

AIダメ!絶対! vs AI推せ!絶対!

ギュられる時代の“AI教育どうやっていきますか?”

中西 恒夫 及川 達裕 vs 山崎 進 久住 憲嗣

「ギユられる」

シンギュラリティにより新卒採用が絞られること

SWEST2026, 2026年8月28日

生成AIのせいで 学校がいろいろ減茶苦茶になっている件

中西 恒夫

福岡大学工学部電子情報工学科
(対外公開用に修正, 2026/09/17)

- 福岡市城南区＝福岡空港, 博多駅から地下鉄で20分ほど
- 九州最大の私立大学, 教員と学生をあわせて2万人ほど
- 学部は人文・法・経・商・商(二), 医・薬・理・工・スポーツ科学
- 工学部は機械, 電気, 電子情報, 土木, 建築, 化学
- 典型的な都市型マンモス中堅私学
- 素直で明るくコミュ力の高い楽天的な学生が多い。
 - あまり考えずに大学, 学科を選んで入学してくる学生は多い。
 - 入学時に必ずしもコンピュータが好きだった訳ではない。(コンピュータを全然知らない学生も。)
 - それでも大多数の学生は電子系コースよりも情報系コースに行きたいと思っている。
 - SEになりたい, ゲーム開発をしたいと漠然と思っている学生は多い。

■ プログラミング演習II(2年生)

- C言語の演習
- 図書館システムを作る。
- 実働は助教の先生たち
- 評価: 課題提出 & 口頭試問

ここが問題

■ 情報処理システム開発(3年生)

- ソフトウェア工学
- ソフトウェア開発プロセス(OOA, OOD)=V字の左側
- UML
- 評価: 小テスト&期末試験

ここも問題だった

■ オブジェクト指向プログラミング(3年生)

- Java
- 評価: 小テスト&課題提出&期末試験→小テスト&期末試験→期末試験

■ プロジェクト型ソフトウェア開発演習(3年生)

- Javaで書かれたなんちゃってPOSシステムをチームでプロセスを踏んで派生開発する演習
- OOA/OOD/OOP, UML, SQL, ...

■ なお,工学部は全学科, BYODにしている。

■ コロナ前（～2019）

- コードが書ける学生の作品をコピペすることが横行
- これはもう昔から（講演者が学生のときから）

■ コロナ中（2020～2022）

- プログラミング演習支援システム（FuPad）を開発, 導入, 科目をまたいでそれが定着
- コピペを簡単にさせない仕組み
- コピペが露骨に「見える化」されるようになった。

■ 生成AI登場後（2023～）

- 生成AIに吐かせたコードを提出する例が続出するようになった。
- 生成AIを禁止したり, 制限をかけて利用を容認したり, 見て見ぬふりしたり。
- 自分の「書いた」コードが説明できない。
- 小テストでの不正横行（2025）
- 2024あたりから全然コードが書けない学生が研究室に配属されるようになってきた。

画像認識のプログラムは「書ける」のに, LEDチカが書けない。

（生成AIにLEDのつながったポートの番号を与えていないから。そういう知識が身についていない。）

- オブジェクト指向プログラミング (Java) の授業
- 評価は10回の小テスト40%, 期末試験60%
- 小テストは Google Forms で実施
 - コードの穴埋め問題
 - 提示したJavaのコードの実行結果を答えさせる問題
- 生成AI不正露見のきっかけとなった小テスト回答

はい、承知いたしました。以下に、プログラム実行後、8を入力した際の出力をコピー＆ペーストできるように記載します。

正の整数を入力してください。> 8

8 / 5 = 1

8 / 4 = 2

8 / 3 = 2

8 / 2 = 4

8 / 1 = 8

ゼロで除算されました!

終了!

本当に終了!

