポート	
システムクロック	CPU コアクロック(ARM Cortex-A7): 528MHz DDR クロック: 396MHz 源発振クロック: 32.768kHz, 24MHz	
RAM	DDR3L: 512MByte バス幅: 16bit	
ROM	eMMC: 約 3.8GB(約 3.6GiB) [a]	
LAN(Ethernet)	100BASE-TX/10BASE-T x 1 AUTO-MDIX 対応	
モバイル通信	LTE CAT-M1（タレス DIS 製 EMS31-J 搭載）[b] [c] SIM スロット: nanoSIM 対応	
USB	USB 2.0（Host）×2（High Speed）	
SD/MMC	microSD スロット×1 [d]	
GPIO	最大 6bit 拡張可能 [e]	コネクタ非搭載
I2C	最大 1 ポート拡張可能 [e] [f]	コネクタ非搭載
SPI	最大 1 ポート拡張可能 [e]	コネクタ非搭載
カレンダ時計	RTC 搭載（外部バックアップ用電源入力対応）	
シリアル(UART)	RS232C（D-Sub 9 ピンオス）x1 3.3V CMOS レベル 1 ポート拡張可能 [e]	RS232C（ピンヘッダ 10 ピン 2.54mm ピッチ）x1
スイッチ	ユーザースイッチ×1	
LED	ユーザー LEDx2	
入力電源	DC5V±5%	
消費電力（参考値）	約 1mW 以下（シャットダウン時）, 約 100mW（スリープ時）, 約 230mW（スリープ時［SMS 起床可能］）, 約 850mW（アクティブ時）, 1950mW（最大消費電力）	
動作温度範囲	-20～＋ 70℃ [g]	-10～＋ 40℃ [g]
外形サイズ	75.0×50.0mm [h] [i]	83.0×58.0×24.3mm [h]

[a]SLC での数値です。出荷時 SLC に設定しています。
[b]外付けアンテナを接続する必要があります。
[c]認証取得済みキャリア: docomo/Softbank/KDDI
[d]microSD カードを脱着する際、基板の取り外し作業が必要となります。
[e]i.MX6ULL のピンマルチプクレスの設定で、優先的に機能を割り当てた場合に拡張可能な最大数を記載しています。
[f]本製品内部で一部アドレス(0x32/0x48/0x50)が使用済みです。
[g]結露無きこと。
[h]突起部、アンテナを除く。
[i]基板の外形サイズです。

3.4. Armadillo-IoT ゲートウェイ A6 C1 モデルの外観

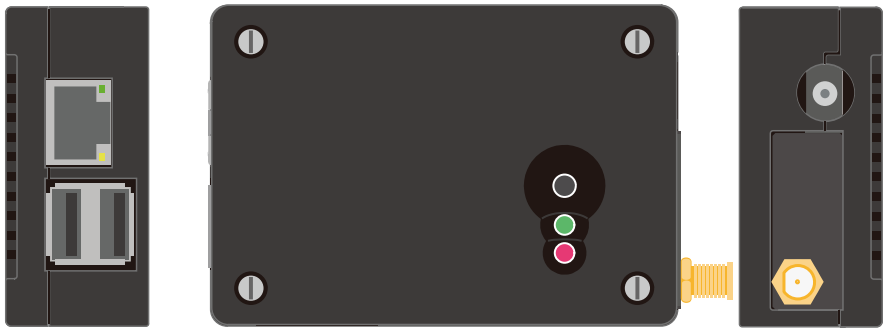


図 3.2 Armadillo-IoT ゲートウェイ A6 C1 モデルの外観

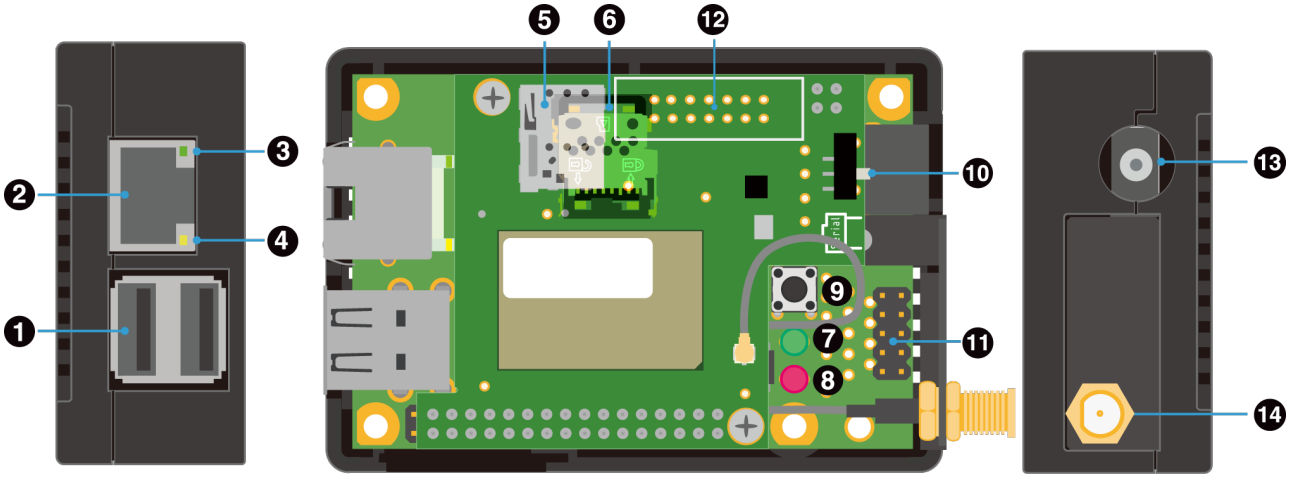


図 3.3 Armadillo-IoT ゲートウェイ A6 C1 モデルの各部名称

表 3.3 C1 モデル各部名称と機能

番号	名称	説明
1	USB コネクタ	USB メモリ等を接続します。
2	LAN コネクタ	LAN ケーブルを接続します。
3	LAN LED1	LAN の通信状況を表す緑色 LED です。
4	LAN LED2	LAN の通信状況を表す黄色 LED です。
5	nanoSIM スロット	nanoSIM カードを接続します。
6	microSD スロット	microSD カードを接続します。
7	ユーザー LED4	ユーザーで自由に機能を設定できる緑色 LED です。
8	ユーザー LED3	ユーザーで自由に機能を設定できる赤色 LED です。
9	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。
10	起動デバイス設定スイッチ	起動デバイスを設定するスイッチです。
11	メンテナンスポート	付属の D-Sub9/10 ピンシリアル変換ケーブルを接続します。
12	拡張インターフェース	機能拡張用のインターフェースです。
13	電源コネクタ	付属の AC アダプタを接続します。
14	アンテナコネクタ	付属の LTE 用外付けアンテナを取り付けます。

3.5. Armadillo-IoT ゲートウェイ A6 U1 モデルの外観

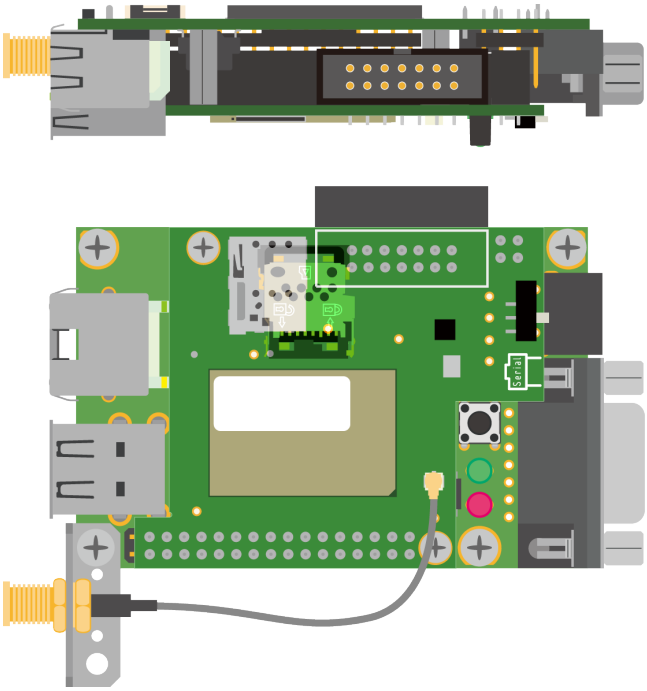


図 3.4 Armadillo-IoT ゲートウェイ A6 U1 モデルの外観

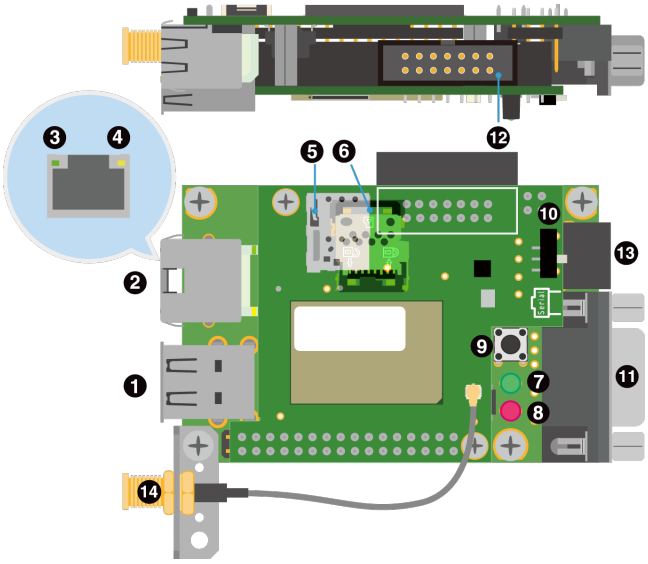


図 3.5 Armadillo-IoT ゲートウェイ A6 U1 モデルの各部名称

表 3.4 U1 モデル各部名称と機能

番号	名称	説明
1	USB コネクタ	USB メモリ等を接続します。
2	LAN コネクタ	LAN ケーブルを接続します。
3	LAN LED1	LAN の通信状況を表す緑色 LED です。
4	LAN LED2	LAN の通信状況を表す黄色 LED です。
5	nanoSIM スロット	nanoSIM カードを接続します。

番号	名称	説明
6	microSD スロット	microSD カードを接続します。
7	ユーザー LED4	ユーザーで自由に機能を設定できる緑色 LED です。
8	ユーザー LED3	ユーザーで自由に機能を設定できる赤色 LED です。
9	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。
10	起動デバイス設定スイッチ	起動デバイスを設定するスイッチです。
11	メンテナンスポート	付属のシリアルクロスケーブルを接続します。
12	拡張コネクタ	機能拡張用のコネクタです。
13	電源コネクタ	付属の AC アダプタを接続します。
14	アンテナコネクタ	付属の LTE 用外付けアンテナを取り付けます。

3.6. Armadillo-IoT ゲートウェイ A6 の基板構成

Armadillo-IoT ゲートウェイ A6 は、2 つの基板で構成されています。

- メインユニット
- サブユニット

各ユニットの外観については、「18.1. インターフェースレイアウト」を参照してください。

3.7. ブロック図

Armadillo-IoT ゲートウェイ A6 のブロック図は次のとおりです。

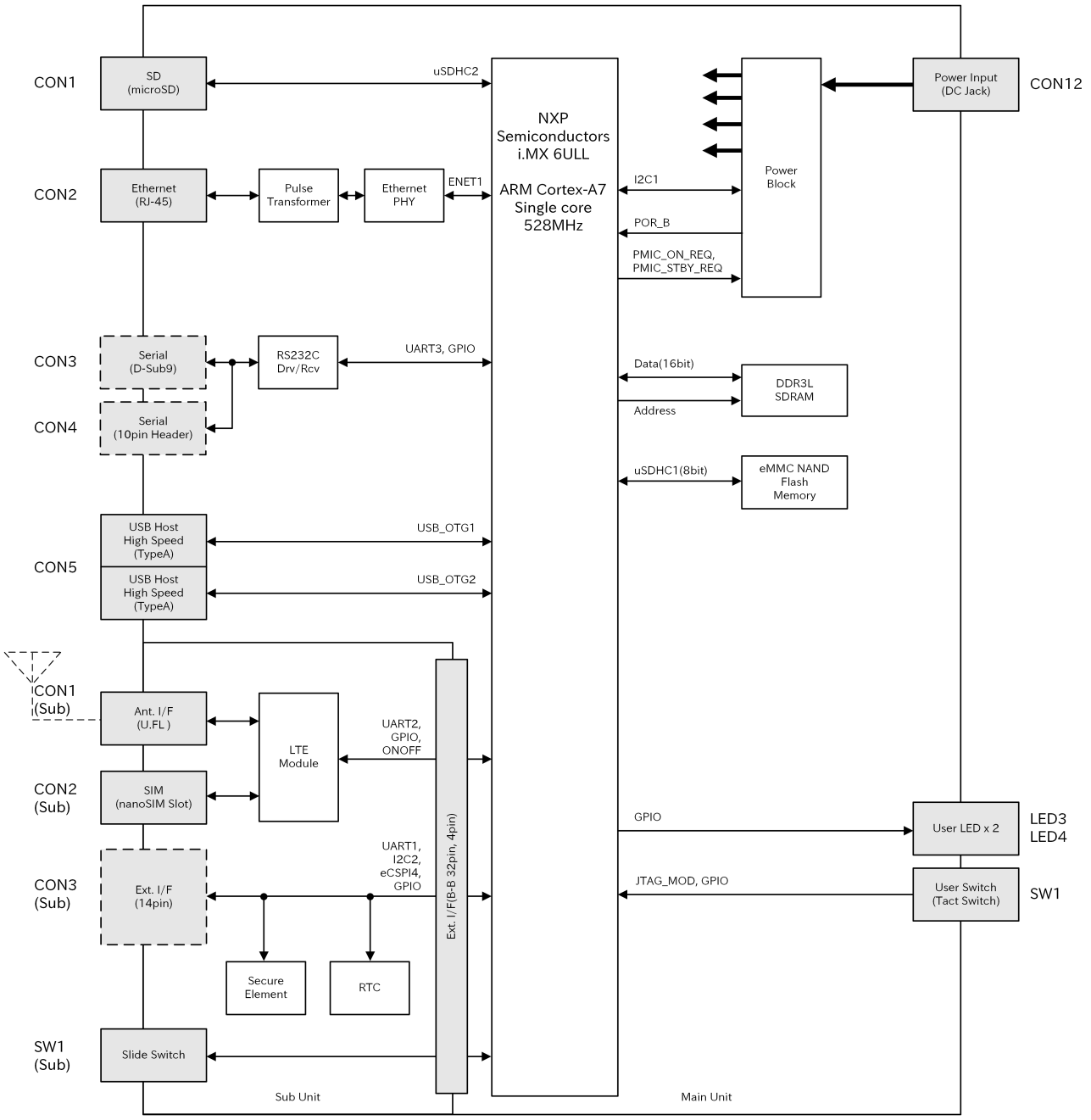


図 3.6 Armadillo-IoT ゲートウェイ A6 ブロック図

3.8. ソフトウェア構成

Armadillo-IoT ゲートウェイ A6 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 ゲートウェイ A6 で利用可能なソフトウェアを「表 3.5. Armadillo-IoT ゲートウェイ A6 で利用可能なソフトウェア」に示します。

表 3.5 Armadillo-IoT ゲートウェイ A6 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は eMMC に配置されています。microSD カードに配置することもできます。ブートローダーが使う環境変数は常に eMMC に保存されます。
Linux カーネル	uImage 形式の Linux カーネルイメージが利用可能です。 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。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 ゲートウェイ A6 の eMMC のメモリマップを「表 3.6. eMMC メモリマップ」に示します。

表 3.6 eMMC メモリマップ

ディスクデバイス	サイズ	説明
/dev/mmcblk0p1	30.6MByte	予約領域
/dev/mmcblk0p2	2.4GByte	Linux カーネルイメージ, Device Tree Blob, Debian GNU/Linux
/dev/mmcblk0p3	122.1MByte	予約領域
/dev/mmcblk0p4	1.0GByte	ユーザー開放領域

Armadillo-IoT ゲートウェイ A6 の eMMC(GPP)のメモリマップを「表 3.7. eMMC(GPP)メモリマップ」に示します。

表 3.7 eMMC(GPP)メモリマップ

ディスクデバイス	サイズ	説明
/dev/mmcblk0gp0	8.389 MByte	ライセンス情報等の保存
/dev/mmcblk0gp1	8.389 MByte	予約領域
/dev/mmcblk0gp2	8.389 MByte	ユーザー領域
/dev/mmcblk0gp3	8.389 MByte	ユーザー領域

4. Armadillo の電源を入れる前に

4.1. 準備するもの

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

作業用 PC	Linux または Windows が動作し、ネットワークインターフェースと 1 つ以上の USB ポートを持つ PC です。「4.5. 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	microSD 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。
nanoSIM(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 nanoSIM(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 で展開できます。Windows では、7-Zip や Lhaz などが対応しています。
プラスドライバー	microSD カードを装着する場合に必要です。推奨するドライバーの先端サイズは No.1 です。
マイナスドライバー	ケースを組み立てる際に必要です。推奨するドライバーの先端サイズは 4.0mm x 0.5mm です。

4.2. microSD カードの装着

microSD カードの装着手順は、以下の通りです。

1. サブユニットを固定しているネジを取り外す
2. サブユニットを取り外す
3. microSD スロットに microSD カードを装着する

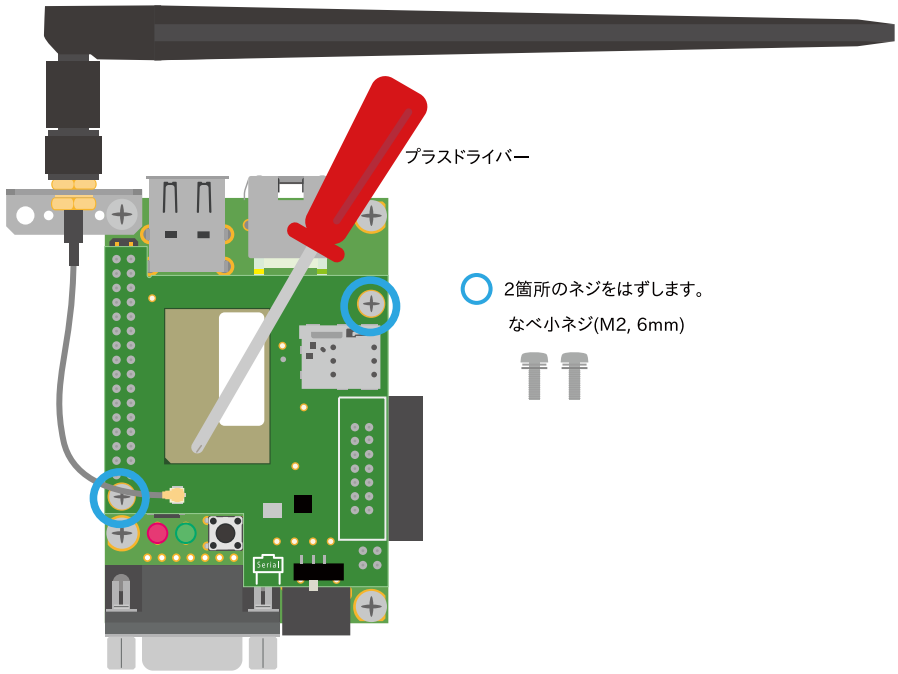


図 4.1 サブユニット固定ネジの取り外し

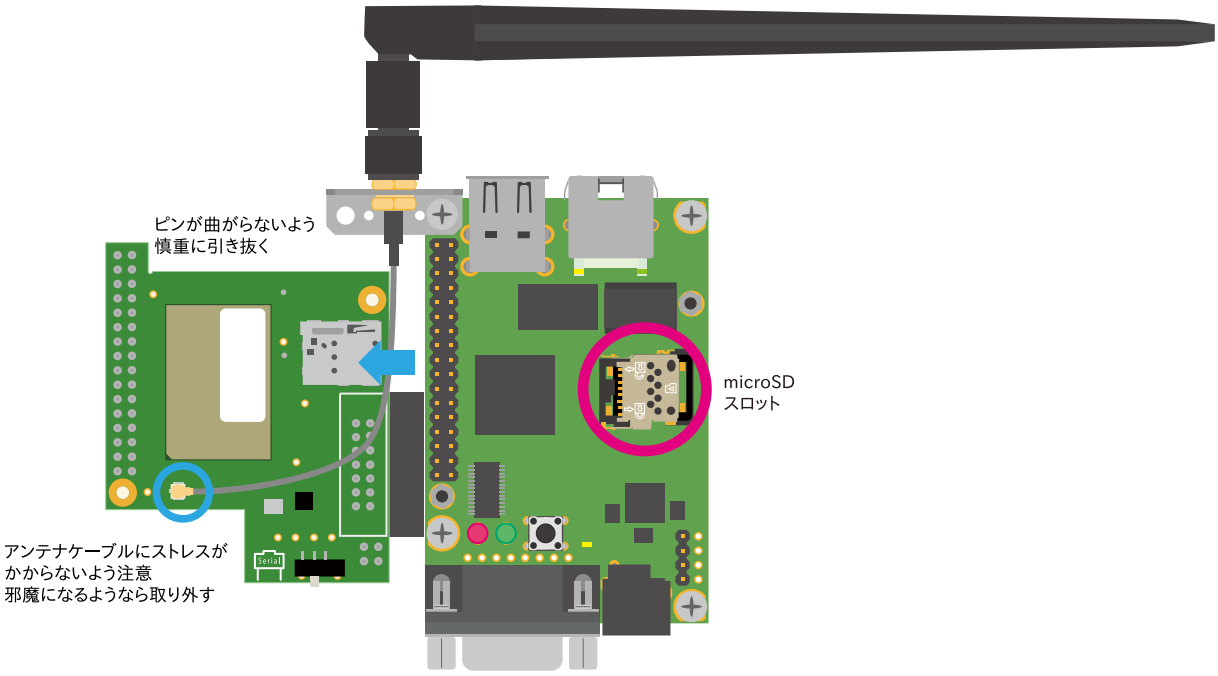


図 4.2 サブユニットの取り外し

サブユニットの外観については、「4.6. インターフェースレイアウト」を参照してください。

4.3. C1 モデル ケースの組み立て

nanoSIM(UIM カード)の取り付けは、ケース組立前に行う必要があります。取り付け手順については、「6.2.4.1. LTE データ通信設定を行う前に」を参照してください。

基板をケースに収め、付属のネジで固定してください。

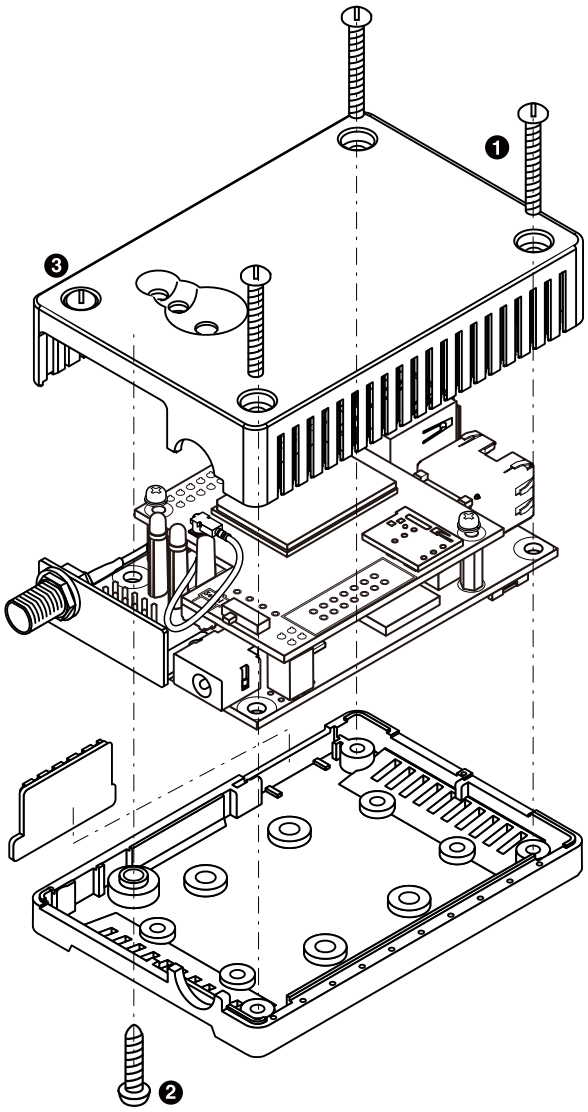


図 4.3 ケースの組み立て

表 4.1 ケースの組み立て

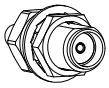
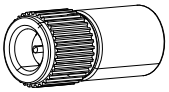
番号	名称	説明
1	タッピングネジ(M2.6、L=21mm)×3	ケーストップとケースボトムを固定するためのねじです。
2	タッピングネジ(M3、L=12mm)×1	ケーストップとケースボトムを固定するためのねじです。

番号	名称	説明
3	飾りネジ	ボンド止めされているので、無理に取り外さないでください。

4.4. アンテナコネクタ形状

アンテナコネクタ形状は、以下の通りです。

表 4.2 アンテナコネクタ形状

	Armadillo のアンテナコネクタ	外付けアンテナコネクタ
タイプ	SMA(FeMale)	SMA(Male)
形状		

4.5. 開発/動作確認環境の構築

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

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

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

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

ATDE は、バージョンにより対応するアットマークテクノ製品が異なります。本製品に対応している ATDE は、ATDE8 の v20201224 以降です。

ATDE8 は Debian GNU/Linux 10 (コードネーム buster) をベースに、Armadillo-IoT ゲートウェイ A6 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 A6 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。

4.5.1. ATDE のセットアップ

4.5.1.1. VMware のインストール

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

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

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

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

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

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

4.5.1.2. ATDE のアーカイブを取得

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

本製品に対応している ATDE のバージョンは ATDE8 v20201224 以降です。

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

4.5.1.3. ATDE のアーカイブを展開

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

Windows での展開方法を「4.5.1.4. Windows で ATDE のアーカイブ展開する」に、Linux での展開方法を手順「4.5.1.5. Linux で tar.xz 形式のファイルを展開する」に示します。

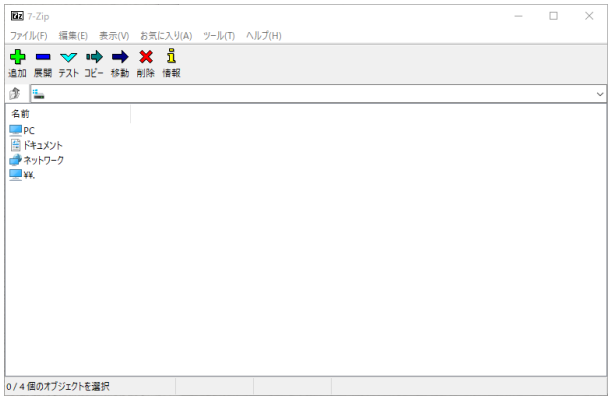
4.5.1.4. Windows で ATDE のアーカイブ展開する

1. 7-Zip のインストール

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

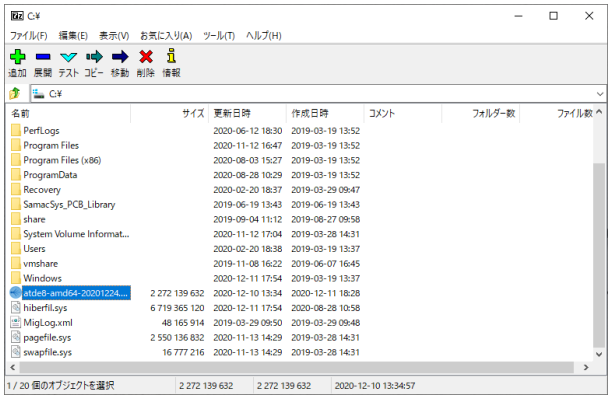
2. 7-Zip の起動

   7-Zip を起動します。



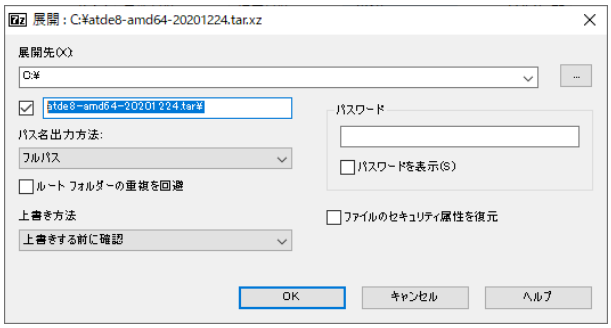
3. xz 圧縮ファイルの選択

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



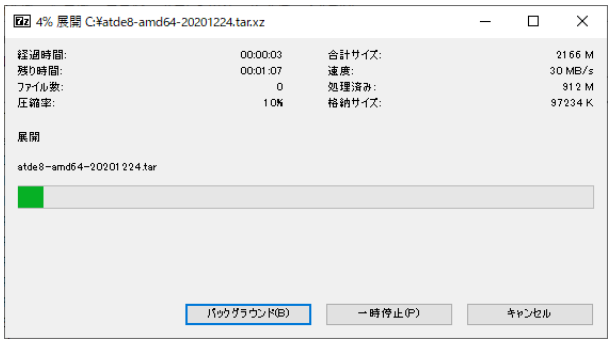
4. xz 圧縮ファイルの展開先の指定

   「展開先」を指定して、「OK」をクリックします。



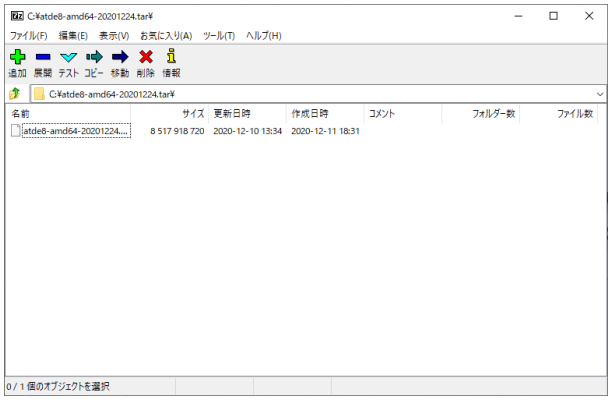
5. xz 圧縮ファイルの展開

   展開が始まります。



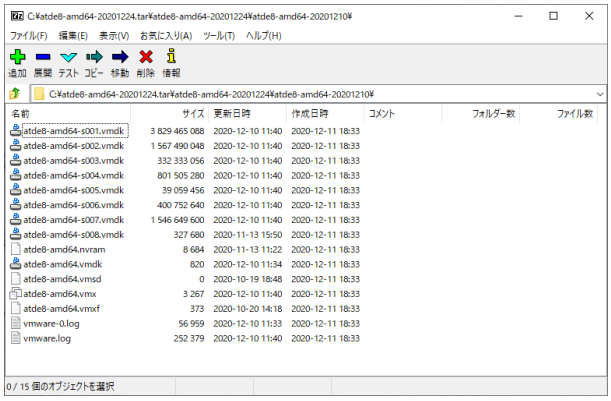
6. tar アーカイブファイルの選択

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



7. 展開の完了確認

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



4.5.1.5. Linux で tar.xz 形式のファイルを展開する

1. tar.xz 圧縮ファイルの展開

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

```
[PC ~]$ tar xf atde8-amd64-[version].tar.xz
```

2. 展開の完了確認

   tar.xz 圧縮ファイルの展開が終了すると、ATDE アーカイブの展開は完了です。 **atde8-amd64-[version]** ディレクトリに ATDE のデータイメージが出力されています。

```
[PC ~]$ ls atde8-amd64-[version]/
atde8-amd64-s001.vmdk  atde8-amd64.nvram
atde8-amd64-s002.vmdk  atde8-amd64.vmdk
atde8-amd64-s003.vmdk  atde8-amd64.vmsd
atde8-amd64-s004.vmdk  atde8-amd64.vmx
atde8-amd64-s005.vmdk  atde8-amd64.vmxf
atde8-amd64-s006.vmdk
atde8-amd64-s007.vmdk
atde8-amd64-s008.vmdk
```

4.5.1.6. ATDE の起動

ATDE のアーカイブを展開したディレクトリに存在する仮想マシン構成(.vmx)ファイルを VMware 上で開くと、ATDE を起動することができます。ATDE8 にログイン可能なユーザーを、「表 4.3. ユーザー名とパスワード」に示します [2]。

表 4.3 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。

4.5.2. 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。

[2]特権ユーザーで GUI ログインを行うことはできません

Armadillo-IoT ゲートウェイ A6 の動作確認を行うためには、「表 4.4. 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 4.4 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

4.5.3. コマンドライン端末(GNOME 端末)の起動

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

GNOME 端末を起動するには、「図 4.4. GNOME 端末の起動」のようにデスクトップ左上のアクティビティから「terminal」と入力し「端末」を選択してください。

図 4.4 GNOME 端末の起動

「図 4.5. GNOME 端末のウィンドウ」のようにウィンドウが開きます。

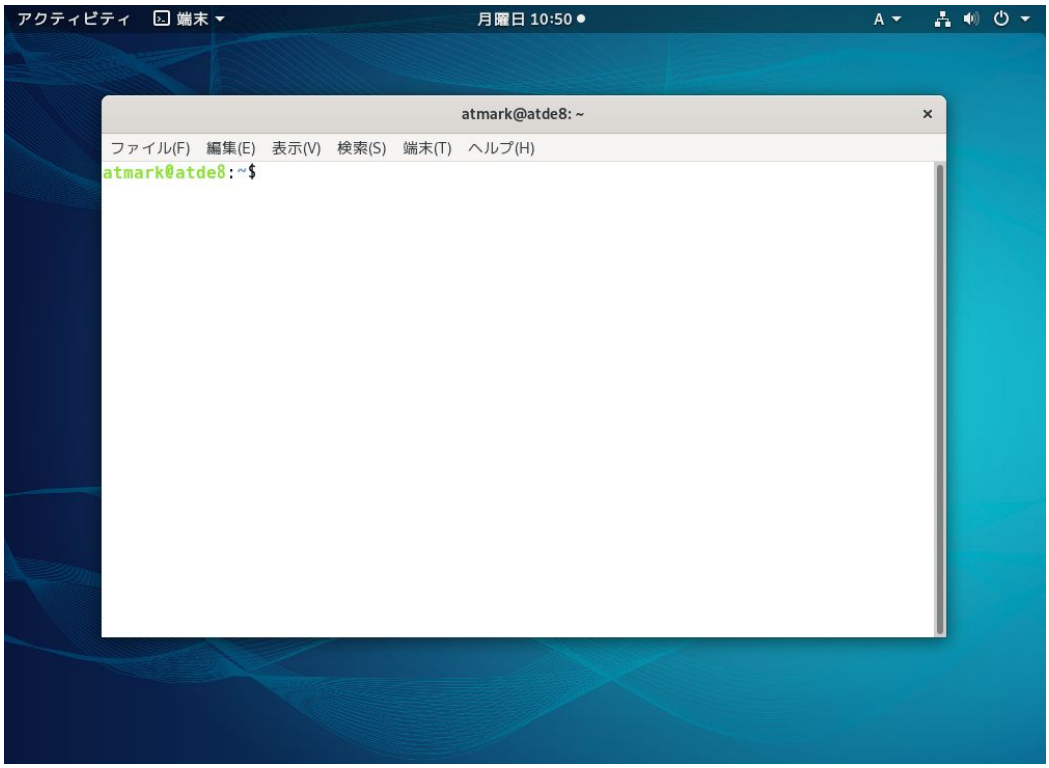


図 4.5 GNOME 端末のウィンドウ

4.5.4. 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 4.5. 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 4.5 シリアル通信設定

