CanSat 電装開発 (β)

FTE 9 期　鄭潤賢

目次

第 1 章

はじめに

第Ⅰ部

電装脳になる

第 2 章

CanSat を作り上げるもの達

2.1 CanSat ってなんだ

CanSat とは何だろうか．ググれば答えは簡単に手に入る．CanSat でググれば，学生が作る擬似人工衛星だとか小型惑星探査機といったような答えが手に入る．もちろんこのような答えも間違いではないだろう．だが，いざ CanSat を作ろうとするとこのような説明はしっくりこない．なぜなら人工衛星の電装を作ったことのある人などほとんどいないからだ．もう少し CanSat というものを電装を作る工程がイメージ出来るような形に変形できないだろうか．

CanSat はもっと大雑把に言えば「宇宙開発を目的とした自律型ロボット」であると言えるだろう．この CanSat の説明でのポイントは 3 つある．1 つ目は宇宙開発を目的としている点である．つまり，製作される CanSat と呼ばれるものは宇宙開発のために使用されることを想定して作られたものである．2 つ目は自律型であるという点である．これは CanSat と呼ばれるものは誰かが操縦することで動作するだけでなく，操縦者なしでも目的の動作をすることが出来るという意味である．最後の 3 つ目はロボットという点である．ロボットの定義は様々だが，ここでは人の代わりに何らかの作業を行う機械という意味で使用した．これで我々は CanSat の電装はロボットの電装に似ているということがわかった．それではロボットの電装とはなんだろうか．

ロボットの電装を知るためにロボットを分解してみることにしよう．例として人型ロボットを考えてみよう．人型ロボットは手足の関節を動かせる．それならきっとロボット内部に関節を動かす何らかの装置があるに違いない．それはなんだろうか．最近の人型ロボットは人が近づくと反応するものもある．それならきっとロボット内部にカメラか何かがあって，人が近づいたことを検知しているに違いない．果たしてどうやっているのだろうか．しゃべる人型ロボットも多い．それならきっとロボット内部にスピーカーが付いているに違いない．だがその声の主はなんだろうか．ロボットを観察すればするほどロボットのある動作を引き起こす何らかの装置があることに気づき，またその装置はとても巧妙に動作していることがわかってくる．そう．ロボットとはあらゆる装置を駆使してプログラムされた総合的なからくりなのである．そう考えると必然的にロボットの電装にはそれらの装置が乗っかってくることがイメージ出来るだろう．そしてあらゆる装置が乗っかった電装にはロボットを思い通りに動作させるための何かがあるはずだ．

さあ，CanSat の実体が少しずつわかってきた．まず CanSat というのはロボットの一種に過ぎない．そしてロボットの電装には様々な装置が乗っかっており，それらを動かす何かがあるのだ．まず電装を理解する第一歩は，その装置が何で，装置を動かすものは何なのかをはっきりさせることだ．本章ではその点をはっきりさせよう．

2.2 役者のご紹介

本節では上述したロボットの装置やその装置を動かす実体について明らかにしていく．簡単に言うと CanSat で使う部品の紹介である．ここで注目してほしいのは，使用する部品には必ず役割があるという点である．それぞれの役割はロボットの動作に直結する．逆に，ロボットの特定の動作にはある部品が不可欠であることが逆算できる．この逆算を出来るようになることこそが，電装設計の第一歩である．そのためにはまず「順算」が出来なければならない．

2.2.1　CanSat の脳

先程はロボットの装置を動かす何かがあると述べた．その何かがまさしくここで説明する部品たちである．CanSat の脳として各部品を動かし制御するのは CPU を持つコンピュータの役目だ．コンピュータといっても一般に使われるパソコンのようなものではない．よく使われるのはシングルボードコンピュータと呼ばれるものやマイクロコントローラと呼ばれるものだ．いずれも小型であるという点がポイントだ．

シングルボードコンピュータとはその名のとおり 1 つのボードに全てが収まったコンピュータである．その類のコンピュータはパソコンのようにディスプレイやキーボードなどは内蔵していない．その代わり小型であり，手のひらに収まる程度のサイズである．シングルボードコンピュータの代表格は Raspberry Pi であり，最近では Jetson Nano などがある．シングルボードコンピュータは後述するマイクロコントローラに比べて高性能である．CPU 性能も高ければメモリの容量も圧倒的に大きい．その反面，値段はシングルボードコンピュータの方が高く，サイズも小型とは言え大きい．また，シングルボードコンピュータのアーキテクチャは基本的にはパソコンのものと似ているため，シングルボードコンピュータでは OS が動く (もっぱら Linux 系が多い)．OS が動くため OS の機能を利用したプログラミングが可能であるという点も魅力の 1 つである．

マイクロコントローラ (通称マイコン) は通常のパソコンとは方向性が若干異なるコンピュータである．パソコンでは主にデータの管理に焦点を充てて処理をすることが多いが，マイコンは周辺機器の制御に焦点を充てて設計されている．このため，大量のデータを扱う必要もなく，メモリの容量もパソコンやシングルボードコンピュータの一般的な容量に比べるとかなり劣る．また，特に OS がないものが多い．これにより，マイコンは制御以外の無駄な処理をすることがない (OS があるものだとプロセスの実行は OS が管理するので，所望のプロセスだけを集中的に実行するということは出来ない)．ただし，マイコンは OS がないので必然的に OS の機能を利用することが出来ない．マイコンはシングルボードコンピュータに比べて安価に手に入ることも考慮すべき点である．

ここまで CanSat の脳となるコンピュータの種類を紹介してきたが，今度はコンピュータという装置についてもう少し踏み込んで考えてみよう．読者のみなさんの中にはパソコンを購入しようとして販売店やサイトを漂いながら製品のスペック (仕様) というものを目にしたことがあるかもしれない．このスペックには主に CPU やメモリの容量，ストレージの容量などが記述されている．なぜそのようなものを表記するのだろうか．その答えは簡単で，それらはこれからパソコンを購入しようとする人に対してヒントを提示するためである．そのヒントの 1 つである CPU とは Central Processing Unit の略で中央処理装置などと訳される．CPU の役目は演算であり，基本的には演算に使われるデータは持っていない (一時的なデータは少し持っている)．それではどこからそのデータが来るのか．その答えはメモリであり，ストレージでもある．もしかしたら，USB メモリのように外部ストレージからデータを取り出す必要があるかもしれない．コンピュータ内部の細かい話を無視すれば，コンピュータがすることはメモリ (もしくは何らかのストレージ) にあるデータを CPU に移して，CPU でなんらかの演算をさせ，演算結果をまたメモリ (もしくは何らかのストレージ) に返す，ということを反復するだけである．これは非常に大雑把な議論であるが，十分本質的であり，開発者が常にイメージすべきことである．この議論から，解決すべき問題は次の 3 つに帰着する．