- 他の答案も精査すると、どうも少なからぬ受講生が生成AI不正をしたものと思われる。
 - 教えていないしプロも覚えていないような例外処理クラスが使われていたり、
 - 出題パターンを変えていないのに、小テストと期末試験の点数の大きな乖離していたり、
 - 小テストの回答時間がやたら短く、点数がやたら高かったり、
 - 授業についてこれなくなっていて不正に手を染めている学生も多い。

- ワクワクする課題を用意して**動機づけ**したら自ず学生はプログラミングをちゃんと学ぶはず。←ホントか？
- だいたい昔からプログラミング演習ではコピペが横行していた。
- プログラミング演習は昔から動機づけ & コピペへの**諦め**／**懲罰**でやってきた。
- 生成AIで一気にめちゃくちゃ。

- 課題提出での加点も小テストでの加点もどちらもやめた。
 - 課題提出で点数をつけるから不正をする。
- 評価は期末試験に全振り。
- 講義のやり方
 - プログラミング演習なのにはほぼ板書する。(たとえばオブジェクト参照の図を描く, 一緒に描かす。)
 - 教えた概念を理解してもらうために小演習をする。
- 結果
 - 過去四半世紀の間に講演者が担当した授業で最高の授業態度。
 - 居眠りをする者はほとんどなし, ゲームをする者は皆無。
 - 学生がノートやメモをしっかりとるようになった。
 - 講義中に「そうだったのか!」という納得した表情を示す学生は増えた。
 - やる気のない学生は最初のほうから授業に出なくなった。
 - 成績は例年よりよい。
- しばらくこれでいきます!
 - とはいえ, 落単の恐怖で学生を縛るのは本当は避けたい。重要性と面白さを理解してほしい。

- 結局,ソフトウェアは国語力。(上流工程は特にそう思う。)
- 検索エンジンと生成AIと動画サイトとでどこまでも劣化してきた「国語力」と「演繹脳」。
 - こう決めた,こう書いた,こういう原理で動いている,だからきっとこうなるはず・・・と推測する力の劣化。
 - 答えを教えてもらうのに慣れきってしまった。(教えないと口を尖らせる,怒る,辞める。)
 - 質問はいつも「これで合ってるでしょうか？」
 - タイパ重視で試行錯誤をしない,失敗から学ばない。(失敗からの学びは定着がよいのだが。)
- 基礎学=国語,数学,三力学(力学,熱力学,電磁気学; 今後は量子力学も?)を徹底的にやってほしい。
 - 数学の証明問題を特にやってほしい。
- 残念な子が使えば,スマホ,生成AIは阿片。ひとりで立てなくなります。
 - 阿片を吸引すると一人で立てなくなるそうです。
 - 生成AIを使いこなす学生と,生成AIに依存する学生の格差。
- 日本人の学びをコントロールするのは,入試と入社と期末試験。
 - これはちょっと悲しい現実。

- 生成AIのリスクを学生がかなり理解するようになったかも。
 - 来年度の新卒採用を2割だの5割だの減らす会社が増えている話をするとう真剣に聞く。
 - どうすれば生成AIに仕事を奪われないかの話も真剣に聞く。(たぶん学生も相当不安に思っている。)
 - 開発演習の位置付け, 重要性を以前よりも聞くようになった。
- 以前からの「大学の勉強をちゃんとやることが最高の就活」という表現が伝わるようになってきたかも。
 - 高校出た当時の君でも書けるプロンプトを書き続けて, それっぽいコードが出るまで「ガチャ」をやっているようだったらまずい。
 - 大学で勉強した専門知識を背景に生成AIを使役できるようになっていればとりあえず大丈夫!