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

1. 「図 4.6. minicom の設定の起動」に示すコマンドを実行し、minicom の設定画面を起動してください。

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

図 4.6 minicom の設定の起動

2. 「図 4.7. minicom の設定」が表示されますので、「Serial port setup」を選択してください。

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

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

図 4.7 minicom の設定

3. 「図 4.8. minicom のシリアルポートの設定」が表示されますので、A キーを押して Serial Device を選択してください。

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

図 4.8 minicom のシリアルポートの設定

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

USB to シリアル変換ケーブル使用時のデバイスファイル確認方法

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

```
usb 2-1.2: new full-speed USB device number 5 using ehci-pci
usb 2-1.2: New USB device found, idVendor=0403, idProduct=6001
usb 2-1.2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
usb 2-1.2: Product: FT232R USB UART
usb 2-1.2: Manufacturer: FTDI
usb 2-1.2: SerialNumber: A702ZLZ7
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
ftdi_sio 2-1.2:1.0: FTDI USB Serial Device converter detected
```

```
usb 2-1.2: Detected FT232RL
usb 2-1.2: Number of endpoints 2
usb 2-1.2: Endpoint 1 MaxPacketSize 64
usb 2-1.2: Endpoint 2 MaxPacketSize 64
usb 2-1.2: Setting MaxPacketSize 64
usb 2-1.2: FTDI USB Serial Device converter now attached to
ttyUSB0
```

図 4.9 例. USB to シリアル変換ケーブル接続時のログ

上記のログから USB to シリアル変換ケーブルが ttyUSB0 に割り当てられたことが分かります。

5. F キーを押して Hardware Flow Control を No に設定してください。
6. G キーを押して Software Flow Control を No に設定してください。
7. キーボードの E キーを押してください。「図 4.10. minicom のシリアルポートのパラメータの設定」 が表示されます。

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

図 4.10 minicom のシリアルポートのパラメータの設定

8. 「図 4.10. minicom のシリアルポートのパラメータの設定」では、転送レート、データ長、ストップビット、パリティの設定を行います。
9. 現在の設定値は「Current」に表示されています。それぞれの値の内容は「図 4.11. minicom シリアルポートの設定値」を参照してください。

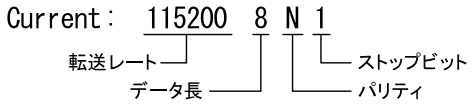


図 4.11 minicom シリアルポートの設定値

10. E キーを押して、転送レートを 115200 に設定してください。

11. Q キーを押して、データ長を 8、パリティを None、ストップビットを 1 に設定してください。
12. Enter キーを 2 回押して、「図 4.7. minicom の設定」に戻ってください。
13.「図 4.7. minicom の設定」から、「Save setup as dfl」を選択し、設定を保存してください。
14.「Exit from Minicom」を選択し、minicom の設定を終了してください。

minicom を起動させるには、「図 4.12. minicom 起動方法」のようにしてください。

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

図 4.12 minicom 起動方法

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

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

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

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

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

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

図 4.13 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。

4.6. インターフェースレイアウト

Armadillo-IoT ゲートウェイ A6 のインターフェースレイアウトです。各インターフェースの配置場所等を確認してください。

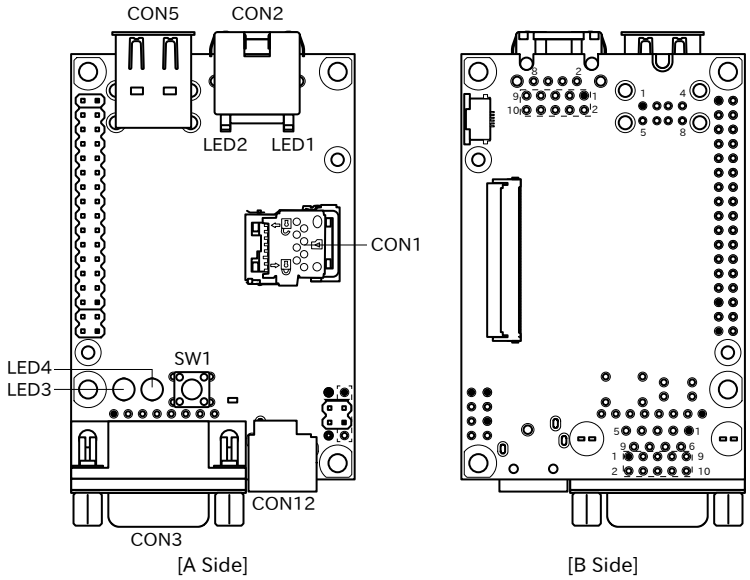


図 4.14 メインユニットインターフェースレイアウト(U1 モデル)

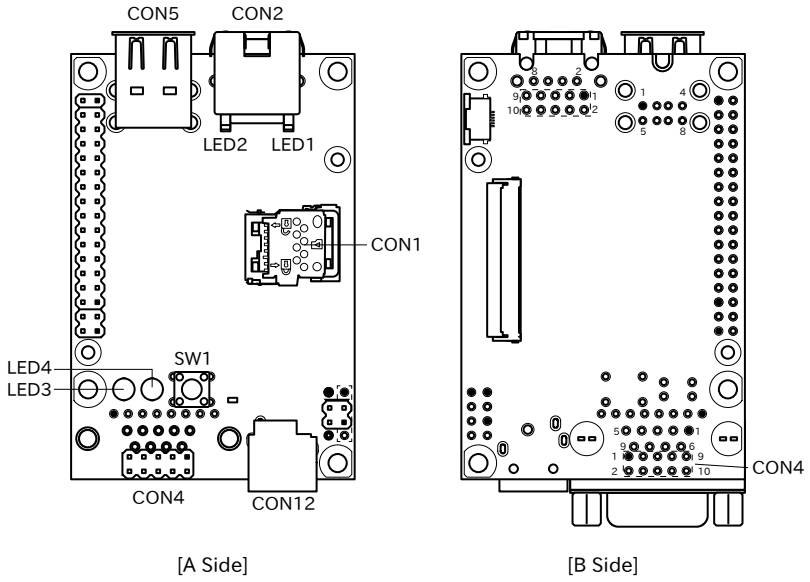


図 4.15 メインユニットインターフェースレイアウト(C1 モデル)

表 4.6 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
CON1	SD インターフェース	microSD スロット(ヒンジタイプ)	
CON2	LAN インターフェース	RJ-45 コネクタ	
CON3	シリアルインターフェース	D-Sub9 ピン(オス)	C1 モデルには、コネクタは実装されていません。

部品番号	インターフェース名	形状	備考
CON4	シリアルインターフェース	ピンヘッダ 10 ピン(2.54mm ピッチ)	U1 モデルには、コネクタは実装されていません。
CON5	USB インターフェース	Type-A コネクタ(2 ポートスタック)	
CON7	LAN インターフェース	ピンヘッダ 10 ピン(2.54mm ピッチ)	コネクタ非搭載
CON12	電源入力インターフェース	DC ジャック	対応プラグ: EIAJ#2
LED1	LAN スピード LED	LED(緑色,面実装)	
LED2	LAN リンクアクティビティ LED	LED(黄色,面実装)	
LED3	ユーザー LED(赤)	LED(赤色,φ3mm)	
LED4	ユーザー LED(緑)	LED(緑色,φ3mm)	
SW1	ユーザースイッチ	タクトスイッチ(h=17mm)	

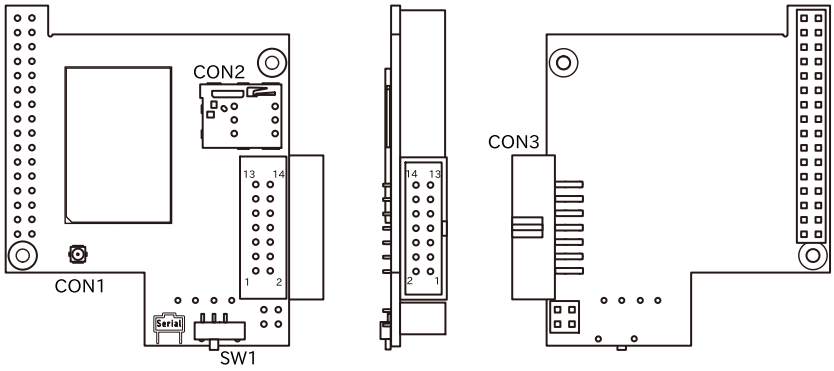


図 4.16 サブユニット インターフェースレイアウト

表 4.7 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
CON1	LTE アンテナインターフェース	小型同軸コネクタ	挿抜寿命: 20 回 [a]
CON2	nanoSIM インターフェース	nanoSIM スロット	
CON3	拡張インターフェース	ピンヘッダ 14 ピン(2.54mm ピッチ)	C1 モデルには、コネクタは実装されていません。
SW1	起動デバイス設定スイッチ	スライドスイッチ	

[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。

4.7. 接続方法

Armadillo-IoT ゲートウェイ A6 と周辺装置の接続例を次に示します。

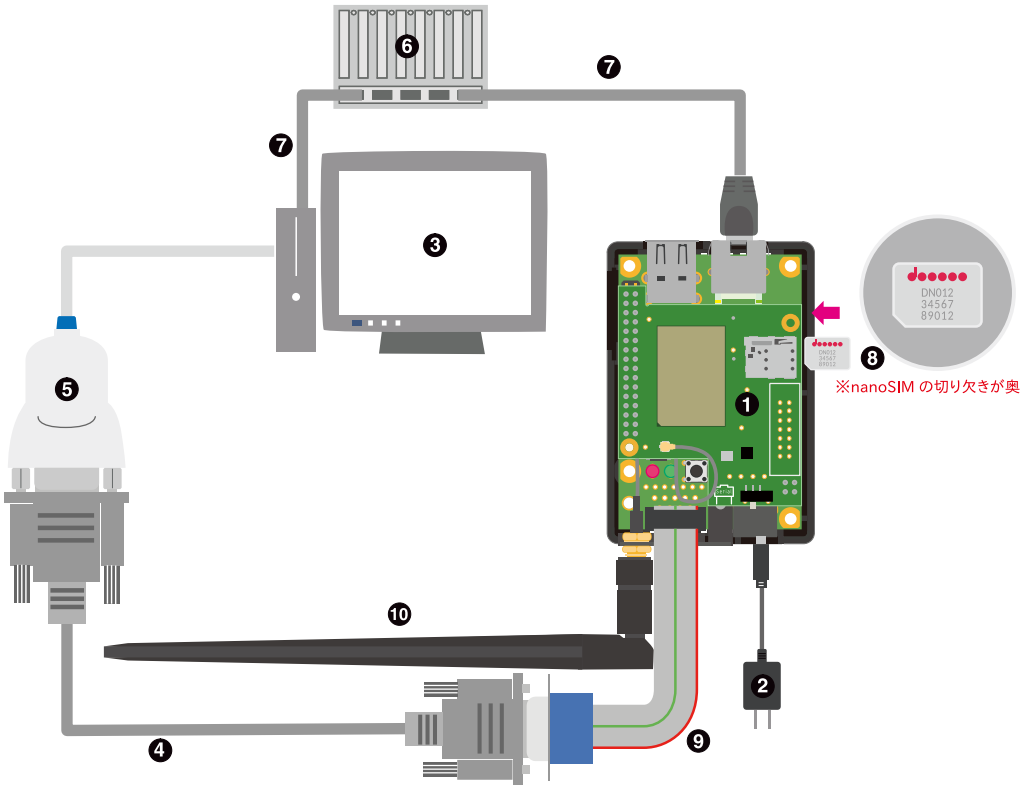


図 4.17 Armadillo-IoT ゲートウェイ A6 C1 モデルの接続例

❶ Armadillo-IoT ゲートウェイ

❷ AC アダプタ(5V/2A)

❸ 作業用 PC

❹ シリアルクロスケーブル

❺ USB-RS232C 変換ケーブル

❻ LAN HUB

❼ Ethernet ケーブル

❽ nanoSIM カード

❾ D-Sub9/10 ピンシリアル変換ケーブル

❿ LTE 用外付けアンテナ

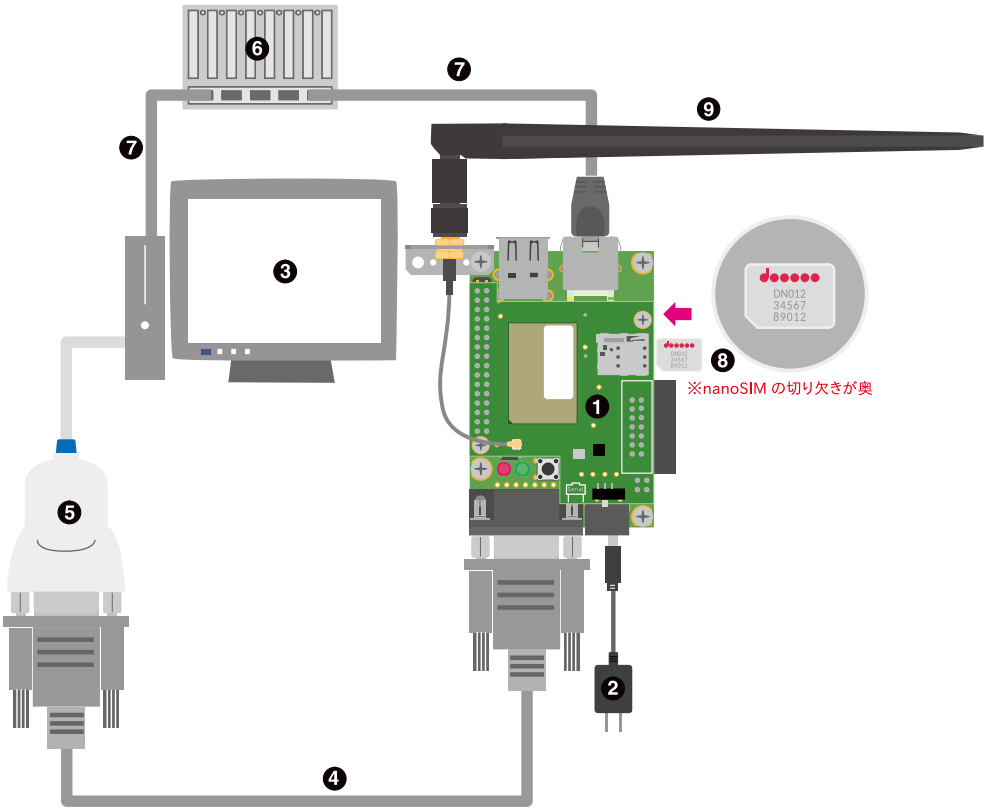


図 4.18 Armadillo-IoT ゲートウェイ A6 U1 モデルの接続例

1. Armadillo-IoT ゲートウェイ
2. AC アダプタ(5V/2A)
3. 作業用 PC
4. シリアルクロスケーブル
5. USB-RS232C 変換ケーブル
6. LAN HUB
7. Ethernet ケーブル
8. nanoSIM カード
9. LTE 用外付けアンテナ

4.8. スライドスイッチの設定について

サブユニットのスライドスイッチを操作することで、起動デバイスを設定することができます。

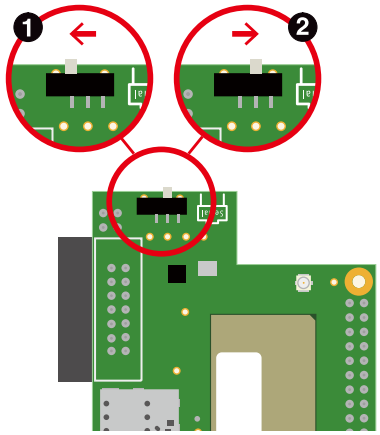


図 4.19 スライドスイッチの設定

❶ 起動デバイスは microSD になります。

❷ 起動デバイスは eMMC になります。

4.9. 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。

4.9.1. vi の起動

vi を起動するには、以下のコマンドを入力します。

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

図 4.20 vi の起動

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

4.9.2. 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 4.8. 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 4.8 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 4.21. 入力モードに移行するコマンドの説明」に示します。

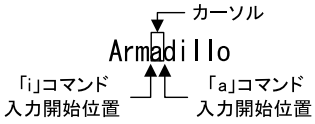


図 4.21 入力モードに移行するコマンドの説明

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、「4.9.4. 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。

4.9.3. 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 4.9. 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 4.9 カーソルの移動コマンド

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

4.9.4. 文字の削除

文字を削除する場合は、コマンドモードで「表 4.10. 文字の削除コマンド」に示すコマンドを入力します。

表 4.10 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 4.22. 文字を削除するコマンドの説明」に示します。

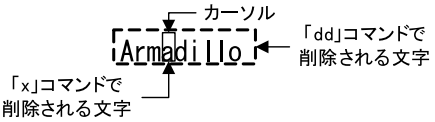


図 4.22 文字を削除するコマンドの説明

4.9.5. 保存と終了

ファイルの保存、終了を行うコマンドを「表 4.11. 保存・終了コマンド」に示します。

表 4.11 保存・終了コマンド

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

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

5. 起動と終了

5.1. 起動

CON12 に電源を接続すると、Armadillo-IoT ゲートウェイ A6 が起動します。

サブユニットのスライドスイッチやユーザースイッチによって起動モードが変わります。 詳しくは 「4.7. 接続方法」 、 「4.8. スライドスイッチの設定について」 、 「10.1. U-Boot の起動モード」 を参照してください。

本章では、保守モードで起動したときの例を示します。 オートブートモードで起動した場合は、途中でコマンドを入力することなく起動が完了します。

初めて起動した時は、 U-Boot の環境変数が eMMC に書き込まれていないために、上記のような Warning が表示されます。

```
U-Boot 2018.03-at8 (Feb 17 2020 - 19:19:00 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... *** Warning - bad CRC, using default environment

Failed (-5)
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

図 5.1 電源投入直後のログ (U-Boot の環境変数が eMMC に無い場合)

env save すると、次の起動から表示されなくなります。 詳しくは 「10. ブートローダー (U-Boot) 仕様」 を参照してください。

```
U-Boot 2018.03-at8 (Feb 17 2020 - 19:19:00 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
```

```
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

図 5.2 電源投入直後のログ (U-Boot の環境変数が eMMC にある場合)

Linux システム (Debian 10 "buster") 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。

```
=> boot
7253776 bytes read in 221 ms (31.3 MiB/s)
26390 bytes read in 54 ms (476.6 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at30-00002-gf0ee53950
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7253712 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02715 ... OK

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.14-at30-00002-gf0ee53950596 (atmark@atde8) (gcc version 8.3.0 ⏎
(Debian 8.3.0-2)) #2 Thu Mar 11 13:21:17 JST 2021
[    0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c53c7d
[    0.000000] CPU: div instructions available: patching division code
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[    0.000000] OF: fdt: Machine model: Atmark Techno Armadillo-IoT Gateway A6
[    0.000000] Memory policy: Data cache writeback
[    0.000000] cma: Reserved 16 MiB at 0x9f000000
[    0.000000] CPU: All CPU(s) started in SVC mode.
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 129920
[    0.000000] Kernel command line: root=/dev/mmcblk0p2 rootwait
[    0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)
[    0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
[    0.000000] Memory: 490896K/524288K available (6144K kernel code, 273K rwdata, 1396K rodata, ⏎
3072K init, 243K bss, 17008K reserved, 16384K cma-reserved)
[    0.000000] Virtual kernel memory layout:
[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
[    0.000000]     fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
[    0.000000]     vmalloc : 0xe0800000 - 0xff800000   ( 496 MB)
[    0.000000]     lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
[    0.000000]       .text : 0xc0008000 - 0xc0700000   (7136 kB)
[    0.000000]       .init : 0xc0900000 - 0xc0c00000   (3072 kB)
[    0.000000]       .data : 0xc0c00000 - 0xc0c44560   ( 274 kB)
[    0.000000]        .bss : 0xc0c49810 - 0xc0c8680c   ( 244 kB)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] NR_IRQS: 16, nr_irqs: 16, preallocated irqs: 16
```

```
[    0.000000] Switching to timer-based delay loop, resolution 41ns
[    0.000016] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 89478484971ns
[    0.000054] clocksource: mxc_timer1: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns:
79635851949 ns
[    0.001770] Console: colour dummy device 80x30
[    0.002379] console [tty0] enabled
[    0.002448] Calibrating delay loop (skipped), value calculated using timer frequency.. 48.00
BogoMIPS (lpj=240000)
[    0.002506] pid_max: default: 32768 minimum: 301
[    0.002926] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.002982] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.004198] CPU: Testing write buffer coherency: ok
[    0.005291] Setting up static identity map for 0x80100000 - 0x80100060
[    0.007326] devtmpfs: initialized
[    0.016729] random: get_random_u32 called from bucket_table_alloc+0x84/0x1a4 with crng_init=0
[    0.017224] VFP support v0.3: implementor 41 architecture 2 part 30 variant 7 rev 5
[    0.017627] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns:
19112604462750000 ns
[    0.017706] futex hash table entries: 256 (order: -1, 3072 bytes)
[    0.019031] pinctrl core: initialized pinctrl subsystem
[    0.020318] NET: Registered protocol family 16
[    0.022221] DMA: preallocated 256 KiB pool for atomic coherent allocations
[    0.030364] vdd3p0: supplied by regulator-dummy
[    0.031204] cpu: supplied by regulator-dummy
[    0.032004] vddsoc: supplied by regulator-dummy
[    0.045197] imx6ul-pinctrl 20e0000.iomuxc: initialized IMX pinctrl driver
[    0.045631] imx6ul-pinctrl 2290000.iomuxc-snvs: initialized IMX pinctrl driver
[    0.074176] SCSI subsystem initialized
[    0.074542] usbcore: registered new interface driver usbfs
[    0.074648] usbcore: registered new interface driver hub
[    0.074762] usbcore: registered new device driver usb
[    0.076088] i2c-gpio i2c-gpio1: using pins 31 (SDA) and 30 (SCL)
[    0.077060] Advanced Linux Sound Architecture Driver Initialized.
[    0.078196] Bluetooth: Core ver 2.22
[    0.078299] NET: Registered protocol family 31
[    0.078334] Bluetooth: HCI device and connection manager initialized
[    0.078388] Bluetooth: HCI socket layer initialized
[    0.078430] Bluetooth: L2CAP socket layer initialized
[    0.078469] Bluetooth: SCO socket layer initialized
[    0.079493] clocksource: Switched to clocksource mxc_timer1
[    0.095413] NET: Registered protocol family 2
[    0.096567] TCP established hash table entries: 4096 (order: 2, 16384 bytes)
[    0.096703] TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
[    0.096830] TCP: Hash tables configured (established 4096 bind 4096)
[    0.097027] UDP hash table entries: 256 (order: 0, 4096 bytes)
[    0.097096] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
[    0.097427] NET: Registered protocol family 1
[    0.098118] RPC: Registered named UNIX socket transport module.
[    0.098173] RPC: Registered udp transport module.
[    0.098206] RPC: Registered tcp transport module.
[    0.098239] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    0.427959] workingset: timestamp_bits=14 max_order=17 bucket_order=3
[    0.444051] squashfs: version 4.0 (2009/01/31) Phillip Lougher
[    0.446126] NFS: Registering the id_resolver key type
[    0.446230] Key type id_resolver registered
[    0.446267] Key type id_legacy registered
[    0.446745] fuse init (API version 7.26)
[    0.454843] NET: Registered protocol family 38
```

```
[    0.455412] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 250)
[    0.455474] io scheduler noop registered
[    0.455508] io scheduler deadline registered
[    0.455954] io scheduler cfq registered (default)
[    0.467114] 21e8000.serial: ttymxc1 at MMIO 0x21e8000 (irq = 49, base_baud = 5000000) is a IMX
[    0.467917] imx-sdma 20ec000.sdma: loaded firmware 3.3
[    0.469075] 21ec000.serial: ttymxc2 at MMIO 0x21ec000 (irq = 50, base_baud = 5000000) is a IMX
[    0.469164] Console IMX rounded baud rate from 114943 to 114900
[    0.998494] console [ttymxc2] enabled
[    1.004581] libphy: Fixed MDIO Bus: probed
[    1.009058] tun: Universal TUN/TAP device driver, 1.6
[    1.014648] CAN device driver interface
[    1.020380] fec 2188000.ethernet: 2188000.ethernet supply phy not found, using dummy regulator
[    1.030428] libphy: fec_enet_mii_bus: probed
[    1.070647] PPP generic driver version 2.4.2
[    1.075427] usbcore: registered new interface driver awl13
[    1.081031] pegasus: v0.9.3 (2013/04/25), Pegasus/Pegasus II USB Ethernet driver
[    1.088539] usbcore: registered new interface driver pegasus
[    1.094355] usbcore: registered new interface driver rtl8150
[    1.100139] usbcore: registered new interface driver r8152
[    1.105729] usbcore: registered new interface driver lan78xx
[    1.111519] usbcore: registered new interface driver asix
[    1.117004] usbcore: registered new interface driver ax88179_178a
[    1.123221] usbcore: registered new interface driver cdc_ether
[    1.129145] usbcore: registered new interface driver net1080
[    1.134923] usbcore: registered new interface driver cdc_subset
[    1.140962] usbcore: registered new interface driver zaurus
[    1.146648] usbcore: registered new interface driver cdc_ncm
[    1.152373] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[    1.158934] ehci-mxc: Freescale On-Chip EHCI Host driver
[    1.164549] usbcore: registered new interface driver cdc_acm
[    1.170288] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN adapters
[    1.178401] usbcore: registered new interface driver usb-storage
[    1.184653] usbcore: registered new interface driver usbserial
[    1.190627] usbcore: registered new interface driver usbserial_generic
[    1.197237] usbserial: USB Serial support registered for generic
[    1.203364] usbcore: registered new interface driver cp210x
[    1.209036] usbserial: USB Serial support registered for cp210x
[    1.215084] usbcore: registered new interface driver ftdi_sio
[    1.220943] usbserial: USB Serial support registered for FTDI USB Serial Device
[    1.228344] usbcore: registered new interface driver mxuport
[    1.234114] usbserial: USB Serial support registered for MOXA UPort
[    1.240506] usbcore: registered new interface driver pl2303
[    1.246155] usbserial: USB Serial support registered for pl2303
[    1.257996] ci_hdrc ci_hdrc.0: EHCI Host Controller
[    1.263041] ci_hdrc ci_hdrc.0: new USB bus registered, assigned bus number 1
[    1.299532] ci_hdrc ci_hdrc.0: USB 2.0 started, EHCI 1.00
[    1.306468] hub 1-0:1.0: USB hub found
[    1.310425] hub 1-0:1.0: 1 port detected
[    1.319581] ci_hdrc ci_hdrc.1: EHCI Host Controller
[    1.324565] ci_hdrc ci_hdrc.1: new USB bus registered, assigned bus number 2
[    1.359534] ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
[    1.366477] hub 2-0:1.0: USB hub found
[    1.370430] hub 2-0:1.0: 1 port detected
[    1.376337] udc-core: couldn't find an available UDC - added [g_cdc] to list of pending drivers
[    1.391520] rtc-nr3225sa 4-0032: registered as rtc0
[    1.397627] snvs_rtc 20cc000.snvs:snvs-rtc-lp: registered as rtc1
[    1.404121] i2c /dev entries driver
```

```
[    1.408090] IR NEC protocol handler initialized
[    1.412729] IR RC5(x/sz) protocol handler initialized
[    1.417814] IR RC6 protocol handler initialized
[    1.422418] IR JVC protocol handler initialized
[    1.426975] IR Sony protocol handler initialized
[    1.431650] IR SANYO protocol handler initialized
[    1.436384] IR Sharp protocol handler initialized
[    1.441136] IR MCE Keyboard/mouse protocol handler initialized
[    1.446997] IR XMP protocol handler initialized
[    1.453924] imx2-wdt 20bc000.wdog: timeout 60 sec (nowayout=0)
[    1.460281] usbcore: registered new interface driver btusb
[    1.466059] sdhci: Secure Digital Host Controller Interface driver
[    1.472336] sdhci: Copyright(c) Pierre Ossman
[    1.476727] sdhci-pltfm: SDHCI platform and OF driver helper
[    1.549538] mmc0: SDHCI controller on 2190000.usdhc [2190000.usdhc] using ADMA
[    1.619969] mmc1: SDHCI controller on 2194000.usdhc [2194000.usdhc] using ADMA
[    1.638398] mmc0: new DDR MMC card at address 0001
[    1.644685] mmcblk0: mmc0:0001 Q2J55L 3.53 GiB
[    1.650508] usbcore: registered new interface driver usbhid
[    1.656129] usbhid: USB HID core driver
[    1.661906] mmcblk0boot0: mmc0:0001 Q2J55L partition 1 2.00 MiB
[    1.672562] Netfilter messages via NETLINK v0.30.
[    1.677694] nf_conntrack version 0.5.0 (8192 buckets, 32768 max)
[    1.684292] nf_tables: (c) 2007-2009 Patrick McHardy <kaber@trash.net>
[    1.690969] nf_tables_compat: (c) 2012 Pablo Neira Ayuso <pablo@netfilter.org>
[    1.698913] ip_tables: (C) 2000-2006 Netfilter Core Team
[    1.705898] NET: Registered protocol family 10
[    1.712129] mmcblk0boot1: mmc0:0001 Q2J55L partition 2 2.00 MiB
[    1.718697] mmcblk0gp0: mmc0:0001 Q2J55L partition 4 8.00 MiB
[    1.725036] mmcblk0gp1: mmc0:0001 Q2J55L partition 5 8.00 MiB
[    1.731355] mmcblk0gp2: mmc0:0001 Q2J55L partition 6 8.00 MiB
[    1.738637] mmcblk0gp3: mmc0:0001 Q2J55L partition 7 8.00 MiB
[    1.746093] mmcblk0rpmb: mmc0:0001 Q2J55L partition 3 4.00 MiB, chardev (248:0)
[    1.758680] Segment Routing with IPv6
[    1.762622] sit: IPv6, IPv4 and MPLS over IPv4 tunneling driver
[    1.769841] NET: Registered protocol family 17
[    1.774357] can: controller area network core (rev 20170425 abi 9)
[    1.780974] NET: Registered protocol family 29
[    1.785464] can: raw protocol (rev 20170425)
[    1.789816] can: broadcast manager protocol (rev 20170425 t)
[    1.795527] can: netlink gateway (rev 20170425) max_hops=1
[    1.803690]  mmcblk0: p1 p2 p3
[    1.810219] Bluetooth: RFCOMM TTY layer initialized
[    1.815216] Bluetooth: RFCOMM socket layer initialized
[    1.820438] Bluetooth: RFCOMM ver 1.11
[    1.824220] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[    1.829600] Bluetooth: BNEP filters: protocol multicast
[    1.834872] Bluetooth: BNEP socket layer initialized
[    1.839896] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[    1.845855] Bluetooth: HIDP socket layer initialized
[    1.851008] Key type dns_resolver registered
[    1.856123] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 900000000, volt:
1275000, enabled: 1. New: freq: 900000000, volt: 1275000, enabled: 1
[    1.870475] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 792000000, volt:
1225000, enabled: 1. New: freq: 792000000, volt: 1225000, enabled: 1
[    1.884754] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 528000000, volt:
1175000, enabled: 1. New: freq: 528000000, volt: 1175000, enabled: 1
[    1.899022] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 396000000, volt:
```

```
1025000, enabled: 1. New: freq: 396000000, volt: 1025000, enabled: 1
[    1.913284] cpu cpu0: _opp_add: duplicate OPPs detected. Existing: freq: 198000000, volt: 950000,
enabled: 1. New: freq: 198000000, volt: 950000, enabled: 1
[    1.928226] ThumbEE CPU extension supported.
[    1.932584] random: fast init done
[    1.980948] input: gpio-keys as /devices/soc0/gpio-keys/input/input0
[    1.988768] input: gpio-wakeup as /devices/soc0/gpio-wakeup/input/input1
[    1.999317] rtc-nr3225sa 4-0032: setting system clock to 2021-03-15 05:01:24 UTC (1615784484)
[    2.008493] ALSA device list:
[    2.011560]   No soundcards found.
[    2.015130] Warning: unable to open an initial console.
[    2.025928] Freeing unused kernel memory: 3072K
[    2.180750] systemd-udevd[83]: starting version 215
[    2.187938] random: systemd-udevd: uninitialized urandom read (16 bytes read)
[    3.344328] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
[    3.878318] systemd[1]: systemd 241 running in system mode. (+PAM +AUDIT +SELINUX +IMA +APPARMOR
+SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD
-IDN2 +IDN -PCRE2 default-hierarchy=hybrid)
[    3.900763] systemd[1]: Detected architecture arm.

Welcome to Debian GNU/Linux 10 (buster)!

[    3.940591] systemd[1]: Set hostname to <armadillo>.
[    5.497729] random: systemd: uninitialized urandom read (16 bytes read)
[    5.518400] random: systemd: uninitialized urandom read (16 bytes read)
[    5.526666] systemd[1]: Listening on Journal Socket (/dev/log).
[  OK  ] Listening on Journal Socket (/dev/log).
[    5.560424] random: systemd: uninitialized urandom read (16 bytes read)
[    5.568229] systemd[1]: Set up automount Arbitrary Executable File Formats File System Automount
Point.
[  OK  ] Set up automount Arbitrary…s File System Automount Point.
[    5.610751] systemd[1]: Listening on udev Kernel Socket.
[  OK  ] Listening on udev Kernel Socket.
[    5.650813] systemd[1]: Listening on fsck to fsckd communication Socket.
[  OK  ] Listening on fsck to fsckd communication Socket.
[    5.693444] systemd[1]: Created slice User and Session Slice.
[  OK  ] Created slice User and Session Slice.
[    5.734281] systemd[1]: Created slice system-serial¥x2dgetty.slice.
[  OK  ] Created slice system-serial¥x2dgetty.slice.
[    5.769946] systemd[1]: Reached target Slices.
[  OK  ] Reached target Slices.
[  OK  ] Created slice system-getty.slice.
[  OK  ] Started Forward Password R…uests to Wall Directory Watch.
[  OK  ] Listening on Journal Socket.
         Starting Remount Root and Kernel File Systems...
[  OK  ] Reached target Remote File Systems.
[  OK  ] Listening on Syslog Socket.
         Starting Load Kernel Modules...
[  OK  ] Listening on udev Control Socket.
[    6.065799] EXT4-fs (mmcblk0p2): re-mounted. Opts: (null)
         Starting udev Coldplug all Devices...
         Starting Journal Service...
[  OK  ] Listening on initctl Compatibility Named Pipe.
[  OK  ] Reached target Swap.
[  OK  ] Started Dispatch Password …ts to Console Directory Watch.
[  OK  ] Reached target Local Encrypted Volumes.
[  OK  ] Reached target Paths.
[  OK  ] Started Remount Root and Kernel File Systems.
```

```
[  OK  ] Started Load Kernel Modules.
         Mounting FUSE Control File System...
         Mounting Kernel Configuration File System...
         Starting Apply Kernel Variables...
         Starting Create System Users...
         Starting Load/Save Random Seed...
[  OK  ] Mounted FUSE Control File System.
[  OK  ] Mounted Kernel Configuration File System.
[  OK  ] Started Apply Kernel Variables.
[  OK  ] Started Load/Save Random Seed.
[  OK  ] Started Journal Service.
[  OK  ] Started Create System Users.
         Starting Create Static Device Nodes in /dev...
         Starting Flush Journal to Persistent Storage...
[  OK  ] Started Create Static Device Nodes in /dev.
[    7.652477] systemd-journald[158]: Received request to flush runtime journal from PID 1
[  OK  ] Reached target Local File Systems (Pre).
[  OK  ] Reached target Local File Systems.
         Starting netfilter persistent configuration...
         Starting udev Kernel Device Manager...
[  OK  ] Started udev Coldplug all Devices.
[  OK  ] Started Flush Journal to Persistent Storage.
[  OK  ] Started netfilter persistent configuration.
[  OK  ] Reached target Network (Pre).
         Starting Create Volatile Files and Directories...
         Starting Helper to synchronize boot up for ifupdown...
[  OK  ] Started udev Kernel Device Manager.
[  OK  ] Started Helper to synchronize boot up for ifupdown.
         Starting Raise network interfaces...
[  OK  ] Started Raise network interfaces.
[  OK  ] Started Create Volatile Files and Directories.
         Starting Network Time Synchronization...
         Starting Update UTMP about System Boot/Shutdown...
[  OK  ] Found device /dev/ttymxc2.
[  OK  ] Started Update UTMP about System Boot/Shutdown.
[  OK  ] Started Network Time Synchronization.
[  OK  ] Found device /dev/mmcblk0gp0.
         Mounting /opt/license...
[  OK  ] Started ifup for eth0.
[   12.412390] squashfs: SQUASHFS error: Can't find a SQUASHFS superblock on mmcblk0gp0
[  OK  ] Reached target System Time Synchronized.
[  OK  ] Reached target System Initialization.
[  OK  ] Started Daily man-db regeneration.
[  OK  ] Started Daily apt download activities.
[  OK  ] Started Daily rotation of log files.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Sockets.
[  OK  ] Reached target Basic System.
[  OK  ] Started Control EMS31 on Armadillo.
[  OK  ] Reached target Network.
[   12.760246] SMSC LAN8710/LAN8720 2188000.ethernet-1:00: attached PHY driver [SMSC LAN8710/ ⏎
LAN8720] (mii_bus:phy_addr=2188000.ethernet-1:00, irq=POLL)
         Starting Restore /etc/reso… the ppp link was shut down...
[   12.826705] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
         Starting OpenBSD Secure Shell server...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started Daily apt upgrade and clean activities.
```

```
[  OK  ] Reached target Timers.
         Starting System Logging Service...
         Starting Deferred execution scheduler...
         Starting Login Service...
[  OK  ] Started Regular background program processing daemon.
         Starting /etc/rc.local Compatibility...
         Starting Permit User Sessions...
[  OK  ] Started System Logging Service.
[  OK  ] Mounted /opt/license.
[  OK  ] Started Restore /etc/resol…re the ppp link was shut down.
[  OK  ] Started Deferred execution scheduler.
[  OK  ] Started /etc/rc.local Compatibility.
[  OK  ] Started Permit User Sessions.
[  OK  ] Started Getty on tty1.
[  OK  ] Started Serial Getty on ttymxc2.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started Login Service.
[   14.890012] fec 2188000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
[   14.897929] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready

Debian GNU/Linux 10 armadillo ttymxc2

armadillo login:
```

