2025 年度

ハッカソン 1

千葉工業大学　情報変革科学部

情報工学科

2025 年 4 月 1 日

目次

第I部　ハッカソン編

第１章　ハッカソンを始める前に

1.1　ハッカソンとは

ハッカソンとは，エンジニア，プログラマー，デザイナーなどがチームを結成し，与えられたテーマに沿って指定の時間内にサービスやプロダクトを開発し，その出来を競うコンペ型のイベントです．この言葉は，Hack（ハードウェアやソフトウェアのエンジニアリング）と Marathon（マラソン）を組み合わせた造語です．米国 IT（Information Technology）企業のサン・マイクロシステムズのマーケティングチームによって，1999 年から「ハッカソン」という言葉が使われ始めたと言われています．IT 企業が発祥ということもあり，ソフトウェア開発に関わるハッカソンが多いですが，現在では，アナログ回路，デジタル回路，組み込みソフトウェア，3D プリンタなどの技術領域を用いたハードウェア分野のハッカソンも実施されています．情報工学科のハッカソン 1 では，**Arduino を利用するシステムの開発**を行うので，ソフトウェアとハードウェアの両方の開発を伴うハッカソンとなります．通常，ハッカソンを実施する目的は，新規事業・新商品の開発，人材交流・コミュニティの構築，人材育成などさまざまですが，これらは**エンジニアリングを楽しむ**ことができて初めて達成されます．システムの開発には苦労も伴いますが，是非，楽しみながら開発を進めてください．

ところで，皆さんは 1 年生のときに**アイデアソン**を経験しました．これは，Idea（アイデア）と Marathon（マラソン）を組み合わせた造語です．特定のテーマに沿った活動を行う点ではハッカソンと同じですが，アイデアソンでは新たな商品企画やビジネスモデルなどの**アイデアを競う**のに対し，ハッカソンではアプリ（アプリケーションソフトウェア）やシステムなどの**プロダクトで成果を競う**ことが大きな違いです．ただし，アイデアを出すこと自体は，ハッカソンを進めるための準

備・計画段階で必ず行いますので，アイデアソンは，その部分のみを切り取って独立したイベントに仕立てたものと言えるでしょう．イベントとしてのアイデアソンは開発を伴わないため，実際にモノを作れるかどうかは置いておいて合理的な思考力を問われますが，プロダクトを開発するハッカソンでは開発力が問われます．したがって，ハッカソン１ではアイデアソンで身につけたことを，エンジニアとして実際に動くモノをつくるという立場で実践することが重要となります．とはいえ，ビジネスコンテストではないので，マネタイズや世の中に与えるインパクトよりも，技術的な課題や興味を優先し，与えられたテーマに対する問題の着眼点・着想点が独創的な完成度・動作性の高いシステムの開発を目指してください．

1.2　ハッカソン１の目的

Mac に搭載されている辞書（スーパー大辞林）によると，工学とは「科学知識を応用して，大規模に物品を生産するための方法を研究する学問．広義には，ある物を作り出したり，ある事を実現させたりするための方法・システムなどを研究する学問の総称」とあります．情報工学科は情報に関係する工学を学ぶ学科です．しかし，カリキュラムに用意できる科目数や履修登録可能な科目数には限りがあります．例えば，情報工学科では，複数のプログラミング言語を扱いますが，ソフトウェア開発におけるプログラミング言語は現在，知名度の低いものも含めると 200 種類以上あります．特定の言語から派生したものを含めると 1,000 種類を超えます．これら全てを科目として用意できないのは自明のことですし，全てを身につけるには時間がいくらあっても足りません．加えて，今後も新しいプログラミング言語が次々に登場するはずです．すなわち，皆さんは社会に出てからも，その都度必要な知識を自ら学んでいかなければなりません．

ハッカソンでは，システムを完成させるのに必要な知識がチームごとに異なりますし，チーム内でも役割によって変わります．ハッカソンで必要となりそうな知識のうち，重要と考えられるものは科目として学べるようにカリキュラムが作られていますが，それでも全てを網羅することはできません．したがって，**必要な知識は**

自ら調べて学んでいく必要があります．それを実践する科目がハッカソンです．学びには終わりがありません．大学での学びというのは「人が教えてくれるのを待つ」のではなく，能動的に「自ら学ぶ」自分なりの方法を身につけることであると考えます．これがハッカソン 1 の**第 1 の目的**です．

ところで，情報工学科の卒業生の多くは製品・システム・サービスなどを開発・構築・販売する企業に就職しますが，個人の趣味でアプリを開発するのとは異なり，企業における製品やシステムの開発は一人では行いません．例えば，日本の代表産業である自動車の開発を例にとれば，開発（企画，デザイン，設計，試作・評価）の後に，生産技術（コスト，工程計画，設備詳細，設備調達，設備テスト，品質確認）が検討されます．その後も開発部署と生産部署のやりとりが幾度となく繰り返されるので，実際の製造が始まるまでには何年もかかります．この間，100 名以上の技術者が関わることも珍しくありません．さらに言えば，使われる個別のパーツ（エンジン，トランスミッション，ヘッドライト，エアコン，ナビゲーションシステム，オーディオ等）ごとにそれらの工程が行われますので，自動車の開発には途方もない人数の技術者が関わります．このときに重要となるスキルがコミュニケーション能力です．

2017 年の調査では，大学生の 54%以上，社会人でも 58%以上の人たちがコミュニケーションを苦手と感じています[1)]．コミュニケーション能力といっても色々な要素があります．分かりやすいところでは，初対面の人とすぐに打ち解けられる能力とか，楽しい話で場を盛り上げる能力などがあります．しかし，技術者が開発に使うコミュニケーション能力とは，自分の考えや事実を正しく伝える能力，相手の考えや事実を正しく理解する能力です．議論で相手を打ち負かすのではなく，お互いを理解して新たなものを創造するためのアイデアを共に産み出すために活かせることが重要です．

ハッカソンでは，企画から要件定義，設計，開発，テストまでをチームで行います．企業での開発とは異なり，チームメンバーと利害関係があるわけではなく，立

1)　「コミュニケーション総合調査＜第 3 報＞」, JTB コミュニケーションデザイン, https://www.jtbcom.co.jp/article/hr/547.html, 2018.

場の上下も存在しません．フラットな関係のなかで，一つの目的に向けて話し合い，役割分担を決め，進捗を確認しつつ，問題があればそれを解決して，最終的に一つの成果物を作り上げます．チーム内である程度の役割分担はあるものの，基本的に全ての工程にチームメンバー全員が関わります．そのため，ハッカソンは**チームワークとコミュニケーションを体験する**最良の機会となります．これがハッカソン 1 の**第 2 の目的**です．**第 3 の目的**は，皆さんが与えられたテーマの内容を良く理解し，自ら立てた計画を実行した後，得られた成果を正しく評価した上で**報告書を書き，プレゼンテーションする経験を積む**ことです．報告書の執筆にもプレゼンテーションにもコミュニケーション能力が問われます．

1.3　ハッカソン 1 の概要

イベントとしてのハッカソンは，通常 1 日〜1 週間程度で短期間で集中的に実施されますが，本学科の科目としてのハッカソン（1 と 2 が同時開講）では，2 年生と 3 年生でチームを構成し，各チームは与えられたテーマに関するシステム開発を 13 週間かけて完了させます．ただし，システムの制作はハッカソンの授業時間内だけでは終わりませんので，授業以外の時間も使ってチームの打ち合わせや個々の分担作業を行う必要があります．他の授業や日々の生活がある中で成果物を完成させるには，各人が割り振られた役割を十分に理解して，ときにはチームメンバーの助けを得ながら計画的に進めることが開発を成功に導きます．このことは準備をはじめ，開発の全ての段階で極めて重要です．次に，一般的なハッカソンと科目としてのハッカソンの進め方の違いについて説明します．

1.3.1　一般的なハッカソンの流れ

ハッカソンは開催する目的や内容によって進め方に多少の違いはありますが，大まかな流れは決まっています．短期間で実施する一般的なハッカソンの進め方は以下のとおりです．

(1) 企画（テーマ決め）

主催者が，どのようなテーマでハッカソンを実施するのか企画を考えます．ハッカソン 1 については，教員が提示するテーマで実施します．ただし，前提として Arduino を用いたシステムを作成するというのが毎年共通の要件です．注意して欲しいのは，できる限り Arduino のマイコン（マイクロコントローラ）上で処理が完結するシステムを構築する必要があるということです．

例えば，Arduino にマイクを繋げて，音声信号を入力するシステムがあるとして，Arduino が入力装置部分のみを担って，入力された信号の解析や処理は Mac で行うといった実装の仕方は認めません．メモリや処理能力に制限がある Arduino のマイコンの中で処理が完結するシステムを作成する必要があるということです．Mac 上で行って構わないのは入出力部分のみと理解してください．Mac 上で行う処理について，判断に迷ったら教員に相談してください．

(2) チーム分け，アイスブレイク

通常のハッカソンでは，主催者が参加者のバランスを見ながらチーム分けをします．あるいはチーム単位で参加を受け付けているハッカソンの場合もあります．ハッカソン 1 については，教員が学生番号などによって事前に機械的にチーム分けします．ハッカソン 1（2 年生）とハッカソン 2（3 年生）は同時開講されますが，1 チームは 2 年生 3 人，3 年生 3 人の計 6 人で構成されます．ただし，年度ごとの受講人数の違いによって 3 で割り切れない場合には，5 人や 7 人のチームになる場合もあります．初年度だけは 2 年生しかいませんので，4 人程度のチームで行います．

(3) インプットセミナー，グループワーク，開発

ハッカソンをスムーズに進めるためには，テーマの目的や意図などを十分に理解する必要があります．まずテーマに対して何を実現するのか，何を解決するのか等の課題をチーム内で洗い出します．その上で課題を一つに絞り込み，複数の実現・解決法を検討し，最終的に実装する機能を決めていきます．実装内容が決まったら，必要となるタスクを洗い出して工数を見積もり，担当者を

割り振ります．以降は，期限までにメンバー同士で協力しながら開発を進めていきます．各人の役割分担をチーム内でよく打ち合わせておく必要があります．

(4) 発表と評価

開発が最後まで終わったら，プレゼンテーションの準備をし，チームごとに成果を披露します．この審査は，主催者側の審査員による採点や参加者の投票により行われるのが一般的です．ハッカソン 1 でも，発表会の評価は教員，TA および参加者で行います．通常のハッカソンの流れは以上ですが，ハッカソン 1 の場合には，情報工学を学ぶ科目という観点から，制作したシステムの性能評価を行い報告書を作成するという内容が加わります．通常のハッカソンでは，ソフトウェアの開発が多いのに加え，短期間で完了する必要があることから，制作物の定性的評価（使いやすさとか満足度のような評価）はあっても定量的な評価（何%改善したとか実行速度が 1 秒以下のような評価）までは行わないことがほとんどです．しかし，ハッカソン 1 はソフトウェアとハードウェアを組み合わせたシステムの開発であるため，報告書をまとめる上でも，定量的評価を必須とします．開発が終わった段階で，報告書をまとめるのに必要な性能評価を行います．