大学教育におけるLLM活用

学年に応じた 「解禁の度合い」の設計

久住 憲嗣

芝浦工業大学 システム理工学部 情報課程

禁止か全面解禁か、ではない

段階的に開く

2年は限定的に、3年はレビューとエージェント、3・4年は研究の全工程へ。

古典も併走

検索、設計、JUnit による自動テスト。手で書いて確かめる工程は残す。

過程を見る

成果物だけでなく、対話ログを提出対象にして使い方そのものを評価する。

学年別の対比

	2年 プログラミング言語	3年 モデリング／情報実験Ⅰ	3・4年 セミナー／卒業研究
使い方	限定的に。エラーの手がかり、書いたコードの改善点を聞く	設計・テスト設計のレビュー。実装ではエージェントを使う	サーベイから実装、論文執筆まで積極的に
道具	Webチャットのみ。コード補完はオフ	Cline（GitHub Codespaces 同梱）	DeepSearch、コーディングエージェント、レビュースキル
提出物	コードとレポート	コード、テスト、レポート＋対話ログ	研究計画、成果発表、論文

2年

限定的な利用

使ってよい

コンパイルエラー・実行時エラーの手がかりを尋ねる

自分が書いたコードの改善点を指摘してもらう

理解を深めるための対話的な辞書、壁打ち相手として

使わない

LLM によるコード生成

Copilot・IDE の AI 補完（オフにする）

古典的なやり方も並走。検索演算子、情報源の見極め、エラーメッセージの読解。

3年

レビュー役から、実装の相棒へ

モデリングの講義

主にレビューで使う

ユースケース記述、クラス図、テストケースを自分で書いたうえで、LLMに見せて指摘をもらう。テスト設計のたたき台にも使う。

情報実験 I (全4回)