5.2. ログイン

5.2.1. 新しいパスワードの準備

初めてログインしたときは、パスワードの変更を促されます。事前に新しいパスワードを用意してください。設定するパスワードには大文字のアルファベット、小文字のアルファベット、0 から 9 までの数字、その他(記号・句読点など)を含める事ができます。

表 5.1 パスワードに設定可能な値

項目	範囲
大文字のアルファベット	A~Z
小文字のアルファベット	a~z
数字	0~9
その他(記号・句読点など)	!"#$%&‘()*+,-./:;<⇒?@[\]^_`{|}~

「表 5.1 パスワードに設定可能な値」の中から、4 つの項目を全て含めた場合は、6 文字以上でなければパスワードの安全性が低いためエラーとなります。項目が 3 つの場合は 7 文字以上。項目が 2 つの場合は 8 文字以上。項目が 1 つの場合は 9 文字以上のパスワードを設定して下さい。また、"ABCDEDCBA" など左右対称のパスワードはエラーとなるため使用しないで下さい。

パスワードを自動生成する pwgen コマンドがあります。作業用 PC に pwgen をインストールして、9 桁のパスワードを生成する例を次に示します。

```
[PC ~]$ sudo apt-get install pwgen
[PC ~]$ pwgen -y 9 1
********* # 9 桁の値が生成されます。
```

5.2.2. 初回ログインと新しいパスワードの設定

初めてログインしたときは、パスワードを変更するようにメッセージが表示されます。以下の手順に従って、パスワードを変更してください。

1. root でログイン

   パスワードを変更します。

```
You are required to change your password immediately (root enforced)
Changing password for root.
(current) UNIX password: root # 初期パスワード”root”を入力
Enter new UNIX password: # 新しいパスワードを入力
Retype new UNIX password: # 新しいパスワードを再入力
```

2. atmark でログイン

   パスワードを変更します。

```
You are required to change your password immediately (root enforced)
Changing password for atmark.
(current) UNIX password: atmark # 初期パスワード”atmark”を入力
Enter new UNIX password: # 新しいパスワードを入力
Retype new UNIX password: # 新しいパスワードを再入力
```

Armadillo-IoT ゲートウェイ A6 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。最終製品として作り上げる場合は、外部からのログインは極力できないようにすることをお勧めします。どうしてもログインが必要な場合は、セキュリティ強度の高いパスワードを設定し、その後も適切にパスワードを運用されることを強くお勧めします。

5.3. Debian のユーザを管理する

例として guest というユーザを作成、パスワードの変更、sudo を許可する方法を紹介します。

```
[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
```

図 5.3 ユーザの作成

```
[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
```

図 5.4 パスワードの変更

```
[armadillo ~]# sudo usermod -a -G sudo guest
```

図 5.5 sudo を許可

```
[armadillo ~]# deluser guest
  or
[armadillo ~]# deluser --remove-home guest
```

図 5.6 ユーザの削除

ホームディレクトリも合わせて消したいときは、 --remove-home オプションをつけます。

5.4. 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、「reboot: 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
[  OK  ] Stopped target Graphical Interface.
[  OK  ] Stopped target Multi-User System.
         Stopping Regular background program processing daemon...
         Stopping Deferred execution scheduler...
         Stopping OpenBSD Secure Shell server...
         Stopping System Logging Service...
[  OK  ] Stopped target Login Prompts.
         Stopping Serial Getty on ttymxc2...
[  OK  ] Stopped Session c1 of user root.
         Stopping Login Service...
[  OK  ] Stopped target Timers.
[  OK  ] Stopped Daily man-db regeneration.
[  OK  ] Stopped Daily apt upgrade and clean activities.
[  OK  ] Stopped Daily Cleanup of Temporary Directories.
         Stopping User Manager for UID 0...
[  OK  ] Stopped Daily apt download activities.
[  OK  ] Stopped Daily rotation of log files.
[  OK  ] Stopped target System Time Synchronized.
[  OK  ] Stopped System Logging Service.
[  OK  ] Stopped Regular background program processing daemon.
```

```
[  OK  ] Stopped Deferred execution scheduler.
[  OK  ] Stopped Getty on tty1.
[  OK  ] Stopped Serial Getty on ttymxc2.
[  OK  ] Stopped OpenBSD Secure Shell server.
[  OK  ] Stopped User Manager for UID 0.
         Stopping User Runtime Directory /run/user/0...
[  OK  ] Removed slice system-serial¥x2dgetty.slice.
[  OK  ] Removed slice system-getty.slice.
[  OK  ] Stopped /etc/rc.local Compatibility.
[  OK  ] Unmounted /run/user/0.
[  OK  ] Stopped Login Service.
[  OK  ] Stopped User Runtime Directory /run/user/0.
         Stopping D-Bus System Message Bus...
[  OK  ] Removed slice User Slice of UID 0.
         Stopping Permit User Sessions...
[  OK  ] Reached target Unmount All Filesystems.
[  OK  ] Stopped D-Bus System Message Bus.
[  OK  ] Stopped Permit User Sessions.
[  OK  ] Stopped target Network.
         Stopping ifup for eth0...
         Stopping Control EMS31 on Armadillo...
         Stopping Raise network interfaces...
[  OK  ] Stopped target Remote File Systems.
[  OK  ] Stopped Raise network interfaces.
[  OK  ] Stopped Control EMS31 on Armadillo.
[  OK  ] Stopped target Basic System.
[  OK  ] Stopped target Sockets.
[  OK  ] Closed Syslog Socket.
[  OK  ] Closed D-Bus System Message Bus Socket.
[  OK  ] Stopped target System Initialization.
         Stopping Network Time Synchronization...
[  OK  ] Stopped target Swap.
         Stopping Load/Save Random Seed...
         Stopping Update UTMP about System Boot/Shutdown...
[  OK  ] Stopped target Local Encrypted Volumes.
[  OK  ] Stopped target Paths.
[  OK  ] Stopped Forward Password R…uests to Wall Directory Watch.
[  OK  ] Stopped Dispatch Password …ts to Console Directory Watch.
[  OK  ] Stopped target Slices.
[  OK  ] Removed slice User and Session Slice.
[  OK  ] Stopped Network Time Synchronization.
[  OK  ] Stopped Load/Save Random Seed.
[  OK  ] Stopped Update UTMP about System Boot/Shutdown.
[  OK  ] Stopped Create Volatile Files and Directories.
[  OK  ] Stopped ifup for eth0.
[  OK  ] Stopped Apply Kernel Variables.
[  OK  ] Stopped target Network (Pre).
         Stopping netfilter persistent configuration...
[  OK  ] Stopped netfilter persistent configuration.
[  OK  ] Stopped Load Kernel Modules.
[  OK  ] Stopped target Local File Systems.
[  OK  ] Stopped target Local File Systems (Pre).
[  OK  ] Stopped Create Static Device Nodes in /dev.
[  OK  ] Stopped Create System Users.
[  OK  ] Stopped Remount Root and Kernel File Systems.
[  OK  ] Reached target Shutdown.
[  OK  ] Reached target Final Step.
         Starting Halt...
```

```
[ 89.022696] systemd-shutdow: 35 output lines suppressed due to ratelimiting
[ 89.125737] systemd-shutdown[1]: Syncing filesystems and block devices.
[ 89.147069] systemd-shutdown[1]: Sending SIGTERM to remaining processes...
[ 89.166150] systemd-journald[158]: Received SIGTERM from PID 1 (systemd-shutdow).
[ 89.186341] systemd-shutdown[1]: Sending SIGKILL to remaining processes...
[ 89.205342] systemd-shutdown[1]: Unmounting file systems.
[ 89.217901] [1085]: Remounting '/' read-only in with options 'data=ordered'.
[ 89.246392] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
[ 89.271636] systemd-shutdown[1]: All filesystems unmounted.
[ 89.277274] systemd-shutdown[1]: Deactivating swaps.
[ 89.282983] systemd-shutdown[1]: All swaps deactivated.
[ 89.288271] systemd-shutdown[1]: Detaching loop devices.
[ 89.294522] systemd-shutdown[1]: All loop devices detached.
[ 89.300153] systemd-shutdown[1]: Detaching DM devices.
[ 89.346219] ci_hdrc ci_hdrc.1: remove, state 4
[ 89.350734] usb usb2: USB disconnect, device number 1
[ 89.356983] ci_hdrc ci_hdrc.1: USB bus 2 deregistered
[ 89.365443] ci_hdrc ci_hdrc.0: remove, state 4
[ 89.369937] usb usb1: USB disconnect, device number 1
[ 89.376110] ci_hdrc ci_hdrc.0: USB bus 1 deregistered
[ 89.381502] reboot: System halted
```

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。

poweroff コマンドでも Armadillo-IoT ゲートウェイ A6 を終了させることができます。

```
[armadillo ~]# poweroff
:（省略）
[   30.356097] reboot: Power down
```

6. 動作確認方法

6.1. 動作確認を行う前に

Armadillo には、 OS として Debian がインストールされています。基本的には PC Linux と同じように動作します。ここではネットワークの設定やストレージのように一般的な動作に加え、GPIO や LED などについて説明します。

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については、「12. イメージファイルの書き換え方法」を参照してください。

6.2. 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。

6.2.1. 接続可能なネットワーク

Armadillo-IoT ゲートウェイ A6 は、 LTE と Ethernet に対応しています。Linux からは、`ppp0` と `eth0` に見えます。

表 6.1 ネットワークとネットワークデバイス

ネットワーク	ネットワークデバイス	出荷時の設定
LTE	`ppp0`	DHCP
Ethernet	`eth0`	DHCP

6.2.2. 有線 LAN の設定方法

ここでは有線 LAN の使用方法について説明します。Armadillo-IoT ゲートウェイ A6 では、通常の Linux システムと同様にネットワーク設定を行います。出荷状態では `eth0` が DHCP [1] でネットワークの設定を行います。 DHCP が無い環境の場合は、 「6.2.3.3. 固定 IP アドレスに設定する」 を参照し設定してください。

6.2.3. 基本的な使い方

最近の GNU/Linux OS では、古くから使われてきた `ifconfig` (net-tools) に代り iproute2 を使用します。 `ifconfig` は Deprecated されています。本書でも `ifconfig` ではなく、 iproute2 に含まれている `ip` コマンドなどを使用します。

[1] Dynamic Host Configuration Protocol

6.2.3.1. IP アドレスの一覧

IP アドレス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。

```
[armadillo ~]# ip address
ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr
oup default qlen 1000
    link/ether 00:11:0c:00:07:a4 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.107/16 brd 172.16.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

図 6.1 インターフェースの一覧確認

ネットワークデバイスの一覧を確認するには link を使います。

```
[armadillo ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
 group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT group default qlen 1000
    link/ether 00:11:0c:00:07:a4 brd ff:ff:ff:ff:ff:ff
```

図 6.2 ネットワークデバイスの一覧確認

他にも ip コマンドではルーティングテーブルの表示やトンネルの設定などもできます。詳しくは ip コマンドの man ページを参照してください。

6.2.3.2. インターフェースの有効化・無効化

インターフェースを有効化するには、次のようにコマンドを実行します。 ifup コマンドは Debian 特有のコマンドで ifupdown パッケージに含まれています。 ifconfig とは関係なく抽象的にネットワークを操作するためのコマンドです。設定は /etc/network/interfaces で行います。

```
[armadillo ~]# ifup eth0
```

図 6.3 インターフェースの有効化

インターフェースを無効化するには、次のようにコマンドを実行します。

```
[armadillo ~]# ifdown eth0
```

図 6.4 インターフェースの無効化

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。

/etc/network/interfaces の変更は、インターフェースを無効化した状態で行ってください。 DHCP が動作している場合など、設定が反映されない場合があります。

6.2.3.3. 固定 IP アドレスに設定する

「表 6.2. 固定 IP アドレス設定例」の内容に設定する例を、以下に示します。

表 6.2 固定 IP アドレス設定例

項目	設定
IP アドレス	192.0.2.10
ネットマスク	255.255.255.0
ネットワークアドレス	192.0.2.0
ブロードキャストアドレス	192.0.2.255
デフォルトゲートウェイ	192.0.2.1

```
[armadillo ~]# vi /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

auto lo eth0
iface lo inet loopback
iface eth0 inet static
    address 192.0.2.10
    netmask 255.255.255.0
    network 192.0.2.0
    broadcast 192.0.2.255
    gateway 192.0.2.1
```

図 6.5 固定 IP アドレス設定

6.2.3.4. DHCP に設定する

DHCP に設定する例を以下にに示します。

```
[armadillo ~]# vi /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

auto lo
iface lo inet loopback
iface eth0 inet dhcp
```

図 6.6 DHCP 設定

6.2.3.5. DNS サーバーを指定する

固定 IP 時、または DHCP で DNS 情報が取得できない場合は、DNS サーバーを指定する必要があります。DNS サーバーを指定する例を、以下に示します。

```
[armadillo ~]# vi /etc/resolv.conf
domain local-network
search local-network
nameserver 192.0.2.1
```

図 6.7 DNS サーバーの指定

6.2.3.6. インターフェースの修正を反映する

有効化されているインターフェースは、修正しないでください。必ず無効化してから設定を変更してください。

```
[armadillo ~]# ifdown eth0
[armadillo ~]# vi /etc/network/interfaces
  :
[armadillo ~]# ifup eth0
```

6.2.3.7. 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、必ず変更したインターフェースを再度有効化してください。

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

```
[armadillo ~]# ping 192.0.2.20
ping -c 3 192.0.2.20
PING 192.0.2.20 (192.0.2.20) 56(84) bytes of data.
64 bytes from 192.0.2.20: icmp_seq=1 ttl=63 time=1.39 ms
64 bytes from 192.0.2.20: icmp_seq=2 ttl=63 time=1.35 ms
64 bytes from 192.0.2.20: icmp_seq=3 ttl=63 time=1.34 ms

--- 192.0.2.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.343/1.365/1.395/0.021 ms
```

図 6.8 有線 LAN の PING 確認

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

6.2.4. LTE の設定

ここでは、Armadillo-IoT ゲートウェイ A6 に搭載されている LTE モジュールの使用方法について説明します。

Armadillo-IoT ゲートウェイの初期イメージには、 LTE モデムを制御するサービス (aiot-modem-controld) とそのサービスを制御するためのアプリケーション(aiot-modem-control) がインストールされています。aiot-modem-controld は、 ppp コネクションの確立と IP アドレスの取得に関して、wvdial というアプリケーションを用いて実行しています。

6.2.4.1. LTE データ通信設定を行う前に

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

> Armadillo-IoT ゲートウェイ A6 の電源が切断されていることを確認してから nanoSIM(UIM カード) を取り付けてください。

nanoSIM(UIM カード) の切り欠きを挿入方向に向け、刻印面を上にして挿入してください。向きと上下を間違えると挿入できませんので、無理に挿入しないでください。破損する可能性があります。

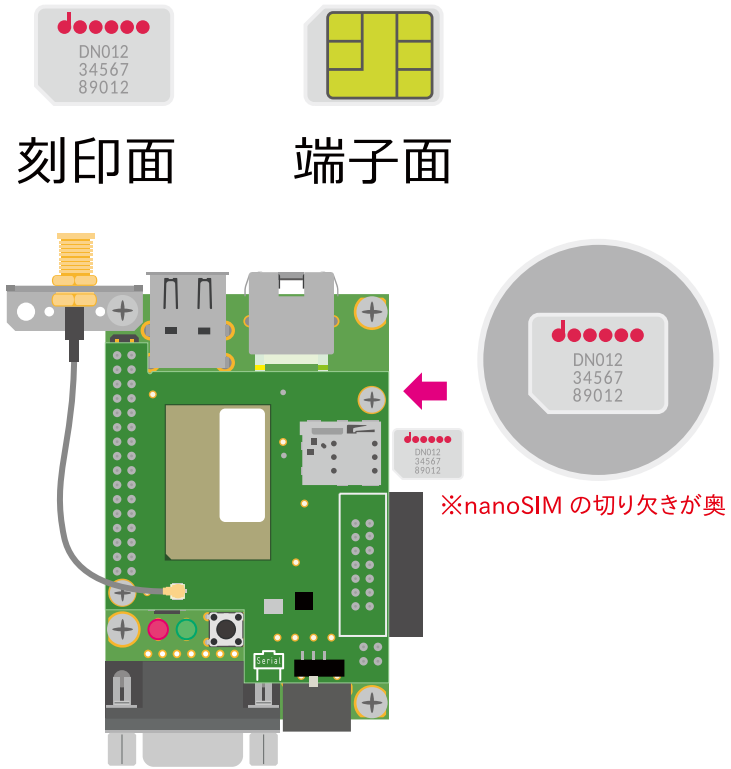


図 6.9 nano SIM の取り付け

APN の設定を行うには、次に示す情報が必要です。

- APN(最大 99 文字)
- ユーザー名(最大 64 文字)
- パスワード(最大 64 文字)
- 認証方式(PAP または CHAP)

6.2.4.2. LTE 設定ファイル (/etc/aiot-modem-control/startup.conf) の編集

標準イメージを利用されている場合、Armadillo-IoT ゲートウェイ A6 起動時、自動的に LTE の通信設定を行うサービスが起動します。このサービスは、 /etc/aiot-modem-control/startup.conf の設定内容を参照し、設定内容に応じた値を LTE モデムに設定します。

全てのパラメーターに関して、設定項目が存在しない場合、該当する項目は設定しませんので、独自のアプリケーションなどで設定したい、もしくは設定をしたくない項目はコメントにしてください。

1. 全て パラメーター=値 の形式で記載してください。
2. 値にはダブルクオテーションを含めて最大 102 文字設定可能です。それ以上長い値は無視します。
3. 1 行の最大文字数はコメントを含めて 256 文字です。それ以降の文字列は無視します。
4. スペース及びタブは無視します。
5. 行内 # 以降はコメントとみなします。

```
apn=APN 名
username=ユーザー名
password=パスワード
auth_prot=pap or chap or none
create_wvdial_conf=true
operator_id=オペレーター ID
fix_profile=プロファイル名
sleep=シリアル(RTS)が指定時間アクティブにならなければスリープする
suspend=サスペンドの有効無効
psm=PSM の設定
edrx=eDRX の設定
auto_dial=true
register_check_interval=監視周期(秒)
```

図 6.10 /etc/aiot-modem-control/startup.conf のフォーマット

- apn: APN 名設定

  APN を設定します。ご利用になられる SIM カードの提供事業者から指定された APN 名を、ダブルクオテーションで囲んで設定してください。

- auth_prot: ppp 認証方式設定

  PPP 認証方式を設定します。ご利用になられる SIM カードの提供事業者から指定された ppp 認証方式を、ダブルクオテーションで囲んで設定してください。"pap"、"chap" または "none" (認証不要) が設定可能です。ユーザー名とパスワードが不要な SIM の場合、 "none" を設定してください。

- username: ユーザー名設定

ユーザー名を設定します。ご利用になられる SIM カードの提供事業者から指定された ユーザー名を、ダブルクオテーションで囲んで設定してください。auth_prot が "none" の場合設定は不要です。

- password: パスワード設定

  パスワードを設定します。ご利用になられる SIM カードの提供事業者から指定された パスワードを、ダブルクオテーションで囲んで設定してください。auth_prot が "none" の場合設定は不要です。

- create_wvdial_conf: wvdial.conf 自動生成設定

  ppp 接続を実行するアプリケーション wvdial が参照する設定ファイル /etc/wvdial.conf を自動生成する場合、 true を設定してください。true 以外の文字列が設定された場合、 /etc/wvdial.conf の自動生成を行いません。自動生成されるファイルの内容は以下の通りです。これ以外の設定にしたい場合は、本パラメーターを無効にし、/etc/wvdial.conf を直接編集してご利用ください。

```
[Dialer Defaults]
Init1 = ATZ
Init2 = AT+CFUN=1
Init3 = AT+COPS?
Modem Type = Analog Modem
Dial Command = ATD
Baud = 460800
Modem = /dev/ttymxc1
New PPPD = yes
Dial Attemps = 0
ISDN = 0
APN = apn
Phone = *99***1#
Password = password の値
Username = username の値
Carrier Check = yes
Auto DNS = 1
Check Def Route = 1
Stupid Mode = yes
Check DNS = 1
```

図 6.11 自動生成される /etc/wvdial.conf

```
[Dialer Defaults]
Init1 = ATZ
Init2 = AT+CFUN=1
Init3 = AT+COPS?
Modem Type = Analog Modem
Dial Command = ATD
Baud = 460800
Modem = /dev/ttymxc1
New PPPD = yes
Dial Attemps = 0
ISDN = 0
APN = apn
Phone = *99***1#
Password = ''
Username = ''
Carrier Check = yes
Auto DNS = 1
```

```
Check Def Route = 1
Stupid Mode = yes
Check DNS = 1
```

図 6.12 自動生成される /etc/wvdial.conf ユーザー名とパスワードが空欄の場合

- operator_id: オペレーター ID 設定

  オペレーター ID を指定します。ローミング対応 SIM などオペレーター自動選択では接続できない場合、本パラメーターを設定することで接続できることがあります。 設定値は、SIM を提供している事象者に確認ください。 例として、 NTT DoCoMo の網を利用している SIM の場合 "44010" になります。 自動設定を指定する場合は、本パラメーターに "auto" を設定してください。 ダブルクォテーションで囲んで設定してください。

- fix_profile: プロファイル指定設定

  LTE モデムで使用するプロファイルを指定します。 通常は LTE モデムが自動でプロファイルを選択する為、設定する必要はありませんが、 ローミング対応 SIM など上手く接続出来ない場合に設定することで接続できることがあります。

表 6.3 fix_profile 設定可能パラメーター

設定値	概要
"docomojp"	NTT DoCoMo のプロファイルを使用する
"none"	自動選択

- sleep: Sleep 設定

  LTE モデムとのシリアルインタフェースが指定時間通信がない時に Sleep モードへ遷移します。 "disable" を設定した場合、この設定が無効となります。 以下の表に示す値以外を設定した場合、LTE モデムへの設定を行いません。

表 6.4 sleep 設定可能パラメーター

設定値	概要
100 … 5000	Sleep モードへ移行する時間(ミリ秒)
"disable"	無効

- suspend: Suspend 設定

  Suspend の有効・無効を設定します。

  LTE モデムの Suspend を有効にすると、LTE モデムは省電力・間欠動作機能のスリープモードに該当する動作となります。詳細は、「7.1.3. スリープモード」を参照ください。

表 6.5 suspend 設定可能パラメーター

設定値	概要
"enable"	有効
"disable"	無効

- psm: PSM 設定

  PSM(Power Saving Mode) の設定を行います。 "disable" を設定すると PSM が無効になります。有効にする場合、tau(periodic Tracking Area Update) と act-time(ACTive TIME) の組み合わせを設定します。

表 6.6 psm 設定可能パラメーター

設定値	概要
tau,act-time	有効 例: psm=3m,1m
"disable"	無効

表 6.7 psm tau,act-time 設定可能パラメーター

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

- edrx: eDRX 設定

  eDRX の設定を行います。 "disable" を設定すると eDRX が無効になります。 有効にする場合、pcl(Paging Cycle Length) と ptw(Paging Time Window eDRX) の組み合わせを設定します。

表 6.8 edrx 設定可能パラメーター

設定値	概要
pcl,ptw	有効 (例: edrx=20.48,5.12)
"disable"	無効

表 6.9 edrx pcl,ptw 設定可能パラメーター

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

- auto_dial: 自動 ppp 接続設定

  このパラメーターに `true` が設定されていると、ppp コネクションを自動確立し、確立出来た後 IP アドレスが付与されると IP 通信が可能となります。 `true` 以外の値が設定されている場合、何もしません。 作成されたアプリケーションのタイミングで ppp 接続を行いたい場合などは、 `true` を設定しないでください。

- register_check_interval: ネットワーク登録確認周期設定

  指定された間隔(秒)でネットワーク登録を確認します。 suspend が "enable" に設定されている場合、確認を実行しません。 また、ネットワーク登録が確認された後、auto_dial の設定が true であれば、自動的に ppp 接続を試みます。 確認周期は 30 秒から 300 秒の間で設定可能です。

表 6.10 register_check_interval 設定可能パラメーター

設定値	概要
30 … 300	ネットワーク登録確認周期(秒)

設定ファイルを更新した場合

LTE 設定ファイル (/etc/aiot-modem-control/startup.conf) を更新した場合、設定を適用するには Armadillo-IoT ゲートウェイ A6 の再起動ま

たは LTE モデムを制御するサービスの再起動が必要です。必要に応じて実施してください。

```
[armadillo ~]# reboot # Armadillo-IoT ゲートウェイ A6 を再起動する
```

```
[armadillo ~]# systemctl restart aiot-modem-controld.service # LTE モデムを制御するサービスの再起動を行う
```

設定ファイルで自動 ppp 接続設定を有効 (auto_dial=true) にしていない場合、「図 6.15. aiot-modem-control dial コマンド」を使用して ppp 接続を確立する必要があります。

6.2.4.3. LTE の接続を確認する

アットマークテクノの Web サーバーと PING 通信を行います。VPN 接続を利用するなどインターネットに接続できない場合は、ネットワーク内の通信機器に読み替えてください。

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

6.2.4.4. コマンドによる LTE モデム制御

コマンド aiot-modem-control を使用することで LTE モデムの制御を実行できます。 コマンド書式とコマンド一覧を以下に示します。 LTE モデムを制御するシリアルインタフェースが 1 本のため、wvdial 実行中または他のアプリケーションなどでシリアル使用中は実行できないコマンドがあります。

```
[armadillo ~]# aiot-modem-control [コマンド] [パラメーター]
```

図 6.13 aiot-modem-control コマンド書式

表 6.11 aiot-modem-control コマンド一覧

command	概要	ppp コネクション確立中実行可否
set-apn	APN 設定	不可
dial	ppp コネクション確立	不可
hangup	ppp コネクション切断	可
get-phone-number	電話番号取得	不可
get-signal-quality	電波品質(RSRQ)取得	不可
wwan-force-restart	LTE モデム再起動	可
poweron	LTE モデム 電源オン	可
poweroff	LTE モデム 電源オフ	可
set-psm	PSM 設定	不可
set-edrx	eDRX 設定	不可
set-sleep	Sleep 設定	不可
set-suspend	Suspend 設定	不可
activate	LTE モデム活性化	可
deactivate	LTE モデム非活性化	可
send-at	AT コマンド送信	不可
send-at-echo	AT コマンド送信(応答表示)	不可

command	概要	ppp コネクション確立中実行可否
status	簡易状態表示	可
help	コマンドヘルプ表示	可

- set-apn: APN 設定

  パラメーターに指定した APN を LTE モデムに設定します。 APN はダブルクオテーションで囲んでください。

```
[armadillo ~]# aiot-modem-control set-apn "exsample.com"
```

図 6.14 aiot-modem-control set-apn コマンドの例

- dial: ppp コネクション確立

  wvdial を使用して ppp コネクションの確立を試みます。 既に wvdial が動作しているときは何もせずコマンドを終了します。

```
[armadillo ~]# aiot-modem-control dial
```

図 6.15 aiot-modem-control dial コマンド

- hangup: ppp コネクション切断

  wvdial を停止して ppp コネクションを切断します。 既に wvdial が動作していないときは何もせずコマンドを終了します。

```
[armadillo ~]# aiot-modem-control hangup
```

図 6.16 aiot-modem-control hangup コマンド

切断後、 /etc/aiot-modem-control/startup.conf で auto_dial に true を設定、かつ register_check_interval に有効な値を設定している場合、次の確認周期でネットワーク登録が出来ている場合、自動的に ppp の接続を行います。

- get-phone-number: 電話番号取得

  SIM カードに設定されている電話番号を取得します。

```
[armadillo ~]# aiot-modem-control get-phone-number
123456789012
```

図 6.17 aiot-modem-control get-phone-number コマンド

- get-signal-quality: 電波品質(RSRQ)取得

  現在の電波品質(RSRQ: Reference Signal Received Quality)を取得します。 Sleep 中などは 255 になります。

```
[armadillo ~]# aiot-modem-control get-signal-quality
21
```

図 6.18 aiot-modem-control get-signal-quality コマンド

表 6.12 get-signal-quality 戻り値の意味

戻り値	意味	戻り値	意味
0	-19.5 dB 未満	18	-11 dB 以上 -10.5 dB 未満
1	-19.5 dB 以上 -19 dB 未満	19	-10.5 dB 以上 -10 dB 未満
2	-19 dB 以上 -18.5 dB 未満	20	-10 dB 以上 - 9.5 dB 未満
3	-18.5 dB 以上 -18 dB 未満	21	- 9.5 dB 以上 - 9 dB 未満
4	-18 dB 以上 -17.5 dB 未満	22	- 9 dB 以上 - 8.5 dB 未満
5	-17.5 dB 以上 -17 dB 未満	23	- 8.5 dB 以上 - 8 dB 未満
6	-17 dB 以上 -16.5 dB 未満	24	- 8 dB 以上 - 7.5 dB 未満
7	-16.5 dB 以上 -16 dB 未満	25	- 7.5 dB 以上 - 7 dB 未満
8	-16 dB 以上 -15.5 dB 未満	26	- 7 dB 以上 - 6.5 dB 未満
9	-15.5 dB 以上 -15 dB 未満	27	- 6.5 dB 以上 - 6 dB 未満
10	-15 dB 以上 -14.5 dB 未満	28	- 6 dB 以上 - 5.5 dB 未満
11	-14.5 dB 以上 -14 dB 未満	29	- 5.5 dB 以上 - 5 dB 未満
12	-14 dB 以上 -13.5 dB 未満	30	- 5 dB 以上 - 4.5 dB 未満
13	-13.5 dB 以上 -13 dB 未満	31	- 4.5 dB 以上 - 4 dB 未満
14	-13 dB 以上 -12.5 dB 未満	32	- 4 dB 以上 - 3.5 dB 未満
15	-12.5 dB 以上 -12 dB 未満	33	- 3.5 dB 以上 - 3 dB 未満
16	-12 dB 以上 -11.5 dB 未満	34	- 3 dB 以上
17	-11.5 dB 以上 -11 dB 未満	255	不明、計測不能

- wwan-force-restart :LTE モデム再起動

  LTE モデムを再起動します。 再起動後、 /etc/aiot-modem-control/startup.conf に記載された設定を実行します。 /etc/aiot-modem-control/startup.conf の設定によっては、コマンドが終了するまでに時間がかかります。

```
[armadillo ~]# aiot-modem-control wwan-force-restart
```

図 6.19 aiot-modem-control wwan-force-restart コマンド

- poweron: LTE モデム 電源オン

LTE モデムを起動します。 起動後、 /etc/aiot-modem-control/startup.conf に記載された設定を実行します。 /etc/aiot-modem-control/startup.conf の設定によっては、コマンドが終了するまでに時間がかかります。 既に起動している場合、何もせずにコマンドを終了します。

```
[armadillo ~]# aiot-modem-control poweron
```

図 6.20 aiot-modem-control poweron コマンド

- poweroff: LTE モデム 電源オフ

  LTE モデムの電源をオフします。 既にオフされている場合、何もせずにコマンドを終了します。

```
[armadillo ~]# aiot-modem-control poweroff
```

図 6.21 aiot-modem-control poweroff コマンド

- set-psm: PSM 設定

  PSM の設定を行います。 パラメーターに enable を指定した場合、 その後ろに tau(periodic Tracking Area Update) と act-time(ACTive TIME) の順に値を指定します。 disable を指定すると、 PSM が無効になります。 default を指定すると、 PSM が無効になり、tau と act-time が初期値に設定されます。 初期値は、 tau が 3 分、 act-time が 1 分です。

```
[armadillo ~]# aiot-modem-control set-psm enable 3m 1m
```

図 6.22 aiot-modem-control set-psm enable コマンド

表 6.13 set-psm tau act-time 設定可能パラメーター

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

```
[armadillo ~]# aiot-modem-control set-psm disable
```

図 6.23 aiot-modem-control set-psm disable コマンド

```
[armadillo ~]# aiot-modem-control set-psm default
```

図 6.24 aiot-modem-control set-psm default コマンド

- set-edrx: eDRX 設定

  eDRX の設定を行います。 パラメーターに enable を指定した場合、 その後ろに pcl(Paging Cycle Length) と ptw(Paging Time Window eDRX) の値を指定します。 disable を指定すると、 eDRX が無効になります。 default を指定すると、 eDRX が無効になり、pcl と ptw が初期値に設定されます。 初期値は、 pcl が 20.48 秒、 ptw が 5.12 秒です。

```
[armadillo ~]# aiot-modem-control set-edrx enable 20.48 5.12
```

図 6.25 aiot-modem-control set-edrx enable コマンド

表 6.14 set-edrx pcl ptw 設定可能パラメーター

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

```
[armadillo ~]# aiot-modem-control set-edrx disable
```