1.3.2　ハッカソン 1 の進め方

次に，より具体的にハッカソン 1 の進め方を示します（2025 年度は 2 年生のみで行うため，2025 年度に限定の流れです）．

第 1 週: 顔合わせ，テーマの理解

(1) 顔合わせ　チームが決まったら，緊張を解し，コミュニケーションが取りやすくなるように，自己紹介等を行ってください．そしてまずはチーム名（後から変更も可）と暫定的でも構わないのでチームリーダーとチームの提出物の提出係を決めましょう（同じ人でも構いません）．提出物の提出係には開発途中におけるチームの進捗報告を毎週 manaba へ提出する

役割を担ってもらいます．

(2)**テーマの理解**　2025 年度はチームに 3 年生がいないので，作成するシステムをある程度絞った 3 種類の課題テーマの中からどれか一つを選んで取り組んでもらいます．テーマによって，作る機能が決まっている（アイデアは必要）ものと，ある程度自由にアイデアを盛り込めるものがあります．まずは各テーマの説明を聞き，チームでどのテーマを選ぶのかを話し合いましょう．授業中に希望調査を取ります（ただし，各テーマに均等にチームを振り分けるので希望通りになるとは限りません）．また，学科から貸し出せる機材等の説明もします．それを元に何を作ろうとするのか，アイデアを話し合うことになります．ハッカソンで重要なのはアイデアを沢山出すことです．そのため，参加者同士がアイデアの否定や批判，討論を行うような雰囲気は避けましょう．ハッカソンを成功させるためには，アイデアが出やすい環境を作ることが大事です．事前にアイデアを出す手法（本テキストを参考にしてください）を相談したり，その際のルールを決めておいたりすると良いでしょう．また，短期間で実施するハッカソンとは異なりますので，単にその場の思いつきのアイデアだけではなく，文献や資料等を調査した上で実現可能な具体的なアイデアを考えることが重要です．4 週目までに分担して技術の調査や資料の収集を行いましょう．

なお，チームの連絡手段には manaba を利用します．成績評価では manaba のチーム内スレッドでのやりとりによるチームへの貢献度合いも考慮しますので，積極的に参加してください．また，チームリーダーがチーム全体の進捗報告を毎週とりまとめるために必要となる各人の進捗の報告も manaba のチーム内スレッドで行ってください（他の連絡手段を使われると評価の対象にできません．その週のチームへの貢献はゼロと見なされます）．これは計画書の作成時も同様です．

第 2 週及び第 3 週: テーマごとの学習と演習

テーマごとに分かれて実施します．まずは，テーマに関係ありそうな技術等の説明を受けた後，演習を行います．チームでの話し合いの時間もありますの

で，資料収集や検討の分担など必要な打ち合わせも行なってください．

第 4 週: アイデアの絞り込みとチームの役割分担

各人が出したアイデアをもとにチームで実際に作成しようとするシステムを絞り込んでいきます．制作に必要な材料や測定機材なども考えておく必要があります．学科には貸し出し可能なセンサやデバイスを，ある程度用意しています（ただし，比較的安価なものばかり．一覧表を参照）．しかし，揃えられるものには限界があります．もし，別途購入するものが出てきた場合，その負担割合はどうするのか，ハッカソン終了後に誰の所有物にするのか等は話し合っておく必要があります．汎用性の高いものであれば，学科で購入して貸し出すことも検討しますので，まずは教員に相談してください．あまりに高価なものや単に楽をするための導入の場合は，使用を制限することがあります．ハッカソンですので，技術課題をお金で解決するのではなく，エンジニアらしく無ければあるもので作る，というスタンスで開発に臨んでください．最終的に実装する内容が決まったら，どのような工程が必要になるのかを話し合います．メンバー個々のスキルを確認し，チーム内での役割分担を決め，それらを計画書にまとめていきます．

第 5 週: アイデアの確定と実施計画

アイデアを実装するのに必要な工程をまとめます．いつまでに，ここまで出来てたらいいな，という希望を書くのではなく，必要な工程を洗い出し，確実に遂行可能な計画を立てることが必要です．また，作業を進める中で計画段階で気づいてなかったことが判明することもあるので，途中で軌道修正ができるよう，ある程度余裕をもった計画にする必要があります．なお，作成した成果物は単に希望通りに動いたというだけでは不十分で，報告書にまとめるためには性能評価を行う必要があります．どのような評価をすべきなのかを評価の妥当性を含めて検討してください．

第 6 週: 計画内容のプレゼンテーション

計画書を提出するとともに，計画内容をプレゼンテーションします．教員や TA により指摘を受けた場合には，それを踏まえて計画内容を修正します．

第 7 週〜第 10 週: 実際の制作

授業時間内にできることには限りがありますので，授業中は制作を進めるというより，チーム内での各人の進捗の確認や，スケジュールの微修正などの打ち合わせ，自宅ではできない動作確認・結合実験や測定器での測定などを行う日であると考えてください．なお，課題としてチームの進捗報告（チームで 1 通）を毎週提出してもらいます．

> ! **事故防止** ハッカソンでの制作で皆さんが扱う電圧はたかだか交流 100 [V] までですが，不用意に扱うと感電事故や火災を起こす恐れがあります．十分に注意してください．また，貸し出し機材は丁寧に扱い，万一異常が起きたときには直ちに担当教員に報告してください．

第 11 週: 成果発表・報告書作成のためのシステム評価の会

最後の成果発表会はプレゼンテーションの時間が限られていますので，その場で作成したシステムの動作を全て見せられない（動作にスペースが必要だったり，特殊な環境が必要だったりする）場合もあります．そのため，予め動画に収めておくなどの工夫が必要になります．

また，測定器を用いる定量評価も自宅ではできませんし，ヒューマンインターフェースなどの定性評価のために，制作物の内容を知らない他のチームのメンバーに被験者として協力してもらいデータを取得することが必要になる場合もあります．11 週目はそれらのことを一斉に行う日とします．そのため，この日までにシステムが完成している必要があります．参加者は必要に応じて他のチームのシステム評価にも協力しましょう．

なお，観測・測定は細心の注意を払い，いつも，これが本番であるとの心構えで行いましょう．結果を予測して測定値を無理に読み換えたり，失敗したらまたやり直せば良いなどと考えていてはよい結果を望めません．**各人がノートを用意し**，測定値やその計算，観測結果などを記録します．ノートには観測・測定の経過を細大もらさず記録し，気付いたこと，疑問点や感想なども書き留めておくと良いでしょう．紙切れに書いてあとからまとめようとするのは無駄であるばかりか，間違いのもとになります．ディジタルカメラ等による記録も

何をどう写したのかの記録がなければ不完全です．例えば，オシロスコープの画面のみを写しても，縦軸・横軸の設定までは記録できません．きちんと記録をとりましょう．それから，測定中もできる限りデータを吟味してください．必要ならばデータをグラフ化し，その傾向をつかんで，測定をどのように進めるべきかを検討しましょう．また，得られた結果が不規則であったとしても，それらを無原則に捨ててはいけません．不規則なデータが別の問題を暗示している場合もあります．

測定や評価が済んだら，使用した機材を元の状態に戻し，必要な手入れをしたのち，数量に誤りの無いことを確認して元の場所に返却してください．

第 12 週: 報告書の提出および成果発表会

午前 9:00 までに manaba を通じて最終報告書を提出します．できるだけ完成度の高い報告書を作成し，1 回で受理されることを目指しましょう．その後，最後の成果発表会を実施します．実施の形態については別途指示します．

第 13 週: 報告書へのコメント返却・修正内容の検討

内容が不十分な報告書については，13 週目の授業時間内までにコメントをつけて返す予定です（教員 3 人で 130 通超の報告書を読むので事情により遅れる場合もあります）．返却された場合には，コメントの内容を十分に吟味して報告書を修正してください．また，測定や評価をやり直すような指示があった場合には，この日に行ってください．完成した報告書は 14 週目までに再提出して全てが完了となります．

1.3.3　成績評価，およびその他の注意

ハッカソン 1 及び 2 は，チームでシステムの制作を行いますが，授業としての成績は個人に対してつける必要があります．基本的に成績は，チームで取りまとめる（計画書/進捗報告など）提出物やプレゼンテーション，個人で執筆する最終報告書によって評価します．全体に対する各成績項目の割合は

- プロジェクト計画書（チーム）【15%，1 回】

- プロジェクト計画発表（チーム）【15%，1 回】
- 各回の進捗報告書（個人・チーム）【14%，7 回】
- 演習課題（個人）【6%，3 回】
- プロジェクト成果報告書（個人）【25%，1 回】
- プロジェクト成果発表（チーム）【25%，1 回】

となっており，評価合計が 60%以上で合格となります．

注意事項

- 提出物は，提出日の指定時刻までに manaba の指定の場所に提出しなければなりません．提出期限に未提出あるいは未完成と認められるものが提出された場合，**不合格**として処理するので，注意してください．
- 正当な理由なき遅刻，欠席，および各種期限が守られない場合にも**不合格**とする場合があるので，注意してください．

第２章　実践編

この章では，テーマの説明を受けたのちにチームで計画を立てるにあたり，各過程で使えるであろう，いくつかのテクニックについて概説します．

2.1　アイデアの出し方（発散）

新しいシステムやサービスを考えるとき，最大の難所となるのは最初のアイデア発想です．アイデアを決定するプロセスは，アイデアの素をひたすら発想する（発散）段階と，それを整理して一つにまとめ上げていく（収束）段階に分けて考えることができます．

アイデアの発想のためには，まず最も基本的な骨組みとなる４つのアイデア構成要素を理解しておく必要があります．

要素①　誰（ヒトやモノ）のため？
要素②　どのような課題？
要素③　解決の方向性は？
要素④　どのような方法で？

この４つの要素に対して，以下の４つのケースに分けて，それぞれに適するアイデア発想方法を考えます．

ケース１　全ての要素が未確定，あるいは漠然としている場合

４つの要素全てが未確定であり，まずはイメージレベルからのスタートでも良いので，本格的な検討のきっかけや足がかりとなるアイデアの種を出したいというケース．

ケース 2　要素①の誰（ターゲット）が確定している場合

4 つの要素のうち「誰」というターゲットが確定している場合，これらを起点としてアイデアを検討していくケース．

ケース 3　要素④の解決の方法（リソース）が確定している場合

4 つの要素のうち「方法論・リソース」すなわちどのようなデバイス，資源（技術やスキルなど）やサービスを使うのかということが確定している場合，これらを起点としてアイデアを検討していくケース．