1. どこからデータを取ってくるのか
2. データをどう演算するのか
3. 演算したデータをどうするのか

上記の問題の 1，3 の問題は本章で解決される．また 2 の問題は第 5 章で解決される．いずれにしても，上記の問題は電装を設計する上で極めて本質的な問題であり，設計における出発点とも言えるものである．

最後にデータについて述べておこう．データとはなんだろうか．データとは情報のことであると答える人がいるかもしれない．では情報とはなんだろうか．実はこの問題は非常に深遠な問題であり，筆者もその答えを知らない．ただ，現在のコンピュータにおけるデータについては述べることが出来る．単にデータといえば抽象的でわかりにくいが，データを扱う対象が定まれば具体的な実体が見えてくるのである．現在のコンピュータにおいてデータとは電圧値である．全てはこの電圧が握っている．よくコンピュータ上のデータは 1 と 0 の羅列だという説明を見かけることがあるが，実際にコンピュータ内部のどこに 1 や 0 があるというのだろうか．1 か 0 は人間の頭の中に概念に過ぎな

いはずだ．実際にはコンピュータ内部に 1 や 0 があるわけではない．コンピュータを開発した賢い先人たちが，この 1 や 0 という概念を物理的に表現する方法を考え，物理現象として 1 や 0 と対応する実体がコンピュータ内部にあるのである．その実体こそが電圧値ということだ．そしてその電圧値はとても雑に扱われる．高いか低いかだ．高ければ 1 を表し，低ければ 0 を表す (逆でも良い)，といった具合に対応付けられるのである．データを維持するためにはこの電圧値を維持しておけばいいわけだ (その役目がメモリだ)．それではこのデータを他の場所に移すにはどうすれば良いのだろうか．これまでの議論から，データを移すことはある場所にある電圧値を他の場所に移すことに対応することがわかるだろう．そっくりそのまま移してもいいし，一度変形して移して受け取り側で復元してもいい．あとは技術や方法の問題だ．基板に配線をして金属のなかを通してデータを輸送する場合もあれば，データを変形し電波として送り受け取り側で復元する方法もある．このようなデータのやり取りについての具体例は後にいくつか扱う予定である．しかしここで最も押さえておくべきことはデータを表すには物理的な実体を必要とすること，そしてそのデータのやり取りもまた物理現象であるということである．そのことさえ理解していれば，どんなデータ表現ややり取りの方法に遭遇しても上手く対処できるだろう．

2.2.2 情報をくれるやつ

前項では CanSat の脳に相当する部分を説明した．そしてコンピュータについて深堀りし，コンピュータを扱う上ではっきりさせなければならない 3 つの問題を提示した．本項ではその 1 つ目の問題である「どこからデータを取ってくるのか」という問題について議論する．その上で，読者は CanSat を構築する上で非常に重要な構成部品について学ぶことになる．より理解を深めるために前項の内容を常に意識しながら本項に臨んでいただきたい．

前項で「どこからデータを取ってくるのか」という問題を提示した．そしてデータとは実際にはどういうものなのかについて説明した．ここでまた疑問が浮かぶ．一体全体データの出発点はどこなのだろうか．少し具体例を考えてみよう．読者はタイヤの付いたローバー型の CanSat を作って，特定のゴールへ向かって走らせたい．さて，特定のゴールの位置の情報はどこから来るのだろうか．ゴールの位置はわかっても自分の位置がわからなければゴールの位置に近づいていることもわからないはずだ．では自分の位置の情報はどこから来るのだろうか．声を掛けてやろうか．あいにく CanSat は言語を理解しなかった．困った困った．

上述した例からわかったことは，どこからデータを取ってくるのかという問題を解決するためには，データの出発点をまず押さえておく必要があるということだ．これは当然のことではあるが，やはり本質的なことである．先人たちはこのような問題を解決するために，データを生み出す方法を考えた．ある人たちは自然からデータを作り出す方法を考え，ある人たちは既存のデータを輸送する方法を考えた．いずれにしても物理現象を上手く利用した方法である．

自然からデータを作り出す装置はセンサと呼ばれる．多くのセンサは物理現象を何らかの方法で電圧値に変換し，その電圧値をコンピュータと交換する手段を持っている．コンピュータの発達していなかった昔ながらのセンサにはアナログ電流計や圧力計などがある (これは物理現象を針の振れ度合いに変換し人間に視覚的に伝えてくれている)．我々はコンピュータを用いてデータを演算したいので，ひとまず対象とするセンサはデータをコンピュータと交換することが出来るものに限定しよう．センサには様々な種類があり，それぞれ対象とする物理量を持っている．センサが作り出した物理量のデータはコンピュータへ送られ，コンピュータはその物理量 (あるいは何らかの変換を施したもの) を使って CanSat の状態を把握することが出来るという仕組みだ[*1]．表 2.1 は CanSat を製作する上でよく利用されるセンサの一覧である．

CanSat が使用するデータは自然から抽出されたものだけではないはずだ．人間が CanSat に伝えたいことだって

[*1] CanSat の状態は，センサの作り出したいくつかの物理量を変数とする何らかの関数であるという見方も出来る．ただし注意が必要なのは，もれなく全ての変数を見つけること，そして見つけたとしても状態関数の具体形を書き下すことは極めて困難であるということだ．例えば複数のセンサを使って 9 種類の物理量のデータを作り出し，それを状態関数の変数に採用した場合，この 9 次元空間 (状態空間) 内のある 1 点が状態に対応することになる．果たしてそのような高次元な空間内の関数を解析して，状態を正確に把握することは可能だろうか．残念ながらこれは容易ではない．これを実現する技術として機械学習，特にディープラーニングを使った学習モデルを利用して解析を行う方法なども近年は発展目覚ましいが，まだまだ CanSat のコンピュータの処理能力と比べて負荷が大きい部分がある (これもまた近年改善されてきているのだが)．いずれにしても，状態空間全体を取り扱おうとするには高度な技術が必要だ．通常はケースに応じて人為的に次元を落として状態を決定する．例えば，暗闇から出たことを照度センサのみ (つまり 1 次元状態空間) を使って判定するなどである．付随する話題は 5 章，6 章で議論する．