図 6.26 aiot-modem-control set-edrx disable コマンド

```
[armadillo ~]# aiot-modem-control set-edrx default
```

図 6.27 aiot-modem-control set-edrx default コマンド

- set-sleep: Sleep 設定

  パラメーターに enable を指定した場合、その後ろに LTE モデムとのシリアルインタフェースが指定時間通信がない時に Sleep モードへ遷移する時間(ミリ秒)を指定します。100 ミリ秒から 5000 ミリ秒の間の値を指定可能です。 disable を設定した場合、この設定が無効となります。

```
[armadillo ~]# aiot-modem-control set-sleep enable 5000
```

図 6.28 aiot-modem-control set-sleep enable コマンド例

```
[armadillo ~]# aiot-modem-control set-sleep disable
```

図 6.29 aiot-modem-control set-sleep disable コマンド

- set-suspend: Suspend 設定

  LTE モデムの Suspend 設定を行います。enable または disable を指定可能です。

  LTE モデムの Suspend を有効にすると、LTE モデムは省電力・間欠動作機能のスリープモードに該当する動作となります。詳細は、「7.1.3. スリープモード」を参照ください。

```
[armadillo ~]# aiot-modem-control set-suspend enable
```

図 6.30 aiot-modem-control set-suspend enable コマンド

```
[armadillo ~]# aiot-modem-control set-suspend disable
```

図 6.31 aiot-modem-control set-suspend disable コマンド

- activate: LTE モデム活性化

  LTE モデムを suspned 状態へ移行させます。 suspend を有効に設定しているときのみ有効なコマンドです。 データー通信中など suspend 状態へ即座に移行出来ないこともあります。

```
[armadillo ~]# aiot-modem-control activate
```

図 6.32 aiot-modem-control activate コマンド

- deactivate: LTE モデム非活性化

  LTE モデムを suspned 状態から復帰させます。 suspend を有効に設定しているときのみ有効なコマンドです。

```
[armadillo ~]# aiot-modem-control deactivate
```

図 6.33 aiot-modem-control deactivate コマンド

- send-at: AT コマンド送信

  パラメーターに指定した AT コマンドを送信します。 AT コマンドはダブルクオテーションで囲んでください。

```
[armadillo ~]# aiot-modem-control send-at "AT"
```

図 6.34 aiot-modem-control send-at の例

- send-at-echo: AT コマンド送信(応答表示)

  パラメーターに指定した AT コマンドを送信します。 LTE モデムからの応答が標準出力に表示されます。 AT コマンドはダブルクオテーションで囲んでください。

```
[armadillo ~]# aiot-modem-control send-at-echo "AT"
AT
OK
```

図 6.35 aiot-modem-control send-at-echo の例

- status: 簡易状態表示

  LTE モデムの状態を表示します。 最新の状態ではなく、それぞれ最後に取得できた値のキャッシュになります。 特に rsrq と rsrp は ppp コネクション確立中は更新できませんので、最新の値にはなりません。

```
[armadillo ~]# aiot-modem-control status
    manufacturer: Cinterion
           model: EMS31-J
        revision: REVISION 04.014
     app revison: A-REVISION 01.000.17
         port(s): /dev/ttymxc1
          status: registerd
    phone number: 01234567890
            imei: 012345678900123
signale quality
            rsrq: 19 [-10.5 dB <= rsrq < -10.0 dB]
            rsrp: 44 [-97 dBm <= rsrq < -96 dBm]
        sim slot: 1
     operator id: "12345"
```

図 6.36 aiot-modem-control status コマンド

表 6.15 status 表示内容

項目	値の説明
manufacturer	LTE モデム製造者
model	LTE モデムのモデル
revision	REVISION 04.014
app revison	LTE モデムも F/W バージョン
port(s)	使用しているシリアルポート
status	LTE モデムの動作状態 nosim: SIM カード未挿入 not registered: ネットワーク未登録 registerd: ネットワーク登録済み dial: ppp コネクション確立中・済み
phone number	SIM カードの電話番号
imei	LTE モデムの IMEI(International Mobile Equipment Identifier)
rsrq	電波品質(Reference Signal Received Quality)
rsrp	電波強度(Reference Signal Received Power)
sim slot	使用している SIM スロット
operator id	ネットワーク登録しているオペレーター ID

・ aiot-modem-control 処理結果

aiot-modem-control の戻り値と意味を「表 6.16. aiot-modem-control 処理結果」に示します。

表 6.16 aiot-modem-control 処理結果

戻り値	エラーメッセージ	意味
0		正常終了
2	on Calling	wvdial 動作中
3	Command Failed	コマンド失敗
4	Socket Error	通信エラー
5	Command Not Found	指定のコマンドが存在しない
6	Memory Error	メモリ不足
7	Serial In Use	他アプリシリアル使用中
8	Serial Error	シリアルエラー
9	Timeout	タイムアウト
10	Parameter Error	パラメーターエラー

6.2.4.5. LTE による時刻同期

ネットワーク登録が完了した時点で、自動的に LTE の NITZ(Network Identity and Time Zone) による日時同期を実行します。

手動での日時の設定・参照に関しては、「6.6. RTC」 を参照ください。

6.2.5. ファイアーウォール

Armadillo-IoT ゲートウェイ A6 では、ファイアーウォールの実現に iptables を使用しています。工場出荷状態の Armadillo-IoT ゲートウェイ A6 では、開発時の利便性のために、すべての通信(送信・受信・転送)を許可する設定になっています。

Armadillo を製品として運用する際には、最低限、踏み台として利用されない程度のファイアーウォールを設定しておかなければいけません。

ここでは、iptables のポリシーの設定と、Armadillo がネットワークに接続される前に自動的に設定を適用する方法を紹介します。

6.2.5.1. iptables のポリシーの設定

送信はすべて許可、受信・転送はすべて破棄するように設定します。

```
[armadillo ~]# iptables --policy INPUT DROP
[armadillo ~]# iptables --policy FORWARD DROP
[armadillo ~]# iptables --policy OUTPUT ACCEPT
```

iptables のポリシーの設定で受信と転送を許可する

iptables のポリシーの設定をもとに戻す(受信・転送を許可する)には次のコマンドを実行します。

```
[armadillo ~]# iptables --policy INPUT ACCEPT
[armadillo ~]# iptables --policy FORWARD ACCEPT
```

6.2.5.2. lo (ローカルループバックインターフェース)の許可

```
[armadillo ~]# iptables --append INPUT --in-interface lo --jump ACCEPT
```

6.2.5.3. iptables の設定確認

設定されている内容を参照するには、次のコマンドを実行します。

```
[armadillo ~]# iptables --list --verbose
Chain INPUT (policy DROP 1 packets, 72 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

図 6.37 iptables 設定確認

「図 6.37. iptables 設定確認」の設定では受信パケットが全て破棄されます。これが最も安全で最小の設定です。

この設定をベースに、SSH や HTTPS などの通信プロトコルから利用するものだけを許可していくことをおすすめします。

6.2.5.4. iptables の設定を保存し自動的に適用する

ここまでの手順で行った iptables の設定は、Armadillo を再起動すると失われてしまいます。そこで、Armadillo-IoT ゲートウェイ A6 では iptables-persistent パッケージを利用して、あらかじめ保存しておいた設定を自動的に適用します。

iptables の設定を保存するには、次のコマンドを実行します。

```
[armadillo ~]# iptables-save > /etc/iptables/rules.v4
```

図 6.38 iptables 設定保存

iptables-persistent がインストールされていなかった場合は、/etc/iptables/ ディレクトリが存在しないため、「図 6.38. iptables 設定保存」が失敗します。次のコマンドを実行して iptables-persistent をインストールし、再度 「図 6.38. iptables 設定保存」 を行ってください。

```
[armadillo ~]# iptables --policy INPUT ACCEPT
[armadillo ~]# apt-get update && apt-get install iptables-persistent
```

図 6.39 iptables のポリシー設定(受信許可)と iptables-persistent のインストール

次回起動時から、Armadillo がネットワークに接続される前のタイミングで、自動的に iptables の設定が適用されます。

6.3. ストレージ

Armadillo-IoT ゲートウェイ A6 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 6.17 ストレージデバイス

デバイス種類	ディスクデバイス	先頭パーティション	インターフェース
オンボード eMMC	/dev/mmcblk0	/dev/mmcblk0p1	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp2	なし	オンボード
オンボード eMMC (GPP)	/dev/mmcblk0gp3	なし	オンボード
SD/SDHC/SDXC カード	/dev/mmcblk1	/dev/mmcblk1p1	microSD スロット(CON1)
USB メモリ	/dev/sd* [a]	/dev/sd*1	USB ホストインターフェース (CON5)

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

GPP(General Purpose Partition)について

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

Armadillo-IoT ゲートウェイ A6 では、8 MiB の GPP を 4 つ作成しています。各領域の用途を「表 6.18. eMMC の GPP の用途」に示します。

表 6.18 eMMC の GPP の用途

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

GPP のユーザー領域を使用する例を「22.5. eMMC の GPP(General Purpose Partition) を利用する」に記載しています。

6.3.1. ストレージの使用方法

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

SDXC/microSDXC カードを使用する場合は、事前に「6.3.2. 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 SDXC/microSDXC カードは exFAT ファイルシステムでフォーマットされています。

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

mount コマンドの典型的なフォーマットは、次の通りです。

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

図 6.40 mount コマンド書式

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

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

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

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

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

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

図 6.41 ストレージのマウント

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

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

図 6.42 ストレージのアンマウント

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

パーティション構成を変更したい場合、 fdisk コマンドを使用します。 fdisk コマンドの使用例として、一つのパーティションで構成されている microSD カードのパーティションを、2 つに分割する例を「図 6.43. fdisk コマンドによるパーティション変更」に示します。一度、既存のパーティションを削除してから、新たにプライマリパーティションを二つ作成しています。先頭のパーティションには 100MByte、二つめのパーティションに残りの容量を割り当てています。先頭のパーティションは /dev/mmcblk1p1 、二つめは /dev/mmcblk1p2 となります。 fdisk コマンドの詳細な使い方は、man ページ等を参照してください。

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

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


Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

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

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

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

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

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

図 6.43 fdisk コマンドによるパーティション変更

FAT32 ファイルシステムでストレージデバイスをフォーマットするには、 mkfs.vfat コマンドを使用します。また、EXT2 や EXT3、 EXT4 ファイルシステムでフォーマットするには、mkfs.ext2 や

mkfs.ext3 、 mkfs.ext4 コマンドを使用します。microSD カードのパーティション 1 を EXT4 ファイルシステムでフォーマットするコマンド例を、次に示します

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

図 6.44 EXT4 ファイルシステムの構築

6.4. LED

Armadillo-IoT ゲートウェイ A6 の LED は GPIO で接続されているため、ソフトウェアで制御することができます。

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

表 6.19 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/red/	ユーザー LED 赤	default-on
/sys/class/leds/green/	ユーザー LED 緑	default-on

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

6.4.1. LED を点灯/消灯する

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

brightness に 0 以外の値を書き込むと LED が点灯します。

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

図 6.45 LED を点灯させる

Armadillo-IoT ゲートウェイ A6 の LED には輝度制御の機能がないため、0(消灯)、1〜255(点灯)の 2 つの状態のみ指定することができます。

brightness に 0 を書き込むと LED が消灯します。

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

図 6.46 LED を消灯させる

brightness を読み出すと LED の状態が取得できます。

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

図 6.47 LED の状態を表示する

6.4.2. トリガを使用する

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

表 6.20 LED トリガの種類

設定	説明
none	トリガを設定しません
mmc0	eMMC のアクセスランプにします
mmc1	microSD スロットのアクセスランプにします
timer	任意のタイミングで点灯/消灯を行います。この設定にすることにより、LED クラスディレクトリ以下に delay_on, delay_off ファイルが出現し、それぞれ点灯時間, 消灯時間をミリ秒単位で指定します
heartbeat	心拍のように点灯/消灯を行います
default-on	主に Linux カーネルから使用します。LED が点灯します

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

```
[armadillo ~]# cat /sys/class/leds/[LED]/trigger
[none] rc-feedback kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shif
tlock kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock kbd-c
trlllock kbd-ctrlrlock mmc0 mmc1 timer oneshot heartbeat default-on
```

図 6.48 対応している LED トリガを表示

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

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

図 6.49 LED のトリガに timer を指定する

6.5. ユーザースイッチ

Armadillo-IoT ゲートウェイ A6 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 6.21 インプットデバイスファイルとイベントコード

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

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

6.5.1. イベントを確認する

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

```
[armadillo ~]# evtest /dev/input/event0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 28 (KEY_ENTER)
Properties:
Testing ... (interrupt to exit)
Event: time 1523249446.289965, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0 ❶
Event: time 1523249446.289965, ---------- SYN_REPORT ------------
Event: time 1523249446.349969, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1 ❷
Event: time 1523249446.349969, ---------- SYN_REPORT ------------
```

図 6.50 ユーザースイッチ: イベントの確認

❶ SW1 のボタン **プッシュ** イベントを検出したときの表示

❷ SW1 のボタン **リリース** イベントを検出したときの表示

6.6. RTC

Armadillo-IoT ゲートウェイ A6 は、サブボード上に温度補償高精度リアルタイムクロック(RTC) 機能を利用しています。

6.6.1. RTC に時刻を設定する

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

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

表 6.22 時刻フォーマットのフィールド

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

2021 年 3 月 2 日 12 時 34 分 56 秒に設定する例を次に示します。

```
[armadillo ~]# date
Sat Jan  1 09:00:00 JST 2000
[armadillo ~]# systemctl stop systemd-timesyncd.service
[armadillo ~]# date 030212342018.56
Fri Mar  2 12:34:56 JST 2021
[armadillo ~]# date
Fri Mar  2 12:34:57 JST 2021
```

図 6.51 システムクロックを設定

Armadillo-IoT ゲートウェイ A6 では、標準で systemd-timesyncd.service が動作しています。systemd-timesyncd.service は、自身が正しいと考えている時刻となるように、自動でシステムクロックおよびハードウェアクロックを設定します。

そのため、date コマンドで過去の時刻を設定しても、すぐに systemd-timesyncd.service によって変更前の正しい時刻に再設定されてしまいます。これを避けるため、システムクロックを設定する前に systemd-timesyncd.service を停止する必要があります。

```
[armadillo ~]# systemctl stop systemd-timesyncd.service
```

Armadillo-IoT ゲートウェイ A6 のタイムゾーンはデフォルトで JST に設定されています。timedatectl コマンドで、これを変更することができます。

タイムゾーンを UTC に変更するには次のようにコマンドを実行します。

```
root@armadillo:~# date
Tue Feb 12 10:32:07 JST 2019
root@armadillo:~# timedatectl set-timezone Etc/UTC
root@armadillo:~# date
Tue Feb 12 01:32:10 UTC 2019
```

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

```
[armadillo ~]# hwclock ❶
2000-01-01 00:00:00.000000+0900
[armadillo ~]# hwclock --utc --systohc ❷
[armadillo ~]# hwclock --utc ❸
2018-03-02 12:35:08.213911+0900
```

図 6.52 ハードウェアクロックを設定

❶ 現在のハードウェアクロックを表示します。

❷ ハードウェアクロックを協定世界時(UTC)で設定します。

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

6.7. GPIO

Armadillo-IoT ゲートウェイ A6 の GPIO は、generic GPIO として実装されています。GPIO クラスディレクトリ以下のファイルによって GPIO の制御を行うことができます。

「表 6.23. サブユニット CON3 ピンと GPIO 番号の対応」の各ピンは GPIO として利用することができます。

表 6.23 サブユニット CON3 ピンと GPIO 番号の対応

ピン番号	ピン名	GPIO 番号
9	GPIO1_IO16	16
11	GPIO1_IO17	17
7	GPIO4_IO06	102
8	GPIO4_IO07	103
5	GPIO4_IO08	104
6	GPIO4_IO09	105

「表 6.23. サブユニット CON3 ピンと GPIO 番号の対応」の 9, 11, 5〜8 ピンは初期出荷状態では GPIO として利用することができません。これらのピンを GPIO として利用する場合は、 at-dtweb を用います。

at-dtweb の利用方法については「22.3. Device Tree をカスタマイズする」を参照してください。

GPIO 番号は次の式より導くことができます。

```
GPIOx_IOy -> (x - 1) * 32 + y
```

例えば、GPIO4_IO8(サブユニット CON3 5 ピン)の場合は、以下のようになります。

```
(4 - 1) * 32 + 8 = 104
```

6.7.1. GPIO クラスディレクトリを作成する

GPIO を利用するには、まず GPIO ディレクトリを作成する必要があります。

GPIO クラスディレクトリは、 /sys/class/gpio/export に GPIO 番号を書き込むことによって、作成することができます。

```
[armadillo ~]# echo 17 > /sys/class/gpio/export
[armadillo ~]# ls /sys/class/gpio/gpio17/
active_low device direction edge subsystem uevent value
```

図 6.53 GPIO クラスディレクトリを作成する

以降の説明では、任意の GPIO を示す GPIO クラスディレクトリを "/sys/class/gpio/[GPIO]" のように表記します。

作成済みの GPIO クラスディレクトリを削除するには、 /sys/class/gpio/unexport に GPIO 番号を書き込みます。

```
[armadillo ~]# echo 17 > /sys/class/gpio/unexport
[armadillo ~]# ls /sys/class/gpio/gpio17/
ls: cannot access '/sys/class/gpio/gpio17/': No such file or directory
```

6.7.2. 入出力方向を変更する

GPIO ディレクトリ以下の direction ファイルへ値を書き込むことによって、入出力方向を変更することができます。 direction に書き込む有効な値を次に示します。

表 6.24 direction の設定

設定	説明
high	入出力方向を OUTPUT に設定します。出力レベルの取得/設定を行うことができます。出力レベルは HIGH レベルになります。
out	入出力方向を OUTPUT に設定します。出力レベルの取得/設定を行うことができます。出力レベルは LOW レベルになります。
low	out を設定した場合と同じです。
in	入出力方向を INPUT に設定します。入力レベルの取得を行うことができますが設定はできません。

```
[armadillo ~]# echo in > /sys/class/gpio/[GPIO]/direction
```

図 6.54 GPIO の入出力方向を設定する(INPUT に設定)

```
[armadillo ~]# echo out > /sys/class/gpio/[GPIO]/direction
```

図 6.55 GPIO の入出力方向を設定する(OUTPUT に設定)

6.7.3. 入力レベルを取得する

GPIO ディレクトリ以下の value ファイルから値を読み出すことによって、入力レベルを取得することができます。"0"は LOW レベル、"1"は HIGH レベルを表わします。入力レベルの取得は入出力方向が INPUT, OUTPUT のどちらでも行うことができます。

```
[armadillo ~]# cat /sys/class/gpio/[GPIO]/value
0
```

図 6.56 GPIO の入力レベルを取得する

6.7.4. 出力レベルを設定する

GPIO ディレクトリ以下の value ファイルへ値を書き込むことによって、出力レベルを設定することができます。"0"は LOW レベル、"0"以外は HIGH レベルを表わします。出力レベルの設定は入出力方向が OUTPUT でなければ行うことはできません。

```
[armadillo ~]# echo 1 > /sys/class/gpio/[GPIO]/value
```

図 6.57 GPIO の出力レベルを設定する

7. 省電力・間欠動作機能

本章では、Armadillo-IoT ゲートウェイ A6 の 省電力・間欠動作機能や動作モード、状態遷移について説明します。

7.1. 動作モードと状態遷移図

Armadillo-IoT ゲートウェイ A6 の動作モードと状態遷移を次に示します。

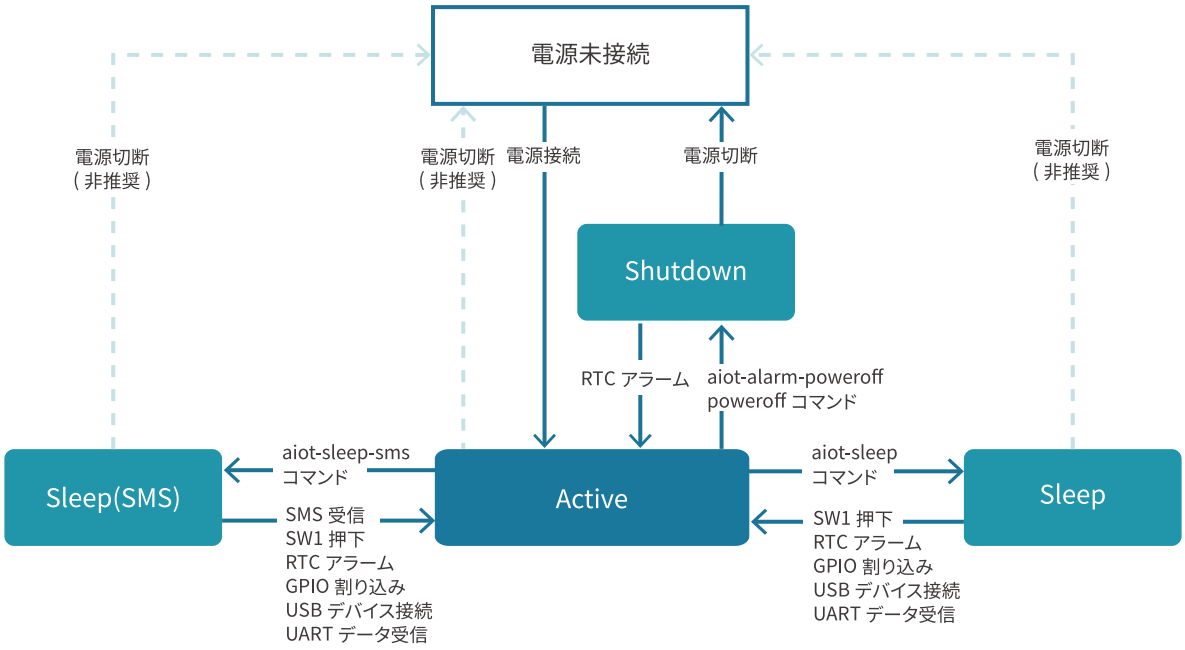


図 7.1 状態遷移図

7.1.1. アクティブモード

「CPU:動作」、「LTE-M モジュール:動作」 状態のモードです。

Armadillo-IoT ゲートウェイ A6 の電源投入後 Linux カーネルが起動し、まずはアクティブモードに遷移します。

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

7.1.2. シャットダウンモード

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

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

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

7.1.3. スリープモード

「CPU:待機」、「LTE-M:停止」 状態のモードです。

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

7.1.4. スリープ(SMS 起床可能)モード

「CPU:待機」、「LTE-M:待機」 状態のモードです。

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

7.2. シャットダウンモードへの遷移と起床

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

7.2.1. poweroff コマンド

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

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

※省略

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

7.2.2. aiot-alarm-poweroff コマンド

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

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

図 7.2 aiot-alarm-poweroff コマンド書式

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

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

現在時刻からの経過秒数は 180 秒以上を指定する必要があります。

7.3. スリープモードへの遷移と起床

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

7.3.1. RTC アラーム割り込み以外での起床

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

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

図 7.3 aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込み以外での起床のとき）

表 7.1 aiot-modem-control TRIGGER 一覧

TRIGGER	説明
usb-lower	メインユニット CON5(USB ホストインターフェース)の上段に USB デバイスを挿抜したとき
usb-upper	メインユニット CON5(USB ホストインターフェース)の下段に USB デバイスを挿抜したとき
uart3	メインユニット CON3、CON4(シリアルインターフェース / dev/ttymxc2)にデータ受信があったとき

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

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

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

※ コンソールに入力
```

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

7.3.2. RTC アラーム割り込みでの起床

RTC アラーム割り込みでの起床を行う場合、パラメーター設定が異なります。

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

図 7.4 aiot-set-wake-trigger コマンド書式 （RTC アラーム割り込みでの起床の場合）

現在時刻からの経過秒数 は 60 秒以上を指定する必要があります。

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

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

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

※ 約 300 秒待つ

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

7.3.3. 起床要因のクリア

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

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

7.4. スリープ(SMS 起床可能)モードへの遷移と起床

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

aiot-sleep-sms コマンドの実行例を次に示します。

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

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

OK

AT+CMGD=1,4
OK

AT+CMGL=”ALL”
OK

[ 3508.609638] PM: suspend entry (deep)

^SIND: message,1,0

OK

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

※ SMS 受信

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

7.5. スリープモードへの遷移・起床時にスクリプトを実行する

/lib/systemd/system-sleep/ に以下のようなスクリプトを配置することで、スリープモードへの遷移時や、スリープモードからの起床時に独自の処理を行うことが可能です。

```
#!/bin/sh
case $1/$2 in
    pre/*)
```

```
        # スリープモードへの遷移時に実行する処理
    ;;
    post/*)
        # スリープモードからの起床時に実行する処理
    ;;
esac
```

図 7.5 スリープモードの遷移・起床時に実行されるスクリプトの例

スクリプト名を example.sh とする場合、以下のように配置してください。

```
[armadillo ~]# cp example.sh /lib/systemd/system-sleep/
```

この機能を利用するには、ems31-utils のバージョンが 1.0.3 以降である必要があります。

```
[armadillo ~]# dpkg -l |grep ems31-utils
ii  ems31-utils                  1.0.3                       armhf
Utilities for Thales EMS31 on Armadillo board
```

8. Linux カーネル仕様

本章では、工場出荷状態の Armadillo-IoT ゲートウェイ A6 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

工場出荷時の Armadillo-IoT ゲートウェイ A6 に書き込まれている Linux カーネルは、デフォルトコンフィギュレーションが適用されています。Armadillo-IoT ゲートウェイ A6 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-v4.14-at[VERSION].tar.gz)に含まれる arch/arm/configs/armadillo-640_defconfig です。

armadillo-640_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
VMSPLIT_3G	3G/1G user/kernel split
AEABI	Use the ARM EABI to compile the kernel
COMPACTION	Allow for memory compaction
MIGRATION	Page migration

8.2. デフォルト起動オプション

工場出荷状態の Armadillo-IoT ゲートウェイ A6 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=ttymxc2	起動ログなどが出力されるイニシャルコンソールに ttymxc2 (CON3 / CON4)を指定します。
root=/dev/mmcblk0p2	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。

8.3. Linux ドライバ一覧

Armadillo-IoT ゲートウェイ A6 で利用することができるデバイスドライバについて説明します。各ドライバで利用しているソースコードのうち主要なファイルのパスや、コンフィギュレーションに必要な情報、及びデバイスファイルなどについて記載します。

8.3.1. Armadillo-IoT ゲートウェイ A6

Armadillo-IoT ゲートウェイ A6 のハードウェアの構成情報やピンのマルチプレクス情報、i.MX6ULL の初期化手順などが定義されています。

関連するソースコード
- arch/arm/mach-imx/
- arch/arm/boot/dts/armadillo-640.dts

- arch/arm/boot/dts/armadillo-iotg-a6.dts
- arch/arm/boot/dts/imx6ull.dtsi
- arch/arm/boot/dts/imx6ul.dtsi

カーネルコンフィギュレーション

```
System Type --->
  [*] Freescale i.MX family  --->                                  <ARCH_MXC>
    [*]   i.MX6 UltraLite support                                <SOC_IMX6UL>
```

8.3.2. UART

Armadillo-IoT ゲートウェイ A6 のシリアルは、i.MX6ULL の UART (Universal Asynchronous Receiver/Transmitter) を利用しています。Armadillo-IoT ゲートウェイ A6 の標準状態では、UART3 (CON3 / CON4) をコンソールとして利用しています。

フォーマット

- データビット長: 7 or 8 ビット
- ストップビット長: 1 or 2 ビット
- パリティ: 偶数 or 奇数 or なし
- フロー制御: CTS/RTS or XON/XOFF or なし
- 最大ボーレート:230.4kbps

関連するソースコード

- drivers/tty/n_null.c
- drivers/tty/n_tty.c
- drivers/tty/pty.c
- drivers/tty/tty_baudrate.c
- drivers/tty/tty_buffer.c
- drivers/tty/tty_io.c
- drivers/tty/tty_ioctl.c
- drivers/tty/tty_jobctrl.c
- drivers/tty/tty_ldisc.c
- drivers/tty/tty_ldsem.c
- drivers/tty/tty_mutex.c
- drivers/tty/tty_port.c
- drivers/tty/serial/earlycon.c
- drivers/tty/serial/serial_core.c
- drivers/tty/serial/serial_mctrl_gpio.c

- drivers/tty/serial/imx.c

Device Tree ドキュメント

- Documentation/devicetree/bindings/serial/fsl-imx-uart.txt
- Documentation/devicetree/bindings/serial/serial.txt

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
UART3	/dev/ttymxc2

カーネルコンフィギュレーション

```
Device Drivers  --->
  Character devices  --->
    [*] Enable TTY                                          <TTY>
        Serial drivers  --->
          [*] IMX serial port support                 <SERIAL_IMX>
          [*]   Console on IMX serial port     <SERIAL_IMX_CONSOLE>
```

8.3.3. Ethernet

Armadillo-IoT ゲートウェイ A6 の Ethernet (LAN) は、i.MX6ULL の ENET(10/100-Mbps Ethernet MAC)を利用しています。

機能

- 通信速度: 100Mbps (100BASE-TX), 10Mbps (10BASE-T)
- 通信モード: Full-Duplex (全二重), Half-Duplex (半二重)
- Auto Negotiation サポート
- キャリア検知サポート
- リンク検出サポート

関連するソースコード

- drivers/net/Space.c
- drivers/net/loopback.c
- drivers/net/ethernet/freescale/fec_main.c
- drivers/net/ethernet/freescale/fec_ptp.c
- drivers/net/phy/fixed_phy.c
- drivers/net/phy/mdio-boardinfo.c
- drivers/net/phy/mdio_bus.c
- drivers/net/phy/mdio_device.c
- drivers/net/phy/phy-core.c
- drivers/net/phy/phy.c
- drivers/net/phy/phy_device.c
- drivers/net/phy/smsc.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/net/fsl-fec.txt
- Documentation/devicetree/bindings/net/phy.txt

ネットワークデバイス
- eth0

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Network device support  --->                    <NETDEVICES>
    [*]   Ethernet driver support  --->                 <ETHERNET>
      [*]   Freescale devices                <NET_VENDOR_FREESCALE>
     [*]    FEC ethernet controller (of ColdFire and some i.MX CPUs)
                                                              <FEC>
    -*-   PHY Device support and infrastructure  --->      <PHYLIB>
      [*]   SMSC PHYs                                     <SMSC_PHY>
```

8.3.4. LTE

Armadillo-IoT ゲートウェイ A6 は LTE モデム EMS31-J を利用しています。 またネットワークコネクション確立に ppp を利用しています。

デバイス
- ppp0

関連するソースコード
- drivers/reset/reset-ems31.c
- drivers/net/ppp/ppp_generic.c
- drivers/net/ppp/ppp_async.c

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] Network device support  --->                    <NETDEVICES>
     [*]   PPP (point-to-point protocol) support      <PPP>
     [*]     PPP support for async serial ports       <PPP_ASYNC>
  -*- Reset Controller Support  --->                <RESET_CONTROLLER>
     [*]   GPIO-based Reset Driver for Tales EMS31    <RESET_EMS31>
```

8.3.5. SD ホスト

Armadillo-IoT ゲートウェイ A6 の SD ホストは、i.MX6ULL の uSDHC (Ultra Secured Digital Host Controller) を利用しています。Armadillo-IoT ゲートウェイ A6 では、オンボード microSD コネクタ (CON1) が uSDHC2 を利用しています。

機能
- カードタイプ: SD/SDHC/SDXC/SDIO
- バス幅: 1bit or 4bit
- スピードモード: Default Speed (24.75MHz), High Speed (49.5MHz)
- カードディテクトサポート

デバイスファイル
- /dev/mmcblk1

関連するソースコード
- drivers/mmc/core/

- drivers/mmc/host/sdhci-esdhc-imx.c
- drivers/mmc/host/sdhci-pltfm.c
- drivers/mmc/host/sdhci.c

Device Tree ドキュメント

- Documentation/devicetree/bindings/mmc/fsl-imx-esdhc.txt
- Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/regulator/fixed-regulator.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  [*] MMC/SD/SDIO card support  --->                              <MMC>
    [*]   MMC block device driver                          <MMC_BLOCK>
    (8)     Number of minors per block device    <MMC_BLOCK_MINORS>
          *** MMC/SD/SDIO Host Controller Drivers ***
    [*]   Secure Digital Host Controller Interface support
                                                           <MMC_SDHCI>
    [*]     SDHCI platform and OF driver helper   <MMC_SDHCI_PLTFM>
    [*]       SDHCI support for the Freescale eSDHC/uSDHC i.MX
controller support
                                                   <MMC_SDHCI_ESDHC_IMX>
```

8.3.6. USB ホスト

Armadillo-IoT ゲートウェイ A6 の USB ホストは、i.MX6ULL の USB-PHY (Universal Serial Bus 2.0 Integrated PHY) および USB (Universal Serial Bus Controller) を利用しています。Armadillo-IoT ゲートウェイ A6 では、USB ホストインターフェース (CON5) が OTG1 (下段) と OTG2 (上段) を利用しています。OTG2 は CON5 と CON9 と排他利用になっており、外部からの信号で切り替えることができるようになっています。詳しくは「18.6. メインユニット CON5(USB ホストインターフェース)」を参照してください。

機能

- Universal Serial Bus Specification Revision 2.0 準拠
- Enhanced Host Controller Interface (EHCI)準拠
- 転送レート: USB2.0 High-Speed (480Mbps), Full-Speed (12Mbps), Low-Speed (1.5Mbps)

デバイスファイル

- メモリデバイスの場合は、デバイスを認識した順番で/dev/sdN (N は 'a'からの連番)となります。
- I/O デバイスの場合は、ファンクションに応じたデバイスファイルとなります。

関連するソースコード

- drivers/usb/chipidea/
- drivers/usb/host/ehci-hcd.c
- drivers/usb/phy/of.c
- drivers/usb/phy/phy-generic.c
- drivers/usb/phy/phy.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/usb/ci-hdrc-usb2.txt
- Documentation/devicetree/bindings/usb/usbmisc-imx.txt
- Documentation/devicetree/bindings/regulator/fixed-regulator.txt

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>
    [*]     Support for Freescale i.MX on-chip EHCI USB controller
                                                        <USB_EHCI_MXC>
    [*]   ChipIdea Highspeed Dual Role Controller       <USB_CHIPIDEA>
    [*]     ChipIdea host controller                <USB_CHIPIDEA_HOST>
          USB Physical Layer drivers  --->
      [*] NOP USB Transceiver Driver                    <NOP_USB_XCEIV>
```

8.3.7. リアルタイムクロック

Armadillo-IoT ゲートウェイ A6 のリアルタイムクロックは、日本電波工業(NDK)製 NR3225SA が搭載されておりこれを利用しています。 NR3225SA は、「8.3.10. I2C」に示す I2C-GPIO1 (I2C ノード: 4-0032) に接続されています。 i.MX6ULL の RTC 機能も存在します。

機能
- アラーム割り込みサポート

デバイスファイル
- /dev/rtc
- /dev/rtc0
- /dev/rtc1

NR3225SA が /dev/rtc0 となるよう、Device Tree でエイリアスを設定しています。そのため、i.MX6ULL の RTC 機能は /dev/rtc1 となります。

エイリアスの設定は、 linux-4.14-at[VERSION]/arch/arm/boot/dts/armadillo-iotg-a6.dts で行っています。