ケース 4　要素②の課題がある程度確定している場合

4 つの要素のうち「課題」が確定している場合，これらを起点としてアイデアを検討していくケース．

2.1.1　ブレインストーミング，ブレインライティング，ゴートン法

これらは，アイデアの内容に関する判断は後回しにして，とにかくアイデアの質よりアイデアの量を求めるときに活用される発想法です．制約条件がほとんどない発想法のため，検討すべきテーマが絞りきれていない場合は，実施効果が低下するというリスクがありますが（例えば漠然と「新規事業のアイデア出し」などのレベルでは良い検討にはならない），アイデアの種や検討すべき方向性が一切定まっていない状態でも実施できるほぼ唯一の発想法とも言えます．これを踏まえてケース適応度を考えると次のようになります．

適応度			
ケース 1	ケース 2	ケース 3	ケース 4
○	△	△	△

ブレインストーミングは，最もシンプルかつ定番の発想法であり，グループでアイデアを引き出す手法です．どのような些細なアイデアでもよいので，とにかく数多くのアイデアを出すことを最大の目的としています．皆さんもアイデアソンで経験しているはずです．

基本的なルールとして，①批判厳禁（他人の意見を批判・批評してはいけない），

②自由奔放（突拍子もない意見でもよい），③質より量（アイデア・意見の質・内容は一切不問），④便乗歓迎（他人の意見・アイデアに乗っかったアイデア・意見を歓迎する）ということを周知徹底することが重要です．あらかじめ制限時間を設定しておき，時間内に終えるようにすることで，高い効果を発揮できます．

ブレインストーミングの方法

- ホワイトボード，付箋，ペンなどの備品をそろえておく
- 司会，書記の役割を決める
- 事前に時間を設定する．時間内に終えるようにすることで，高い効果を発揮できる
- ブレインストーミングの時間は，長くとりすぎないのが理想的．テーマに応じて 20〜30 分が適切で長くても 1 時間にする．
- 設定した時間の中で，アイデアを発想・量産（50 以上が理想，100 くらいあってもいい）する作業と，量産したアイデアをまとめる作業にプロセスを分けて行う

ブレインライティングは，ブレインストーミングを紙上で展開する発想法であり，一人いくつと決めておけば，参加者全員に均等にアイデアを求めることが可能となります．また，他人の意見を聞く時間を割愛できるため，ブレインストーミングと比較して，アイデアを考える時間を多くとることが可能です．

ゴートン法は，ファシリテータ（司会・進行役）以外は本当のテーマを知らない状態で参加者にブレインストーミングを実施してもらう発想法です．本当のテーマは伏せるものの，検討して欲しいテーマは明確にする必要があります．本質的な検討テーマとは異なるテーマ設定をしつつ，最終的に本来の目的につなげていく必要があるため，ファシリテータの手腕が問われる発想法となります．本来目的とする検討テーマを隠すことによって，参加者の検討の視野や思考が「一般的な常識」などに縛られ，斬新なアイデアが出難くなることを回避できるという効果を期待できます．

2.1.2　マンダラート法，ハチの巣ノート，マトリックス法

これらは，アイデアを検討していくべき方向性について大枠が定まっていれば実施できる発想法ですが，検討テーマの共有や枠の軸の設定が甘いと，検討が上手く行かなくなる可能性が高まります．ケース適応度は次のようになります．

適応度			
ケース 1	ケース 2	ケース 3	ケース 4
×	○	○	○

マンダラート法は，9 つのセル/マスを使ってアイデアを出していく発想法です．マンダラートとは仏教に登場する曼荼羅模様に由来する言葉で，曼荼羅とアートを組み合わせた造語です．縦横 3 つの行と列からなる 9 つのマス目を作成して，中央のセルに検討テーマを書き，その周辺のセル/マスに検討テーマに沿ったアイデアを書き出していきます．セル/マスを全て埋めることを必須とすること以外の制約条件はありません．

この発想法のポイントは 2 つあり，1 つは 8 つのマス目を漏れなく埋めるようにすることと，これを 2〜3 度程度繰り返し実行することにあります．ブレインストーミング等とは異なり，「限定された数のマス目を埋める」という強制性がアイデア発想の思考を助けると言われています（全く自由ににアイデアを出せといわれるよりも，周囲 8 つのマスを全て埋めろと言われる方が取組みやすいことを利用します）．

ハチの巣ノートは，ハチの巣状のセル/マスを使ってアイデアを出していく発想法です．中央に検討テーマを書き，その周辺のセル/マスに検討テーマに沿ったアイデアを書き出していくのはマンダラートと同様ですが，類比・連想により検討しているアイデアをどんどん自由に発展・展開させることにポイントがあります．

マトリックス法は，マトリックス表を作成して縦軸（表側），横軸（表頭）に任意の変数を設定し，それら変数の組合わせによるアイデアの検討を行う方法です．縦軸，横軸の変数の設定が有意義な検討になるかどうかを決めるポイントになります．

2.1.3　シックスハット法

シックスハット法とは，その名のとおり 6 色の帽子を使って行うアイデア発想法です．アイデアの検討視点を 6 つ準備して，それぞれに従った視点でのアイデア発想を行います．多様な視点から物事を見るための手段として開発されたのがシックスハット法です．参加者どうしでアイデア検討の論点が定まり難い時などに，視点を揃えるのに有効です．原則的には検討の方向性・テーマ，すなわち検討すべきアイデアのテーマがある程度定まっている必要があります．ケース適応度は次のようになります．

適応度			
ケース 1	ケース 2	ケース 3	ケース 4
○	×	×	×

シックスハット法では帽子の色によって話し合う視点が決められています．

チェックリスト（シックスハット）		
白	客観的，中立的	データに基づく客観的な事実を述べます．自分の意見や判断は言わないようにします．
赤	直感的，主観的	直感的な意見を述べます．好き嫌いなど感覚的，感情的な意見で OK です．
黒	否定的，悲観的	論理的にネガティブな意見を言います．あえて最悪の状態を想像し，リスクや懸念点などを見つけ出します．
黄	肯定的，楽観的	ポジティブな視点でどうすれば実現できそうかを話し合います．大げさでもいいので長所や利点を探します．
緑	革新的，創造的	クリエイティブに新しいアイディアを提案します．黒のアイデアを乗り越えられるような革新的なアイディアを考えます．
青	俯瞰的，統括的	全体の流れを俯瞰し，次に何をすべきか結論を出します．

2.1.4　オズボーンのチェックリスト，SCAMPER 法，TRIZ 法

これらは，検討の視点・着眼点を恣意的に多角化/多様化，かつ論点をフォーカスしてアイデアを検討する発想法です．個別にフォーカスされた既存ビジネス，商品，サービスの改良・改善に関するアイデアの発想に適しています．ケース適応度

は次のようになります．

適応度			
ケース 1	ケース 2	ケース 3	ケース 4
○	×	×	△

オズボーンのチェックリストは，予め準備/用意されているチェックリスト（質問リスト）への回答によりアイデアを発想させていく発想法です．アイデア発想のテーマがフォーカスされていなければ有意義な検討になりません．商品・サービスの企画アイデアの検討に適しています．

チェックリスト（オズボーン）	
転用	他に使い道はないか？新しい用途はないか？他の利用方法はないか？
応用	他・過去からアイデアを持ってこれないか？近似・類似のものはないか？
変更	色，形，動作，匂い，意味，重量，長さ，等々の何か・どこかを変えてみてはどうか？
拡大	大きくしたらどうか？加えたらどうか？多くしたらどうか？拡張したらどうか？
縮小	小さくしたらどうか？減らしたらどうか？少なくしたらどうか？
代用	他の人にさせたらどうか？他のものにしたらどうか？他の場所ではどうか？
再編成	要素，配列・配置・順序，組合せ，スピード，因果等を変えてみればどうか？
逆転	逆にできないか？正比例・正反対ではどうか？前後を入れ替えられないか？
結合	組合せられないか？統合できないか？合体できないか？

SCAMPER 法は，オズボーンのチェックリストを簡略，発展させた発想法です．9 つのチェックリスト項目（質問事項）を 7 つに整理・集約し，その頭文字をとって SCAMPER 法と呼ばれます．それぞれに複数の質問事項を準備/用意して臨む場合もあります．製品や事業のアイデア出しに向いています．

チェックリスト（SCAMPER）	
Substitute（代用する）	何かで代用できないか？何かを代用できないか？
Combine（組み合わせる）	何かと何かを組み合わせることはできないか？
Adapt（応用する）	他の何かを適用・応用できないか？
Modify（修正する）	何かを修正（変更）することはできないか？
Put to other uses（転用する）	何かに転用できないか？他のことに使えないか？
Eliminate（削ぎ落とす）	何かを削除・減らすことはできないか？
Reverse, Rearrange（再構成する）	何かを逆転・組み替えることはできないか？

SCAMPER 法は 0 からアイデアを生み出すのには適しておらず，既存のアイデアを拡張・派生させて多くのアイデアを量産する発想法です．したがって，SCAMPER 法を使用する前に最低限，核となるいくつかのアイデアを用意しておくことが重要です．

TRIZ 法は，特許データの分析から導き出された 40 の発明原理のパターンに基づく端的な質問事項による発想法です．とくに技術アイデアについての発想に向いています．

40 の発明原理（TRIZ）		
1	分割	分割したらどうか？
2	分離	分離したらどうか？
3	局所性質	一部を変更したらどうか？
4	非対称	非対称にしたらどうか？
5	組み合わせ	2 つ以上を合わせたらどうか？
6	汎用性	他でも使えるようにしたらどうか？
7	入れ子	中に入れたらどうか？
8	つりあい	バランスをよくしたらどうか？
9	先取り反作用	先に反動をつけたらどうか？
10	先取り作用	先に予想したらどうか？
11	事前保護	重要なところを保護したらどうか？
12	等ポテンシャル	同じ高さにしたらどうか？
13	逆発想	逆にしたらどうか？
14	曲面	回転させたらどうか？
15	ダイナミック性	環境に合わせたらどうか？
16	アバウト	おおざっぱにしたらどうか？
17	他次元移行	垂直方向を使ったらどうか？
18	機械的振動	振動を与えたらどうか？
19	周期的作用	繰り返しにしたらどうか？
20	連続性	継続的に続けたらどうか？
21	高速実行	高速で実行したらどうか？
22	災い転じて福となす	マイナス点からプラスを引き出せないか？
23	フィードバック	基準値に戻したらどうか？
24	仲介	仲介したらどうか？
25	セルフサービス	自分で行ないようにしたらどうか？
26	代替	コピーしたらどうか？
27	高価な長寿命より安価な短寿命	安くてすぐダメになるものを作ったらどうか？
28	機械的システム代替	別のシステムを使ったらどうか？
29	流体利用	流体を使ったら（にしたら）どうか？
30	薄膜利用	薄い膜を利用したらどうか？
31	多孔質利用	スキマを利用したらどうか？
32	変色利用	色を変えたらどうか？
33	均質性	質を統一したらどうか？
34	排除／再生	排除したらどうか？再生させたらどうか？
35	パラメータ	形や条件を変更したらどうか？
36	相変化	形状を変更したらどうか？
37	熱膨張	熱を加えてふくらませたらどうか？
38	高濃度酸素利用	濃度を濃くしたらどうか？
39	不活性雰囲気利用	反応しないものを入れたらどうか？
40	複合材料	違う質のものを合わせたらどうか？