表 2.1: CanSat でよく用いるセンサの種類と概要

種類	物理量

あるだろうし，他の CanSat が CanSat にデータをあげたいこともある．つまり通信だ．通信は他の地点にあったデータを別の地点に移動させることを指す．通信を行える装置は通信機器などと呼ばれ，特に電磁波 (電波も電磁波の一種だ) を利用した通信機器は無線機と呼ばれたりする[*2]．通信というものはデータを送る装置とデータを受け取る装置があって初めて成立するが，一般的な通信機器はそれらを両方備えている場合がほとんどである．我々がよく利用する通信機器には例えば PC やスマートフォンなどがある．ただしこれらの通信方法はとても複雑である．例えば，スマートフォンのブラウザアプリで検索をしてあるホームページを見ようとすると，そのときの通信は決してスマートフォンとホームページの HTML ファイルを送ってくるサーバー間だけの通信ではない．スマートフォンから送信された電波は最寄りの基地局にたどり着き，基地局から局舎へ，局舎からバックボーンと呼ばれる通信網へと接続される．海外サイトなら海底ケーブルを通ってデータが送信されるかもしれない．そしてサーバーにたどり着いたデータはサーバー側で適切に処理され，その結果がスマートフォンに送られてくるのである．しかし，CanSat を製作する上でこのようなインフラを構築することは難しい．そこで，通常はデバイス間だけで通信を行う通信モジュールを利用する (ブロードキャスト通信[*3]をするものもある)．CanSat でよく利用する通信モジュールを表 2.2 にまとめた．

表 2.2: CanSat でよく用いる通信モジュール

通信モジュール	周波数帯	概要

実はセンサにも通信の要素が存在する．コンピュータとのデータのやりとりだ．データをやりとりするとはすなわち通信をすることなのである．だからといって，センサに無線通信モジュールが組み込まれているわけではない．通信は金属内を通って行われる．したがって，センサとコンピュータを適切に配線してあげる必要がある (詳細は 2.4 項，第 4 章で述べる)．この方法は上述した無線通信モジュールの通信方法とは異なる方法だが，れっきとした通信である．もっと言えば，通信という言葉は人間が勝手に作り出した頭の中だけで有効な概念なのであり，その概念を具体的に (つまり物理的に) 表現する方法が異なるだけの問題なのである．このような些細な問題よりも重要なことは，データをやりとりするためには通信という動作をする必要があり，通信の実現方法はデバイス (装置) によって異なるということを知っておくことだ．このことさえ知っていれば，コンピュータの外部デバイスを利用するときにそれがコンピュータとどのように通信するのかを必ずチェックするようになるはずだ．

最後に荒業とも言える情報の伝え方を伝授しよう．その方法とは，キーボードをつなげてコンピュータに直接データを入力してしまうという方法である．荒業ではあるがこれも立派な通信である．キーボードには押されると電気信号が発生するような装置が仕込まれており，発生した電気信号は有線のキーボードならその線を通って，Bluetooth 対応のキーボードなら Bluetooth 用の通信モジュールによって電波に変換されコンピュータに送信される．コンピュータは受け取った信号を解析して文字を特定し，ファイルの指定された位置にその文字を書き込む．ただし，この方法は本番環境では通用しないことは読者のみなさんも承知のとおりである．「こういう方法もあるということを忘れるな」ということを強調することが真の目的である．

さて，CanSat の電装 (ロボットの電装でもある) が段々と明るみになってきた．本項では「情報をくれるやつ」にはどのようなものがあるか，そして情報が移動させることが通信であることを学んだ．これによって我々はコンピュータに情報をあげることが出来る役者の存在を知ることが出来た．ここまでの流れを確認してみよう．コンピュータにはデータを使って演算をする，演算するためのデータをくれる役者を知った．これで演算は出来ることになるが，演

[*2] この業界の用語の定義は曖昧極まりないので微妙な違いをあまり気にしてはならない．用語の定義が曖昧になる理由としては，ベンダー達がワードパワーで自社製品を売りつけようと企み様々な造語を作成することで，結果的に本質的には似たようなデバイスや概念でありながら，名前の異なる名称が生まれてしまったためと筆者は考えている．筆者の意見としては，小さな問題に引っかからないためには，細かいことよりも本質的な部分を理解することに努めるべきだと思う．

[*3] 他の端末へ一斉送信する通信方法．端末とはネットワークの終端点を意味する言葉である．

算したらその後はどうするのだろうか．今のところその答えは出ていない．その詳しい議論は次項に譲ることにしよう．

2.2.3 情報を欲しがるやつ

前項までで 2.2.1 で提示した 3 つの問題点の 1 つ目が解決した．残す 2 つのうち，本項では 3 つ目の問題である「演算したデータをどうするのか」について考えることにする．この問いは一見馬鹿馬鹿しく思えるが，よく考えてみるともっともな意見である．データを外部から取得してまでして演算をしたのに，演算したデータをどうするのかが決まっていないなんてことはないだろう．では実際にそのデータをどうするのだろうか．答えは簡単だ．「情報を欲しがるやつ」にあげればいいのだ．それでは「情報を欲しがるやつ」とは一体全体何者なのだろうか．もっとわかりやすく言えば情報を受け取ることで機能するデバイスとは何だろうかという問いである．これに答えるためにはまずデータの使いみちを知る必要がある．使いみちが分かればそこには必ずデータを受け取るものがあり，それが答えとなるはずだからだ．

それでは前項で見た具体例を再度考えてみよう．その具体例の状況は，読者がいまタイヤ付きのローバー型 CanSat を作って，特定のゴールへ向かって走らせたいと考えているというものであった．前項で学んだことを活用すればゴール地点の位置情報はは事前にキーボードでゴール地点の緯度および経度を直接入力してしまえば済むし，自身の位置データは GPS 受信モジュールなどを使って得ることが出来る．つまり，この時点で演算する準備は整っているわけだ．そしてこの状態から例えば現在地からゴールまでの距離を演算によって得たとしよう．この距離のデータは何に使おうか．当然 CanSat の状態を決定するために使いたい．それだけではなく，CanSat をゴールまで移動させるためにも使いたいはずだ．簡単のため CanSat はゴール方向を向いているとして話を進めると，ゴールまでの距離が残り 100m もあるのなら，CanSat を直進させるべきであろう．また，ゴールまでの距離が 0.1m 程度であればかなり十分と言える距離まで近づいているので場合によっては停止してもいいかもしれない．つまり，演算して新しく生成されたデータ (もしくはセンサなどから得られた生データ) は CanSat を次の状態へと変化させるために使われる可能性がある．

データの使いみちはまだある．データを人に見せることだ．コンピュータが受け取ったデータ，そのデータから新たに作り出したデータを人間が見たい場合は多々ある．これらのデータは今後の開発や CanSat の動作をより深く理解するためには有益であるし，データを見たい人たちに提供することでビジネスになったりする．データを人に見せることというのは案外価値のあるものなのだ．

さて，データの使いみちがわかった．データを受け取る装置を整理しよう．まず CanSat の状態 (主に位置や姿勢) を変化させるために使う装置はアクチュエータと呼ばれものである．アクチュエータとは入力されたデータに対し物理的な運動によって応答をする装置である．アクチュエータの代表格としてはモータ (電動機) と呼ばれるものがあり，モータは電磁誘導とローレンツ力を組み合わせて電気的なエネルギーを力学的なエネルギーへと変換する．モータには様々な種類があり，用途によって使い分けることが重要である．CanSat でよく利用されるモータを表 2.3 に示す．