関連するソースコード
- drivers/rtc/rtc-lib.c
- drivers/rtc/rtc-core.c
- drivers/rtc/hctosys.c
- drivers/rtc/systohc.c
- drivers/rtc/nvmem.c
- drivers/rtc/rtc-sysfs.c

- drivers/rtc/rtc-proc.c
- drivers/rtc/rtc-dev.c
- drivers/rtc/rtc-nr3225sa.c
- drivers/rtc/rtc-snvs.c

カーネルコンフィギュレーション

```
Device Drivers --->
  [*] Real Time Clock                                                 <RTC_CLASS>
    [*]   Set system time from RTC on startup and resume                        ⏎
<RTC_HCTOSYS>
    (rtc0)  RTC used to set the system time                                    ⏎
<RTC_HCTOSYS_DEVICE>
    [*]   Set the RTC time based on NTP synchronization                        ⏎
<RTC_SYSTOHC>
    (rtc0)  RTC used to synchronize NTP adjustment                             ⏎
<RTC_SYSTOHC_DEVICE>
    [*]   RTC non volatile storage support                                     ⏎
<RTC_NVMEM>
          *** 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>
          *** I2C RTC drivers ***
   [*]  NDK NR3225SA                                      <RTC_DRV_NR3225SA>
          *** on-CPU RTC drivers ***
    [*]   Freescale SNVS RTC support                                           ⏎
<RTC_DRV_SNVS>
```

アラーム割り込みは、デバイスファイル経由で利用することができます。

詳細な情報については、Linux カーネルのソースコードに含まれているドキュメント(Documentation/rtc.txt)やサンプルプログラム(tools/testing/selftests/timers/rtctest.c)を参照してください。

8.3.8. LED

Armadillo-IoT ゲートウェイ A6 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/red
- /sys/class/leds/green

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.c
- drivers/leds/trigger/

Device Tree ドキュメント
- Documentation/devicetree/bindings/leds/leds-gpio.txt

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 Default ON Trigger             <LEDS_TRIGGER_DEFAULT_ON>
```

8.3.9. ユーザースイッチ

Armadillo-IoT ゲートウェイ A6 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 8.3 キーコード

ユーザースイッチ	キーコード	イベントコード
SW1	KEY_ENTER	28

デバイスファイル
- /dev/input/event0 [1]

関連するソースコード
- drivers/input/evdev.c
- drivers/input/input-compat.c
- drivers/input/input.c
- drivers/input/keyboard/gpio_keys.c

Device Tree ドキュメント
- Documentation/devicetree/bindings/input/gpio-keys.txt

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>
    [*]   Event interface                                <INPUT_EVDEV>
          *** Input Device Drivers ***
    [*]   Keyboards  --->                              <INPUT_KEYBOARD>
      [*]   GPIO Buttons                                <KEYBOARD_GPIO>
```

[1] 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

8.3.10. I2C

Armadillo-IoT ゲートウェイ A6 の I2C インターフェースは、GPIO を利用した I2C バスドライバ(i2c-gpio)を利用します。また、i2c-gpio を利用することで、I2C バスを追加することができます。

Armadillo-IoT ゲートウェイ A6 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 8.4 I2C デバイス

I2C バス	I2C デバイス	
	アドレス	デバイス名
4(I2C-GPIO1)	0x32	NR3225SA
4(I2C-GPIO1)	0x48	SE050

Armadillo-IoT ゲートウェイ A6 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-4 (I2C-GPIO1)

関連するソースコード
- drivers/i2c/i2c-core.c
- drivers/i2c/i2c-boardinfo.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

Device Tree ドキュメント
- Documentation/devicetree/bindings/i2c/i2c-imx.txt
- Documentation/devicetree/bindings/i2c/i2c-gpio.txt

カーネルコンフィギュレーション

```
Device Drivers  --->
  I2C support  --->
    [*] I2C support                                              <I2C>
    [*]   Enable compatibility bits for old user-space  <I2C_COMPAT>
    [*]   I2C device interface                         <I2C_CHARDEV>
      I2C Algorithms  --->
        -*- I2C bit-banging interfaces                 <I2C_ALGOBIT>
      I2C Hardware Bus support  --->
        [*] GPIO-based bitbanging I2C                     <I2C_GPIO>
        [*] IMX I2C interface                              <I2C_IMX>
```

8.3.11. パワーマネジメント

Armadillo-IoT ゲートウェイ A6 のパワーマネジメント機能は、Linux の SPM(System Power Management)および DPM(Device Power Management)を利用しています。パワーマネジメント状態を省電力モードに遷移させることにより、Armadillo-IoT ゲートウェイ A6 の消費電力を抑えることができます。

パワーマネジメント状態を省電力モードに遷移させると、アプリケーションの実行は一時停止し、Linux カーネルはサスペンド状態となります。起床要因が発生すると、Linux カーネルのリジューム処理が行われた後、アプリケーションの実行を再開します。

sysfs ファイル　・ /sys/power/state

関連するソースコード　・ kernel/power/

カーネルコンフィギュレーション

```
Power management options  --->
  [*] Suspend to RAM and standby                                <SUSPEND>
  -*- Device power management core functionality                    <PM>
```

Armadillo-IoT ゲートウェイ A6 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 8.5 対応するパワーマネジメント状態

パワーマネジメント状態	文字列	説明
Suspend-to-RAM	mem	最も消費電力を抑えることができる
Power-On Suspend	standby	Suspend-to-RAM よりも短時間で復帰することができ、Suspend-to-Idle よりも消費電力を抑えることができる
Suspend-to-Idle	freeze	最も短時間で復帰することができる

起床要因として利用可能なデバイスは次の通りです。

UART2（LTE モデムと接続）　起床要因　データ受信

有効化

```
[armadillo ~]# echo enabled > /sys/bus/platform/
drivers/imx-uart/21e8000.serial/tty/ttymxc1/power/
wakeup
```

UART3（CON3 / CON4）　起床要因　データ受信

有効化

```
[armadillo ~]# echo enabled > /sys/bus/platform/
drivers/imx-uart/21ec000.serial/tty/ttymxc2/power/
wakeup
```

USB OTG1（下段）　起床要因　USB デバイスの挿抜

有効化

```
[armadillo ~]# echo enabled > /sys/bus/platform/
devices/2184000.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 OTG2（上段）	起床要因	USB デバイスの挿抜

	有効化	`[armadillo ~]# echo enabled > /sys/bus/platform/devices/2184200.usb/power/wakeup`
		`[armadillo ~]# echo enabled > /sys/bus/platform/drivers/ci_hdrc/ci_hdrc.1/power/wakeup`
		`[armadillo ~]# echo enabled > /sys/bus/platform/drivers/ci_hdrc/ci_hdrc.1/usb2/power/wakeup`

RTC(i.MX6ULL)	起床要因	アラーム割り込み
	有効化	`[armadillo ~]# echo enabled > /sys/bus/platform/devices/20cc000.snvs¥:snvs-rtc-lp/power/wakeup`
RTC(NR3225SA)	起床要因	アラーム割り込み
	有効化	`[armadillo ~]# echo enabled > /sys/devices/soc0/i2c-gpio1/i2c-4/4-0032/power/wakeup`

9. Debian ユーザーランド仕様

本章では、Armadillo-IoT ゲートウェイ A6 の Debian 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 ゲートウェイ A6 の標準ルートファイルシステムとして、 32-bit hard-float ARMv7(「armhf」)アーキテクチャ用の Debian GNU/Linux 10(コードネーム「buster」)を書き込んだ場合、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 ゲートウェイ A6 シリーズにインストールされた Debian GNU/Linux 10 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なってください。

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、「6.2. 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「6.6. RTC」を参照してシステムクロックを合わせてください。

apt update

パッケージインデックファイルを最新の状態にアップデートします。

引数　なし

使用例

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

apt 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　なし

[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。

使用例

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

apt 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 install gcc
```

apt 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 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 (U-Boot) 仕様

本章では、 Armadillo-IoT ゲートウェイ A6 のブートローダーである U-Boot 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。

U-Boot は Open Source で開発されているブートローダーで、特に組み込み機器に良く使われています。 U-Boot のマニュアルは、 Denx Software Engineering の U-Boot のページ (https://www.denx.de/wiki/U-Boot/WebHome) からアクセスできます。

10.1. U-Boot の起動モード

U-Boot はブートローダーなので、OS を起動するのが仕事です。しかし OS を起動する以外にも、いろいろと便利な機能が U-Boot には備わっています。

Armadillo-IoT ゲートウェイ A6 の U-Boot には 2 つの起動モードがあります。「保守モード」と「オートブートモード」です。 このモード切り換えは、 GPIO によって実現しています。 U-Boot 本家にはまだマージされておらず、 Armadillo-IoT ゲートウェイ A6 用の U-Boot に独自実装されている機能です。

ユーザースイッチ(SW1) を押しながら電源を投入した場合、保守モードでブートローダーが起動します。

表 10.1 ブートローダー起動モード

起動モードの種別	説明
保守モード	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	電源投入後、自動的に Linux カーネルを起動させます。

表 10.2 各種スイッチの状態とブートローダー起動モード

起動モードの種別	ユーザースイッチ
保守モード	押す
オートブートモード	押さない

U-Boot が起動すると、U-Boot のバージョンや、ビルド時間、CPU の情報、DRAM のサイズなどボード情報が表示されます。

```
U-Boot 2018.03-at8 (Feb 17 2020 - 19:19:11 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

図 10.1 U-Boot の起動

⇒ が U-Boot のプロンプトです。プロンプトが出るのは保守モードの時だけです。Armadillo-IoT ゲートウェイ A6 では U-Boot のプロンプトが表示され、コマンド入力を受け付ける状態を「保守モード」と呼んでいます。

10.2. U-Boot の機能

U-Boot の 機能を使うには U-Boot のコマンドプロンプトからコマンドを入力します。コマンドプロンプトは保守モードにすることで表示されます。

U-Boot の保守モードでは、U-Boot のバージョン番号を表示したり、あるメモリアドレスの値を表示したり Linux カーネルの起動オプションの設定などを行うことができます。保守モードで利用できる有用なコマンドは、プロンプトで help と入力すると表示されます。

```
=> help
?       - alias for 'help'
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootefi - Boots an EFI payload from memory
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
clocks  - display clocks
cmp     - memory compare
config  - print .config
cp      - memory copy
crc32   - checksum calculation
dcache  - enable or disable data cache
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
editenv - edit environment variable
env     - environment handling commands
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
ext4load- load binary file from a Ext4 filesystem
ext4ls  - list files in a directory (default /)
ext4size- determine a file's size
ext4write- create a file in the root directory
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
fatsize - determine a file's size
fdt     - flattened device tree utility commands
fstype  - Look up a filesystem type
fsuuid  - Look up a filesystem UUID
fuse    - Fuse sub-system
grepenv - search environment variables
help    - print command description/usage
icache  - enable or disable instruction cache
load    - load binary file from a filesystem
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loadx   - load binary file over serial line (xmodem mode)
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
loopw   - infinite write loop on address range
```

```
ls      - list files in a directory (default /)
md      - memory display
md5sum  - compute MD5 message digest
meminfo - display memory information
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub system
mmcinfo - display MMC info
mw      - memory write (fill)
nm      - memory modify (constant address)
part    - disk partition related commands
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
save    - save file to a filesystem
saveenv - save environment variables to persistent storage
setenv  - set environment variables
sha1sum - compute SHA1 message digest
size    - determine a file's size
strings - display strings
tftpboot- boot image via network using TFTP protocol
usb     - USB sub-system
version - print monitor, compiler and linker version
=>
```

図 10.2 U-Boot コマンドのヘルプを表示

各コマンドのヘルプを表示するには 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 10.3 U-Boot コマンドのヘルプを表示

良く使うと思われるコマンドを以下で説明します。

boot	環境変数 bootcmd に指定されているコマンドを実行。デフォルトでは Linux を起動。オートブートモード時はこのコマンドが呼ばれている
env	U-Boot の環境変数に関連したコマンド (下記で詳しく説明)
ext4load	Ext4 ファイルシステムからファイルをメモリにロード
ext4ls	Ext4 ファイルシステムにあるファイルをリスト
fuse	CPU の内部 Fuse の値の読み書き
help	コマンド一覧、または指定されたコマンドのヘルプを表示
mmc	MMC/SD 関連のコマンド群 「10.2.2. mmc コマンド」 で詳しく説明
ping	ICMP ECHO_REQUEST を送信
run	環境変数に登録されているコマンドの実行
tftpboot	TFTP による起動

usb	USB 関連のコマンド群
version	U-Boot のバージョン番号表示

help で表示されるコマンドには、Git のようにサブコマンドを持つものがあります。 env や usb などがそうです。 help env とすることで、 指定したコマンドのサブコマンドが表示されます。

10.2.1. env コマンド

```
=> help env
env - environment handling commands

Usage:
env default [-f] -a - [forcibly] reset default environment
env default [-f] var [...] - [forcibly] reset variable(s) to their default values
env delete [-f] var [...] - [forcibly] delete variable(s)
env edit name - edit environment variable
env exists name - tests for existence of variable
env export [-t | -b | -c] [-s size] addr [var ...] - export environment
env grep [-e] [-n | -v | -b] string [...] - search environment
env import [-d] [-t [-r] | -b | -c] addr [size] - import environment
env print [-a | name ...] - print environment
env run var [...] - run commands in an environment variable
env save - save environment
env set [-f] name [arg ...]

=>
```

図 10.4 env コマンドのヘルプを表示

env default	環境変数をリセット
env delete	指定した環境変数を削除
env grep	指定した文字列を環境変数から検索
env print	指定した環境変数を表示します。指定が無ければ、すべて表示。 printenv コマンドと同じ
env save	環境変数を eMMC に保存。 saveenv コマンドと同じ。「10.3. U-Boot の環境変数」 で詳しく説明
env set	環境変数を設定。 setenv コマンドと同じ

10.2.2. mmc コマンド

```
=> mmc
mmc - MMC sub system

Usage:
mmc info - display info of the current MMC device
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
```

```
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition]
mmc list - lists available devices
mmc hwpartition [args...] - does hardware partitioning
  arguments (sizes in 512-byte blocks):
    [user [enh start cnt] [wrrel {on|off}]] - sets user data area attributes
    [gp1|gp2|gp3|gp4 cnt [enh] [wrrel {on|off}]] - general purpose partition
    [check|set|complete] - mode, complete set partitioning completed
  WARNING: Partitioning is a write-once setting once it is set to complete.
  Power cycling is required to initialize partitions after set to complete.
mmc setdsr <value> - set DSR register value

=>
```

図 10.5 mmc コマンドのヘルプを表示

`mmc info`　　現在指定されている MMC デバイスの情報を表示

`mmc list`　　ボード上の MMC デバイスのリストを表示

`mmc dev`　　現在指定されている MMC デバイスを表示。または指定された番号で示される MMC デバイスを選択

10.3. U-Boot の環境変数

U-Boot は、環境変数を持つことができます。デフォルトの環境変数は、U-Boot をビルドした時に値が決定します。実行に環境変数を変更したり、変更した環境変数を保存したりすることも可能です。

env print コマンドで、現在設定されているすべての環境変数を表示できます。

```
=> env print
baudrate=115200
bootcmd=run setup_mmcargs; ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load m
mc 0:2 0x83000000 /boot/${fdt_file}; bootm ${loadaddr} - 0x83000000;
bootdelay=0
enable_pf3000_lpm=no
ethact=FEC
fdt_file=a640.dtb ❶
loadaddr=0x82000000
setup_bootcmd_usb=setenv bootcmd run setup_usbargs¥¥; usb start¥¥; ext4load usb
0:2 ¥¥${loadaddr} /boot/uImage¥¥; ext4load usb 0:2 0x83000000 /boot/¥¥${fdt_file
}¥¥; usb stop¥¥; bootm ¥¥${loadaddr} - 0x83000000¥¥;
setup_mmcargs=setenv bootargs root=/dev/mmcblk0p2 rootwait ${optargs};
setup_usbargs=setenv bootargs root=/dev/sda2 rootwait rw ${optargs};
stderr=serial
stdin=serial
stdout=serial
stop_nr3225sa_alarm=no;
tftpboot=tftpboot uImage; tftpboot 0x83000000 ${fdt_file}; bootm ${loadaddr} - 0
x83000000;
usbboot=run setup_bootcmd_usb; boot;

Environment size: 826/524284 bytes
=>
```

❶　fdt_file 変数のデフォルト値は、"a640.dtb" となります。

または env print コマンドで変数を指定すると、その変数の値だけを表示することも可能です。

```
=> env print loadaddr
loadaddr=0x82000000
=>
```

新しく変数を追加したり、すでに設定されている値を変更するには env set を使います。

```
=> env set hello world
=> env print hello
hello=world
=> env set hello armadillo
=> env print hello
hello=armadillo
=>
```

不要な変数を削除するには env delete を使います。

```
=> env delete hello
=> env print hello
## Error: ”hello” not defined
=>
```

環境変数を保存するには env save コマンドを使います。

```
=> env set hello armadillo
=> env save
Saving Environment to MMC... Writing to MMC(0)... OK
=>

....電源入れなおし....

U-Boot 2018.03-at8 (Feb 14 2020 - 10:21:57 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC

=> env print hello
hello=armadillo
=>
```

表示された文字からも分るように、Armadillo-IoT ゲートウェイ A6 では環境変数を MMC の 0 番、つまりオンボード eMMC に保存します。U-Boot の環境変数が保存される場所は、オンボード eMMC の先頭から 512 KByte オフセットです。eMMC の 1 MByte オフセットから第 1 パーティションが始まっているので、環境変数を保存できるのは 512 KByte になります。 eMMC のアドレスマップについては「表 3.6. eMMC メモリマップ」 を参照してください。

環境変数をデフォルト値に戻したい場合は `env default` コマンドを使います。

```
=> env print bootdelay
bootdelay=0
=> env set bootdelay 1
=> env print bootdelay
bootdelay=1
=> env default bootdelay
=> env print bootdelay
bootdelay=0
=>
```

もし、すべての環境変数をデフォルト値に戻したい場合は、 オプション `-a` を付けてください。

```
=> env default -a
## Resetting to default environment
=>
```

図 10.6 全ての環境変数をデフォルト値に戻す

特定の変数は、値を変更するタイミングで U-Boot の関数が実行されます。環境変数 `baudrate` もそのうちの 1 つです。つまり値が変更されるだけでなく、実際にボーレートが変更されます。

他にも値の変更できなくなっていたり、変更回数が決まっているものもあります。環境変数 `ethaddr` もその 1 つです。それらの変数は、変更する必要はあまり無いと思いますが、もし変更する場合には U-Boot のマニュアルを参照してください。

10.4. U-Boot が Linux を起動する仕組み

U-Boot は、 `boot` コマンドによって、OS を起動します。 `boot` コマンドは、 環境変数 `bootcmd` に登録されているコマンドを実行するコマンドです。つまり `run bootcmd` と同じ意味です。

```
=> env print loadaddr
loadaddr=0x82000000
=> env print bootcmd
bootcmd=run setup_mmcargs; ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load m
mc 0:2 0x83000000 /boot/${fdt_file}; bootm ${loadaddr} - 0x83000000;
=> run bootcmd
7229976 bytes read in 221 ms (31.2 MiB/s)
28291 bytes read in 55 ms (502 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
```

```
   Image Name:   Linux-4.14-at28
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7229912 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02e82 ... OK

Starting kernel ...
```

環境変数 bootcmd には複数のコマンドが ; で区切られて並んでいます。 U-Boot の run コマンドは、並んでいるコマンドを順次実行していきます。

```
bootcmd=ext4load mmc 0:2 ${loadaddr} /boot/uImage; ext4load mmc 0:2 0x83000000 /boot/${fdt_file};
bootm ${loadaddr} - 0x83000000;
```

最初のコマンドは ext4load です。このコマンドは、**0** 番目の **MMC** デバイスにある、 **2** 番目パーティションにアクセスし **/boot/uImage** を、環境変数 **loadaddr** で指定されているメモリアドレスにロードします。コマンドで環境変数を参照するには、変数を ${…} でくくります。出荷状態では、オンボード eMMC の第 2 パーティションが EXT4 でフォーマットされており、 /boot/ ディレクトリに uImage というファイル名で Linux カーネルが配置されています。つまり最初のコマンドは、Linux カーネルをメモリにロードしているわけです。

```
=> help ext4load
ext4load - load binary file from a Ext4 filesystem

Usage:
ext4load <interface> [<dev[:part]> [addr [filename [bytes [pos]]]]]
    - load binary file 'filename' from 'dev' on 'interface'
      to address 'addr' from ext4 filesystem
=>
```

同じコマンドを U-Boot のプロンプトで手で実行しても、同じ動作結果になります。コマンドを 1 つだけ入力するときは ; を付けても付けなくても問題ありません。

```
=> ext4load mmc 0:2 ${loadaddr} /boot/uImage
6601520 bytes read in 206 ms (30.6 MiB/s)
=>
```

次のコマンドも同じく ext4load で、環境変数 fdt_file で指定されているファイルをメモリアドレス 0x83000000 にロードしている事が分ります。環境変数 fdt_file は、他の環境変数とは異なり、デフォルト値が Armadillo の起動時に決まります。Armadillo-IoT ゲートウェイ A6 では fdt_file=a640.dtb となります。このファイルは「Device Tree Blob (dtb)」というもので、ボードのデバイス情報が記録されています。最近の Linux カーネルでは必須のファイルです。このファイルのロードアドレスは、 uImage と異なり変数になっておらず値 (0x83000000) がそのまま記載されています。ファイルがある場所は、uImage と同じくオンボード eMMC の第 2 パーティションで /boot/ です。

```
=> ext4load mmc 0:2 0x83000000 /boot/${fdt_file}
27838 bytes read in 55 ms (494.1 KiB/s)
=>
```

これで、Linux を起動するために必要なファイル 2 つ (uImage と a640.dtb) をメモリに配置することができました。次のコマンド bootm で実際に Linux をブートします。 bootm コマンドは、引数に 3 つのアドレスを取ります。

```
bootm ${loadaddr} - 0x83000000
```

1 つ目が、Linux カーネルを置いたアドレス (${loadaddr} == 0x82000000)、2 つ目が initrd のアドレスですが Armadillo-IoT ゲートウェイ A6 では使用してないので - を書きます。3 つ目が Device Tree Blob を置いたアドレス (0x83000000) です。 U-Boot は Linux 以外も起動することができるので、 bootm のヘルプでは Linux カーネルを "Application image" と記載しています。

```
=> help bootm
bootm - boot application image from memory

Usage:
bootm [addr [arg ...]]
    - boot application image stored in memory
        passing arguments 'arg ...'; when booting a Linux kernel,
        'arg' can be the address of an initrd image
        When booting a Linux kernel which requires a flat device-tree
        a third argument is required which is the address of the
        device-tree blob. To boot that kernel without an initrd image,
        use a '-' for the second argument. If you do not pass a third
        a bd_info struct will be passed instead
   :
   :
=>
```

実際に bootm コマンドを使って Linux を起動してみます。ここではあえて loadaddr ではなく 0x82000000 を入力してみます。

```
=> bootm 0x82000000 - 0x83000000
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at28
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7229912 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02e82 ... OK

Starting kernel ...
```

このように、手で入力しても手順さえ間違わなければ、Linux を起動することができます。もちろん uImage の場所やファイル名を変更しても、 U-Boot で正しく指定すれば同様に動作します。

10.5. U-Boot から見た eMMC / SD

起動コマンドから分るように Armadillo-IoT ゲートウェイ A6 では、オンボード eMMC が 0 番目の MMC デバイスです。これは mmc list コマンドでも確認できます。

```
=> mmc list
FSL_SDHC: 0 (eMMC)
FSL_SDHC: 1
```

MMC デバイス情報は mmc info コマンドで表示できます。 mmc info コマンドは、引数でどのデバイスかを指定するのではなく、事前に mmc dev コマンドで使うデバイスを指定しておく必要があります。

```
=> mmc dev
switch to partitions #0, OK
mmc0(part 0) is current device
```

mmc dev コマンドでデバイス番号を指定しないと、現在指定されているデバイスを表示します。

```
=> mmc info
Device: FSL_SDHC
Manufacturer ID: 13
OEM: 14e
Name: S0J35
Bus Speed: 52000000
Mode : MMC High Speed (52MHz)
Rd Block Len: 512
MMC version 5.1
High Capacity: Yes
Capacity: 3.5 GiB
Bus Width: 8-bit
Erase Group Size: 512 KiB
HC WP Group Size: 8 MiB
User Capacity: 3.5 GiB ENH WRREL
User Enhanced Start: 0 Bytes
User Enhanced Size: 3.5 GiB
Boot Capacity: 31.5 MiB ENH
RPMB Capacity: 4 MiB ENH
GP1 Capacity: 8 MiB ENH WRREL
GP2 Capacity: 8 MiB ENH WRREL
GP3 Capacity: 8 MiB ENH WRREL
GP4 Capacity: 8 MiB ENH WRREL
```

これが、オンボード eMMC の情報です。次に 1 番目のデバイスを指定してみます。microSD カードが装着されていれば、次のように表示されます。

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
```

```
=> mmc info
Device: FSL_SDHC
Manufacturer ID: 2
OEM: 544d
Name: SA08G
Bus Speed: 50000000
Mode : SD High Speed (50MHz)
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 7.2 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
=>
```

カードの種類によって表示される値は異なります。もし SD カードに入れている Linux カーネルを起動する場合は、先の ext4load コマンドでデバイス 1 を指定すれば良いことが分ります。

```
=> ext4load mmc 1:1 ${loadaddr} /boot/uImage
```

上記の例は、 microSD の 1 番目のパーティションにある /boot/uImage をロードする例です。

10.6. Linux カーネル起動オプション

Linux カーネルは、ブートローダーから「カーネルパラメーター」と呼ばれる起動オプションを受けとる事ができます。U-Boot では 環境変数 bootargs に入っている文字列を Linux に渡します。

Armadillo-IoT ゲートウェイ A6 では bootargs の値は setup_mmcargs 変数の内部で一旦 setenv することで以下のように設定しています

```
setup_mmcargs=setenv bootargs root=/dev/mmcblk0p2 rootwait ${optargs};
```

この文字列で、ルートファイルシステムが /dev/mmcblk0p2 であり、mmcblk0 がみつかるまで待つ (rootwait) ように指示しています。 前述の通り Armadillo-IoT ゲートウェイ A6 では オンボード eMMC が 0 番目の MMC デバイスです。Armadillo-IoT ゲートウェイ A6 では初期出荷時に、オンボード eMMC の第 2 パーティションに Debian をインストールして出荷しているので 「オンボード eMMC、つまり 0 番目の MMC デバイスの第 2 パーティション」を表わす /dev/mmcblk0p2 を指定しています。

もし microSD カードに、Debian を構築し、そのパーティションをルートファイルシステムとして指定したい場合、

```
=> setenv setup_mmcargs setenv bootargs root=/dev/mmcblk1p1 rootwait ${optargs};
```

としてください。 前述の通り microSD は 1 番目の MMC デバイスなので root=/dev/mmcblk1p1 で microSD の第 1 パーティションという意味になります。

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。

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。

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、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 A6 ソフトウェアダウンロード
>
> https://armadillo.atmark-techno.com/armadillo-iot-a6/resources/software

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。

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。

   ```
   [ATDE ~]$ tar xf u-boot-a600-console-uart3-v2018.03-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、デフォルトコンフィギュレーションを適用します。

   ```
   [ATDE ~]$ cd u-boot-a600-v2018.03-at[version]
   [ATDE ~/u-boot-a600-console-uart3-v2018.03-at[version]]$ make ARCH=arm
   armadillo-640_console-uart3_defconfig
   ```

3. ビルド

   ビルドには make コマンドを利用します。

   ```
   [ATDE ~/u-boot-a600-console-uart3-v2018.03-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。

```
[ATDE ~/u-boot-a600-console-uart3-v2018.03-at[version]]$ ls u-boot.imx
u-boot.imx
```

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から、イメージファイルを作成する手順を説明します。

ビルドに必要なファイル
- linux-v4.14-at[version].tar.gz
- initramfs_a600-[version].cpio.gz

11.2.1. 手順：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。

```
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version].tar.gz
[ATDE ~]$ tar xf linux-v4.14-at[version].tar.gz
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version]  linux-v4.14-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

```
[ATDE ~]$ cd linux-v4.14-at[version]
[ATDE ~/linux-v4.14-at[version]]$ ln -s ../initramfs_a600-[version].cpio.gz
initramfs_a600.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。

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm armadillo-640_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。

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x82000000 uImage
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

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。

```
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/uImage
arch/arm/boot/uImage
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-iotg-a6.dtb
arch/arm/boot/dts/armadillo-iotg-a6.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、at-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。at-debian-builder は ATDE 等の PC で動作している Linux 上で Armadillo 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 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 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 at-debian-builder を使って構築することをお勧めします。

Debian GNU/Linux 10(コードネーム buster) ルートファイルシステムを構築するには、at-debian-builder のバージョンが v2.x.x (x は 0 以上の数字)である必要があります。

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。

```
[ATDE ~]$ tar xf at-debian-builder-[VERSION].tar.gz
[ATDE ~]$ cd at-debian-builder-[VERSION]
[ATDE ~]$ sudo ./build.sh aiota6
```

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

at-debian-builder-**[VERSION]**/aiota6_resouces 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

aiota6_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

aiota6_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 上で "apt 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。

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 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. イメージファイルの書き換え方法

本章では、Armadillo-IoT ゲートウェイ A6 の内蔵ストレージ(eMMC)に書き込まれているイメージファイルを書き換える手順について説明します。

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、"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、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 ゲートウェイ A6 ソフトウェアダウンロード
>
> https://armadillo.atmark-techno.com/armadillo-iot-a6/resources/software

12.1. インストールディスクを使用する

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イメージをまとめて書き換えることができます。 Armadillo がソフトウェアの問題により起動しなくなった場合の復旧方法としてもご使用頂けます。

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「12.2. 特定のイメージファイルだけを書き換える」を参照してください。

インストールディスクは ATDE で作成します。インストールディスクの作成に使用するファイルを次に示します。

表 12.1 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-buster-aiota6-**[version]**.img

12.1.1. インストールディスクの作成

1. 512 MB 以上の microSD カードまたは USB メモリを用意してください。
2. ATDE に microSD カードまたは USB メモリを接続します。詳しくは「4.5.2. 取り外し可能デバイスの使用」を参照してください。
3. microSD カードまたは USB メモリがマウントされている場合、アンマウントします。

```
[ATDE ~]$ mount
(省略)
/dev/sdb1 on /media/atmark/B18A-3218 type vfat
```

```
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=437,iocharset=u ⏎
tf8,shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)
[ATDE ~]$ sudo umount /dev/sdb1
```

4. microSD カードまたは USB メモリにインストールディスクイメージを書き込みます。

```
[ATDE ~]$ sudo dd if=install-disk-sd-buster-aiota6-[version].img of=/dev/sdb bs=4M ⏎
conv=fsync
127+1 レコード入力
127+1 レコード出力
536870400 バイト (537 MB) コピーされました、 65.6377 秒、 8.2 MB/秒
```

12.1.2. インストールの実行(microSD カード)

1. Armadillo の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. インストールディスクを microSD ユニットに装着します。装着方法は「4.2. microSD カードの装着」を参照してください。

3. サブユニット SW1(ユーザースイッチ)のスライドスイッチが「4.8. スライドスイッチの設定について」の microSD 側に設定されている事を確認してください。

4. Armadillo に電源を投入すると microSD カードからブートローダーが起動し、次に示すログが表示され、インストールが始まり、自動的に eMMC が書き換えられます。

```
U-Boot 2018.03-at8 installer+ (Mar 05 2021 - 18:01:07 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
Hit any key to stop autoboot:  0
7253576 bytes read in 368 ms (18.8 MiB/s)
26390 bytes read in 54 ms (476.6 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at31
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7253512 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eefa000, end 9ef03715 ... OK

Starting kernel ...
```

```
:（省略）
*+**+ Install Start!! *+**+
```

5. 以下のようにメッセージが表示されるとインストール完了です。電源を切断してください。

```
*+**+ Install Completed!! *+**+
```

12.1.3. インストールの実行(USB メモリ)

1. Armadillo の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. インストールディスクを書き込んだ USB メモリを、 Armadillo の USB インタフェースに差し込んでください。
3. SW1(ユーザースイッチ)を押しながら Armadillo に電源を投入すると、次に示すログが表示されます。

```
U-Boot 2018.03-at8 installer+ (Feb 17 2020 - 19:19:21 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

4. 次のように"run usbboot"コマンドを実行すると USB メモリから起動しインストールが始まり、自動的に eMMC が書き換えられます。

```
=>run usbboot
starting USB...
USB0:   USB EHCI 1.00
scanning bus 0 for devices... 1 USB Device(s) found
USB1:   USB EHCI 1.00
scanning bus 1 for devices... 2 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
7253576 bytes read in 316 ms (21.9 MiB/s)
26390 bytes read in 130 ms (198.2 KiB/s)
stopping USB..
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at31
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7253512 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
```

```
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02715 ... OK

Starting kernel ...
:（省略）
*+*+* Install Start!! *+*+*
```

5. 以下のようにメッセージが表示されるとインストール完了です。電源を切断してください。

```
*+*+* Install Completed!! *+*+*
```

12.1.4. LED 点灯パターンによるインストールの進捗表示

LED 点灯パターンによって、インストールの進捗状況を確認することができます。

インストールの進捗と LED 点灯パターンの関係を次に示します。

表 12.2 インストールの進捗と LED 点灯パターン

進捗	ユーザー LED 赤	ユーザー LED 緑
実行中	消灯	点滅
正常終了	点灯	点灯
異常終了	点滅	消灯

12.2. 特定のイメージファイルだけを書き換える

Armadillo-IoT ゲートウェイ A6 が起動した状態であれば、特定のイメージファイルだけを書き換えることができます。

イメージファイルと書き込み先の対応を次に示します。

表 12.3 イメージファイル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-a600-console-uart3-v2018.03-at[version].imx	eMMC	/dev/mmcblk0
Linux カーネルイメージ	uImage-a600-v4.14-at[version]		/dev/mmcblk0p2
Device Tree Blob	armadillo-iotg-a6-v4.14-at[version].dtb		/dev/mmcblk0p2
Debian GNU/Linux ルートファイルシステム	debian-buster-armhf-aiota6-[version].tar.gz		/dev/mmcblk0p2

12.2.1. ブートローダーイメージの書き換え

ブートローダーイメージの書き換え方法を次に示します。

```
[armadillo ~]# dd if=u-boot-a600-console-uart3-v2018.03-at[version].imx of=/dev/mmcblk0 bs=1k ⏎
seek=1 conv=fsync ❶
275+0 records in
275+0 records out
281600 bytes (282 kB, 275 KiB) copied, 0.0310393 s, 9.1 MB/s
```

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。

製品アップデートでリリースされた最新のブートローダーイメージに書き換えを行った場合は「図 10.6. 全ての環境変数をデフォルト値に戻す」を参照して環境変数をデフォルト値に戻してください。

新しくリリースされた最新のブートローダーイメージは、環境変数のデフォルト値が更新されることがあります。書き替え前のブートローダーによって生成された環境変数で、最新のブートローダーを動作させると不具合が起こる可能性があります。

12.2.2. Linux カーネルイメージの書き換え

Linux カーネルイメージの書き換え方法を次に示します。

```
[armadillo ~]# mount /dev/mmcblk0p2 /mnt ❶
[armadillo ~]# cp uImage-a600-v4.14-at[version] /mnt/boot/uImage ❷
[armadillo ~]# umount /mnt ❸
```