2.1.5　欠点列挙法，希望点列挙法，特性（属性）列挙法

これらは，検討の視点・着眼点を欠点や希望点，特性・属性などにフォーカスしてアイデアを検討する発想法です．個別にフォーカスされた既存ビジネス，商品，サービスの改良・改善に関するアイデアの発想に適しています．ただし，実際に検討する際はターゲットをできるだけ明確化した方が良いでしょう．ケース適応度は

次のようになります．

適応度			
ケース 1	ケース 2	ケース 3	ケース 4
○	△	△	×

欠点列挙法は，アイデアを検討する商品やサービスについての欠点にフォーカスした発想法です．欠点列挙法では，些細な欠点や問題点を可能な限り探し出して列挙します．あえてマイナス面を見つけ出すことで普段は見逃している問題点を洗い出し，解決策やアイデアを拾い出す手法です．

希望点列挙法は，アイデアを検討する商品やサービスについての希望点にフォーカスした発想法です．希望点列挙法では，「こうすればもっと良くなる」という理想や夢を可能な限り書き出します．その中から現実味のあるものを選び，革新的なアイデアを得るという手法です．

特性（属性）列挙法は，アイデアを検討する商品やサービスの特性や属性など，構成要素にフォーカスした発想法で，モノが持つ属性を 3 つの観点に分けて列挙し，分析をしてアイデアを生み出す発想法です．新商品の開発や商品の改良など，モノの問題を分析する時に役立てることが出来ます．この手法のベースになっているのは，次の 2 つの考え方です．

1. 全てのアイデアはそれ以前にあったアイデアを何らかの方法で変化を加えたものであり，従来のものをきちんと研究することで新しいアイデア発想につながる．
2. 課題を小さく分解していけばいくほどアイデアが出やすくなる．

属性列挙法（特性列挙法）では，特定のモノについて考えられる全ての属性を下記の 3 つに分けて列挙していきます．

1. 名詞的属性（全体，部分，材料，製法）
2. 形容詞的属性（色，性質，形，デザイン）
3. 動詞的属性（メカニズム，機能）

属性の一つ一つについて改善案を出来るだけたくさん出し，アイディアを発想していきます．

2.2　アイデアの整理法（収束）

新しいシステムやサービスを考えるときの最大の難所，最初のアイデア発想は，できるだけ自由な発想を妨げずに実行することが重要です．しかし，自由な発想で出したアイデアは多種多様であるため，それらをそのまま並べて眺めているだけでは，意味・意義あることを読み取ることはできません．そこで，次にアイデアを整理していく方法について考えます．整理法は大きく 3 つに分類できます．

整理法 1：ツリー構造化する　ツリーあるいはヒエラルキー構造による整理によって可視化していく整理方法です．出されたアイデアの質やバランスに，比較的バラつきが大きい場合には，まずこの方法でアイデアの構造化を図ることが先決です．その後，必要に応じて以下にあるポジショニングやフレームワークによる整理を行います（どうしても使えないアイデアがあれば切り捨ててから整理しても構いません）．

整理法 2：ポジショニング化を図る　出されるアイデアが，ある一定の階層水準でそれなりに揃っている場合は，同一階層内におけるポジショニング整理を試みます．この場合，平面での配置（ポジショニング）となりますので，1〜2 軸に沿ったマトリックス配置をイメージして整理していきます．この軸設定が最も重要であり，顧客や利用・購買シーン，商品・サービス，価格，等，任意に設定した上で，そのカテゴリーにおける構成要素を軸要素として配置します．

整理法 3：フレームワーク的に整理する　整理法 1 や整理法 2 とは異なり，アイデア発想の結果，ある程度ビジネスアイデアが見えてきているような場合に適用する整理方法です．ビジネスアイデアとして必要な構成要素を並べておき（フレーム），それら一つ一つについてアイデアを埋めていくスタイルで整理します．その際，要素が全て埋まらなくても問題ありません．

2.2.1　KJ 法

KJ 法は，カード（付箋）に思いつくままにアイデアを書き出して，カードを並べ変えたり，グループ分けしたりすることにより，断片的なアイデアを整理する手法です．考案者の文化人類学者である川喜田二郎氏のイニシャルを取って KJ 法と名付けられています．KJ 法では，整理の対象となるアイデアが必要であり，ブレインストーミングでアイデアを出してから行うと，アイデアを効率的に整理できます．そのため，KJ 法とブレインストーミングはセットで行われることが多いです（タイプ：整理法 1）．

KJ 法の方法

- **アイデアを書き出す**

  KJ 法を実施する前に，テーマを参加者に伝え，テーマに沿ってブレインストーミングを行い，多くのアイデアを出し合います．アイデアは，1 枚のカード（付箋）に 1 つ書き出すようにします．

- **アイデアをグルーピング**

  カードを整理し分類します．まずは無造作にカードを並べ，どのようなことが書かれているかをよく読み，印象が似ているアイデアや同じような意見を集めて小さなグループを作ります．

  その後，小グループをよく観察し，カードのアイデアのなかで，共通して表現していることを小グループの見出しとします．小グループの見出しを見て，親近性のある小グループ同士を中グループにまとめます．さらに作業を繰り返し，数個の大きなグループができたら，グルーピングは終了です．

- **関係性を図解化**

  その後，各アイデアの関係性を見出しながら論理的整序を行い，図解化していきます．論理的整序のポイントは，以下の通りです．

  1. 各カードのグループごとに関連性があるもの同士を近くに配置する（空間配置）．

2. 線でカード同士をつないだり，囲んだりして関係性を可視化する．
3. 対立，因果関係，原因・結果などの関係性を矢印などで書き表す．

- **文章化（叙述化）**
  前ステップで図解化した関係を言語化します．言語化する際には，グループのカードに書かれている言葉（ワード）をできるだけ多く使うことがポイントです．グループごとの関係性を図解化した後，グループの重要度に応じた優先順位を付けて参加者で議論・共有し，その結果を文章化します．

2.2.2　セブンクロス法

多数のアイデアについて，ある軸により 7 つ程度に分類し，さらにその分類の中での重要度について 7 段階で整理する方法です．これにより軸に設定した 7 つの項目ごとに重要なアイデアが抽出できます．ただし，7 つというのは一つの目安であり，多少前後しても構いません．この方法では，軸の設定がポイントであり，ある程度，同一水準，基準の中でアイデアが揃っている場合に適用できる整理法です（タイプ：整理法 2）．

2.2.3　5 W 1 H法

ビジネスアイデアとして必要不可欠な基本的要素について，5W1H のスタイルでフレームワーク的に整理する方法です．ある程度ビジネスアイデアのイメージが出来つつある場合に適する整理法です（タイプ：整理法 3）．

2.2.4　マインドマップ

マインドマップは，人間の自然な思考プロセスを可視化する手法です．まず，紙やホワイトボードの中心にテーマとなる問題，もしくは関連するキーワードを書きます．その周りに解決策やアイデアを書き込み，中心にあるテーマと線でつなぎま

す．さらに細分化させ，アイデアを出し切るまで連想する要素をつないでいくという流れです．マインドマップの作成により漠然としたアイデアを分解し，具体的な形にしていくことが可能です．

2.2.5　ストーリーボード

ストーリーボードは，イラストでアイデアを可視化するフレームワークです．アイデアとそこから想定される結果を，イラストや画像を使ってストーリーとして可視化します．発想したばかりのアイデアを整理する際に役立つ手法です．思いついたばかりのアイデアは抽象的で，言葉で説明してもイメージしにくいことがあります．このとき，イラストによるストーリーにすれば，言語化が難しいものでもイメージが可能です．

2.3　アイデア出しに困ったら？考え方のポイント

フレームワークをいろいろ試してもアイデアがなかなか出なかったり，出尽くしてしまったりした場合は，視点を変えてみるのも手です．以下は，アイデアの出し方に悩んだときに試せる方法です．

2.3.1　消去法

消去法とは，とても採用できないようなアイデアから順に消去していく方法です．これを何度も繰り返し，最後に比較的良いアイデアを残します．各フレームワークで出したアイデアがどれも今ひとつで，どれを採用すればいいかわからないといった場合に役立ちます．積極的に良いアイデアを求めている場合には向きませんが，現在出ている時点で最も良いアイデアを選びたいときに効果的です．

2.3.2　逆張り思考

逆張り思考とは，アイデアがなかなか浮かばないとき，自分の中で常識としていることの逆の思考・行動をしてみる方法です．逆張り思考は本来，投資で使われる戦略を指します．市場では見向きもされない安い株式を安値で購入し，高値がついたときに利益を得るという戦略で，市場の流れとは逆の発想をする投資方法です．アイデアの出し方にも使うことができ，固定観念や先入観を新しい発想に転換することで，思わぬアイデアが生まれる可能性があります．

2.4　システム開発の手法

システム開発といっても，その手法はさまざまです．プロジェクトの規模やリリースまでの納期など，開発するシステムに適した手法を選択する必要があります．ここでは，一般的に用いられるシステム開発の手法とそのメリット・デメリットを概説します．ただし，企業におけるシステム開発は顧客がいるのが普通ですが，ハッカソンでは自分たちをアピールするためにシステム開発を行うので，その立ち位置や目的が異なります．ここで説明するのはあくまで一般的な開発手法であり，ハッカソンにそのままマッチするとは限らないので，注意が必要です．それぞれの特性を理解してチームなりの方法を構築すれば良いと思います．

システム開発で使用される開発手法として，「ウォーターフォール型開発」，「アジャイル型開発」，「プロトタイピング型開発」などが存在します．このほかにも，「スパイラル型開発」，「DevOps」，「MVC モデル」などが使用されることもあります．業界や企業によっては，このほかにも独自の開発手法を取り入れるケースも普通ですが，手法によって，開発工程は大きく変化します．各手法の特徴を把握しておけば，開発案件に合わせた手法を取り入れやすくなるでしょう．