表 2.3: CanSat でよく用いるモータ

モータ	概要

アクチュエータは自由気ままに動くわけではない．アクチュエータには制御というものがつきまとう．アクチュエータはあるデータ (もっぱら電気信号) を受け取り，そのデータにしたがって出力を出すようになっている．アクチュエータが暴走しないためには電気信号を適切に設定し，出力を意図したものに近い形にする必要がある．このことを制御と呼ぶ[*4]．最終的に制御を決めるのは入力であるデータであることから，アクチュエータはデータを必要とする装置であると言える[*5]．また，アクチュエータによってはコンピュータからの直接の信号を受け付けないもの

[*4] 制御という言葉自体はアクチュエータに対してのみ使われるものではない．基本的に入力と出力があるような対象に対して，出力を統制することを制御する言う．

[*5] ここではデータを概念的な情報と解釈するのではなく，物理的実体のある情報 (電圧，電流，磁気など) として捉えてほしい．

も存在する．そのようなアクチュエータを操作するためにはドライバと呼ばれる IC [*6]を利用する必要が出てくる．モータに対するドライバはモータドライバと呼ばれ，モータの代わりにモータドライバが信号を受け付け，モータドライバはその入力をもとに適切な出力をモータへ送る．

データは人に見せるために使われるとも述べた．そのためにはデータをとりあえず保管する必要があるかもしれない．データを保管しておくための装置はストレージと呼ばれる[*7]．ストレージにも様々な種類があり，これは読者のみなさんにも馴染み深いものも多いだろう．よく使われるストレージを表 2.4 に示す．

表 2.4: CanSat でよく用いるストレージ

ストレージ	概要

データを人に見せるために使う装置はまだある．代表的なのはディスプレイ (モニター) である．実際に見ているのはディスプレイ上に表示されたデータであるが，そこに行き着くために無線通信モジュールなどの通信手段を取ったかもしれない．これもデータを受け取る装置である．もっと言えば，演算を行う CPU もデータを受け取る装置だし，メモリも同様である．センサにも設定を書き換えるためにデータを受け取る仕組みがある．もうここまで来るとデータを与えるだけのデバイスやデータを受け取るだけのデバイスはあまりないことに気がつくだろう．ここまで，便宜上データを与えるものとデータを受け取るもので説明してきたが，ここで限界が来てしまった．だからといって焦ることはない．データを与える，データを受け取る，そしてデータを処理するという動作は依然として保たれている．このうち複数の動作をするデバイスがあることがわかっただけだ．

ここまでで CanSat の電装に乗っかる多くの装置たちについて見てきた．そして本項でも継続的にデータの所在・行き先を強調しながら議論を進めてきた．データがどこにあるのか，データがどこに行くのかというのはデバイス間の接続，そして回路を設計する上で非常に重要なことである．また，このことはソフトウェア設計をする際にも極めて本質的な事項として浮かび上がってくる．その点については本書の残りの部分を読み進めることで次第に実感するであろう．

2.2.4 元気の源

最後に縁の下の力持ちを紹介して役者紹介の締めくくりとしよう．今まで CanSat の電装に必要な部品を見てきたが，重要なことを忘れている．それは電源である．電源がなければコンピュータを起動することも演算をすることも通信をすることもできない．電源は CanSat にとって (当然他の電装にとっても) なくてはならないものである．

電源を構成するのに必要なものは主にバッテリーと電圧を調整する電源 IC と呼ばれる部品である．いずれの部品も選定を間違えれば深刻な問題を引き起こすので慎重に選定する必要がある (選定については 2.5 節を参照)．バッテリーにはいくつか種類があり，CanSat 製作時によく使われるバッテリーを表 2.5 に示す．

表 2.5: CanSat でよく用いるバッテリー

バッテリー	概要

バッテリーと同時に検討されるべきなのが電源 IC である．電源 IC には様々な種類があるが，基本的な用途は降圧[*8]である場合が多い．降圧が出来る電源 IC の代表格は 3 端子レギュレータや DC/DC コンバータ[*9]がある．3 端子レギュレータは文字通り端子が 3 つ存在する IC で，各端子は入力，出力，GND 用である．入力端子に電圧を印加すると決まった電圧に降圧し出力端子に出力されるのが特徴である．3 端子レギュレータは入力端子と出力端子の差分である電圧によって発生する電気エネルギーが熱エネルギーに変換されて放出される．このようなことから後に述

[*6] Integrated Circuit の略．直訳すると集積回路．複雑な回路がチップのようなパッケージにまとめられているものを指す．
[*7] もっぱら不揮発性メモリに対して使われることが多い．不揮発性とは電力の供給がなくともデータが消失しないという性質を表す言葉である．
[*8] 電圧を下げること．
[*9] スイッチングレギュレータとも言う．

べる DC/DC コンバータに比べて変換効率が劣っており，発熱対策が必要となる場合もある．ただし長所として回路が単純であり，DC/DC コンバータで発生するようなスイッチングノイズがないことが利点である．一方で DC/DC コンバータは入力電圧を所望の出力電圧に変換するために，電力をスイッチングする (つまり付けたり切ったりする)．このようにスイッチングした電圧波形は矩形波の形をしている．そしてこの矩形波を電圧波形をコイルとダイオードを用いて整流[*10]する．これにより，3 端子レギュレータに比べ降圧による電力消費を押さえながら効率よく降圧を行うことが出来る．ただし DC/DC コンバータの周辺回路は 3 端子レギュレータの周辺回路に比べ複雑であり，基板の空間的コストが生じる．また，スイッチングを行うため出力電圧にノイズが発生する可能性があり，ノイズ対策を怠ると部品が破損する可能性がある．

電源は回路において非常に重要であり，最も慎重に組むべき回路である．だがその分理解すべきことも多くあり，ちゃんとやろうと気負うと容易にノイローゼになってしまう分野でもある．電源回路にあまり慣れていないうちは，そういうものなのかと受け入れ，先人たちの回路を丸パクリしたほうが身のためである．ただ，精神的余裕のあるときに少しづつ深めていくと非常に面白い分野であることに気づくだろう．具体的な部品選定の方法や回路設計の方法は後に再度述べるので，今はこのくらいにしておこう．

2.3 データシート

前節では CanSat の電装に乗っかることになる部品について述べた．だが実際問題，読者のみなさんが知りたいことはおそらく「どのようにその部品を使うか」であることだろう．しかし先に述べておくと，その答えは簡単であるようで非常に難しい．なぜ簡単なのか．それは製造元がドキュメントを作成してくれているので，それを読めば大体のことはわかるからである．なぜ難しいのか．それはドキュメント以上の知識をくれる人がいないからである．これはすなわち，ドキュメントを読めばある程度のことはわかるようになり，部品を扱うことも可能になるが，完全にその部品を理解するにはその部品に長期間向き合う必要があるということである．本節ではこのうち希望に満ちた側面について述べていこう[*11]．