❶　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。

❷　Linux カーネルイメージを/mnt/boot/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 2 パーティションをアンマウントします。

12.2.3. DTB の書き換え

DTB の書き換え方法を次に示します。

```
[armadillo ~]# mount /dev/mmcblk0p2 /mnt ❶
[armadillo ~]# cp armadillo-iotg-a6-v4.14-at[version].dtb /mnt/boot/a640.dtb ❷
[armadillo ~]# umount /mnt ❸
```

❶　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。

❷　DTB を a640.dtb にリネームして/mnt/boot/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 2 パーティションをアンマウントします。

12.2.4. 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には、SD ブートまたは USB ブートを行う必要があります。ブートディスクの作成方法や SD/USB ブートの実行方法ついては「16. SD ブートの活用」「15. USB ブートの活用」を参照してください。

SD/USB ブートした Armadillo で、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。

1. Debian GNU/Linux ルートファイルシステムアーカイブを Armadillo にコピーします。例として USB メモリを利用する方法を記載します。

ATDE にある Debian GNU/Linux ルートファイルシステムアーカイブを USB メモリの第 1 パーティションにコピーするには次のコマンドを実行します。

```
[ATDE ~]$ sudo umount /dev/sdb1 ❶
[ATDE ~]$ sudo mount /dev/sdb1 /media ❷
[ATDE ~]$ sudo cp debian-buster-armhf-aiota6-[version].tar.gz /media ❸
[ATDE ~]$ sudo umount /media ❹
```

❶ USB メモリ接続時に自動で USB メモリの第 1 パーティションがマウントされることがあるため、一旦アンマウントします。

❷ USB メモリの第 1 パーティションを/media/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/media/ディレクトリ以下にコピーします。

❹ /media/ディレクトリにマウントした USB メモリの第 1 パーティションをアンマウントします。

USB メモリの第 1 パーティションにある Debian GNU/Linux ルートファイルシステムアーカイブを Armadillo にコピーするには次のコマンドを実行します。

```
[armadillo ~]# mount /dev/sda1 /media ❶
[armadillo ~]# cp /media/debian-buster-armhf-aiota6-[version].tar.gz . ❷
[armadillo ~]# umount /media ❸
```

❶ USB メモリの第 1 パーティションを/media/ディレクトリにマウントします。

❷ ルートファイルシステムアーカイブをカレントディレクトリ以下にコピーします。

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

2. 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 ~]# mount -t ext4 /dev/mmcblk0p2 /mnt ❶
[armadillo ~]# cd /mnt
[armadillo /mnt]# ls | grep -v -E 'boot|lost¥+found' | xargs rm -rf ❷
[armadillo /mnt]# cd -
[armadillo ~]# tar zxf debian-buster-armhf-aiota6-[version].tar.gz -C /mnt ❸
[armadillo ~]# umount /mnt ❹
```

❶ 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。

❷ eMMC の第 2 パーティションの boot、lost+found ディレクトリ以外のファイル・ディレクトリを削除します。

❸ 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。

❹ /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。

13. 開発の基本的な流れ

この章では Armadillo 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 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。

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、PC と同じように開発を進めることができます。

この章では、APT を使用して必要なソフトウェアを ATDE8 および Armadillo にインストールします。そのため、あらかじめ ATDE8 および Armadillo をインターネットに接続できる状態にしてください。

13.1. 軽量スクリプト言語によるデータの送信例(Ruby)

ここでは、サンプルとして Armadillo の現在時刻を定期的に HTTP POST でパラメータ名 "time" に格納した値として送信する例を示します。

現在時刻は date コマンド、もしくは Ruby の Time クラスを使用して取得します。

ここで作成するアプリケーションは Armadillo 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。

13.1.1. テスト用サーバーの実装

最初に ATDE8 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。

```
[ATDE ~]$ sudo apt install ruby
[ATDE ~]$ sudo gem install sinatra
```

図 13.1 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 "Current Time is #{params[:time]}"
  "Thanks!¥n"
end
```

図 13.2 テスト用サーバー (server.rb)

13.1.2. テスト用サーバーの動作確認

Armadillo 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 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、ATDE8 の IP アドレス を確認しておきます。下記の例では、ip コマンドで確認すると ATDE8 の IP アドレス が 172.16.2.117 であることがわかります。

```
[ATDE ~]$ ip addr show eth0
     2: eth0: &lt;BROADCAST,MULTICAST,UP,LOWER_UP&gt; 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 13.3 IP アドレスの確認 (ip 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。

```
[ATDE ~]$ ruby server.rb -p 8081 -o 0.0.0.0
[2018-07-18 17:47:21] INFO  WEBrick 1.3.1
[2018-07-18 17:47:21] INFO  ruby 2.3.3 (2016-11-21) [i386-linux-gnu]
== Sinatra (v2.0.3) has taken the stage on 8081 for development with backup from WEBrick
[2018-07-18 17:47:21] INFO  WEBrick::HTTPServer#start: pid=4610 port=8081
```

ここで、Armadillo 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 ATDE8 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 ~]# apt install curl
```

図 13.4 curl のインストール

```
[armadillo ~]$ curl -d "time=$(date "+%H:%M:%S")" 172.16.2.117:8081
Thanks!
```

図 13.5 curl によるテストデータの送信

正しく受信できた場合は、ATDE8 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。

```
Current Time is 07:44:19
```

図 13.6 ATDE8 におけるテストデータの受信表示

13.1.3. クライアントの実装

Armadillo 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、ATDE8 上で作成しても Armadillo 上で、vi 等を使って作成しても構いません。ATDE8 で作成した場合は次の手順で、Armadillo へ転送します。

```
require 'net/http'

uri = URI.parse(ARGV[0])
time = Time.now.strftime("%H:%M:%S")
response = Net::HTTP.post_form(uri, {"time" => time})

puts response.body
```

図 13.7 時刻送信クライアント(client.rb)

13.1.4. Armadillo へのファイルの転送

ATDE8 上で作成したソースコードを Armadillo 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 install openssh-server
```

図 13.8 Armadillo 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 13.9 ATDE8 から Armadillo への client.rb の転送

13.1.5. クライアントの実行

Armadillo に Ruby をインストールしてから、作成した時刻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 ATDE8 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 ~]# apt install ruby
```

図 13.10 ruby のインストール

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

図 13.11 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、ATDE8 で動作しているサーバーのコンソールには時刻が表示されます。

```
Current Time is 07:44:19
```

図 13.12 ATDE8 における時刻データの受信表示

13.2. 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。

13.2.1. 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 install build-essential
```

図 13.13 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 install libssl-dev
```

図 13.14 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。

14. i.MX6ULL の電源制御方法

本章では、ソフトウェアによる i.MX6ULL の電源制御方法について説明します。

14.1. poweroff コマンドによる制御

poweroff コマンドを利用して、i.MX6ULL 自身で電源を OFF にすることができます。

電源を OFF にするには、次のようにコマンドを実行します。

```
[armadillo ~]# poweroff
```

図 14.1 poweroff コマンドによる電源 OFF

14.2. RTC による制御

RTC のアラーム割り込みによって、i.MX6ULL の電源を ON にすることができます。

また、RTC の割り込み信号はアラームを解除するまで出力され続けるため、i.MX6ULL の電源が ON になってから 5 秒以内に、ブートローダーもしくは Linux カーネルがアラームを解除する必要があります。

ブートローダーの起動時に RTC のアラームを解除するには次のようにコマンドを実行します。

```
=> setenv stop_nr3225sa_alarm yes
=> saveenv
```

i.MX6ULL の電源を OFF にし、1 分後に電源を ON にするには、次のようにコマンドを実行します。

```
[armadillo ~]# echo +60 > /sys/class/rtc/rtc0/wakealarm ❶
[armadillo ~]# poweroff ❷
:（省略）
[   32.428051] reboot: Power down


U-Boot 2018.03-at4 (Dec 18 2018 - 18:34:38 +0900) ❸

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
```

```
Net:   FEC
=>
```

図 14.2 i.MX6ULL の電源を OFF にし、1 分後に電源を ON にする手順

❶ 1 分後にアラームを設定します。

❷ i.MX6ULL の電源を OFF にします。

❸ 1 分経過後、自動的に i.MX6ULL の電源が ON になります。

サブユニットに搭載されている RTC NR3225SA は、毎分 0 秒にしかアラーム割り込みを発生させることができません。

0 時 0 分 30 秒の時に、1 秒後にアラームが鳴るように設定したとしても、実際にアラーム割り込みが発生するのは 0 時 1 分 0 秒となります。

15. USB ブートの活用

本章では、USB メモリから起動(以降「USB ブート」と表記します)する手順を示します。

USB ブート時、 eMMC のブートローダーを利用しています。その為、eMMC のブートローダーの修復は USB ブートではできません。eMMC ブートローダーの復旧は SD ブートで実施ください。SD ブートに関する詳細は、「16. SD ブートの活用」を参照ください。

USB ブートを活用すると、USB メモリ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 USB メモリを必要とします。例として Debian GNU/Linux 10(コードネーム buster)を USB ブートする手順を示しますが、他の OS を USB ブートすることも可能です。

USB メモリに対する作業は、ATDE で行います。そのため、ATDE に USB メモリを接続する必要があります。詳しくは「4.5.2. 取り外し可能デバイスの使用」を参照してください。

ATDE に USB メモリを接続すると、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 USB メモリをアンマウントしておく必要があります。

```
[ATDE ~]$ mount
(省略)
/dev/sdb1 on /media/atmark/D07E-969E type vfat
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,sho
rtname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2)
[ATDE ~]$ sudo umount /dev/sdb1
```

図 15.1 自動マウントされた USB メモリのアンマウント

本章で使用するブートローダーイメージファイルの最新版ファイルは、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、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 Gateway A6 ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6/resources/software

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。

「表 15.1. ブートディスクの構成例」に示します。

表 15.1 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	第 2 パーティションにルートファイルシステムを構築するため、第 1 パーティションを作成します。
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。

15.1.1. 手順：ブートディスクの作成例

1. USB メモリに 2 つのプライマリパーティションを作成します。

```
[ATDE ~]$ sudo fdisk /dev/sdb ❶

Welcome to fdisk (util-linux 2.33.1).
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 0xc659ba45

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 (2048-60868607, default 2048): ❻
Last sector, +sectors or +size{K,M,G,T,P} (2048-60868607, default 60868607): +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 (264192-60868607, default 264192): ⓫
Last sector, +sectors or +size{K,M,G,T,P} (264192-60868607, default 60868607): ⓬

Created a new partition 2 of type 'Linux' and of size 28.9 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
```

```
Changed type of partition 'Linux' to 'W95 FAT32'.

Command (m for help): w ⑯
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[ATDE ~]$
```

❶ USB メモリのパーティションテーブル操作を開始します。他にリムバーブルデバイスなどを接続している場合は、USB メモリ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 USB メモリに書き込みます。

2. 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。

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

Disk /dev/sdb: 29 GiB, 31164727296 bytes, 60868608 sectors
Disk model: USB DISK
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: 0xc659ba45

Device     Boot  Start      End  Sectors  Size Id Type
```

```
/dev/sdb1          2048   264191   262144  128M  b W95 FAT32
/dev/sdb2        264192 60868607 60604416 28.9G 83 Linux
```

3. 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。

```
[ATDE ~]$ sudo mkfs.vfat -F 32 /dev/sdb1 ❶
mkfs.fat 4.1 (2017-01-24)
[ATDE ~]$ sudo mkfs.ext4 /dev/sdb2 ❷
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 7575552 4k blocks and 1896832 inodes
Filesystem UUID: 0eb75985-a5c5-497c-8939-a3f96be2000e
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[ATDE ~]$
```

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。

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 15.2 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-buster-armhf-aiota6-[version].tar.gz	ARM(armhf)アーキテクチャ用 Debian GNU/Linux 10(コードネーム buster) 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。

15.2.1.1. 手順：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. 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。

```
[ATDE ~]$ mkdir work ❶
[ATDE ~]$ sudo mount -t ext4 /dev/sdb2 work ❷
[ATDE ~]$ sudo tar zxf debian-buster-armhf-aiota6-[version].tar.gz -C work ❸
```

```
[ATDE ~]$ sudo umount work ❹
[ATDE ~]$ rmdir work ❺
```

❶ USB メモリをマウントするための work/ディレクトリを作成します。

❷ 第 2 パーティションを work/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 work/ディレクトリに展開します。

❹ work/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 work/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 USB メモリを作業用 PC から取り外すと、USB メモリ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 15.3 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	uImage-a600-v4.14-at_[version]_
DTB	armadillo-iot-a6-v4.14-at_[version]_.dtb

USB メモリに 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 15.4 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	ext4
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	a640.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。

15.3.1. 手順：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。

```
[ATDE ~]$ ls
uImage-a600-v4.14-at[version]  armadillo-iotg-a6-v4.14-at[version].dtb
```

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

```
[ATDE ~]$ mkdir work ❶
[ATDE ~]$ sudo mount -t ext4 /dev/sdb2 work ❷
[ATDE ~]$ sudo cp uImage-a600-v4.14-at[version] work/boot/uImage ❸
[ATDE ~]$ sudo cp armadillo-iotg-a6-v4.14-at[version].dtb work/boot/a640.dtb ❹
[ATDE ~]$ sudo umount work ❺
[ATDE ~]$ rmdir work ❻
```

❶ USB メモリをマウントするための work/ディレクトリを作成します。

❷ 第 2 パーティションを work/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 work/ディレクトリにコピーします。

❹ DTB を work/ディレクトリにコピーします。

❺ work/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 work/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 USB メモリを作業用 PC から取り外すと、USB メモリのデータが破損する場合があります。

15.4. USB 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。

1. Armadillo に電源を投入する前に、ブートディスクを CON5(USB Type-A コネクタ)に挿入します。
2. SW1(ユーザースイッチ)を押しながら電源を投入します。

```
U-Boot 2018.03-at8 (Feb 17 2020 - 19:19:11 +0900)

CPU:   Freescale i.MX6ULL rev1.0 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_USBHC: 0, FSL_USBHC: 1
Loading Environment from MMC... OK
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
=>
```

3. 以下のコマンドを入力し USB メモリからの起動を実行します。

```
=> run usbboot
```

4. 以下のようにログが表示され USB メモリからの起動が実行されます。

```
starting USB...
USB0:   USB EHCI 1.00
scanning bus 0 for devices... 1 USB Device(s) found
USB1:   USB EHCI 1.00
scanning bus 1 for devices... 2 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
7249760 bytes read in 314 ms (22 MiB/s)
26390 bytes read in 130 ms (198.2 KiB/s)
stopping USB..
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at28-ge97480002d52-di
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7249696 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eef9000, end 9ef02715 ... OK

Starting kernel ...
```

16. 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、SD 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。例として Debian GNU/Linux 10(コードネーム buster)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 eMMC 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「4.5.2. 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。

```
[ATDE ~]$ mount
(省略)
/dev/sdb1 on /media/52E6-5897 type ext2
(rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0077,codepage=cp437,iocharset=utf8,sh
ortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks)
[ATDE ~]$ sudo umount /dev/sdb1
```

図 16.1 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の最新版ファイルは、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、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 Gateway A6 ソフトウェアダウンロード

https://armadillo.atmark-techno.com/armadillo-iot-a6/resources/software

16.1. 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 16.1 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
ブートローダーイメージ	u-boot-a600-console-uart3-v2018.03-at_[version]_.imx

「表 16.2. ブートディスクの構成例」に示します。

表 16.2 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	第 2 パーティションにルートファイルシステムを構築するため、第 1 パーティションを作成します。
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。

16.1.1. 手順：ブートディスクの作成例

1. ブートローダーイメージファイルを取得し、ATDE 内に配置しておきます。

```
[ATDE ~]$ ls
u-boot-a600-console-uart3-v2018.03-at[version].imx
```

2. SD 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。

```
[ATDE ~]$ sudo fdisk /dev/sdb ❶

Welcome to fdisk (util-linux 2.33.1).
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.

[ATDE ~]$
```

❶ SD 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 SD 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。

```
[ATDE ~]$ 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。

```
[ATDE ~]$ sudo mkfs.vfat -F 32 /dev/sdb1  ❶
mkfs.fat 4.1 (2017-01-24)
[ATDE ~]$ sudo mkfs.ext4 /dev/sdb2  ❷
mke2fs 1.44.5 (15-Dec-2018)
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

[ATDE ~]$
```

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。

5. 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。

```
[ATDE ~]$ ls
u-boot-a600-console-uart3-v2018.03-at[version].imx
[ATDE ~]$ sudo dd if=u-boot-a600-console-uart3-v2018.03-at[version].imx of=/dev/sdb bs=1k ⏎
seek=1 conv=fsync
```

16.2. ルートファイルシステムの構築

「16.1. 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 16.3 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-buster-armhf-aiota6-[version].tar.gz	ARM(armhf)アーキテクチャ用 Debian GNU/Linux 10(コードネーム buster) のルートファイルシステムアーカイブ

16.2.1. 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。

16.2.1.1. 手順：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. 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。

```
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t ext4 /dev/sdb2 sd ❷
[ATDE ~]$ sudo tar zxf debian-buster-armhf-aiota6-[version].tar.gz -C sd ❸
[ATDE ~]$ sudo umount sd ❹
[ATDE ~]$ rmdir sd ❺
```

❶ SD 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 SD カードを作業用 PC から取り外すと、SD カードのデータが破損する場合があります。

16.3. Linux カーネルイメージと DTB の配置

「16.1. 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 16.4 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	uImage-a600-v4.14-at_[version]_
DTB	armadillo-iot-a6-v4.14-at_[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 16.5 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	ext4
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	a640.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。

16.3.1. 手順：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。

```
[ATDE ~]$ ls
uImage-a600-v4.14-at[version]  armadillo-iotg-a6-v4.14-at[version].dtb
```

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

```
[ATDE ~]$ mkdir sd ❶
[ATDE ~]$ sudo mount -t ext4 /dev/sdb2 sd ❷
[ATDE ~]$ sudo cp uImage-a600-v4.14-at[version] sd/boot/uImage ❸
[ATDE ~]$ sudo cp armadillo-iotg-a6-v4.14-at[version].dtb sd/boot/a640.dtb ❹
[ATDE ~]$ sudo umount sd ❺
[ATDE ~]$ rmdir sd ❻
```

❶ SD 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 SD カードを作業用 PC から取り外すと、SD カードのデータが破損する場合があります。

16.4. SD ブートの実行

「16.1. ブートディスクの作成」で作成したブートディスクから起動する方法を説明します。

1. Armadillo に電源を投入する前に、ブートディスクを CON1(microSD スロット)に挿入します。また、サブユニット SW1(ユーザー スイッチ) を microSD 側に設定します。 サブユニット SW1(ユーザースイッチ) の設定に関しては「4.8. スライドスイッチの設定について」を参照ください。

2. 電源を投入すると、以下の様にブートディスクから起動します。

```
U-Boot 2018.03-at8 installer+ (Mar 05 2021 - 18:01:07 +0900)

CPU:   Freescale i.MX6ULL rev1.1 at 396 MHz
Reset cause: POR
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
In:    serial
Out:   serial
Err:   serial
PMIC: PFUZE3000 DEV_ID=0x30 REV_ID=0x11
Net:   FEC
Hit any key to stop autoboot:  0
7253576 bytes read in 367 ms (18.8 MiB/s)
26390 bytes read in 54 ms (476.6 KiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-4.14-at31
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7253512 Bytes = 6.9 MiB
   Load Address: 82000000
   Entry Point:  82000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 83000000
   Booting using the fdt blob at 0x83000000
   Loading Kernel Image ... OK
   Loading Device Tree to 9eefa000, end 9ef03715 ... OK

Starting kernel ...
```

17. 電気的仕様

17.1. 絶対最大定格

表 17.1 絶対最大定格

項目	記号	Min.	Max.	単位	備考
電源電圧	VCC_5V	-0.3	5.25	V	-
入出力電圧(USB 信号以外)	VI,VO	-0.3	OVDD +0.3	V	OVDD=VCC_3.3V
入力電圧(USB 信号)	VI_USB	-0.3	3.63	V	USB_OTG1_DP, USB_OTG1_DN, USB_OTG2_DP, USB_OTG2_DN
RTC バックアップ電源電圧	RTC_BAT	-0.3	5.25	V	-
使用温度範囲(U1 モデル)	Topr	-20	70	℃	結露なきこと
使用温度範囲(C1 モデル)	Topr	-10	40	℃	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。

17.2. 推奨動作条件

表 17.2 推奨動作条件

項目	記号	Min.	Typ.	Max.	単位	備考
電源電圧	VCC_5V	4.75	5	5.25	V	-
RTC バックアップ電源電圧	RTC_BAT	2.75	-	5.25	V	Topr=+25℃
使用温度範囲(U1 モデル)	Ta	-20	25	70	V	結露なきこと
使用温度範囲(C1 モデル)	Ta	-10	25	40	V	結露なきこと

17.3. 入出力インターフェースの電気的仕様

表 17.3 入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
5V 電源電圧	VCC_5V	4.75	5	5.25	V	-

表 17.4 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
5V 電源電圧	USB_OTG1_VBUS USB_OTG2_VBUS	4.75	5	5.25	V	-
3.3V 電源電圧	VCC_3.3V	3.102	3.3	3.498	V	-

表 17.5 入出力インターフェースの電気的仕様(OVDD = VCC_3.3V)

項目	記号	Min.	Max.	単位	備考
ハイレベル出力電圧	VOH	OVDD-0.15	OVDD	V	IOH = -0.1mA, -1mA
ローレベル出力電圧	VOL	0	0.15	V	IOL = 0.1mA, 1mA
ハイレベル入力電圧[a]	VIH	0.7×OVDD	OVDD	V	-
ローレベル入力電圧[a]	VIL	0	0.3×OVDD	V	-
ローレベル入力電圧(ONOFF 信号)	VIL	0	0.9	V	-
ローレベル入力電圧 (PWRON 信号)	VIL	0	0.5	V	-
ローレベル入力電圧 (EXT_RESET_B 信号)	VIL	0	0.19	V	-
入力リーク電流(no Pull-up/ Pull-down)	IIN	-1	1	μA	-
Pull-up 抵抗(5kΩ)	-	4	6	kΩ	-
Pull-up 抵抗(47kΩ)	-	37.6	56.4	kΩ	-
Pull-up 抵抗(100kΩ)	-	80	120	kΩ	-
Pull-down 抵抗(100kΩ)	-	80	120	kΩ	-

[a]オーバーシュートとアンダーシュートは 0.6V 以下でかつ 4ns を超えないようにしてください。

17.4. 電源回路の構成

電源回路の構成は次のとおりです。電源入力インターフェース(メインユニット CON12 またはサブユニット CON3)からの入力電圧をパワーマネジメント IC(PMIC)等で各電圧に変換し、内部回路および各インターフェースに供給しています。各インターフェースや電圧レギュレータの最大出力電流値を超えないように、外部機器の接続、供給電源の設計を行なってください。

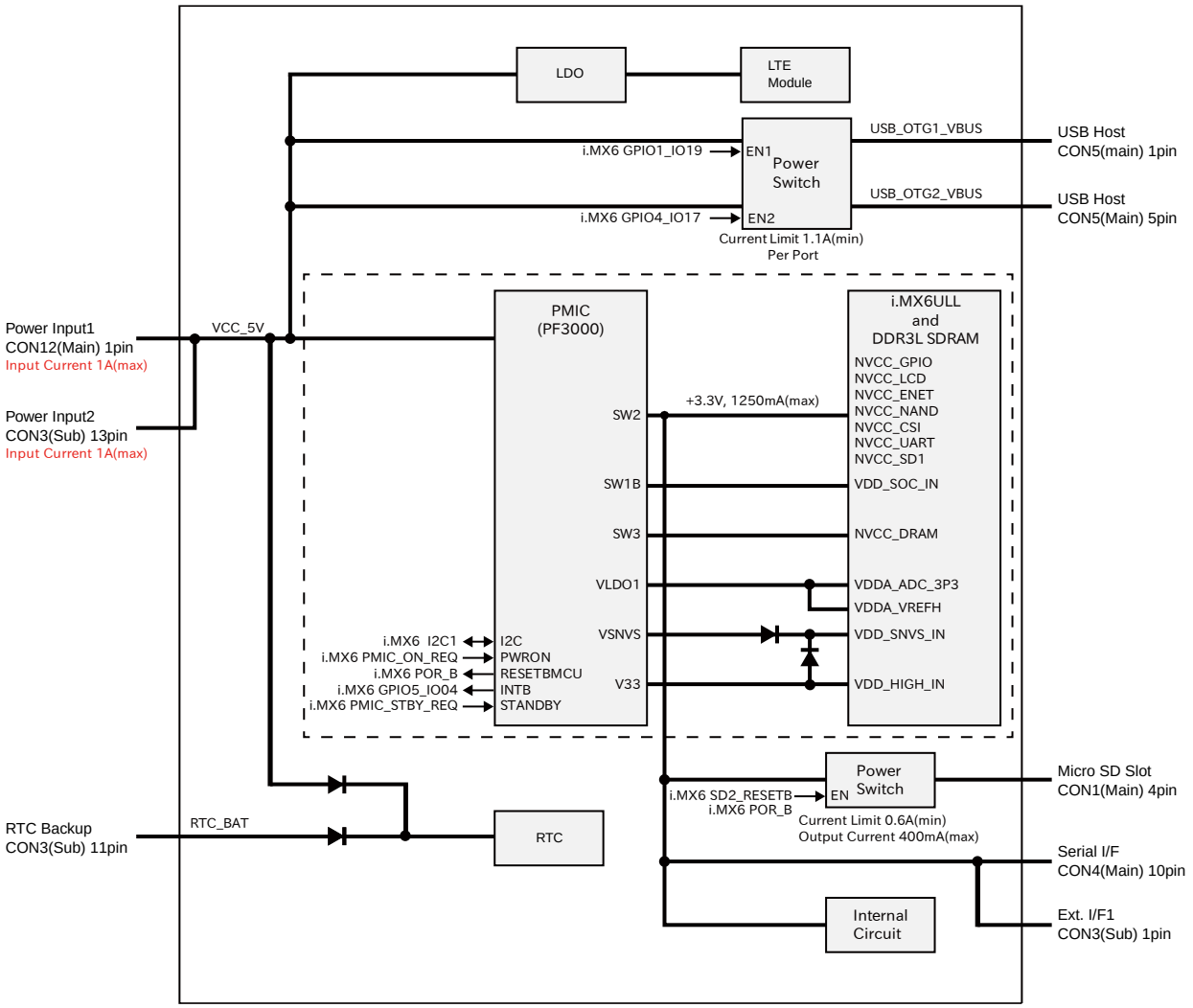


図 17.1 電源回路の構成

18. インターフェース仕様

Armadillo-IoT ゲートウェイ A6 のインターフェース仕様について説明します。

18.1. インターフェースレイアウト

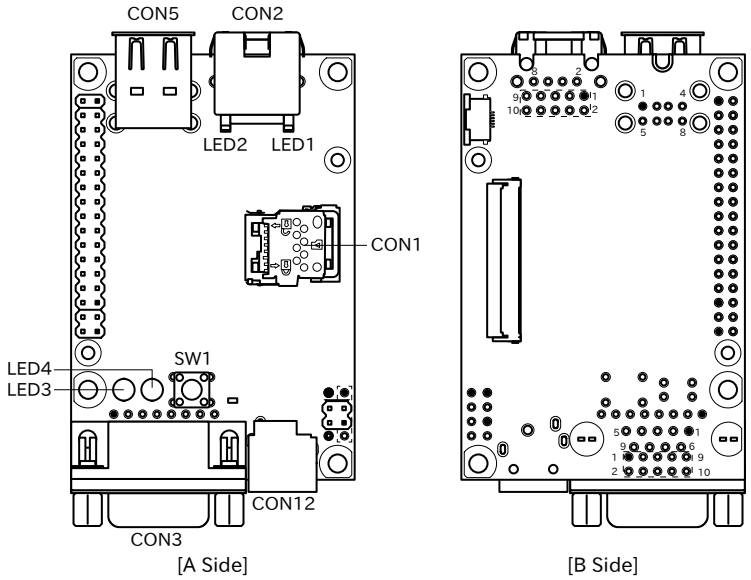


図 18.1 Armadillo-IoT A6 メインユニット インターフェースレイアウト(U1 モデル)

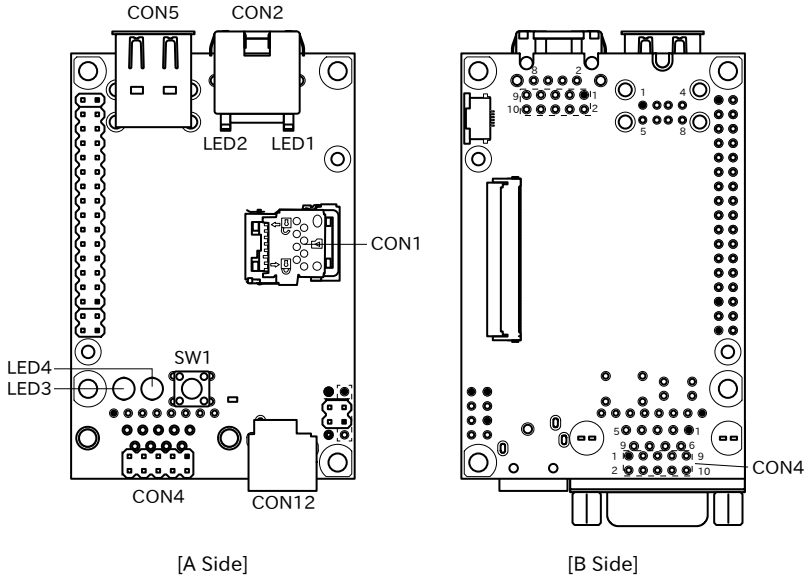


図 18.2 Armadillo-IoT A6 メインユニット インターフェースレイアウト(C1 モデル)

表 18.1 Armadillo-IoT ゲートウェイ A6 メインユニット インターフェース一覧 [a]

部品番号	インターフェース名	型番	メーカー
CON1	SD インターフェース	SDHK-8BNS-K-303-TB(HF)	日本圧着端子製造株式会社
CON2	LAN インターフェース	TM11R-5M2-88-LP	ヒロセ電機株式会社
CON3	シリアルインターフェース	XM2C-0942-132L	オムロン株式会社
CON4	シリアルインターフェース	A1-10PA-2.54DSA(71)	ヒロセ電機株式会社
CON5	USB インターフェース	UBA-4RS-D14T-4D(LF)(SN)	日本圧着端子製造株式会社
CON12	電源入力インターフェース	HEC3690-015210	ホシデン株式会社
LED1	LAN スピード LED	SML-310MTT86	ローム株式会社
LED2	LAN リンクアクティビティ LED	SML-310YTT86	ローム株式会社
LED3	ユーザー LED(赤)	SLR-342VC3F	ローム株式会社
LED4	ユーザー LED(緑)	SLR-342MC3F	ローム株式会社
SW1	ユーザースイッチ	SKHHDJA010	ALPS ELECTRIC

[a]部品の実装、未実装を問わず、搭載可能な部品型番を記載しています。

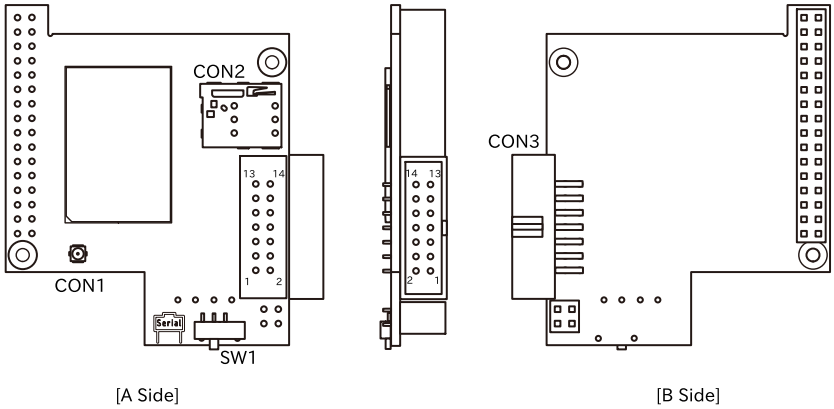


図 18.3 Armadillo-IoT ゲートウェイ A6 サブユニット インターフェースレイアウト

表 18.2 Armadillo-IoT ゲートウェイ A6 サブユニット インターフェース一覧 [a]

部品番号	インターフェース名	型番	メーカー
CON1	LTE アンテナインターフェース	U.FL-R-SMT-1(10)	ヒロセ電機株式会社
CON2	nanoSIM インターフェース	SF72S006VBDR2500	日本航空電子工業株式会社
CON3	拡張インターフェース	XG4C-1434	オムロン株式会社
SW1	起動デバイス設定スイッチ	CSS-1210TB	日本電産コパル電子株式会社

[a]部品の実装、未実装を問わず、搭載可能な部品型番を記載しています。

18.2. メインユニット CON1(SD インターフェース)

CON1 はハイスピード(最大クロック周波数: 49.5MHz)に対応した SD インターフェースです。信号線は i.MX6ULL の SD ホストコントローラ(uSDHC2)に接続されています。

SD カードに供給される電源は i.MX6ULL の NAND_ALE ピン(GPIO4_IO10)で制御が可能です。High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

CON1 は活線挿抜に対応していません。microSD カードの挿抜は、電源を切断してから行ってください。

表 18.3 メインユニット CON1 信号配列

ピン番号	ピン名	I/O	説明
1	DAT2	In/Out	SD データバス(bit2)、i.MX6ULL の NAND_DATA02 ピンに接続
2	CD/DAT3	In/Out	SD データバス(bit3)、i.MX6ULL の NAND_DATA03 ピンに接続
3	CMD	In/Out	SD コマンド/レスポンス、i.MX6ULL の NAND_WE_B ピンに接続
4	VDD	Power	電源(VCC_3.3V)
5	CLK	Out	i.MX6ULL の NAND_RE_B ピンに接続
6	VSS	Power	電源(GND)
7	DAT0	In/Out	i.MX6ULL の NAND_DATA00 ピンに接続
8	DAT1	In/Out	i.MX6ULL の NAND_DATA01 ピンに接続

18.3. メインユニット CON2(LAN インターフェース)

CON2 は 10BASE-T/100BASE-TX に対応した LAN インターフェースです。カテゴリ 5 以上の Ethernet ケーブルを接続することができます。AUTO-MDIX 機能を搭載しており、ストレートケーブルまたはクロスケーブルを自動認識して送受信端子を切り替えます。

信号線は Ethenet PHY(LAN8720AI-CP/Microchip Technology) を経由して i.MX6ULL の Ethernet コントローラ(ENET1)に接続されています。

表 18.4 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(+)
4	-	-	CON2 の 5 ピンと接続後に 75Ω 終端
5	-	-	CON2 の 4 ピンと接続後に 75Ω 終端
6	RX-	In/Out	受信データ(-)
7	-	-	CON2 の 8 ピンと接続後に 75Ω 終端
8	-	-	CON2 の 7 ピンと接続後に 75Ω 終端

18.4. メインユニット LED1、LED2(LAN LED)

LED1、LED2 は LAN インターフェースのステータス LED です。CON2 の上部に表示されます。信号線は Ethenet PHY(LAN8720AI-CP/Microchip Technology)の LED ピンに接続されています。

表 18.5 LAN LED の動作

LED	名称(色)	状態	説明
LED1	LAN スピード LED(緑)	消灯	10Mbps で接続されている、もしくは Ethernet ケーブル未接続
		点灯	100Mbps で接続されている
LED2	LAN リンクアクティビティ(黄)	消灯	リンクが確立されていない
		点灯	リンクが確立されている
		点滅	リンクが確立されており、データを送受信している