2.4.1　ウォーターフォール型開発

ウォーターフォール型開発は，要件定義・外部設計・内部設計といったシステム開発の各工程を，上流工程から下流工程へ順に行っていく手法です．「完了したら次の工程に進む」という，分かりやすい開発手法であり，システム開発の経験が少

ない方もイメージしやすいでしょう．

リリースまでの作業手順を完成させてから，次に進みます．計画性をもって前進できるのが特徴です．流れる水のように開発工程が進む様子から，ウォーターフォール（滝）と言われています．計画通りに開発が進みやすい分，組み込みソフトウェアや通信システムのように，「仕様変更を考慮しないシステム」の開発に向いています．

メリット:　システム開発を計画通りに進めやすい，必要な人材を確保しやすい
デメリット: 後工程に戻ることが難しい，開発が長期化しやすい

まずは，開発スケジュールが綿密に組み立てられており，順番に進められる点．1 つの工程が終わったら次に進み，後戻りが発生しないため計画通りに進めやすいメリットがあります．2 つ目は「人材が確保しやすい」点．企画〜リリースまでの計画が立てられているため，費用やプロジェクトにかかる人数を予想しやすく必要な人材を集めやすいのです．さらに，採用率の高い手法である分，経験者も多いのもメリットです．

デメリットは，作業ミスが発生した場合の手間や大きいこと．前段階の作業をベースに進行し，ミスが見つかった場合は，前の工程から見直しが入ります．初めの工程で見つかれば，比較的手間が少なく済むでしょうが，後ろの段階であるほど，時間・コストのロスが大きくなります．

2.4.2　アジャイル型開発

日本語で「素早い」という意味をもつアジャイルですが，その名の通り，アジャイル型開発は「スピード重視のシステム開発」に向いている手法です．作業は，「計画」→「設計」→「実装」→「テスト」の工程を反復していきます．アジャイル型開発は，開発工程を小さくまとめる分，リリースまでの期間が短くできるのが特徴です．

依頼者の要望を取り入れながら，作業を反復することによりシステムの品質を上げやすい手法と言えます．ウォーターフォールとは反対に，仕様変更を前提とした

Web サービスやゲームアプリなどのシステム開発に適した開発手法です．

メリット:　開発期間を短くしやすい，不具合が発生した際に戻る工数が少ない
デメリット: 開発の方向性がずれやすい，進捗確認がしにくい

アジャイル型開発は，「8 割の要件が決まっているが，残りの 2 割の要件はプロジェクトの進行状況に応じて定めていきたい」といったような，柔軟性を要するプロジェクトに向いている手法です．リリースまでに優先順位の高いシステムから開発ができるため，短納期でユーザーに自社サービスを届けられるのがメリットです．柔軟性・スピード感のある仕事を全うできるアジャイル型開発は，流行に左右されやすいゲームアプリ開発や新規事業で，メリットを発揮してくれるでしょう．

反面，仕様変更を前提とした開発手法のため，開発の方向性がずれやすいという欠点もあります．スケジュールや進捗具合が把握しにくいのも，アジャイル型のデメリットと言えます．

2.4.3　プロトタイピング型開発

プロトタイピング型開発とは，開発の早い段階で「試作品（プロトタイプ）」を完成させ，依頼者チェックのもとに詳細な仕様やプログラムを肉付けするスタイルのことです．システム開発の経験が少ない依頼者の中には，実現させたいシステムはあっても，具体的な案が思いつかないケースもあります．漠然とした全体像を，言語化させるのに役立つのが，プロトタイピング型開発です．試作提供という形で可視化することにより，追加したい仕様や本当に求めていた完成ビジュアルを，依頼者から提案しやすくするのです．

メリット:　具体的なイメージがなくても開発を進めやすい
デメリット: 試作品をつくるのに時間や費用がかかる

プロトタイピング型開発のメリットは，試作品に修正を加えて完成に近づける手法のため，柔軟性の高い開発を可能とするところです．また，システムの稼働イ

メージが具体的でない依頼者に対しても，現物を見せることにより具体的なイメージを共有しやすくなるのもメリットと言えます．

デメリットは，システムの方向性が変わっても柔軟に対応できる分，要望が増えることにより，当初のスケジュールとズレてしまう可能性があります．試作品をつくるのに時間や費用がかかるというデメリットも抱えており，大規模なシステム開発には不向きです．

2.4.4　スパイラル型開発

スパイラル型開発とは，アジャイル型開発と同様に作業を反復してシステム開発の品質を上げていく手法です．アジャイル型開発とよく似た開発手法ですが，スパイラル型は「品質がまだ保証されていない」段階で依頼者へ披露するという違いがあります．工程が終わるごとに，依頼者の評価や声を反映していきながら，システム開発のクオリティを高めていく流れです．

メリット:　流行を反映させやすい，臨機応変に仕様変更しやすい
デメリット: 修正の反映に時間とコストがかさみやすい

スパイラル型開発は，開発工程を繰り返しながらアップデートすることにより，質を高められるのが特徴です．その時期の流行をシステムに反映しつつ，クオリティ向上を目指せるのが最大のメリットと言えます．依頼者の声を反映させながら，修正を加えていくため，満足度の高いシステムを提供できるのが魅力です．ただし，修正が発生した際，都度システム開発をしなければなりません．修正反映までに時間とコストがかさんでしまうのが，デメリットです．

2.4.5　DevOps

DevOps とは，開発チームを意味する「Development」と，運用チームを意味する「Operations」を組み合わせた造語です．開発チームと運用チームが連携し合うことにより，品質向上・余分な工数削減など，システム開発・運用を高品質かつ

効率よく進めていきます．

アジャイル型開発のように，「計画」→「設計」→「実装」→「テスト」など，短いスパンで作業が必要な手法に，DevOps の考え方をのせると，「良いサービスを提供する」を目的としたシステム開発へコントロールがしやすくなるでしょう．

手法としては，小さなサイクルで製品をリリース→ユーザーの声を反映させながら，開発→実装→テスト→展開の流れを繰り返しに行っていく流れになります．開発手法としてだけではなく，開発における概念として使われることもしばしばです．

メリット:　システムの利便性と安定性を両立しやすい
デメリット: 開発と運用の連携が乱れると進捗や品質に影響しやすい

DevOps は，開発チームと運用チームが連携し合うことにより，「システムの利便性」「リリース後の安定性」どちらも重視したシステム開発を可能とできるのがメリットです．また，両チームの密なやり取りにより，余計な作業を削減により，リリースまでの道のり短縮化も実現できるのが，DevOps を起用するメリットです．ただ，開発チーム・運用チームが協力し合うことにより，真価を発揮するシステムなので，どちらか一方に問題が生じると，全体の進捗にズレが発生してしまう可能性があります．作業がずれた分，スケジュールに影響が出てしまうのがデメリットといえるでしょう．

2.4.6　システム開発の手法の特徴を知るのが大切

これだけ開発手法がたくさんあると，「結局どの手法が一番いいのか」と感じてしまうでしょう．しかし，システム開発は，案件ごとに予算や納期などが大きく異なるため，「最も良い方法」を断言しにくい側面があります．

そのため，繰り返しになりますが「特徴を把握したうえで開発手法を選ぶ」のが最適と言えます．例えば，大規模なシステム開発ならば，計画性を立てて進められる「ウォーターフォール型開発」を採用する，リリースまでにかけられる時間が少ないのであれば「アジャイル型開発」など，プロジェクトに合わせて開発手法を選

ぶのが，システム開発を円滑に進めるコツです．ハッカソンの場合は，完成していなければならない日が決まっているので，そこが最優先になります．どの手法を採用しても，**スケジュールに影響が出ないようにする**ことが必要です．

2.5　ハッカソンの成果物の評価について

科目としてのハッカソン１の評価については，シラバスにある通り，日々の進捗報告等も含めてトータルに判断しますが，その中にはハッカソン自体の審査も含まれます．一般に，ハッカソンの審査は，

- **問題着眼点・着想点**
  - ・何の問題を解いているのかが明確で，その問題の着眼点が新しい
  - ・問題自体は新しくないが，解き方が斬新，技術的に優れている
- **実行・実現可能性**
  - ・実際に世の中で利用される可能性が高い
  - ・スケーラビリティが高い
- **完成度・動作性**
  - ・コンセプトで提示された機能が実際に動作している
  - ・開発前との差分が明確である
- **プレゼンテーション力**
  - ・問題点，解決方法などが理解できる
  - ・動作デモができている

などの観点から評価されますが，本科目での審査項目については，毎年のテーマによって少しずつ変わりますので，別途配布するルーブリックで確認してください．他にも技術に特化したハッカソンでは，

- **新規性**（技術や組み合わせのオリジナリティがあるか）
- **技術性**（利用している技術は高度か）

- **発展性**（将来どの程度の波及効果が期待できるか）
- **再現性**（アイデア，プレゼンテーションだけではなく実際に動くものがあるか）
- **機能性**（使い勝手や性能が良いか）
- **テーマ性**（テーマがある場合，そのテーマを実現しているか）

といった観点での評価もあります．

この中で**新規性**については，毎年悩むチームが多いです．ハッカソンでは，技術面で研究のような意味での新規性を発案できるチームはほとんどいません．今までの常識を覆す斬新な発想や，独自の新技術を含むシステムを実装できるなら理想的ですが，現実には短期間で仕上げなければいけない性質上，そのクオリティのものはなかなかできません．大抵の成果物は似たようなものを「見たことがある」「聞いたことがある」「考えたことがある」ものになってしまっています．それが悪いということではなく，そうであれば既存の似たような製品やシステムがあっても，それらと何が違うのかの工夫を考えて，それをアピールできることが重要です．例えば，誰かが不満を感じているところを解決するアイデアであったり，不特定多数ではなく特定の人に重宝されるシステムといった観点で考えてみるのも一つの方法でしょう．研究のような新規性（Novelty）はなくとも，**独創性（Originality）を盛り込んで**ください．

> ! 研究論文における**新規性**とは，その内容が公知・既発表でなく，また既知のことから容易には導き得ない新しいことですが，「アプローチ」，「事象やトピック」，「理論」，「方法」，「データ」，「結果」のいずれかに新しさがあるなら，その部分も新規性とみなされます．

2.5.1　ハッカソン 1 と卒業研究の違い

本学の情報工学科では，ICT 応用技術の修得を教育の目標としています．そのために，皆さんは情報工学の基礎となる理論や技術を学んできました．やがて始められる卒業研究では，皆さんはそれらの知識が実際にどのように活用されるかを理解し，それらをさらに発展させるための探求をすることになります．そのためにはソ