気になっているであろうことにお答えしよう．データシートとは何なのか．それは部品の製造元が部品の説明書として作成してくれたドキュメントのことである．データシートは知識の宝庫であり，その部品に対するこれ以上の知識はおそらく得ることは難しい．つまり，データシートは我々にとって感謝すべき存在だ．読者も今後回路設計やソフトウェア設計をする際によくお世話になるだろう．たまにデータシートを読むことを嫌がる人がいるが，これは理解不能だ[*12]．データシートに答えがあるのになぜ見ないのか．ただし，気をつけてほしい．データシートはあなたの英文読解能力を試してくる．それも専門用語てんこ盛りの英語だ．ある人にとってはデータシートは天使のような存在になるかもしれないが，ある人にとってはノイローゼを発症させる悪魔のような存在になるかもしれない[*13]．

データシートを幸福を呼ぶ PDF だ．ある宗教の教えによるとデータシートを集めれば集めるほど幸福になれるという．データシートはたいてい部品の販売サイトで手に入る．もしデータシートのリンクが貼っていなかった場合は，直接型番で検索しデータシートを見つけよう．それでも見つからないならその部品は採用すべきではない (抵抗などではない限り)．理由は簡単だ．カンニングペーパー付きの問題と普通の問題をどちらを選ぶか考えてみればいい (当然試験官はいない)．相当疲れていない限りあなたは前者を選ぶはずだ．何かを製作する上で重要なことは，早い段階で不安要素を取り除くことである．ずるずる引きずり基板テスト段階に入って重大な問題が発覚するのが一番困る出来事なのである．

具体的にデータシートには何が書かれているのかを説明しておこう．ほとんどの部品で共通に記述されていることは，部品の電気的特性，部品概形，ピン配置，回路設計例などである．これだけでも涙を流して喜ぶべき内容である．部品の電気的特性を見れば，その部品が何 V の電圧で動作するのかやどれくらいの電流が流れ込むのかなどを知ることが出来る．これは電力設計に活かせる情報だ．次に部品概形やピン配置は部品の EAGLE ライブラリ (4.4 節で

[*10] 交流のような時間的に変動する電圧を直流電圧のように定常的な電圧信号に変換すること．

[*11] 負の側面は開発が進めばいずれ遭遇するかもしれない．

[*12] データシートを読む決心がつくまでに時間がかかることには同意する．

[*13] 大学生なら今後嫌と言うほど英語の文献を読む羽目になるのでデータシートで免疫を付けておくのもいいかもしれない．ソフトウェアの勉強をするときもたまに英語の文献しかない場合などもあるので，この際英語縛りをしておいた方が良い．機械翻訳並の下手くそ翻訳本を読むよりはよっぽどマシである．

詳しく述べる) を作成する上で欠かせない情報である．また，回路設計例は用途さえ合っていればその回路をそのまま採用してしまえば正常に動作することを保証できる．素晴らしい．素晴らしすぎる．これに加えてコンピュータと通信するようなセンサ，通信モジュール，モータドライバのような部品たちのデータシートには通信の仕方なども書いてくれている．通常この情報を参考にしてコードを書く．特にセンサの場合にはセンサ内部にレジスタと呼ばれるメモリが存在し，そのレジスタにセンサのデータが格納されるのだが，データシートにはどのレジスタにどのような種類のデータがあるのかということももちろん記載されている．おわかりだろうか．データシートがなければ，ハードウェア設計もソフトウェア設計もままならないのである．

本節ではわざわざデータシートに 1 節を割いてその重要性を強調した．読者のみなさんもその重要性を実感していただけたと信じている．何か問題が起きたときに友人に聞いても構わない，Google 先生に聞いても構わない，しかし最も正確な答えはデータシートにあるということだけ心に留めておいていただきたい．

2.4 コミュニティの形成

2.2 節では何度か通信について言及した．通信とは概念的には情報をやりとりすることを意味し，物理的な表現方法としては電磁波を金属内で伝播させる方法であったり，空気中を伝播させる方法があるということを話した．この説明では何やら回路の実装は難しそうな印象を受けるかもしれない．しかし心配無用．そのような難しい回路は部品内に内蔵されている．部品を使用する側の我々は最低限要求される定石的な回路を構成するだけで良い[*14]．本節では通信について再度疑問を投げかけ，情報のやりとりに必要なものは本質的に何なのか，そして電装を設計する上では具体的にどうすればいいのかについて見ていく．

2.4.1 通信の物理的側面

何度か述べたように情報や通信のような抽象概念の実体は物理現象である．したがって具体的な通信を知るまず第一歩はどのような物理現象によって通信が行われるかを知ることである．そして，その通信手段では物理的にどういう操作をするべきなのかを考えれば良い．例えば金属内で電気信号を送る手段をとるならば，通信の始点と終点を金属の線で繋いであげる必要があるだろう．また，電波で無線通信をするのであれば電磁放射を行えるようなアンテナが必要だし，アンテナに電流を流す前に送信するデータを D/A 変換[*15]する必要があり，受信側ではアンテナに発生した電流を A/D 変換をしてあげる必要がある (このあたりは部品側でやってくれる)．

上記のような話はなんだか物理学や数学の話が絡んできそうで一部の読者には馴染めない話題かもしれない．しかし幸いにも，我々は実際の数式や計算などはほとんど考えることなくこのような物理現象を利用できる．このような低レベル[*16]な実装は難しいため部品内で解決されているケースがほとんどであるからである．なのでこのような細かいことを考慮しなくても，データシートに書いてあるとおりに配線しておけば正常に動作する．ただ，何かトラブルが発生したときや独自の改良を施したいときなどはこれらの知識はなくては解決方法を見出す見通しが立たなくなってしまうため，最低限の原理は知っておいたほうが良い[*17]．

2.4.2 コミュニケーションを実現するためには

前節では通信の物理的側面を簡単に見た．ここからは通信をもう少し抽象化して考えてみる．このためには通信という言葉より，コミュニケーションという言葉の方が馴染み深いかもしれない．まずは簡単に我々人間のコミュニケーションについて考えてみよう．我々のコミュニケーション手段は様々だ．例えば話すという手段があるし，書く

[*14] もちろん内部の詳細を知っておいて損はない．

[*15] デジタル信号からアナログ信号へと変換すること．逆を A/D 変換という．

[*16] この業界で使う低レベルという用語は何か蔑む気持ちをはらんだものではない．物理的な側面に近い話題のことを低レベルといい，抽象的な側面に近い話題のことを高レベルと呼ぶのである．ハードウェアとソフトウェアで言ったらハードウェアのほうが物理的な側面が強いため，より低レベルであるということになる．

[*17] このことは意外と重要なことである．浅くても広く知識を持っていれば解決方法を模索しやすくなる．知識の幅が狭いと解決方法を見つける方法すらわからなくなり八方塞がりになってしまう．電装開発初心者の方には玄人の方は何でも知っているように思うかもしれないが，実際はそうでない場合も多い．そういう人たちは知識を持っているのではなく，どこに行けばその知識があるかを知っているのである．