18.5. メインユニット CON3、CON4(シリアルインターフェース)

CON3、CON4 は非同期(調歩同期)シリアルインターフェースです。信号線は RS232C レベル変換 IC を経由して i.MX6ULL の UART コントローラ(UART3)に接続されています。

- 信号入出力レベル: RS232C レベル
- 最大データ転送レート: 230.4kbps

・フロー制御: CTS、RTS、DTR、DSR、DCD、RI

表 18.6 メインユニット CON3 信号配列

ピン番号	ピン名	I/O	説明
1	DCD	In	キャリア検出、i.MX6ULL の SNVS_TAMPER2 ピンに接続、CON4 の 1 ピンと共通
2	RXD	In	受信データ、i.MX6ULL の UART3_RX_DATA ピンに接続、CON4 の 3 ピンと共通
3	TXD	Out	送信データ、i.MX6ULL の UART3_TX_DATA ピンに接続、CON4 の 5 ピンと共通
4	DTR	Out	データ端末レディ、i.MX6ULL の GPIO1_IO00 ピンに接続、CON4 の 7 ピンと共通
5	GND	Power	電源(GND)
6	DSR	In	データセットレディ、i.MX6ULL の SNVS_TAMPER0 ピンに接続、CON4 の 2 ピンと共通
7	RTS	Out	送信要求、i.MX6ULL の UART3_CTS_B ピンに接続、CON4 の 4 ピンと共通
8	CTS	In	送信可能、i.MX6ULL の UART3_RTS_B ピンに接続、CON4 の 6 ピンと共通
9	RI	In	被呼表示、i.MX6ULL の SNVS_TAMPER1 ピンに接続、CON4 の 8 ピンと共通

表 18.7 メインユニット CON4 信号配列

ピン番号	ピン名	I/O	説明
1	DCD	In	キャリア検出、i.MX6ULL の SNVS_TAMPER2 ピンに接続、CON3 の 1 ピンと共通
2	DSR	In	データセットレディ、i.MX6ULL の SNVS_TAMPER0 ピンに接続、CON3 の 6 ピンと共通
3	RXD	In	受信データ、i.MX6ULL の UART3_RX_DATA ピンに接続、CON3 の 2 ピンと共通
4	RTS	Out	送信要求、i.MX6ULL の UART3_CTS_B ピンに接続、CON3 の 7 ピンと共通
5	TXD	Out	送信データ、i.MX6ULL の UART3_TX_DATA ピンに接続、CON3 の 3 ピンと共通
6	CTS	In	送信可能、i.MX6ULL の UART3_RTS_B ピンに接続、CON3 の 8 ピンと共通
7	DTR	Out	データ端末レディ、i.MX6ULL の GPIO1_IO00 ピンに接続、CON3 の 4 ピンと共通
8	RI	In	被呼表示、i.MX6ULL の SNVS_TAMPER1 ピンに接続、CON3 の 9 ピンと共通
9	GND	Power	電源(GND)
10	VCC_3.3V	Power	電源出力(VCC_3.3V)

CON3 と CON4 は、共通の信号が接続されており、同時に使用することはできません。どちらか一方のコネクタでのみ、ご使用ください。

18.6. メインユニット CON5(USB ホストインターフェース)

CON5 は USB ホストインターフェースです。2 段のコネクタを実装しており、下段の信号線は i.MX6ULL の USB コントローラ(USB OTG1)接続されています。上段の信号線はマルチプレクサを経由して、i.MX6ULL の USB コントローラ(USB OTG2)に接続されています。

マルチプレクサのセレクトピンは CON9 の 24 ピンで制御することが可能で、オープンもしくは High レベルを入力することで CON5 の上段、Low レベルを入力することで CON9 に USB OTG2 の接続先が変更されます。

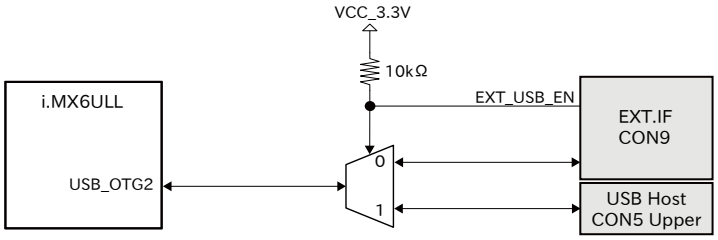


図 18.4 USB OTG2 の接続先の変更

下段に供給される電源(USB_OTG1_VBUS)は i.MX6ULL の UART1_RTS_B ピン(GPIO1_IO19)、上段に供給される電源(USB_OTG2_VBUS)は i.MX6ULL の CSI_MCLK ピン(GPIO4_IO17)で制御が可能で、High レベル出力で電源が供給され、Low レベル出力で電源が切断されます。

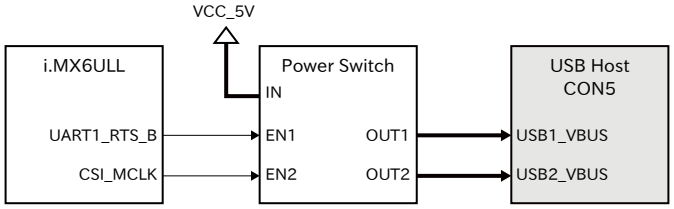


図 18.5 USB ホストインターフェースの電源制御

- データ転送モード
  - High Speed(480Mbps)
  - Full Speed(12Mbps)
  - Low Speed(1.5Mbps)

表 18.8 メインユニット CON5 信号配列

ピン番号	ピン名	I/O	説明
1	USB1_VBUS	Power	電源(USB_OTG1_VBUS)、i.MX6ULL の USB_OTG1_VBUS ピンに接続
2	USB1_DN	In/Out	USB1 のマイナス側信号、i.MX6ULL の USB_OTG1_DN ピンに接続
3	USB1_DP	In/Out	USB1 のプラス側信号、i.MX6ULL の USB_OTG1_DP ピンに接続
4	GND	Power	電源(GND)
5	USB2_VBUS	Power	電源(USB2_OTG2_VBUS)、i.MX6ULL の USB_OTG2_VBUS ピンに接続
6	USB2_DN	In/Out	USB2 のマイナス側信号、i.MX6ULL の USB_OTG2_DN ピンに接続
7	USB2_DP	In/Out	USB2 のプラス側信号、i.MX6ULL の USB_OTG2_DP ピンに接続
8	GND	Power	電源(GND)

18.7. メインユニット CON12(電源インターフェース)

CON12 は電源供給用インターフェースです。

メインユニット CON12 とサブユニット CON3 の電源(VCC_5V)供給ラインは接続されていますので、同時に電源を供給することはできません。どちらか一方からのみ電源を供給してください。

CON12 には DC ジャックが実装されており、「図 18.6. 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。AC アダプタのジャック形状は JEITA RC-5320A 準拠(電圧区分 2)です。

図 18.6 AC アダプタの極性マーク

AC アダプタを使用する際は、AC アダプタの DC プラグを Armadillo-IoT ゲートウェイ A6 に接続してから AC プラグをコンセントに挿してください。

18.8. メインユニット LED3、LED4(ユーザー LED)

LED3、LED4 は、ユーザー側で自由に利用できる LED です。

表 18.9 メインユニット LED3、LED4

部品番号	名称(色)	説明
LED3	ユーザー LED(赤)	i.MX6ULL の GPIO1_IO05 ピンに接続、(Low: 消灯、High: 点灯)
LED4	ユーザー LED(緑)	i.MX6ULL の GPIO1_IO08 ピンに接続、(Low: 消灯、High: 点灯)

18.9. メインユニット SW1(ユーザースイッチ)

SW1 は、ユーザー側で自由に利用できる押しボタンスイッチです。

表 18.10 メインユニット SW1 信号配列

部品番号	名称	説明
SW1	ユーザースイッチ	i.MX6ULL の JTAG_MOD ピンに接続、(Low: 押されていない状態、High: 押された状態)

18.10. サブユニット CON1(LTE アンテナインターフェース)

サブユニット CON1 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外付アンテナと接続します。

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。

18.11. サブユニット CON2(nanoSIM インターフェース)

サブユニット CON2 は、LTE モジュール(EMS31-J)用の nanoSIM インターフェースです。

表 18.11 サブユニット CON2 信号配列

ピン番号	ピン名	I/O	説明
C1	SIM_VCC	Power	SIM 電源、LTE モジュールの CCVCC に接続
C2	SIM_RST	Out	SIM リセット、LTE モジュールの CCRST に接続
C3	SIM_CLK	Out	SIM クロック、LTE モジュールの CCCLK に接続
C5	GND	Power	電源(GND)
C6	SIM_VPP	-	未接続
C7	SIM_I/O	In	SIM データ、LTE モジュールの CCIO に接続

サブユニット CON6 は活線挿抜に対応しておりません。SIM カードの挿抜は、本製品の電源を切断してから行ってください。

18.12. サブユニット CON3(拡張インターフェース)

サブユニット CON3 は、機能拡張用のインターフェースです。複数の機能(マルチプレクス)をもった i.MX6ULL の信号線が接続されています。

拡張インターフェースに接続する回路を設計される際は、シャットダウンモード時、I2C または拡張入出力ピンへ電圧が印可されない様にしてください。(例：入力トレラントのバッファ IC を介して、拡張インターフェースに接続等）シャットダウンモード時に電圧を印可した場合、製品が故障するおそれがあります。

メインユニット CON12 とサブユニット CON3 の電源(VCC_5V)供給ラインは接続されていますので、同時に電源を供給することはできません。どちらか一方からのみ電源を供給してください。

基板上に記載されているピン番号は、CON3 に実装されたコネクタのピン配列と相違がありますのでご注意ください。本マニュアルでの CON3 信号配列は、コネクタのピン配列に合わせて記載しています。

拡張できる機能の詳細につきましては、「アットマークテクノ Armadillo サイト」 [https://armadillo.atmark-techno.com/]からダウンロードできる『Armadillo-IoT ゲートウェイ A6 マルチプレクス表』をご参照ください。

表 18.12 サブユニット CON3 信号配列

ピン番号	ピン名	I/O	説明
1	VCC_5V	Power	電源(VCC_5V)
2	GND	Power	電源(GND)
3	RTC_BAT	Power	電源(RTC_BAT)、リアルタイムクロックの電源ピンに接続
4	GND	Power	電源(GND)
5	GPIO4_IO08	In/Out	拡張入出力、i.MX6ULL の NAND_DATA06 ピンに接続
6	GPIO4_IO09	In/Out	拡張入出力、i.MX6ULL の NAND_DATA07 ピンに接続
7	GPIO4_IO06	In/Out	拡張入出力、i.MX6ULL の NAND_DATA04 ピンに接続
8	GPIO4_IO07	In/Out	拡張入出力、i.MX6ULL の NAND_DATA05 ピンに接続
9	GPIO1_IO16	In/Out	拡張入出力、i.MX6ULL の UART1_TX_DATA ピンに接続
10	I2C2_SCL	In/Out	I2C クロック出力、i.MX6ULL の UART5_TX_DATA ピンに接続、基板上で 4.7kΩ プルアップされています。Armadillo-IoT 内の I2C デバイスに接続されており、I2C 以外の機能を割り当てて使用することはできません。また、一部の I2C アドレス(0x32, 0x48, 0x50)は使用できません。
11	GPIO1_IO17	In/Out	拡張入出力、i.MX6ULL の UART1_RX_DATA ピンに接続
12	I2C2_SDA	In/Out	I2C データ入出力、i.MX6ULL の UART5_RX_DATA ピンに接続、基板上で 4.7kΩ プルアップされています。Armadillo-IoT 内の I2C デバイスに接続されており、I2C 以外の機能を割り当てて使用することはできません。また、一部の I2C アドレス(0x32, 0x48, 0x50)は使用できません。
13	VCC_3.3V	Power	電源(VCC_3.3V)
14	GND	Power	電源(GND)

18.12.1. 動作モードと拡張インターフェース各機能の状態

Armadillo-IoT ゲートウェイ A6 の動作モードにより、拡張インターフェースの状態が変わります。

表 18.13 動作モードと拡張インターフェース各機能の状態

項目	アクティブモード時	スリープモード時	スリープ(SMS 起床可能)モード時	シャットダウンモード時
VCC_3.3V 電源出力状態	ON	ON	ON	OFF
I2C 入出力状態	任意	アクティブモード時の入出力状態を維持	アクティブモード時の入出力状態を維持	電源 OFF のため、Low レベル [a]
拡張入出力状態	任意	アクティブモード時の入出力状態を維持	アクティブモード時の入出力状態を維持	電源 OFF のため、Low レベル [a]

[a]シャットダウンモード時、外部から電圧は印可できません。

18.13. サブユニット SW1(ユーザースイッチ)

SW1 は、起動デバイス設定スイッチです。スイッチの状態で、起動デバイスを設定することができます。

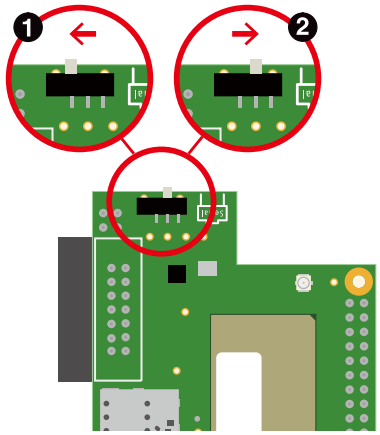


図 18.7 スライドスイッチの設定

❶ 起動デバイスは microSD になります。

❷ 起動デバイスは eMMC になります。

19. 寸法図

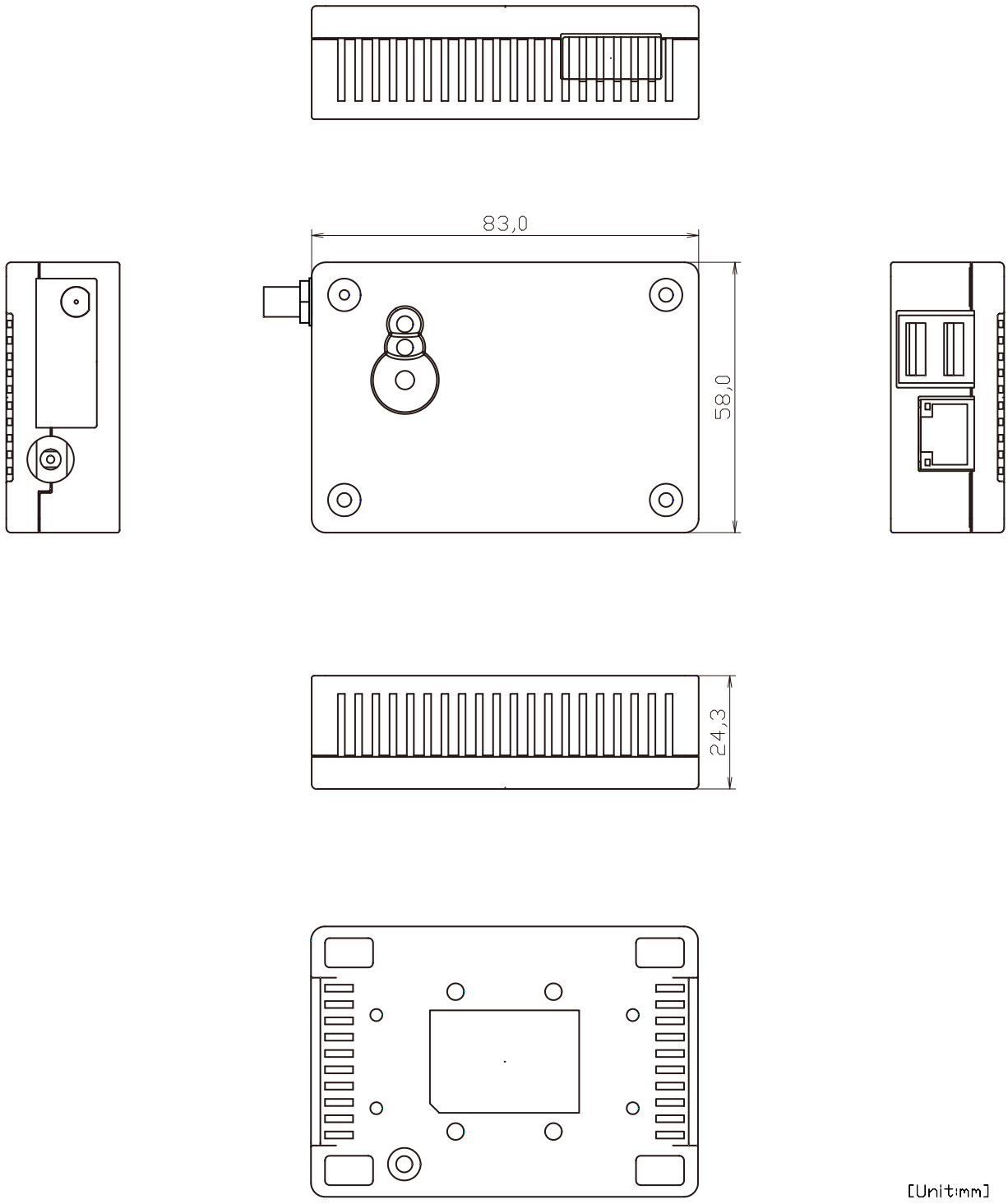


図 19.1 筐体寸法図

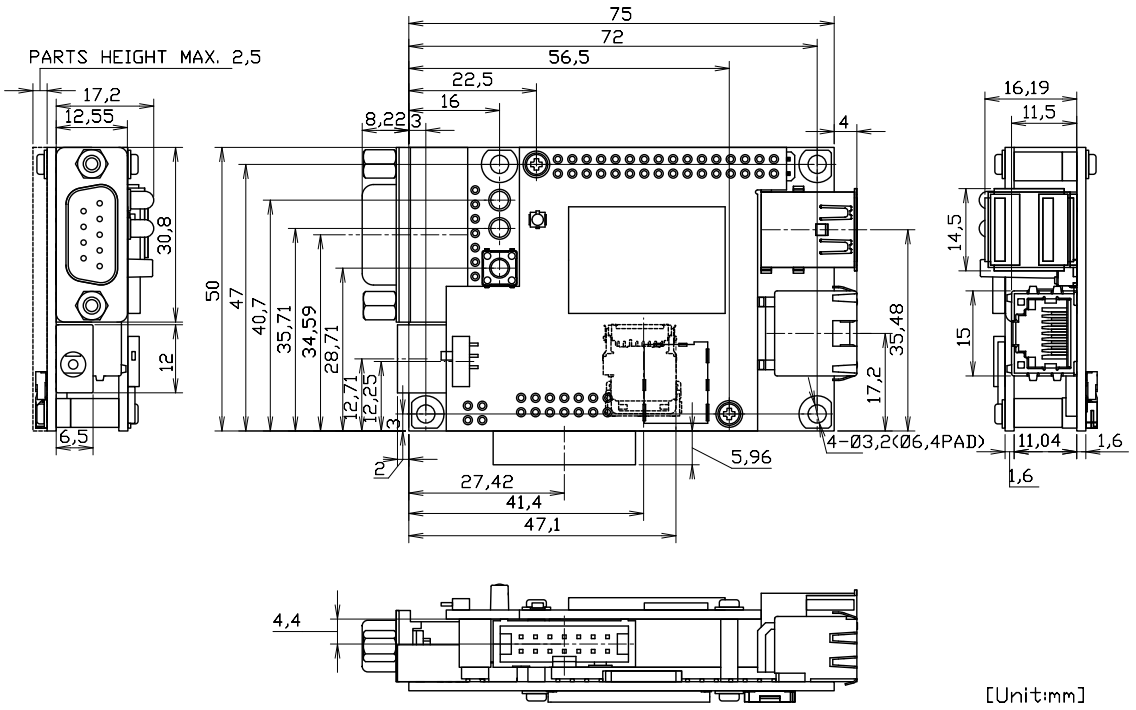


図 19.2 基板寸法図

DXF 形式の基板形状図を「アットマークテクノ Armadillo サイト」から「購入者向けの限定公開データ」としてダウンロード可能です。

20. オプション品

本章では、Armadillo-IoT ゲートウェイ A6 のオプション品について説明します。

表 20.1 Armadillo-IoT ゲートウェイ A6 関連のオプション品

名称	型番
3G/LTE 用 外付けアンテナセット 04	OP-ANT-3GLTE-04K
D-Sub9/10 ピン シリアル変換ケーブル（ロックなし）	OP-SCDSUB-00
AC アダプタ（5V/2.0A EIAJ#2）	OP-AC5V5-00

20.1. 3G/LTE 用 外付けアンテナセット 04

20.1.1. 概要

3G/LTE 用 外付けアンテナセット 04 は、LTE モジュール(EMS31-J)対応のアンテナセットです。

20.1.2. セット内容

- 外付けアンテナ
- アンテナケーブル

20.1.3. 組み立て

アンテナケーブルは、LTE アンテナインターフェース(サブユニット CON1)に取り付けます。

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。

アンテナケーブルを引き抜く際は、専用の引き抜き治具(U.FL-LP-N-2/ヒロセ電機 等)を用いて行うことを推奨します。 引き抜き治具を用いずに引き抜いた場合に、コネクタの変形やケーブルの断線等の原因となります。

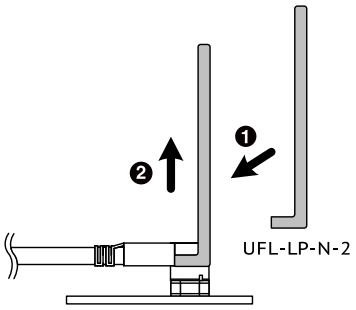


図 20.1 外付けアンテナケーブルの引き抜き方法

20.1.4. 形状図

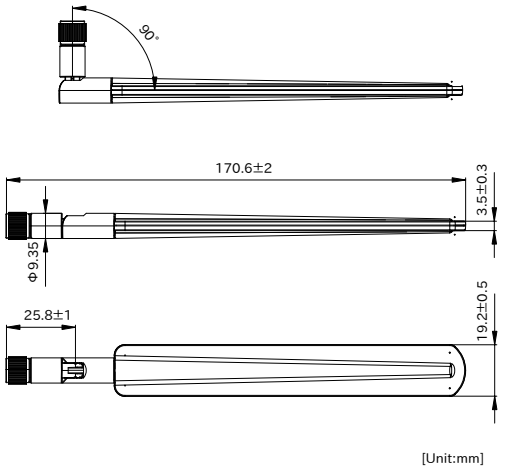


図 20.2 アンテナ形状

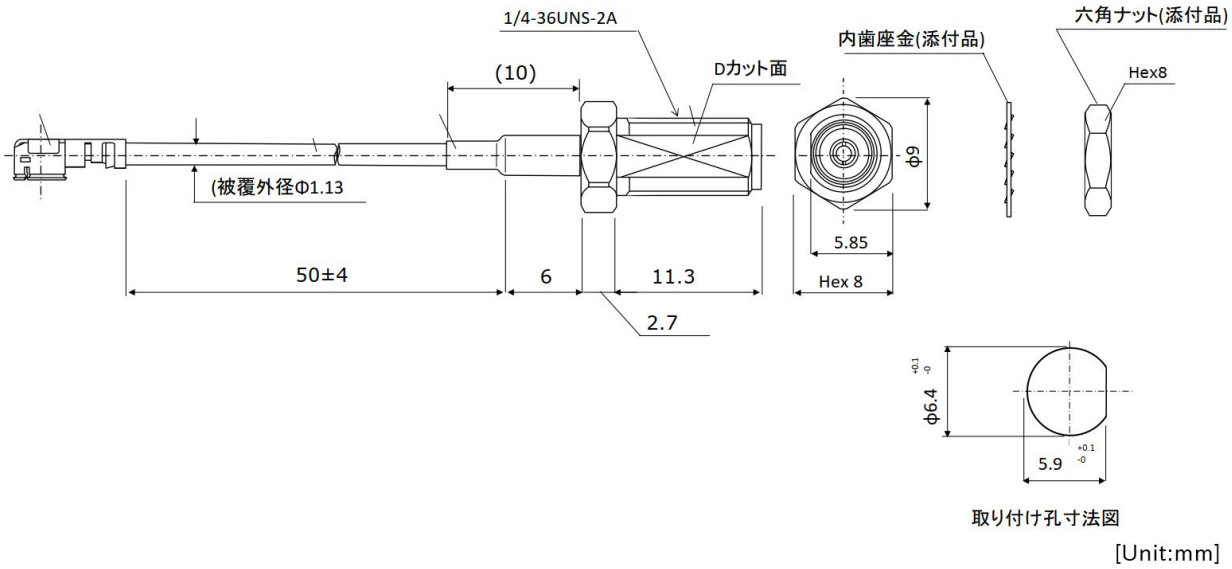


図 20.3 アンテナケーブル形状

21. 設計情報

本章では、Armadillo-IoT ゲートウェイ A6 の信頼性向上のための設計情報について説明します。

21.1. 放射ノイズ

サブユニット CON3(拡張インターフェース)を使用して、Armadillo-IoT ゲートウェイ A6 と拡張基板を接続すると、放射ノイズが問題になる場合があります。特に、オーディオアンプのような電力が大きく変動するデバイスを拡張基板に搭載する場合、コネクタの GND 線の接続のみでは強い放射ノイズが発生する可能性があります。放射ノイズを減らすために、以下の対策が効果的です。

- シールド付ケーブルを使用する
  - 長さが余る場合は、ケーブルを折りたたむ
  - シールドは拡張基板の GND に接続する
- Armadillo-IoT ゲートウェイ A6 の GND(固定穴等)と拡張基板の GND を太い導線や金属スペーサ等で接続する
- 未使用の拡張ピンは Low レベル出力とする
- 使用する拡張ピンはコンデンサ(1000pF 程度)を介して GND と接続する

21.2. ESD/雷サージ

Armadillo-IoT ゲートウェイ A6 の ESD 耐性を向上させるために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6 を金属筐体に組み込み、GND(固定穴等)を金属ねじ等で接続する
- 金属筐体を接地する

また、Armadillo-IoT ゲートウェイ A6 に接続されたケーブルが屋外に露出するような設置環境では、ケーブルに侵入した雷サージ等のストレスによりインターフェース回路が破壊される場合があります。ストレスへの耐性を向上させるために、以下の対策が効果的です。

- Armadillo-IoT ゲートウェイ A6 と通信対向機の GND 接続を強化する
- シールド付きのケーブルを使用する

22. Howto

本章では、Armadillo-IoT ゲートウェイ A6 のソフトウェアをカスタマイズする方法などについて説明します。

22.1. Device Tree とは

Device Tree とは、ハードウェア情報を記述したデータ構造体です。ハードウェアの差分を Device Tree に記述することによって、1 つの Linux カーネルイメージを複数のハードウェアで利用することができるようになります。

Device Tree に対応しているメリットの 1 つは、ハードウェアの変更に対するソフトウェアの変更が容易になることです。例えば、サブユニット CON3(拡張インターフェース) に接続する拡張基板を作成した場合、主に C 言語で記述された Linux カーネルのソースコードを変更する必要はなく、やりたいことをより直感的に記述できる DTS(Device Tree Source)の変更で対応できます。

ただし、Device Tree は「データ構造体」であるため、ハードウェアの制御方法などの「処理」を記述することができない点に注意してください。Device Tree には、CPU アーキテクチャ、RAM の容量、各種デバイスのベースアドレスや割り込み番号などのハードウェアの構成情報のみが記述されます。

Device Tree 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/devicetree/`)、devicetree.org で公開されている「Device Tree Specification」を参照してください。

> **DeviceTree: The Devicetree Specification**
>
> https://www.devicetree.org/

Linux カーネルのソースコードに含まれている初期出荷状態での DTS、及び関連するファイルを次に示します。

初期出荷状態での DTS、及び関連するファイル
- `arch/arm/boot/dts/armadillo-640.dts`
- `arch/arm/boot/dts/armadillo-iotg-a6.dts`
- `arch/arm/boot/dts/imx6ull.dtsi`
- `arch/arm/boot/dts/imx6ul.dtsi`

22.2. 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。

22.2.1. 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。

```
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version].tar.gz
[ATDE ~]$ tar xf linux-v4.14-at[version].tar.gz
[ATDE ~]$ ls
initramfs_a600-[version].cpio.gz linux-v4.14-at[version]  linux-v4.14-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。

```
[ATDE ~]$ cd linux-v4.14-at[version]
[ATDE ~/linux-v4.14-at[version]]$ ln -s ../initramfs_a600-[version].cpio.gz
initramfs_a600.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。

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm armadillo-640_defconfig
[ATDE ~/linux-v4.14-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 4.14-at1 Kernel Configuration
--------------------------------------------------------------------------------
 ----------------- Linux/arm 4.14-at1 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  [ ]
   ---------------------------------------------------------------------
            General setup  --->
        [ ] Enable loadable module support  ----
        [*] Enable the block layer  --->
            System Type  --->
            Bus support  --->
            Kernel Features  --->
            Boot options  --->
            CPU Power Management  --->
            Floating point emulation  --->
            Userspace binary formats  --->
            Power management options  --->
        [*] Networking support  --->
            Device Drivers  --->
            Firmware Drivers  --->
            File systems  --->
            Kernel hacking  --->
```

```
            Security options  --->
        -*- Cryptographic API  --->
            Library routines  --->
        [ ] Virtualization  ----



  ------------------------------------------------------------------------
------------------------------------------------------------------------------
        <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。

1. ビルド

```
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
LOADADDR=0x82000000 uImage
[ATDE ~/linux-v4.14-at[version]]$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

2. 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。

```
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/uImage
arch/arm/boot/uImage
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-iotg-a6.dtb
arch/arm/boot/dts/armadillo-iotg-a6.dtb
```

22.3. Device Tree をカスタマイズする

at-dtweb を利用して Device Tree をカスタマイズする方法を説明します。at-dtweb では、Web ブラウザ上のマウス操作で DTS および DTB を生成することができます。カスタマイズの対象はサブユニット上の拡張インターフェース(CON3)です。

22.3.1. at-dtweb のインストール

ATDE8 に at-dtweb パッケージをインストールします。