フトウェア技術およびハードウエア技術を実地に修得しておくことが望まれます．ハッカソン１には，それらの経験を積むことも目的に含まれます．ただし，**ハッカソン１で行う内容は，大抵の場合，研究とは認識されない**ことを知っておくべきです．誤解を避けるために，ここでは**システム開発と研究の違い**について簡単に触れておきます．

工学の分野で研究と呼ばれるものは，通常最後に論文として発表するか，特許の取得を目指します（そうしないとその技術が世間に認知されません）が，論文や特許では**新規性**が問われます．過去の技術から容易に推測できない，新しいアイデアや知見が提示されている必要があります．ここで，研究は一般に基礎研究，応用研究，開発研究に分類できます．**基礎研究**とは，特別な応用，用途を直接に考慮することなく，仮説や理論を形成するため若しくは現象や観察可能な事実に関して新しい知識を得るために行われる理論的又は実験的な探究を指します．一方，**応用研究**とは，基礎研究によって発見された知識等を利用して，特定の目標を定めて実用化の可能性を確かめる研究，および既に実用化されている方法に関して，新たな応用方法を探究することをいいます．基礎研究が知識の枠組みを広げることを目的とするのに対し，応用研究は個別の問題に対する解決策を明らかにすることを目的としています．これらはどちらも新規性が重視されます．これらに対して，**開発研究**とは，基礎研究，応用研究，および実際の経験から得た知識の利用であり，新しい材料，装置，製品，システム，工程等の導入または既存のこれらのものの改良を狙いとする研究開発のことを言います．

例として，今，自動車の衝突防止のための検知技術を考えます．自動車が前方の車両や歩行者を検知する方法は，大きく 3 種類に分類できます．

1. カメラ（単眼・複眼）での画像による検出
   画像認識アルゴリズムにより，人や車両を判別する手法であり，夜間や悪天候時の検知が苦手．
2. ミリ波（電波）レーダーによる検出
   ミリ波帯の電波を発し，対象物に反射して戻ってくるまでの時間を計測するこ

とで物体までの距離や方向を検出する手法であり，遠距離や夜間でも検知可能であるが，人の検知が苦手であり価格も高い．

3. レーザーレーダーによる検出

電波ではなく，レーザー光を照射し，物体に当たって跳ね返ってくるまでの時間を計測することで，物体までの距離や方向を検出する手法であり，価格が安く夜間でも検知可能であるが，人の検知や遠距離は検知できない．

これらの手法に対して，基礎研究と呼べるのは，例えば，新しい原理の画像認識アルゴリズムを考える，画像認識の精度を改善する手法を考える，複数の反射波があっても適切に分離可能なレーダーの送信方法を考える，等が該当します．これらは基本的な原理なので，応用先は自動車の検知技術に限りません．ですから，特許として出願可能ですし，研究論文として発表することができます．次に，応用研究に相当するのは，人や車両に特化した認識アルゴリズムを考える，夜間での認識精度向上手法を考える，車の材質や形状を考慮してアンテナ形状を工夫する，等が該当します．これらも原理的な工夫があれば特許として出願可能ですし，論文化が可能です．

一方，開発研究と呼ばれるのは，画像検出とミリ波レーダーを組み合わせて検知精度をあげる，特定のハードウェアに特化して検知手順を修正する，等が該当します．これらは，既存の要素技術（基礎研究や応用研究）の組み合わせや修正で実現されることも多く，製品化を考える場合，個別の要素技術が特許で権利化されている場合には，特許使用（ライセンス）料を払う必要があります．ただし，要素技術の組み合わせ以上の工夫や効果があるなら論文化はできる場合もあります．

以上が，研究と呼ばれるものの大まかな分類ですが，皆さんがハッカソン 1 で行うのは，システムの開発が主であり，上記で言えば開発研究に近い内容です．多くの場合，**十分に調べた既存技術を活用することでシステムを実現**できますので，新たな基礎原理を考案したりすることまでは要求しません（Novelty は要求しませんが，Originality は必要です）．一方，卒業研究では，上記の基礎研究，応用研究に相当することを行いますので新規性が必要です．その点がハッカソンと卒業研究が

大きく異なる部分であると理解してください.

報告書を書くときに文献として載せられる信頼のおけるソースを調べて活用しよう.
文献情報の記録も忘れずに！

第 3 章　報告書の書き方

3.1　報告書の意義

実験・調査や研究・開発などを行ったとき，我々はその内容を報告書にまとめるのが普通です．その目的は，どのようなことを実施して，その結果はどうなったのかということを共有し，今後のために活かすことです．せっかく新たな技術が生まれても知見が共有されなければ，その技術は活用されずに宝の持ち腐れとなります．また，ある方法を試して，その方法では上手くいかなかったという結果であっても，なぜ上手くいかないのかが客観的にまとめられていれば[1)]，同じことを目指す人たちが同じ轍を踏まずに済むことになります．したがって，結果の良し悪しに関わらず，後の技術者の参考となるよう報告書は書かれるべきです．このことは 4 年生になって執筆する卒業論文にも当てはまります．論文を書くということは世の中に有益な情報を残し，専門の分野に貢献するということなのです．

また，「文章を書く」ことは，自分が考えたことや思いつきを頭の外に出して見えるように（可視化）し，客観的に読み返すことで自分の思考を整理することにも役立ちます．皆さんは，1 年生の「技術文書作成」で技術文章を書くことの基礎を学び，「アジャイルワーク 1」でそれを実践しました．ハッカソンでは，さらに経験を積むことで，文章執筆の練度を高めていくことになります．

---

1)　当然のことながら，きちんと調べれば上手くいかないのは当たり前というレベルでは意味がありません．試した方法が合理的に考えて上手くいくと考えられるのに期待通りにならないのは何故なのか，その原因や要因の部分に新たな知見が含まれていることに意味があります．

3.2　報告書の内容

ハッカソン 1 で求められる報告書は，

(1) 与えられたテーマをどのように理解したのか
(2) どのようなことを目指してシステム開発をしたのか
(3) どのような結果を得たのか
(4) そして，最も大切なことはその結果を適切に評価・検証した上で，何故そのような結果が得られたのか，目的は達成できたのか

といった内容が読者に分かり易い文章で，要点を逃さず，簡潔明瞭[2)]に書かれている必要があります．

技術報告書の基本的な流れについては，「技術文書作成」で学んだと思いますが，ハッカソンの場合，テーマやチームによって内容が大きく変わるので一般化し難い側面があります．そこで，このようなことは盛り込むべきという項目を示します．

はじめに:

- テーマの背景
- システム開発の目的
- 採用した技術的工夫
- 得られた結果とそこから導き出せる知見

既存の方法や理論:

- 似たような（あるいは同じ）目的のシステムと目指したシステムの原理的・技術的な違い
- 開発したシステムの原理の採用理由や長所・短所など
- 盛り込んだ技術的工夫の新規性や独自性

2)　文章量は少なければ少ないほど読み手の負担を減らせます．ただし，それは必要なことが漏れなく書かれていることが前提です．多くの場合，学生の書く報告書は必要な情報が欠落しています．書く内容や項目を減らすのではなく，冗長な表現を省いて文章の長さを短くすることを心がけましょう．

システム概要:

- 開発したシステムの概要と原理や工夫点（ハードウェアやソフトウェアの観点）
- 使用したセンサや部品の特性，器具や装置などの説明
- チーム内での役割

評価方法・結果・考察:

- 開発したシステムをどのように評価するのか
- 評価の目的や方法の妥当性
- 評価で得られた結果（表やグラフ）
- 評価結果の理論との比較による検証・考察

おわりに:

- 報告書の総括
- （ある場合は）開発手法を発展させる際に有益となる未解明点や課題等

ただし，これらは（「はじめに」と「おわりに」を除き）節構成の見出しの例ではないことに注意してください．あくまで盛り込むべき内容を分類したものです．「システム概要」という節見出しより，「磁気センサを用いたドアの開閉判断システムの概要」とする方が，読者にとって分かりやすいのはどちらであるかは明らかでしょう．

これらの項目の中で「はじめに」は，実は最も書き方が難しい節であると言え，最後に書かれることも多いです．ここでは，「はじめに」をまとめるにあたり，どのように思考を整理をするのかということを考えます．例として，ハッカソンとして与えられたテーマが「ドアが自動ロックされてしまう部屋で，鍵を持たずに部屋を出るのを防止するためのシステム」であるとします．このとき，その実現にはさまざまな方法が考えられます．

- ドアの開放を検知したら「鍵を持ちましたか？」というアナウンスを流すだけのシステム
- 鍵を持たずに人が通過すると警告するシステム

- 個人使用の部屋であれば，鍵を特定の場所に置くことをルールとして，鍵が置かれたままドアの開放を検知したら警告するシステム＋鍵を使ってドアを開けたのに特定の場所に鍵が置かれたことが検知できないときに警告するシステム
- 鍵を身につけていないと警告するシステム

他にもまだまだ考えられますので，報告書の目的として「鍵を持たずに部屋を出るのを防ぐためのシステム」とするだけでは不十分です．大きな目的（要求テーマ）は全チーム同じですが，その実現のために使われる技術はチームごとに異なるはずですので，その部分に特化した目的や目標が必要です．上記の例では，要求テーマに対応する大テーマとして以下のシステムが考えられます．

- ドアの開放検知システムの実現（検知時間 0.5 秒以下）
- 人が通過するときに鍵の所持を判定するシステムの実現（検知精度 85%以上）
- 鍵の置き場での鍵の検知システムの実現（検知精度 95%以上）
- 鍵が人から離れると警告するシステムの実現（検知精度 90%以上，検知時間 2 秒以下）

これらのうち，どの目的（や目標）を採用するのかに応じて記述する内容を決めなければいけません．

> ! 目標値は具体的であるほどいいですが，その場合，その数値を選んだ合理的な根拠（適当に決めた数値でないこと）まで書かれている必要があります．

ただし，報告書の目的を書くには，これでもまだ不十分です．例えば，ドアの開放検知システムを採用するにしても，その手法にも複数の種類が考えられます（磁気センサ，赤外線遮断センサ，赤外線距離センサ，衝撃センサ等）．さらには，磁気センサを採用する場合でも，磁気センサにも複数の手法が存在します（コイル方式，リードスイッチ方式，磁気抵抗素子，ホール素子，SQUID 方式等）．このようにテーマを細分化していくと，実現しようとしているシステムでは，どの方式のどの部分に着目して，どのようなことを達成しようとするのかによって，書くべき内容が大きく異なります．これを整理すると，以下のようになります．