という手段もある．手話やジェスチャーなどの見るという手段もあるだろう．それぞれ音 (空気の振動) や光のような物理的実体が情報を伝達する手段となっていることは，先程述べた部品間の通信と同じことである．ただ，人間が話したことを機械が理解することが出来るだろうか．最近なら AI 技術が発達し，そのようなことも可能にはなってきているが，そのような技術のない時代にはそんなことは出来なかった．でも人間なら出来る．なぜ出来るのだろうか．それは当然であるが言葉を知っているからである．

ここでこんな反論もあるだろう．日本人が日本語で無作為にアメリカ人に話しかければ，ほとんどのアメリカ人には通じないはずだと．全くそのとおりだ．なぜ通じないのか．それはそのアメリカ人が日本語を知らないからだ．想像力豊かな人間でさえ，知らない言葉を聞いてその意味を想像することが出来ない．キーワードが見えてきただろうか．アメリカ人でも日本語を理解できる人もいるのだ．なぜ理解できるのか．日本語を知っているからだ．つまり，知っていることがコミュニケーションを行うための十分条件なのである[*18]．

アメリカ人が日本人と円滑にコミュニケーションするためには日本語を知っていれば十分であることがわかった．では，実際に日本語を知っているというのはどういうことなのだろうか．これは我々が英語を勉強するときを思い浮かべれば良い．英語を勉強するときは文法や単語，イディオムなどを勉強するだろう．これらはいわば英語を使う上での決まりごとである．これらを間違えれば適切に受け取り手に表現したい内容が伝わらない可能性が出てくるのだ[*19]．

また，次のようなケースはどうだろうか．すなわち話し手である日本人側が間違った日本語を話している場合である．これは受け取り手であるアメリカ人がいくら日本語を完璧にマスターしていようと通じない可能性がある．このケースでもコミュニケーション失敗ということになるわけだ[*20]．

さて，ここまでの議論で人間が言葉でコミュニケーションを行うためには互いに言葉の決まりごと (文法や単語，イディオムなど) を知っておけば十分であることがわかった．ただし注意すべきことはコミュニケーションをする人たちの中では言葉の決まりごとに対する認識は必ず一致している必要がある．このことがわかれば十分だ．これを電装にも適用すれば良い．

2.4.3 物理的なプロトコル

前項の続きをしよう．言葉の決まりがあればコミュニケーションが成り立つということだった．この結論をコミュニケーションというアナロジーのもとで電子部品間のコミュニケーション (通信) に適用してみよう[*21]．その前にはっきりさせておくべきことがある．一体電子部品間で用いる「言葉の決まり」とは何なのかということだ．

そもそも電子部品における「言葉」とは何なのだろうか．実は言葉というのは情報や通信と同じく人間の頭の中にある概念でしかない．つまり，我々は言葉をより具体的な表現に落とし込む必要がある[*22]．その結果生まれたものが，文字であり言葉の音である．つまり「言葉の決まり」というのは，文字や音の並び方の規則のことである．例えば多くの人間の言語はある 1 つの文字の並び (文章) が複数の小さな文字の並び (単語など) に分割でき，小さな文字の並びの並べ方 (文法) で意味が異なるような規則を持っている．これがコミュニケーションをもたらす「言葉の決まり」の実装になっている．

では電子部品ではどうだろうか．電子部品における情報は多くの場合電圧である．つまり言葉が電圧なのである．

[*18] もう少し噛み砕いて言うと，言語を知っていればコミュニケーションが成り立つということだ．もちろんセンセティブな話題になれば意見の食い違いでコミュニケーションが成り立たない場合もあるだろうが，機械にはそんな感情はないのでここでは無視する．

[*19] 実は人間の言語はよく出来ていて，少し文法や単語が間違っていたとしても受け取り手に意図通りの内容が通じてしまう場合がある．これは人間の言語が冗長性を持っているからである．冗長性についての詳細は情報理論の本を参照してほしい．

[*20] 勘の鋭い人はアメリカ人側も日本人側と全く同じ間違いをしていれば 2 人のコミュニケーションは成立するのではないかと反論してくるかもしれない．これは全くそのとおりである．そしてまさにこれこそが言語の変遷なのである．コミュニケーションにおいて辞書的な意味はさほど意味をなさない．コミュニケーションする人の間で共通の認識であればいいのだ．日本語が汚れていくなどという人たちはこのことについて理解すべきである．

[*21] もちろん人間同士のコミュニケーションの具体例から得られた結論が電子部品間でのコミュニケーションの際にも適用できるという根拠は存在しない．しかし実際には本文で見るように，電子部品間でのコミュニケーションにおいても人間のように「言葉の決まり」に対応するものがあればコミュニケーションが上手く行くことがわかっている．だからといって他の種類のコミュニケーションでも同様に上手く行くかどうかはわからないし，「言葉の決まり」があることがコミュニケーションの必要条件であることもわからない．これ以上この話題に踏み込むと本書の趣旨からずれてしまうので，興味がある読者は自分で考えてみてほしい．

[*22] 本書ではこのように抽象概念を具体的な表現に落とし込むことを実装と呼ぶことにする．

そして電子部品においての文章は (高いか低いかだけの) 電圧の並びとでも言えるだろう．そして人間の言語のように，並びをまたいくつかの並びに分割することも出来るだろう．それによって小さな電圧の並びの順序が定義できることになる．そしてある決まった順序を定め，その順序に従って小さな電圧の並び (人間語では単語) を 1 つの大きな電圧の並び (人間語では文章) にして送るのである．もちろん前項で述べたように受け取り手もそのような電圧の並びの規則を知っていて，その規則に従って受け取った電圧の並びを小さな電圧の並びに分割して全体の電圧の並びの意味を解釈するのである．

上述したような電子部品間で通信を行う際の「言葉の規則」は通信プロトコルと呼ばれている[*23]．通信プロトコルには様々な種類があるが，よく使われるプロトコルは限られている[*24]．CanSat でよく利用される通信プロトコルを表**??**に示す．

表 2.6: CanSat でよく用いる物理的なプロトコル

プロトコル	概要

プロトコルの概念の導入が終わったところで次のステップのお話をしよう．実は 1 つの通信において使用されるプロトコルは 1 つとは限らないのだ．それら 1 つ 1 つのプロトコルは全く異なる決まり事なのである．いくつのプロトコルを使うのかは用途によって異なるのだが，今回は説明の関係上プロトコルを大きく次の 2 つの種類に分けて考えることにする:

- 物理的なプロトコル
- 概念的なプロトコル

表 2.6 に示したのは物理的なプロトコルの方だ．物理的なプロトコルは物理的な実体について規則を設ける．例えば，表 2.6 に示したプロトコルは全て電圧の並びという物理的実体に対しての規則だ．一方で概念的なプロトコルは物理的実体のない (というよりもその実体を意識しなくても良い) ものについて規則を設ける．後者については次項で述べることにしよう．