コーディングエージェントを導入

設計 → コアの実装と JUnit → Web 化。コアを自分で作りきったあと、Cline に「皮」を実装させる。

対話ログも提出。コード・テスト・レポートに加えて、エージェントとのやりとりを提出対象にしている。

3・4年

サーベイから論文執筆まで

サーベイ

DeepSearch を2サイクル

1回目で分野を俯瞰し、2回目で深掘り。会議を指定して検索し、論文が実在するか必ず確認する。

精読・研究計画

対話相手として

重要論文は3パス・リーディングと落合フォーマットで自分で読む。RQ の検討では sparring partner に。

執筆

レビュースキルを配布

論文の体裁やロジックのつながりなど、指導でいつも指摘している項目をまとめた Claude のスキル。

指摘の観点を渡してしまえば、学生は自分で何周も回せる。教員のレビューは、その先の議論に使える。

まとめ

解禁の度合いの設計と、 過程の評価

2年は限定的に、3年はレビューとエージェント、3・4年は研究の全工程へ。

設計・自動テスト・精読といった古典的なやり方は残す。

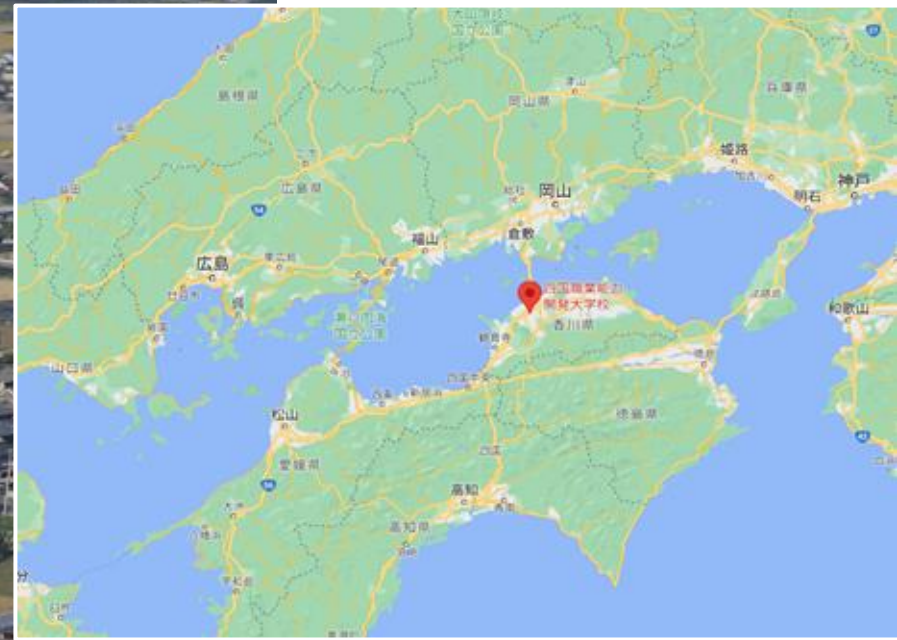
対話ログを提出対象にして、成果物ではなく使い方を見る。

AIダメ絶対派の ポジショントーク

近畿職業能力開発大学校

及川 達裕

四国職業能力開発大学校



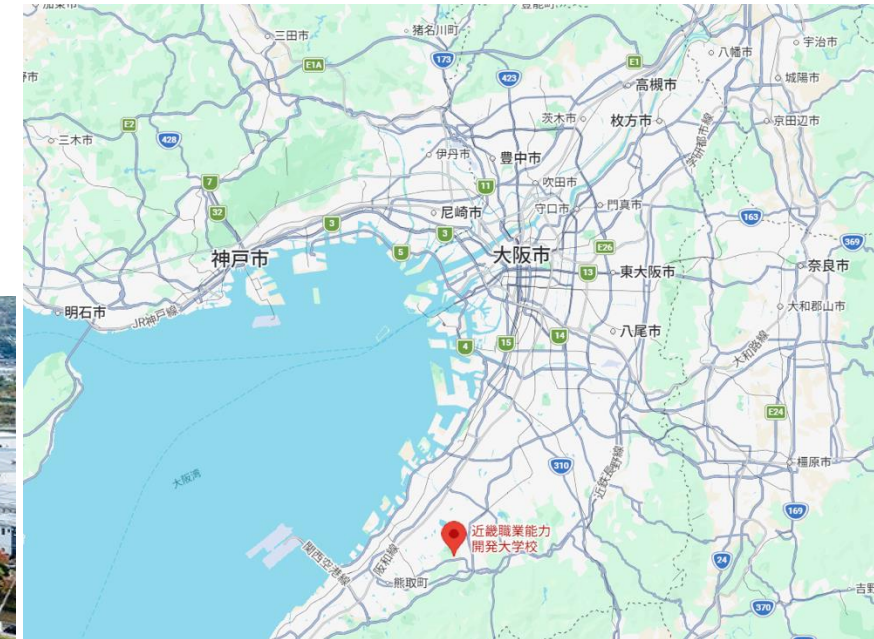
厚生労働省所管の省庁大学校
全国に10校，附属短期大学校13校
→ 高知に附属短期大学校

- 香川県丸亀市
- 学生数 約280名
- ユニークな教育システム
2年制+2年制
- 短期大学校から40年超
(1981年 開校)→約2000人超の卒業生
- 機械系、電子情報系、電気系、
建築系の専攻科