- **【要求テーマ】**「鍵を持たずに部屋を出るのを防ぐためのシステム」の実現
  - **[大テーマ]** ドアの開放を検知して「鍵を忘れていませんか」というアナウンスを流すシステム（採用の理由は？）
    - 〈中テーマ〉 磁気センサによるドアの開閉判断システム（採用の理由は？）
      - (小テーマ) リードスイッチ型磁気センサによるドアの開閉判断システム（採用の理由は？）
      - (小テーマ) コイル方式磁気センサによるドアの開閉判断システム
    - 〈中テーマ〉 赤外線遮断センサによるドアの開閉判断システム
  - **[大テーマ]** 鍵を持たずに人が通過すると警告するシステム
    - 〈中テーマ〉 ・・・

このような整理がされると，元々与えられた要求テーマは「鍵を持たずに部屋を出るのを防ぐためのシステム」であるのに対して，実際に制作するのは「リードスイッチ型磁気センサによるドアの開閉判断システム」ということになり，そこで何を実現したいのか（・センサ部分が無電源で動作可能なため省電力化を図る・SQUID 型センサ並みの好感度検出を目指す・小型化を目指す）という具体的な目的に落とし込めます．

例えば，省電力化を目指すのであれば，ドア以外への応用先も含めて省電力化を達成している方式を調べる必要がありますし，小型化もドアへの応用に限る必要はありません．また，目的に応じて報告書のタイトルも「リードスイッチ型磁気センサによる省電力なドア開閉判断システム」や「リードスイッチ型磁気センサを用いたドア開閉判断システムの小型化」といったタイトルになるでしょう．また，目標性能についても，センサ単体の性能に対して，システム全体でどれくらいの性能を目指すのか，という議論ができるようになります．このように報告書にまとめる際には，具体的な項目に落とし込んで整理することが重要です．

3.3　報告書の作成

ここでは一般的な注意事項を挙げておきます．

(1) 配布されるルーブリックおよびチェックシートを参照し，チェックシートの全ての項目にチェックが入れられるように報告書を作成すること．

(2) 報告書については，TEX のテンプレートを配布します．テンプレート内の設定は改変せず，そのまま使用すること．

> ! 近年，評価手順が箇条書きのみだったり，評価結果の図表を載せるのみで節を構成している報告書が散見されますが，**きちんと文章を書く**こと．

(3) 報告書の作成にあたり，グラフは軸に目盛と数値を入れ，数値が読み取れるようにすること．また，複数の線が書かれるときは，それぞれの線が何を表すのか（凡例）を忘れずに書くこと．図・表には通し番号をつけて標題をつける．このとき，図の番号・標題は図の下に，表の番号・標題は表の上に書く．

(4) 数式には番号をつける．また，数式や文章中で，変数は斜体，ベクトルや行列は太字の斜体を用いる．

> ! 例えば，$\cos x_1$ における cos は関数なので斜体にはしない．また，数字も変数ではないので斜体にはしない．同様に，しきい値電圧 v_{T} の T は，Threshold の T で変数ではないので，v_T とはしない．

(5) 測定などの使用機器がある場合は，製造会社名と型番号をメモすること．ただし，コンデンサや抵抗器などは定格値のみを記入すること．

(6) 評価によって得られた測定値等は，統計的に処理する必要があることを意識すること．

(7) 評価結果に対する検討は，**皆さんが自身の言葉で工夫してまとめる部分**であり，この種の報告書の中心になります．労を惜しまずによく考えて，分かりやすい表現で簡潔に，しかし粗略にならぬように書くことを心がけること．

> ! 測定または観測のデータのみを書いて，そのデータから何を知ることができたか，問題点はなかったか，などの検討がなされていない報告書が多くみられます．これでは報告書としては落第です．また，期待通りの結果が得られなかった場合でも，誤差とか測定ミスといった安易な言葉で片付けるのではなく，なぜそうなったのかの理由を根拠のある言葉できちんと考察すること．

(8) 文献調査は必須であるので，参考文献を必ず挙げること．

(9) 再提出の場合には，指摘された部分のみならず，全体を見直して修正すること．

(10) 報告書の初回提出期限は 12 週目の 9:00，再提出の場合の提出期限は返却された翌週の水曜日 9:00 とします（提出日が祝日の場合は別途指示します）．

3.3.1　参考文献の書き方

参考文献は [1] [2] のように番号をつけて最後にリストし，全ての文献を本文中で参照すること．本文中での文献の参照は文献番号で行う．また，Web ページは改版や消滅の可能性があるので，URL を参照することはできるだけ避けること（ブログ等，誰が書いたのか特定できないサイトは，情報の信頼性が保証されないので不適切ですらあります）．ただし，標準化団体やメーカーなどが文書公表の場を Web ページにしている場合などはこの限りではありません．文献のリストの仕方は下記の例に従うこと．

[1]著者名（複数名の場合は全員），書名，出版社，発行年月．
[2]著者名，“標題，”書名，pp.21–31（ページ番号），出版社，発行年月．
[3]著者名，“Web ページタイトル，”サイト（管理者）名等，URL，文書年月または参照年月日．

なお，参考文献とは議論の裏付けや，読者に内容を更に深く理解してもらうために挙げるものです．自分が報告書を執筆する際に参考にしたものであっても構いませんが，単に文章を参考にした（写した）文献を挙げることではないことを理解してください．文献の文章や図を無断で使うなどはもっての外です（ただし，文章の

明示的な引用は可)．このことは，**本テキストについても同様**です．

3.3.2　報告書提出に関する注意

- 報告書は，提出日の指定時刻までに manaba の指定の場所に提出しなければなりません．提出期限に未提出あるいは未完成と認められるものが提出された場合，**不合格**として処理するので，注意してください．
- 成果報告書が受理された場合，manaba を通じてその旨が通知されます．不受理の場合は報告書の全てを見直し，指摘された点に特に注意して報告書を修正し，指定の期日までに再提出してください．
- 報告書は，1 度だけ修正の機会が与えられますが，2 回目の提出時には内容が不十分であっても受理され，その報告書の内容をもって成績判定に利用されます．また，再提出にあたり，内容の改善がほとんど認められない場合には，**不合格**として処理します．

第II部　資料編

第 4 章　回路を組むときの Tips

4.1　外部回路をつかうときの注意

皆さんは，1 年生のときに「電気回路」「論理回路」を学び，Arduino の入力や出力の信号を扱う方法の基礎を学びました．また，「フィジカルコンピューティング」でセンサ等の概要を学び，「アジャイルワーク 1」でそれらを一部実践しました．「ハッカソン」では，制作するシステムがチームごとに変わるため，必要な知識やノウハウがチームによって変わりますが，ここでは電子回路を組むときの一般的な Tips をまとめます．トランジスタ等のアナログ素子については，昨年のアジャイルワーク 1 の資料を参照してください．

4.2　チャタリング

Arduino の制御などのために機械式スイッチを使うことがあります．スイッチには様々な種類がありますが，基本的には信号のオン・オフを操作したり，信号を切り替えたりする動作をします．このとき，スイッチに機械的な機構が含まれる場合には，チャタリングに気をつける必要があります．

今，スイッチの操作により，図 4.1 のように電圧のオン・オフを切り替えることを考えます．このような切り替えは図 4.2 のようなスイッチで簡単に実現できますが，この方法には問題があります．実際にスイッチをオン・オフした場合に，スイッチの接点は一度で安定せず，接点で跳ねることで図 4.2 のようにオン・オフを何度も繰り返した後に安定するのが普通です．この現象を**チャタリング**といいます．

デジタル回路はパルスの立ち上がりや立ち下がりで動作するものがほとんどなの

図 4.1　電圧のオン・オフの切り替え

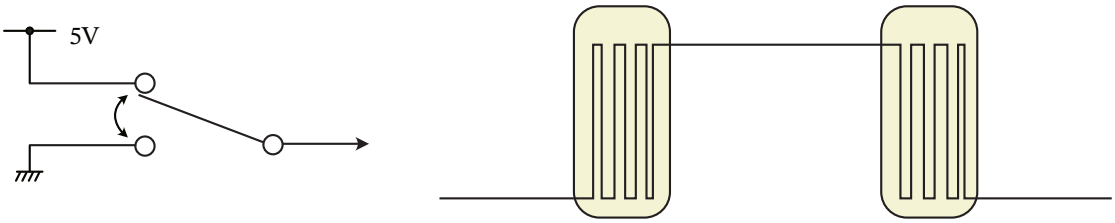

図 4.2　スイッチによるオン・オフの切り替え

で，チャタリングにより，不要な切り替わりを発生させてしまう場合があります．そこで，チャタリングを防止する回路がいくつも考案されています．その中の代表的な方法として，図 4.3 にフリップフロップを用いた方法を示します．この回路でどのようにチャタリングを防止できるかは各自で考えてください．また，Arduino のプログラミングでタイミングを制御することにより，チャタリングの影響を避ける方法もよく用いられます．

! デジタルオシロスコープによるハザードの観測を考えます．ナイキストの標本化定理によると，高周波成分を正確に再現し，エイリアシング（アンダーサンプリング）を防ぐためには，計測する信号の最も高い周波数成分の少なくとも 2 倍の周波数でサンプリングする必要があります．しかし、ナイキスト周波数は絶対最小であり，正弦波のみに適用され，連続信号を想定しています．ハザードは連続して起きないため，最高周波数成分の 2 倍のサンプリングレートでは事実上不十分です．
そこで，デジタルオシロスコープでは **5 倍ルール**と言って，観測周波数の 5 倍のサンプリングレートを採用することが望ましいとされています．このルールに従うと振幅誤差を 2%程度に抑えることができます．また，5 次高調波程度まで再生するなら 10 倍以上が必要になります．オシロスコープの周波数帯域が十分にないと，高周波成分の変化を表示できません．波形は歪み，立上りは鈍り，信号の詳細は失われます．

4.3　ハザード

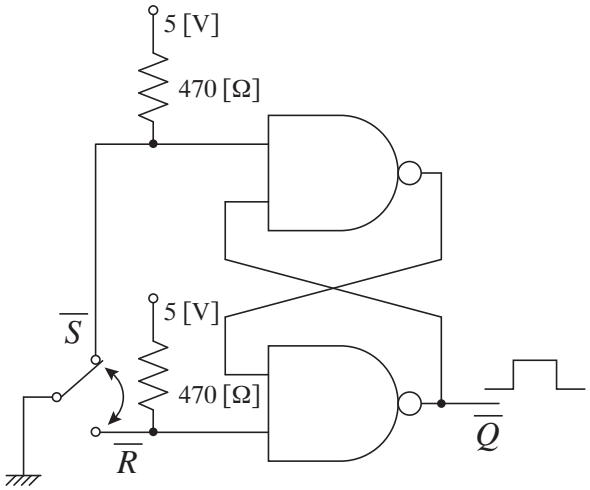

図 4.3　チャタリング防止回路

デジタル回路を確実に動作させる条件は 2 つあります．1 つは適切な信号の電圧レベルであり，もう 1 つは適切なタイミングです．しかるべき電圧の信号がしかるべき時間に入りさえすれば，回路は確実に動作します．とこで，一般に IC では入力が加わってから出力に反映されるまでの間にわずかな遅延を生じます．この遅れ時間は使用する IC や負荷，配線の状況によっても変化しますが，普通の TTL-IC では数 ns〜100 ns 程度です．使用する IC の規格表を調べると，**伝搬遅延**（t_{pd}: Propagation Delay Time）の項目に代表値が出ています．例えば，1 MHz のクロック信号を考えると 1 周期は 1 μs であるため，数 ns くらいの遅れが大きな問題を起こすとは考え難いです．取り扱う信号のパルス幅や周期が数 ms もあれば，素子間で伝搬遅延が数十 ns くらいあっても何の問題もないように思えます．