```
[ATDE ~]$ sudo apt update
[ATDE ~]$ sudo apt install at-dtweb
```

22.3.2. at-dtweb の起動

1. at-dtweb の起動開始

   at-dtweb の起動を開始するには、デスクトップ左上のアクティビティから「at-dtweb」と入力し、at-dtweb のアイコンをクリックしてください。

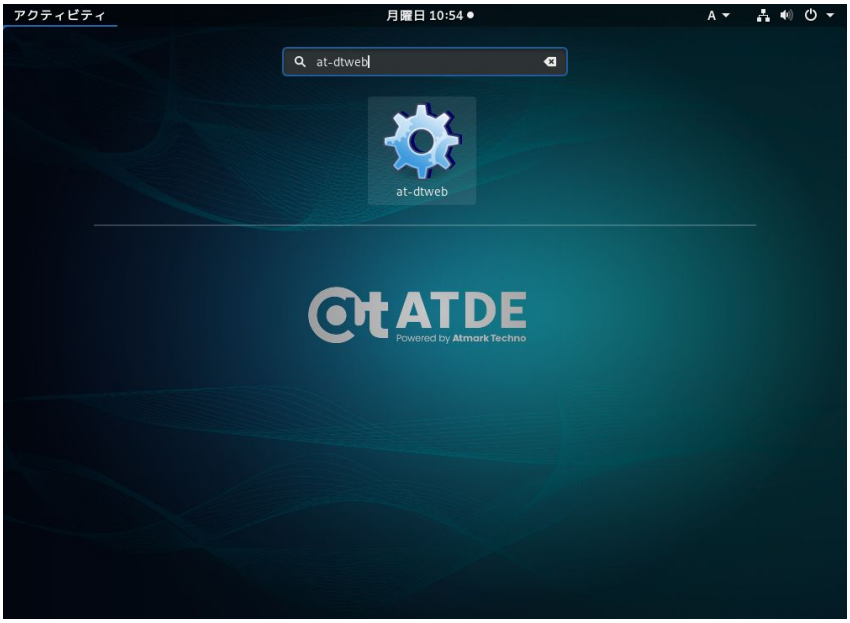


図 22.1 at-dtweb の起動開始

2. ボードの選択

   ボードを選択します。 `Armadillo-IoT_A6` を選択して、「OK」をクリックします。

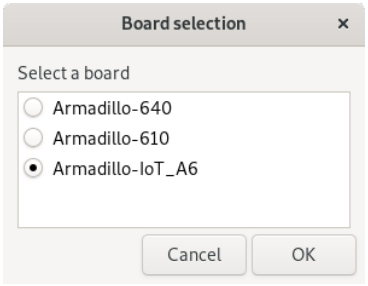


図 22.2 ボード選択画面

3. Linux カーネルディレクトリの選択

   Linux カーネルディレクトリを選択します。コンフィギュレーション済みの Linux カーネルディレクトリを選択して、「OK」をクリックします。

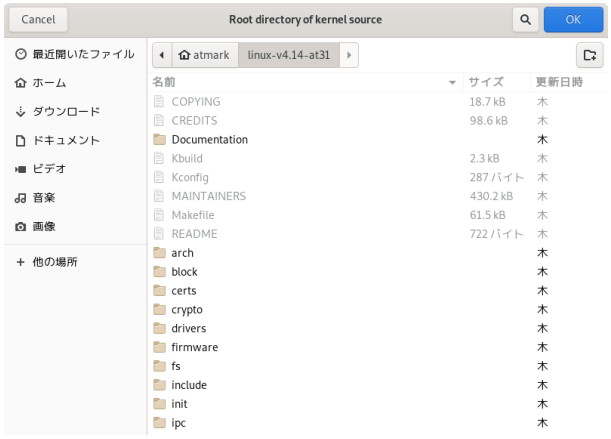


図 22.3 Linux カーネルディレクトリ選択画面

4. at-dtweb の起動完了

   at-dtweb が起動し、次のように画面が表示されます。

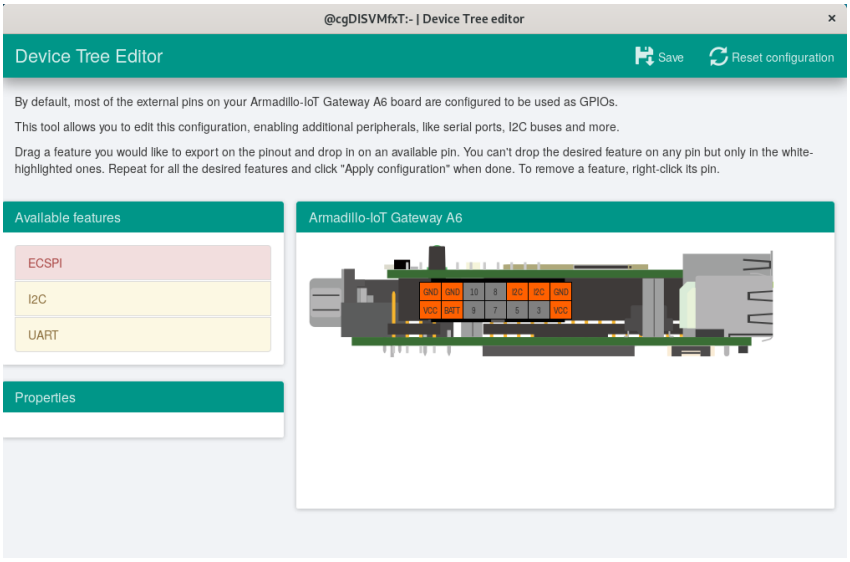


図 22.4 at-dtweb 起動画面

Linux カーネルは、事前にコンフィギュレーションされている必要があります。コンフィギュレーションの手順については「11.2.1. 手順：Linux カーネルをビルド」を参照してください。

22.3.3. Device Tree をカスタマイズ

22.3.3.1. 機能の選択

機能の選択は、ドラッグ&ドロップで行います。画面左上の「Available features」から有効にしたい機能をドラッグし、画面右側の「Armadillo-IoT Gateway A6」の白色に変化したピンにドロップします。例としてサブユニット CON3 9,11 ピンを I2C3(SCL/SDA)に設定します。

何も機能が選択されていないピンには GPIO の機能が割り当てられます。

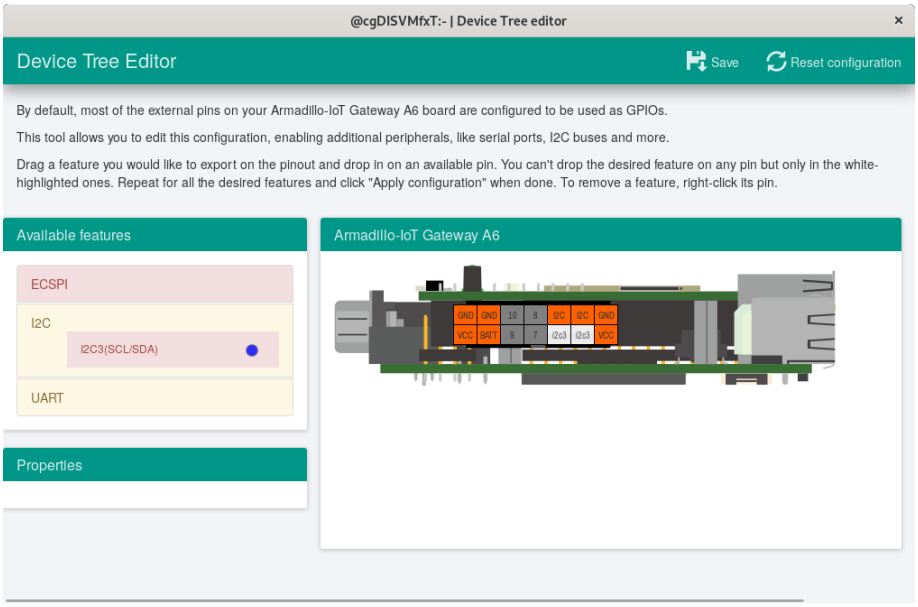


図 22.5 I2C3(SCL/SDA)のドラッグ

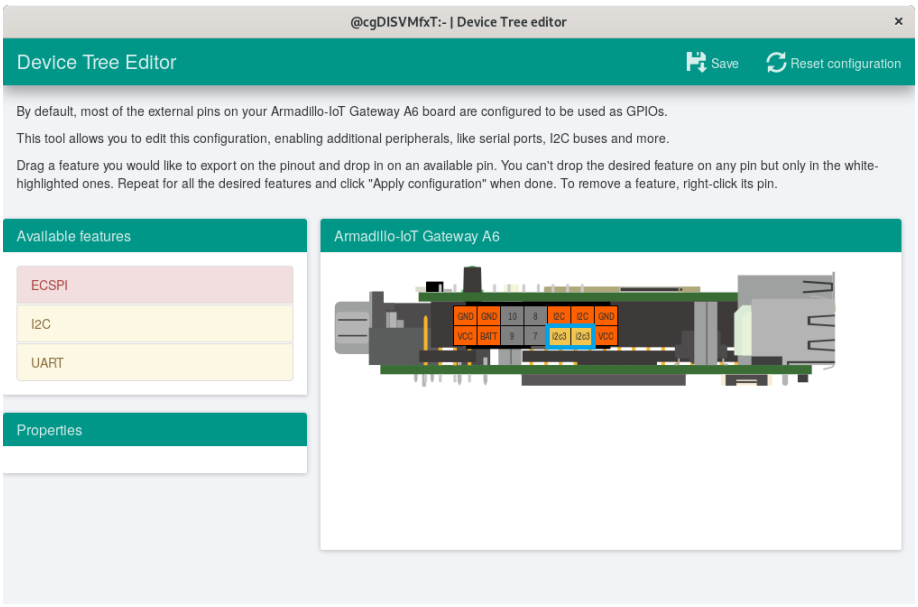


図 22.6 サブユニット CON3 9,11 ピンへのドロップ

22.3.3.2. プロパティの設定

いくつかの機能にプロパティを設定することができます。画面右側の「Armadillo-IoT Gateway A6」に選択した機能を左クリックすると、画面左下の「Properties」からプロパティを選択することができます。

例としてサブユニット CON3 5,6,7,8 ピンの ECSPI4(SCLK/MOSI/MISO/SS0) の spl-max-frequency プロパティを設定します。

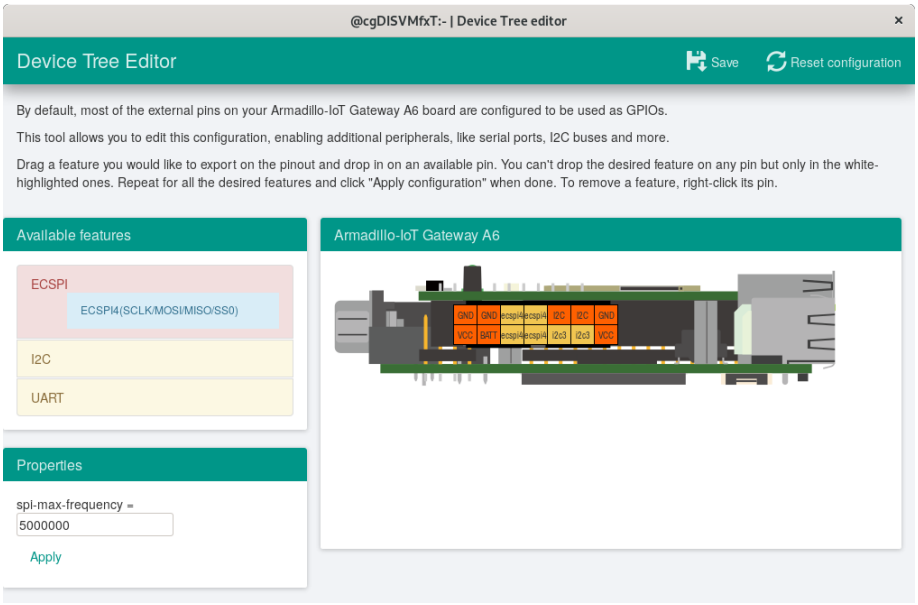


図 22.7 プロパティの設定

設定したプロパティを確定させるには「Apply」をクリックします。

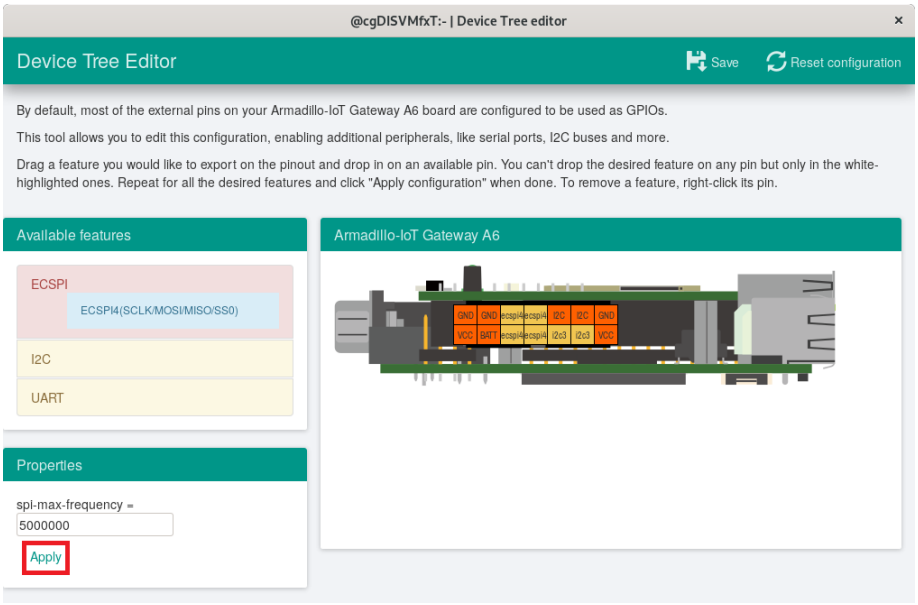


図 22.8 プロパティの保存

22.3.3.3. 機能の削除

全ての機能を削除する場合は、画面右上の「Reset configuration」をクリックします。機能ごとに削除する場合は、画面右側の「Armadillo-IoT Gateway A6」のピンを右クリックして「Remove」をクリックします。

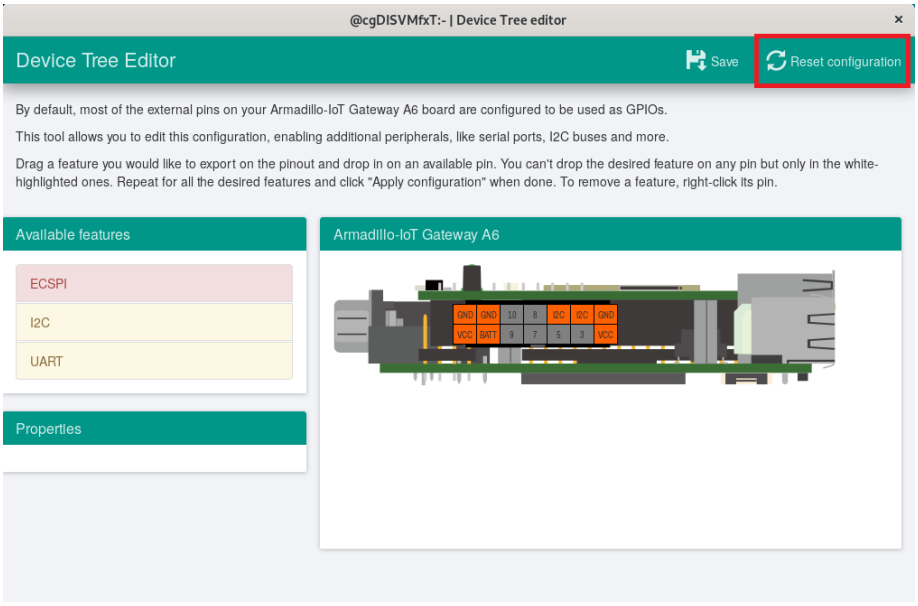


図 22.9 全ての機能の削除

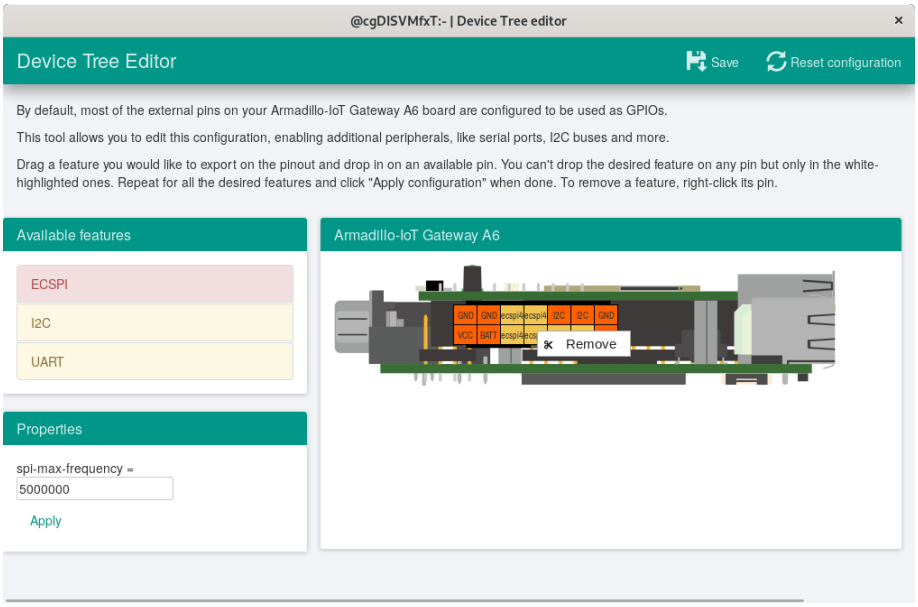


図 22.10 ECSPI4(SCLK/MOSI/MISO/SS0)の削除

22.3.3.4. DTS/DTB の生成

DTS および DTB を生成するには、画面右上の「Save」をクリックします。

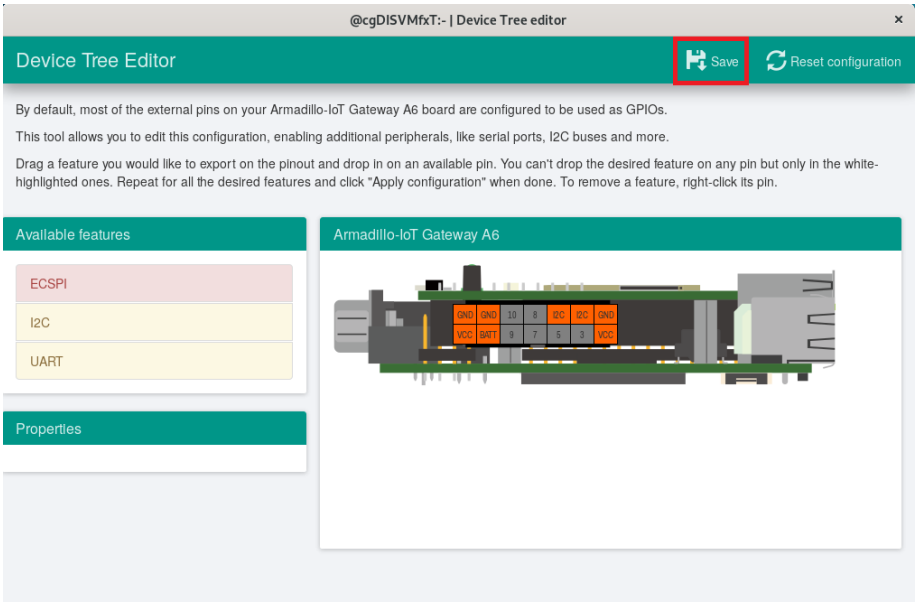


図 22.11 DTS/DTB の生成

「Device tree built!」と表示されると、DTS および DTB の生成は完了です。

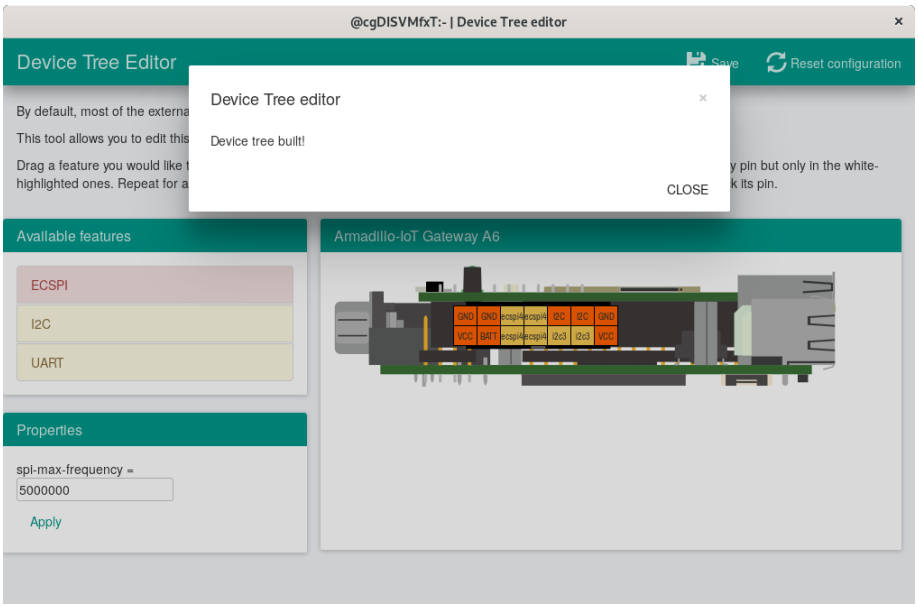


図 22.12 DTS/DTB の生成完了

ビルドが終了すると、arch/arm/boot/dts/以下に DTS/DTB が作成されています。

```
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-iotg-a6-expansion-interface.dtsi
armadillo-iotg-a6-expansion-interface.dtsi
[ATDE ~/linux-v4.14-at[version]]$ ls arch/arm/boot/dts/armadillo-iotg-a6-at-dtweb.dtb
armadillo-iotg-a6-at-dtweb.dtb
```

22.4. 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 ゲートウェイ A6 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 が稼働している間は、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 ゲートウェイ A6 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。

22.4.1. 保護機能の使用方法

eMMC への書き込み保護を行うには、kernel の起動オプションに"overlay=50%"("=50%"は省略可、"overlay"のみ書くと RAM を 256MByte 使用)というパラメータを追加するだけです。

パラメータを追加すると、debian の起動前に initramfs 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 は起動前と変わ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 ゲートウェイ A6 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
```

また、オプションの指定を解除するには次のようにコマンドを実行してください。

```
=> setenv optargs
=> saveenv
```

22.4.2. 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 ゲートウェイ A6 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。

22.5. eMMC の GPP(General Purpose Partition) を利用する

GPP に squashfs イメージを書き込み、Armadillo の起動時に自動的にマウントする方法を紹介します。

22.5.1. squashfs イメージを作成する

この作業は ATDE8 上で行います。

squashfs-tools パッケージに含まれている mksquashfs コマンドを使用して squashfs イメージを作成します。

```
[ATDE]$ mkdir sample
[ATDE]$ echo "complete mounting squashfs on eMMC(GPP)" > sample/README
[ATDE]$ mksquashfs sample squashfs.img
```

図 22.13 squashfs イメージの作成

22.5.2. squashfs イメージを書き込む

以降の作業は Armadillo 上で行います。

「22.5.1. squashfs イメージを作成する」で作成した squashfs イメージを、USB メモリ利用するなどして Armadillo-IoT ゲートウェイ A6 にコピーし、GPP に書き込みます。

ユーザー領域として使用可能な GPP は /dev/mmcblk0gp2 および /dev/mmcblk0gp3 です。

GPP への書き込みを行う際は、誤って /dev/mmcblk0gp0 や /dev/mmcblk0gp1 に書き込みを行わないよう、十分に注意してください。

```
[armadillo]# mount /dev/sda1 /mnt
[armadillo]# dd if=/mnt/squashfs.img of=/dev/mmcblk0gp2 conv=fsync
[armadillo]# umount /mnt
```

22.5.3. GPP への書き込みを制限する

GPP の全ブロックに対して Temporary Write Protection をかけることにより、GPP への書き込みを制限することができます。Temporary Write Protection は電源を切断しても解除されません。

Temporary Write Protection をかけるには、mmc-utils パッケージに含まれている mmc コマンドを使用します。

```
[armadillo]# apt install mmc-utils
```

図 22.14 mmc-utils のインストール

GPP の全ブロックに対して Temporary Write Protection をかけるには、次のようにコマンドを実行します。

```
[armadillo]# mmc writeprotect user get /dev/mmcblk0gp2 ❶
Write Protect Group size in blocks/bytes: 16384/8388608
Write Protect Groups 0-0 (Blocks 0-16383), No Write Protection
[armadillo]# mmc writeprotect user set temp 0 16384 /dev/mmcblk0gp2 ❷
```

図 22.15 eMMC の GPP に Temporary Write Protection をかける

❶ /dev/mmcblk0gp2 のブロック数を確認します。コマンドの出力を見ると /dev/mmcblk0gp2 が 16384 ブロックあることがわかります。

❷ /dev/mmcblk0gp2 の全ブロックに Temporary Write Protection をかけます。

Temporary Write Protection を解除するには、次のコマンド実行します。

```
[armadillo]# mmc writeprotect user set none 0 16384 /dev/mmcblk0gp2
```

22.5.4. 起動時に squashfs イメージをマウントされるようにする

/etc/fstab を変更し、起動時に squashfs イメージがマウントされるようにします。

```
[armadillo]# mkdir -p /opt/sample ❶
[armadillo]# vi /etc/fstab
:
:(省略)
:
/dev/mmcblk0gp2 /opt/sample squashfs defaults,nofail 0 0 ❷
```

❶ squashfs イメージをマウントするディレクトリを作成します

❷ 最終行にこの行を追加します。これで、/dev/mmcblk0gp2 が /opt/sample にマウントされるようになります。

Armadillo の再起動後、 /opt/sample/README の内容が正しければ完了です。

```
[armadillo]# reboot
:
: (省略)
:
Debian GNU/Linux 10 armadillo ttymxc0

armadillo login:
[armadillo]# ls /opt/sample
README
[armadillo]# cat /opt/sample/README
complete mounting squashfs on eMMC(GPP)
```

23. ユーザー登録

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

ユーザー登録すると次のようなサービスを受けることができます。

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

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

アットマークテクノ Armadillo サイト

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

23.1. 購入製品登録

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

購入製品登録の詳しい手順は以下の URL をご参照ください。

Armadillo-IoT ゲートウェイ A6 購入製品登録

https://armadillo.atmark-techno.com/armadillo-iot-a6/register

付録 A eFuse

Armadillo-IoT ゲートウェイ A6 で採用している CPU (i.MX6ULL) には、一度しか書き込むことのできない eFuse が搭載されています。 eFuse には、 CPU がブートする時の設定や MAC アドレスなどが書かれます。Armadillo-IoT ゲートウェイ A6 は組み込み機器を作り込むエンジニアを対象にした製品ですので、 eFuse もユーザーに開放し、細かな制御を可能にしています。しかし eFuse はその性質上、一度書き間違うと直すことができません。十分に注意してください。

eFUSE は一度書き込むと元に戻すことができません。eFUSE の設定によっては Armadillo-IoT ゲートウェイ A6 が正常に動作しなくなる可能性がありますので、書き込みを行う際には細心の注意を払うようお願いいたします。eFUSE の設定によって異常が起こった場合は保証対象外となります。

MAC アドレスは Armadillo-IoT ゲートウェイ A6 の出荷時に書き込まれているので、新たに書き込む必要はありません。この章では U-Boot を使って eFuse の書き換えを行い、ブートモードを制御する方法を説明します。

eFuse を変更する場合は、必ず「i.MX 6ULL Applications Processor Reference Manual [https://www.nxp.com/docs/en/reference-manual/IMX6ULLRM.pdf]」を参照してください。重要な章は、以下の 4 つです。

- Chapter 5: Fusemap
- Chatper 8: System Boot
- Chapter 37: On-Chip OTP Controller
- Chapter 58: Ultra Secured Digital Host Controller

以降、本章では i.MX 6ULL Applications Processor Reference Manual を「リファレンスマニュアル」と呼びます。

章番号や章タイトルは、i.MX 6ULL Applications Processor Reference Manual Rev. 1, 11/2017 現在の情報です。異るリビジョンのリファレンスマニュアルでは、章番号およびタイトルが異なる場合があります。

A.1. ブートモードとジャンパーピン

Armadillo-IoT ゲートウェイ A6 では サブユニット SW1(ユーザースイッチ) を microSD 側に設定することで、 JP1 及び JP2 双方ともショートさせることができます。詳細は、「4.8. スライドスイッチの設定について」を参照ください。

A.1.1. ブートモードと JP2

i.MX6ULL にはブートモードを決める BOOT_MODE0 と BOOT_MODE1 というピンがあります。 BOOT_MODE0 は GND になっているので必ず 0 になります。BOOT_MODE1 は JP2 に接続されています。 JP2 がショー

トされていると BOOT_MODE1 は 1 になり Internal Boot モードになります。開放されていると 0 となり、Boot From Fuses というモードになります。

Internal Boot モードでは、 on-chip boot ROM に書き込まれているコードが実行し、ブート可能なデバイスを検索します。リファレンスマニュアル「8.5 Boot devices (internal boot)」に、i.MX6ULL がブートできるデバイスの一覧が記載されています。Armadillo-IoT ゲートウェイ A6 では、そのうちオンボード eMMC と microSD カードに対応しています。Internal Boot モードでは、GPIO によって eFuse の設定を上書き (override) できるようになっています。つまり eFuse の設定がどうなっていようと、GPIO のピンでブートデバイスを決めることができます。

Boot From Fuses モードでは、単純に言えば GPIO による override が禁止され eFuse に書き込まれた状態でしかブートしません。この機能を有効にすることで、フィールドに出した製品が悪意ある人によって意図していないブートをし、被害が出ることを防ぐことができます。(もちろん、ブート後に root アカウントを乗っ取られるような作りでは、意味がありませんが…)

A.1.2. ブートデバイスと JP1

Internal Boot モードでは、GPIO によって eFuse の設定を上書き (override) できるようになってると紹介しましたが、JP1 はまさにこの機能を使っています。 JP1 は LCD1_DATA05 と LCD1_DATA11 の制御をしていますが、これらのピンはそれぞれ BOOT_CFG1[5] と BOOT_CFG2[3] を override しています。「8.3.2 GPIO boot overrides」の 表「8-3. GPIO override contact assignments」を確認してください。

ややこしい事に、この BOOT_CFG で始まる eFUSE は、リファレンスマニュアルの中では eFuse のアドレスでも表記されています。 BOOT_CFG1 は eFuse のアドレスで言うと 0x450 の下位 8 bit つまり 0x450[7:0] であり、 BOOT_CFG2 は上位 8 bit つまり 0x450[15:8] にあたります。これは「5.1 Boot Fusemap」の表「5-5. SD/eSD Boot Fusemap」または表「5-6. MMC/eMMC Boot Fusemap」を確認することでわかります。

さらにややこしい事に、eFuse を書き込む場合にはこれら全ての値が使えず、On-Chip OTP Controller の bank と word の値が必要になります。これらの値は リファレンスマニュアルの「On-Chip OTP Controller」を参照してください。後で出てきますが Boot From Fuses で使用する BT_FUSE_SEL という eFuse のように GPIO による override ができないものもあります。

表 A.1 GPIO override と eFuse

信号名	eFuse 名	eFuse アドレス	OCOTP 名	Bank	Word
LCD1_DATA05	BOOT_CFG1[5]	0x450[5]	OCOTP_CFG4	0	5
LCD1_DATA11	BOOT_CFG2[3]	0x450[11]	OCOTP_CFG4	0	5
N/A	BT_FUSE_SEL	0x460[4]	OCOTP_CFG5	0	6

Armadillo-IoT ゲートウェイ A6 では SD カード または eMMC からのブートになるので、ブートデバイスを選択する eFuse BOOT_CFG1[7:4] は、010x または 011x になります。

リファレンスマニュアル「8.5.3.1 Expansion device eFUSE configuration」には、さらに詳しく SD/MMC デバイスの設定について記載されています。テーブル「8-15. USDHC boot eFUSE descriptions」によれば、eFuse の 0x450[7:6] が 01 の場合に SD/MMC デバイスからブートすることを決めています。さらに 0x450[5] が 0 なら SD が、 0x450[5] が 1 なら MMC が選択されます。つまり、4 から 7 bit までの間で 5 bit 目だけが MMC か SD かを決めています。 BOOT_CFG1[5] が 0 の場合はコントローラーは SD デバイスが繋がっている前提で、 BOOT_CFG1[5] が 1 の場合は MMC デバイスが繋っている前提で動作します。

i.MX6ULL には、SD/MMC のコントローラーである uSDHC が 2 つ搭載されています。 Armadillo-IoT ゲートウェイ A6 では、eMMC が uSDHC1 に、 microSD カードが uSDHC2 に接続されています。ブート時にどちらのコントローラーからブートするかを決めている eFuse が 0x450[12:11] です。0x450[12:11] が 00 であれば uSDHC1 つまりオンボード eMMC から、01 であれば uSDHC2 つまり

microSD カードからブートします。言い換えると Armadillo-IoT ゲートウェイ A6 でオンボード eMMC からブートしたい場合は、0x450[5] を 1 に、 0x450[12:11] を 00 にします。逆に microSD カードから起動したい場合は 0x450[5] を 0 に、0x450[12:11] を 01 にします。

表 A.2 ブートデバイスと eFuse

ブートデバイス	eFuse 0x450[5]	0x450[12:11]
オンボード eMMC	1	00
microSD カード	0	01

A.2. eFuse の書き換え

Armadillo-IoT ゲートウェイ A6 では、U-Boot のコマンドによって eFuse の書き換えをサポートしています。 U-Boot については 「10. ブートローダー (U-Boot) 仕様」 を参照してください。

eFuse の書き換えは、 fuse コマンドを使います。

U-Boot の fuse コマンドのソースコードは、以下の 2 つです。

- cmd/fuse.c
- drivers/misc/mxc_ocotp.c

```
=> help fuse
fuse - Fuse sub-system

Usage:
fuse read <bank> <word> [<cnt>] - read 1 or 'cnt' fuse words,
    starting at 'word'
fuse sense <bank> <word> [<cnt>] - sense 1 or 'cnt' fuse words,
    starting at 'word'
fuse prog [-y] <bank> <word> <hexval> [<hexval>...] - program 1 or
    several fuse words, starting at 'word' (PERMANENT)
fuse override <bank> <word> <hexval> [<hexval>...] - override 1 or
    several fuse words, starting at 'word'
=>
```

fuse read　eFuse の値を Shadow Register から読み出します。i.MX6ULL の eFuse は、すべて Shadow Register を持ち、起動時に eFuse から Shadow Register に値がコピーされます。詳しくはリファレンスマニュアル「37.3.1.1 Shadow Register Reload」を確認してください。

fuse sense　eFuse の値を eFuse から読み出します

fuse prog　eFuse の値を書き換えます

fuse コマンドは、 bank 、 word 、cnt 、 hexval を引数に取ります。

bank　eFuse のバンク番号

word　eFuse のワード番号

cnt　eFuse を読み出す個数

hexval 書き込む値

A.3. Boot From Fuses モード

A.3.1. BT_FUSE_SEL

Boot From Fuses を有効にするには、 eFuse に書き込んだ値が正しいことを i.MX6ULL に教える必要があります。そのための eFuse が BT_FUSE_SEL (0x460[4]) です。BOOT_MODE が 00 、つまり JP2 がオープンで、且つこのビットが 1 であれば Boot From Fuses モードになります。 BOOT_MODE が 00 でもこのビットが 0 であれば Boot From Fuses モードにはならず、 SD/MMC マニュファクチャリングモードやシリアルダウンロードモードになってしまいます。SD/MMC マニュファクチャリングモードについては 「8.12 SD/MMC manufacture mode」に、シリアルダウンロードモードについては「8.9 Serial Downloader」に記載されています。

Armadillo-IoT ゲートウェイ A6 では BOOT_MODE が 00 で、且つ BT_FUSE_SEL が 0 の場合は SD/MMC マニュファクチャリングモードで eMMC から起動します。Internal Boot モードで起動する場合は、JP2 をショートしてください。

A.3.2. eMMC からのブートに固定

オンボード eMMC からだけブートさせたい場合は、ブートデバイスの種類で MMC と、コントローラーで uSDHC1 を選択することで可能です。忘れずに BT_FUSE_SEL を 1 にします。

オンボード eMMC のスペックは、以下の通りです。リファレンスマニュアル 8.5.3 Expansion device および 表「5-6. MMC/eMMC Boot Fusemap」を確認してください。「可変」列が「不」となっている値は、変更しないでください。例えば、オンボード eMMC は 1.8 V に対応していません。 bit 9 の SD Voltage Selection で 1 の 1.8 V では動作しません。

表 A.3 オンボード eMMC のスペック

名前	Bit	eFuse	値	bit 列	可変
BOOT_CFG2	[15:13]	Bus Width	8 bit	010	不
	[12:11]	Port Select	uSDHC1	00	不
	[10]	Boot Frequencies	500 / 400 MHz	00	可
	[9]	SD Voltage Selection	3.3 V	0	不
	[8]	-	-	0	-
BOOT_CFG1	[7:5]	eMMC	-	011	不
	[4]	Fast Boot	Regular	0	可
	[3]	SD/MMC Speed	High	0	不
	[2]	Fast Boot Acknowledge Disable	Enabled	0	可
	[1]	SD Power Cycle Enable	Enabled	1	可
	[0]	SD Loopback Clock Source Sel	SD Pad	0	不

値を見易いように、 BOOT_CFG2 を上にしています。 BOOT_CFG1 と BOOT_CFG2 は、OCOTP_CFG4 にマップされており Bank 0 Word 5 です。つまり 010000000 01100010 の 16 bit (0x4062) を Bank 0 Word 5 に書き込めば良いことが分ります。 BOOT_CFG3 と BOOT_CFG4 はここでは無視します。

BT_FUSE_SEL は Bank 0 Word 6 の 4 bit 目になるので 0x10 を書き込みます。

```
=> fuse read 0 5
Reading bank 0:

Word 0x00000005: 00000000
=> fuse prog 0 5 0x4060
Programming bank 0 word 0x00000005 to 0x00004060...
Warning: Programming fuses is an irreversible operation!
         This may brick your system.
         Use this command only if you are sure of what you are doing!

Really perform this fuse programming? <y/N>
y
=> fuse read 0 6
Reading bank 0:

Word 0x00000006: 00000000
=> fuse prog -y 0 6 0x10
Programming bank 0 word 0x00000006 to 0x00000010...
=> fuse read 0 6
Reading bank 0:

Word 0x00000006: 00000010

(電源入れなおしても、SD からブートしない)
```

fuse prog にオプション -y を付けると 「 Really perform this fuse programming? <y/N> 」と聞かれません。

これで eMMC からしか起動しない Armadillo-IoT ゲートウェイ A6 ができあがりました。

eMMC からしか起動しないので、あやまって eMMC に書き込まれている U-Boot を消してしまうと、二度と起動しないようになります。注意してください。

eMMC Fast Boot 機能を使う場合や Power Cycle を Enable にする場合は、当該ビットを 1 に変更してください。

同じ要領で、SD からだけしかブートしないようにすることも可能です。しかし eFuse によるブートデバイスの固定は、意図しないブートを防ぐことが目的です。 Armadillo-IoT ゲートウェイ A6 で

microSD からのブートに固定することは可能ですが、別の microSD カードを挿入されてしまうと、その別の microSD カードからブートしてしまうので目的を達成できません。理解してお使いください。

A.3.3. eFuse のロック

書き込んだ eFuse の値を変更されてしまっては、Boot From Fuse モードにしている意味がありません。i.MX6ULL では eFuse を変更できなくするビットも用意されています。

リファレンスマニュアル「5.3 Fusemap Descriptions Table」を確認してください。

Armadillo-IoT ゲートウェイ A6 製品マニュアル

改訂履歴

バージョン	年月日	改訂内容
1.0.0	2021/03/25	・1.0.0 発行
1.1.0	2021/04/09	・「6.2.4.3. LTE の接続を確認する」 を追加 ・「図 3.6. Armadillo-IoT ゲートウェイ A6 ブロック図」 から CON11 を削除 ・誤記を修正
1.2.0	2021/05/28	・「7.5. スリープモードへの遷移・起床時にスクリプトを実行する」 を追加 ・「17.4. 電源回路の構成」 を追加 ・「表 18.12. サブユニット CON3 信号配列」 の表記を修正 ・「17.3. 入出力インターフェースの電気的仕様」 について、入力と出力の仕様を分けて記載するよう変更 ・誤記、わかりにくい記載の修正
1.2.1	2021/06/04	・CON3 の信号配列をコネクタのピン配列に合わせるよう変更 ・拡張ボードを作成する際に必要な情報を追加

Armadillo-IoT ゲートウェイ A6 製品マニュアル
Version 1.2.1
2021/06/04