2.4.4 概念的なプロトコル

2.4.5 プロトコル階層

2.5 部品の選び方

2.5.1 使いたいもの・必要なもの

2.5.2 電力設計は大事

2.6 まとめ

[*23] プロトコルという言葉は雑に言うと規則という意味の言葉である．したがって，プロトコルという言葉が使われるのは通信の分野だけではない．しかし前項で見たように「言葉の規則」があれば通信が出来るということがわかっているため，通信の分野には必ずといっていいほどプロトコルの概念が存在する．ググってよくヒットするプロトコルはインターネット関連のプロトコル (TCP/IP など) であるが，それだけがプロトコルであると勘違いしないように注意してほしい．それはプロトコルの一種なのである．

[*24] よく使われるプロトコルが限られているのには理由がある．あなたは何ヶ国語話すことが出来るだろうか．多くても 3 ヶ国語という人がほとんどだろう．それなのにもし世界中で数万種類の言語が非常に活発に使われていたらあなたはどう感じるだろうか．非常に嫌気が指すだろう．これと同じように，人間よりは簡単なプロトコルを扱う電子部品であっても同じような理屈でプロトコルが多すぎるのは嫌なのである．そこで一般的には標準化という手続きを踏んで，多くの開発者たちに標準化されたプロトコルを使ってくれるように懇願するわけである．

第 3 章

全体を俯瞰してみる

この章では，具体的な製作の説明に入る前に製作全体についての説明を行う．本章では読者が製作全体および電装全体に対して俯瞰的な立場からイメージできるようになることを目指す．例えるなら，この章では電装製作の骨組みと製作のスケジュールについて述べる．スケジュールを知っている状態とスケジュールを知らない状態ではその日の臨み方がかなり異なってくるのは理解していただけると思う．ただし，そのスケジュールの中身の詳細については後の章に譲ることにする．もし読者が具体的な製作の説明に急ぎたい場合は本章を飛ばして次章から読み進めてしまっても構わない．そして具体的な製作の章を読み終えてから，本章に立ち戻って読んでもらうと今まで本書で述べた内容についてよく整理がいくことになるだろう．

3.1 概念と実装

前章では CanSat によく用いられる部品を見てきた．また，データや通信，プロトコルなど部品を組み合わせて電装を構成していく上で重要な概念についても触れてきた．前章でも少し触れたが，我々が扱う対象は概念的なものではなく，実際に存在する物理的対象だ．このように抽象的な概念でしかないものを具体化することを実装という．実装という用語にも色々な使われ方があるが，大体は前述のような意味で使われる[*1]．

実装はいわば概念に対する表現である[*2]．「CanSat 電装」という概念的な言葉に対して，千人が実装を行えば千人とも同じ基板やコードを作ってくることは稀であろう．つまり，それぞれが「CanSat 電装」に対して自分なりの表現をしているのである．したがって，何か概念的な用語に遭遇したとき，その実装には様々な方法がある可能性があるということをまずは念頭に置いておくべきだ．このことは，抽象概念に対して様々な実装がある中で，読者は既存の実装を選ぶことも出来るし，自らその抽象概念を実装してみるという選択肢もあることを意味している．

最後に抽象概念を実装したとしても，必ずしもそれは十分に具体的ではないということを注意しておこう．例えば前章で述べた「概念的なプロトコル」というのは十分に具体的ではない．十分に具体的ではないというは，それだけでは機能することは出来ないという意味である．概念的なプロトコルでは主に文字の羅列などについて規則を設けることをしていたが，電子部品にとって文字というのは具体的ではなく認識出来るものではない．このため，文字を電圧の配列として表現 (実装) してあげる必要がある．すると，物理的なプロトコルによって実際に通信が出来るようになる．このように，実装というのは概念を具体化するが最後まできっちり具体化することを強制するものではない．このような性質から，前章で述べたプロトコルの階層化のような，実装を段階的に細分化することでシステムを柔軟に構築するという手法が取れるのである[*3]．

本節で説明した内容は CanSat 電装製作というよりもあらゆる開発をしていく上で非常に重要なことである．というのも，多くの開発は「概念を作り，実装をする」というプロセスを経て進んでいくからである．もしかすると概念

[*1] ハードウェア開発において実装は実際に人間の頭の中にある概念的なもの (回路図，基板概形など) を物理的に作り出すことを指す場合が多い (例: 基板の実装)．ソフトウェア開発の場合は企画書や設計案から実際にコードを書いていくことを実装という場合が多い．どちらも抽象的な概念を具体化するという意味では共通である．

[*2] 筆者は概念から具体的な対象への写像のようなものと考えている．

[*3] もちろん実装を段階的に細分化するのであれば，最終的には十分具体的であることが要求される．例えば，概念的なプロトコルだけを実装し物理的なプロトコルを実装しないということは許されない．なぜなら本文で述べたように概念的なプロトコルは電子部品にとって十分具体的ではないからである．最終的に機能するは電子部品であることを考慮すると，これは当然のことである．

と実装という用語は**要件**と**仕様**という用語で現れるかもしれない．どちらにしろ，本章で述べたプロセスを拡張し複雑化した結果が現在の多くの開発で採用されているプロセスなのである．

3.2　CanSat でやりたいこと

2.1 節では CanSat は「宇宙開発を目的とした自律型ロボット」であるという説明をした．とは言っても本書の読者が実際にそのようなロボットを作ることはないだろう[*4]．しかし「それに近いもの」を作ろうと思っている人は多いだろう．「それに近いもの」とは何だろうか．例えば，宇宙では実際に動作することはないが地球上では模擬的に動作できるもの，宇宙で動作できるものであるが宇宙に輸送することが出来ないので動作環境は地球になってしまうもの，などであるだろう．

実際に CanSat を製作するにあたってまずは CanSat でやりたいことを確認しておこう．「宇宙開発を目的とした自律型ロボット」を作ろうとしているのだから，「どんな宇宙開発」であるかははっきりしておく必要があるし，その宇宙開発が「それに近いもの」として製作できるのかについても検討する必要がある．その理由は製作したロボットが CanSat と呼べるものかどうかということよりも，CanSat に何をやらせたいかによって部品の選定やその上のハードウェア・ソフトウェア設計は大きく変化する可能性があるからである．つまり「CanSat でやりたいこと」というのは CanSat 製作の出発点なのである．

3.3　コミュニティの形成と掌握

電装にはコミュニティが存在する．部品間のコミュニティだ．電装のコミュニティの構造は少々複雑だ．なぜなら「部品」と思っているものの中にも実はコミュニティが存在する場合があるからだ．つまりコミュニティの中にコミュニティがあるときがある．また，コミュニティはコミュニケーションがあることも忘れてはならない[*5]．やや大袈裟に言うと，開発者がすることは「部品間のコミュニティを形成し，そのコミュニティでのコミュニケーションを掌握すること」である．