うどん県にある
工科系の大学校です



近畿職業能力開発大学校



厚生労働省所管の省庁大学校
全国に10校，附属短期大学校13校
→ 京都，滋賀に附属短期大学校

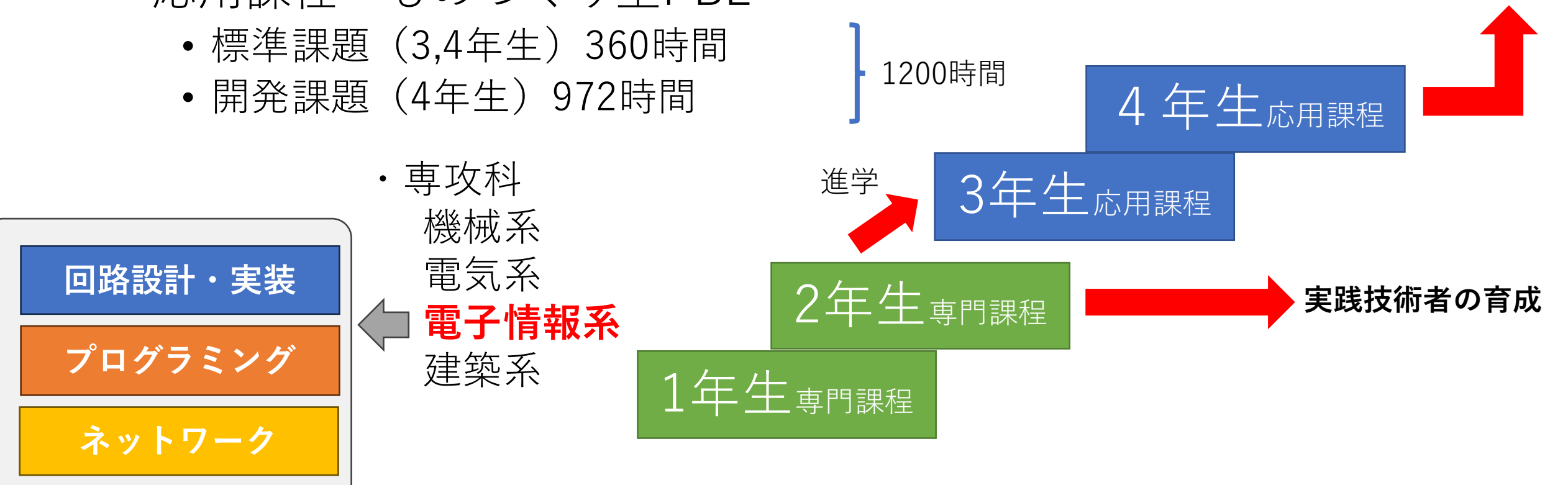
- 大阪府岸和田市 約330名
- ユニークな教育システム
2年制+2年制
- 機械系、電子情報系、電気系、
建築系の専攻科

たこやき府にある
工科系の大学校です



能開大の教育システム(厚労省所管)

- 1 時限 100分
- 4 年間 5600時間（実習3600時間）
- 応用課程 ものづくり型PBL
 - 標準課題（3,4年生）360時間
 - 開発課題（4年生）972時間



AIダメ絶対派な理由

- 初学者向けの教育において
 - 0→1のところに制約なく／使い方を適切に指導なくAIを活用すると、思考することなく何となく「プログラムが動いた！おっけー！」な学生が量産される点
- 0→1を超えた先である4年次の卒業研究等では、AI活用もありだと思える
- よって、初学者向けには生成AIの利用は否定的

AI教育のスタンス

- 初学者に生成AIのアシストは不要
- 特に基礎的なプログラミング
- 章立てごとに**動機付け**→**提示**→**適用**→**評価**の流れで授業を展開
 - **適用は 課題提出** を提示
 - **評価は 動作確認 と コード説明させるor口頭試問**
- 禁止しても隠れて結局使うので、コード説明させるor口頭試問で縛る
- コード説明or口頭試問は一對一の個別指導
 - とっても時間が溶ける

AI教育のスタンス

- 4年次の卒業研究（当校の場合は卒業開発）では、AI活用もアリのスタンス
- どのタイミングからAIダメからAI推せにするか悩ましい
- 加えて、学科内の他の教員とのスタンスの違いもある

AI推せ!絶対!

ギュられる時代の“AI教育どうやっていきますか?”

SWEST実行委員 & 北九州市立大学 山崎 進

ソフトウェア設計論

学部3年生

- ・ 達人プログラマー（第2版）
 - ・ 熟達に向けたあなたの旅
 - ・ 著者
 - ・ David Thomas 著
 - ・ Andrew Hunt 著
 - ・ 村上 雅章 訳
- ✓ 実務の例が学生にはわかりにくいので
学生のプログラミング体験に寄り添い解説