しかし，実際には遅延を無視して回路を設計した場合，出るはずのないところに細かい髭状の信号が現れて誤動作の原因となることが珍しくありません．このような髭状の信号が発生する現象を**ハザード**（hazard）[1] と呼びます．ハザードの発生は不確実で，使用する素子や時間，温度によって出たり出なかったりするためトラブルとなりやすいです．ハザードの対策法は各種ありますが，各自で調べてみてください．

1）グリッチ（glitch），スパイク（spike），ヒゲ（髭）などと呼ばれることもあります．

4.4　プルアップ抵抗

デジタル回路の信号の電圧レベルは，“H"または “L"の状態でなければなりません．“H"と “L"の中間の電圧状態では，内部状態が不安定になり誤動作を起こす場合があります．また，デジタル回路の入力端子は，内部インピーダンスが高いため，入力端子がどこにも接続されていない場合[2)]，周囲の静電気や電磁誘導による電流の侵入によって，予期せぬ電圧が印加されることもあります．その対策として，電源電圧または接地電圧と，入力端子との間に，**数 kΩ から数百 kΩ 程度**の値をもつ抵抗器を挿入します．電源電圧側に抵抗器を挿入することをプルアップと呼び，入力が無い場合に “H"レベルの電圧をかける働きをします．一方，接地電圧側（GND）に抵抗器を挿入することをプルダウンと呼び，入力が無い場合に “L"レベルにしておくことができます．

4.5　ファンアウト

デジタル回路では，1 つの IC の出力信号が複数の後段 IC へ分岐して入力することが多くあります．このとき，出力端子に繋がれた入力端子の数のことを**ファンアウト**（Fanout）と呼びます．デジタル回路では電圧レベルにより情報を伝達しますが，このときには当然電流の流れも伴います．具体的には，出力のレベルが “L"になると，出力端子に後段の回路から電流が流れ込み，出力が “H"になると，出力端子から後段の回路に電流が流れ出ます．TTL-IC のように入力端子に流れ込んだり入力端子から流れ出る電流が比較的大きな IC の場合，前段の出力側 IC の電流駆動能力に応じて，**ファンアウトの数が制限されます**．これを超える数の IC を接続した場合，回路が正常に動作しないことがあるので注意が必要です．この対策としてバッファを挿入して駆動能力を上げる方法がよく取られます．バッファは波形整形のために用いることもできます．ただし，バッファでも遅延が起こるので，ハ

2)　「浮いている状態」と表現されたりします．

ザードの原因とならないよう注意が必要です[3)]．

4.6　バイパスコンデンサ

バイパスコンデンサとは，電源や信号線と GND の間に設置し，ノイズを迂回（バイパス）して逃がす役目をもつコンデンサのことです．特定の種類のコンデンサを意味するのではなく，ノイズをバイパスするために使われるコンデンサの総称です．ノイズを逃がすことで，電子回路への安定した電源供給が実現できます．バイパスコンデンサがないと，電子回路によっては正しく動作できない場合があるほどに重要な部品です．バイパスコンデンサは略してパスコンとも呼ばれます．また，直流と交流両方の成分からノイズなどの交流成分をバイパスして逃がすことから，デカップリングコンデンサとも呼ばれます．

バイパスコンデンサは高周波ノイズ対策なのか，電源電圧の安定化を目指すのかによって必要な容量が異なります．高周波ノイズ対策を目的とする場合は数 100 pF〜0.01μF 程度，電源電圧の安定化を目的とする場合は数 10μF〜数 100μF の容量が使われます．

なお，高周波ノイズを取り除く場合には，バイパスコンデンサをできるだけ IC の電源ピンに近い位置に配線します．IC とコンデンサが離れている場合，「インダクタンス成分」が大きくなるため，ノイズ除去効果が十分に得られません．また，バイパスコンデンサを IC の近くに置くことで，ノイズが流れるループが短くなります．これにより放射ノイズの発生を抑えられる効果もあります．バイパスコンデンサを配置する場合は，必ず IC の電源ピンに近い位置に配線しましょう．

4.7　電位・電圧・電圧降下

3)　逆に遅延を揃えるためにバッファを使うこともあります．

これらの用語の違いについて，正しい理解をしないまま報告書を書いている人が非常に多いです．そこで，これらの用語についておさらいしておきます．ただし，以下は直流回路に限定した話となります．

まず，基本となるのは**電位**ですが[4]，これは回路中のある地点が，基準となる電位（0V）の位置から何ボルト高い位置にあるかを表し，単位はボルト（V）です．このとき，基準となる位置をグラウンド（GND）と呼びます[5]．また，電位は静電ポテンシャルとも呼ばれ，ある地点にどれだけ電荷があるのかを表しています．

次に，**電位差**とは，回路中の異なる 2 点間の電位の差のことです．こちらも単位はボルト（V）です．一般には，この電位差を電圧と呼びます．回路中の 2 点間に電位差があると，正の電荷が電位の高いところから低いところに移動します[6]．この電荷が移動する流れのことを電流と呼び，単位はアンペア（A）を用います（1A は 1 秒間に移動する電荷の量が 1 クーロンであることを意味します）．

すなわち，電位とは回路のある 1 点と GND との電位差であり，電圧とは任意の異なる 2 地点間の電位差（もちろん GND との電位差でも構わない）です．どちらも電位差ですが基準となる点が異なることに注意してください．したがって，回路中のある 1 点を指して電圧と言うのはおかしいですし，回路中の 2 点を特定することなく電圧（電位差）と言うこともできません．報告書に「電圧が流れる」などと書く人が毎年若干名いますが，これは用語を全く理解できていない人の表現です．

一方，回路に電源を繋ぎ電流が流れると，導線や素子の内部抵抗により，電位が下がっていく（熱エネルギーに変換されて消費される）現象を電圧降下と呼びます．実験で使う数 cm 程度のジャンプワイヤではまったく問題ない（計測不能なレ

4） 概念として，電位とは物理（力学）における位置エネルギーに相当します（英語ではどちらも potential）．

5） 一般的には地面が 0V であるため，そう呼ばれます．ちなみに，電子レンジやエアコン等の電気機器では，接地（アース）が必要ですが，これには漏電等の発生時に大電流が流れて機器が破損したり，人が感電したりすることを防ぐために地面に電流を逃がす役目があります．ただし，小電力な機器では金属製の筐体にアースをとる場合も多いです．

6） 実際には，負の電荷をもつ電子が GND から電位の高い方向に移動するため，電流の流れる方向と電子の移動方向は逆となります．これは電流の実体が判明していなかった時代に電流の向きを決めたが，後に電子の流れが発見されたことによります．

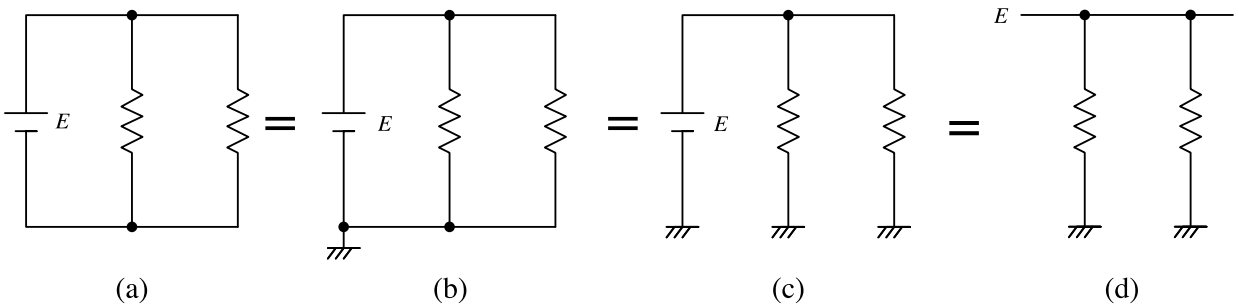

図 4.4　すべて同じ回路

ベル）ですが，導線の長さが数 10m 以上になると導線の太さによっては導線自体が一種の抵抗になり影響がでます．また，ダイオード，トランジスタ，IC などの半導体でも電圧降下は生じます．

半導体とは，電流を良く通す「導体」と電流をほとんど通さない「絶縁体」との中間の性質をもち，特定の条件のときに電流を通すシリコンなどの物質や材料のことです．ダイオードは P 型半導体と N 型半導体を接合した構造をもちますが，P 型半導体側から N 型半導体方向に（順方向）電圧を印加する（電位差を与える）ことで電流が流れます．このとき，電圧に関係なく電流が流れるのではなく，実際にはある閾値以上の印加が必要となります．これがダイオードにおける電圧降下です．すなわち，ダイオードは閾値以下の電圧では抵抗値が非常に高く，閾値以上の電圧では抵抗値が非常に低くなる素子と考えることができ，このような素子は非線形素子と呼ばれます．

4.8　回路図の基本

回路を視覚的に理解するためには，回路図を用いますが，図 4.4 に示す回路図はすべて同じ回路を表しています．(a) は広く一般的に用いられる描き方で，(b) は GND を明示的に示した描き方です．(c) は (b) と同様に GND を使って描いています．(d) は電源を省略した描き方です．また，回路図の描き方の基本は，電流が左から右，あるいは上から下に流れるように描きます（当然，複雑な回路では，そうならない線も出てきます）．また，基準電位を表す GND の記号には，図 4.5 に示

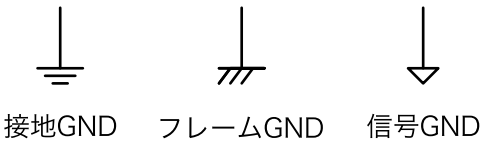

図 4.5　グラウンド（GND）の記号

す 3 種類が主に用いられます．接地 GND は，本来のグラウンド（地面）でアースとも呼び，感電防止などのために実際に地面と接続する場合の記号です．フレーム GND は，電子機器の筐体（フレーム）に接続されるもので，信号 GND は回路基板の内部における基準電位に接続することを表します．日本ではフレーム GND を用いることが多いです．