3.3.1　コミュニティの形成

コミュニティを形成し，さらにそれを掌握するとは一体どういうことなのか．前章を読んだ方は少しピンと来るものがあるかもしれない．コミュニティを形成するとは言わばコミュニケーションを行うことの出来る土台を作ることである．つまりコミュニティはコミュニケーションに依存する．我々は既にコミュニケーションの方法はプロトコルと呼ばれるものが定めていることを知っているので，コミュニティを形成するにはあるコミュニケーションのプロトコルを満足させるような環境を整えれば良いということになる．つまり，コミュニティを形成するにはプロトコルについてしっかり知っておく必要がある．

具体例を挙げていこう．いまセンサとコンピュータがコミュニケーションを取れるようなコミュニティを形成したいとする．ならばそのコミュニケーションのプロトコルを調べる必要があるというのは上述のとおりである．今回はそのプロトコルが I2C と呼ばれるものであったとしよう．I2C というプロトコルについて調べると，ある決まった配線方法があることがわかる[*6]．我々はその配線方法に従ってセンサとコンピュータを配線すれば良いのである．これでコミュニケーションの準備は整った．つまり 2 つの部品間にコミュニティが形成されたのだ．

この具体例のように，我々はプロトコルに則ったコミュニティ作りをすることで部品間のコミュニケーションを実現する準備を整えることが出来る．ここまで来ればわかるように，コミュニティの形成とはもっぱらハードウェア製作において行われることである．ハードウェア製作者は部品間のコミュニケーションを可能にするために，まずは最低限プロトコルに則ったコミュニティを形成する必要がある．もちろん，自身のスキルを高めることでより良いコミュニティの構築を行うことも出来る．

[*4] もし本当に作る用事がある人なら本書を読む必要がある人ではないからだ．

[*5] 部品間のコミュニケーションについては 2.4 を参照．

[*6] これもプロトコルの一部である．

最後にここまでの内容について注意しておくべき事項について話しておこう．まず 1 つ目は，コミュニティの形成にはプロトコルが必要だがプロトコルは部品に依存しているということである．つまり，コミュニティの形成を行う方向性は部品選定によって決まる．逆の場合はなくもないが，使用できる部品を極端に制限することに繋がりかねない．2 つ目の注意点は，データシートなどでプロトコルが明確に書かれていない部品については，製作者自身でプロトコルを考える必要があるということだ[*7]．例えば MOSFET [*8]を利用する場合，MOSFET のデータシートを見てもどこにもプロトコルなど載っていない．載っているのはゲート・ソース間電圧に対するドレイン電流の特性などのグラフだ．設計者はそのグラフから MOSFET の挙動を推測し，その挙動によってプロトコルを作成する必要がある[*9]．3 つ目の注意点は配線をすることのみがコミュニティを形成することとは限らないという点である[*10]．これは少し考えてみればすぐにわかる．なぜなら上述したようにコミュニティはコミュニケーションを行うための土台の役割をし，コミュニケーションの方法は前章で述べたように配線された線路上での電圧の変化だけではなく，空気中を伝播する電波による方法であったり様々であるからである．結局，コミュニティを形成するには行おうとしているコミュニケーションについてしっかり理解しておくことが重要であると言える．

3.3.2 コミュニティの掌握

ここまでの説明で「部品間のコミュニティの形成」についてはざっくりとイメージが出来たのではないだろうか．読者のみなさんが理解できたことを信じ，次の話題である「コミュニティの掌握」について話を移そう．

我々の最終目標はコミュニティの形成，つまり回路・基板の製作ではないはずだ．我々の最終目標は「CanSat でやりたいこと」を実現することであり，基板が出来たところでその目標は達成されない[*11]．我々はそのコミュニティの中でコミュニケーションを活発化させ，その最終目標を達成したいのである．

ここで一つ疑問が生じる．それはコミュニケーションを活発化させることでどんなメリットがあるのか，ということである．そもそもコミュニケーションとはどういうものであっただろうか．前章の内容を思い出すと，コミュニケーションとは情報のやりとりを指す概念的な言葉であった．つまり，ある部品はある部品とコミュニケーションをすることにより新たな情報を得ることが出来る．情報を活発化させるということは，すなわち外部からたくさんの情報を収集することを指すのである．

議論をもう少し進めるためにもう一つ問題提起しておこう．コミュニケーションによって情報を収集することが出来ることはわかったが，その情報はどこに集められるのだろうか．また，その情報はどのように使われるのだろうか．その答えは半ば前章で出ているが改めて述べるなら，大多数の情報の行き先はコンピュータであり，コンピュータに行き着いた情報は何らかの演算によって違う形の情報に変換され CanSat の状態を決めるために使われる，というのが答えになる．

最後にこの節の結論を述べよう．部品間のコミュニケーションはハードウェア製作者によって形成されたコミュニティによって実現される．そしてコンピュータは周辺部品とコミュニケーションを行い，情報を集め，その情報を演算し，CanSat の状態を決める．そして必要があればアクチュエータなどの「情報を欲しがる部品」にコミュニケーションによって情報を与え動かす．つまり，ハードウェア製作者によって作られた部品間のコミュニティの支配者はコンピュータである．そして，コンピュータを支配するのは実行されているコードであり，コードを書くのはソフトウェア製作者である．つまり，このコミュニティを掌握しているのはまさしくソフトウェア製作者なのである．この構図こそが電装開発の開発フローを定め，ハードウェア担当やソフトウェア担当のように役割分担を行うことになる根源と言える．

[*7] この点がハードウェア開発者が最も工夫を施す部分の一つだろう．

[*8] MOSFET については 2.2.3 節および 4.2.4 を参照．

[*9] と言っても，MOSFET の基本的な使い方を習得すればそれほど難しくないし，大抵の場合センサなどに用いられるプロトコルよりも極めて単純なものになる場合がほとんである．MOSFET の基本的な使い方については 4.2.4 を参照．

[*10] そのためコミュニティの形成という抽象的な概念で表した．その概念の実装が配線などの具体的な手段である．

[*11] コミュニケーションが出来る環境にも関わらずコミュニケーションを行わない社会を思い浮かべてほしい．コミュニティを形成したところでコミュニケーションがなければ，その先に何も生まれないのである．

3.4　段階的な実装

ここまでで電装製作全体のイメージが出来るようになってきただろうか.

3.5　まとめ

第 II 部

電装を製作する

第 4 章

ハードウェアの製作

第 5 章

ソフトウェアの製作

第 6 章

制御

第 III 部

電装製作者の気持ち

第 7 章

設計思想

第 8 章

立ち止まって振り返る

第 9 章

この先にあるもの

第 10 章

最後に

参考文献