AI時代における高等教育のあり方についての一考察

<https://zacky1972.github.io/blog/2023/04/10/learning-with-ai.html>

レポートの方針

- 本授業において、レポート作成にChatGPT等AIを使用して良いものとする。本授業以外の授業については、各担当教員の指示に従うこと。
- 設問をそのまま問うただけのAIの使用は認めない。AIを使用した場合には、プロンプト(AIに与えた指示)を添付すること。添付ファイルをAIに与えた場合には、レポートの参考文献として添付ファイルの出典を明記し、レポート中で適切に引用すること。また、AIが生成した文書を丸写ししただけのレポートは認めず、筆者が責任を持って執筆・編集することを求める。講義動画をAIに入力として与えることは認めない。AIでプロンプトを生成させた場合には、まず、生成の大元となるプロンプトの添付が必須である。また、生成して得たプロンプトは必要に応じて適切な方法で引用すること。
- 文体を「だである体」に統一すること。引用部分を除き「ですます体」が混在する文体で提出した場合には、評価の対象としない。
- 教科書・参考書を適切に引用することを求める。また、引用することを指示されている場合で、適切な方法で引用がなかった場合には、評価の対象としない。
- AIの利用を禁止している設問でAIの利用が疑われる場合には、成績評価の対象にしない。また、AIの利用が疑われる場合でプロンプトを提出していない場合も、成績評価の対象にしない。

AIを使ってレポート書いてくる学生が「自分の疑問を明確化して、それに関してAIに問う学生」と「レポート課題をそのままAIに問う学生」に分かれており、ここに差が現れるのかと思った話

<https://togetter.com/li/2526788>

AI時代の大学教員は、何を教える人になるのか
https://note.com/tarolabo_0110/n/n5524ce4195aa



【任意レポート課題】 わからない点をAIで解消しよう(第2回)

第2回の授業内容について、疑問に思ったことはありませんか？ ChatGPT等AIに尋ねる質問をたくさん考えてみましょう。さらなる疑問が湧いたら、どんどん質問していきましょう。

次のことをまとめてレポートにしてください。

1. どのような質問をしたか、全ての質問をレポートにまとめてください。
2. AIの回答から、どのようなことがわかりましたか？ 要点をまとめてレポートしてください。
3. AIの回答で腑に落ちない点や矛盾した点はありませんでしたか？ 気づいた点をレポートにまとめてください。
4. AIの回答について、人間の先生に質問するのに比べて、どのような傾向や特性があると思いましたか？ 気づいた点をレポートにまとめてください。

なお、1はAIを用いて良いですが、2,3,4はAIに作文させることを禁じます。



【必須レポート課題】 今までのプログラミング経験をふりかえろう(第1回)

1. 今まで授業やサークル活動等でしてきたプログラミング経験のうち1つをとりあげて、どのような体験だったかを説明してください。要点を与えてAIに詳細を説明させるというスタイルで構いません。この場合、プロンプトは提出しなくて良いです。また、出力させた結果をそのまま添付するというのでよく、だである体への統一等もしなくて良いです。ただし、出力結果の妥当性は確認してください。
2. その体験のうち、本日の授業で習ったことに当てはまる事象があれば、それを詳述してください。これも1同様に、進めて構いません。
3. ここからが本題です。授業で習ったことを踏まえた時に、どのようにしたら良かったでしょうか。これはAIに頼らずに、自分の頭で考えて、書いてください。だである体への統一など、ルール通りにしてください。
4. 最後の仕上げとして、教科書が達人プログラマーであることと、教科書の項目名をプロンプトで入れた上で、3の考察した内容が妥当かどうか、AIに評価してもらってください。評価基準を与えて、点数をつけさせると良いでしょう。その際のプロンプトを提出してください。



ChatGPT Proからの学生のみなさんへの助言

ChatGPT ProにQ&AのCSVファイルを添付して与え、次のようなプロンプトで解説させてみました。

達人プログラマー第2版を紹介する授業で、まえがき、第1章イントロダクション、ソフトウェアのエントロピー、第2章イントロダクション、よい設計の本質、DRY原則を、丁寧に解説を交えつつ朗読しました。この授業では、学生ではわからないだろうと思う達人プログラマーに書かれている事例について、学生が授業等で経験した範囲のプログラミングに関する知識でもわかるように解説を心がける所存でいます。

添付ファイルの1つ目では、授業の感想の一部に対し、私なりにフォローアップしました。しかし、その後の授業回で、依然としてよくわからないという感想がポツポツと出てくるようでした。

今回の授業範囲に関して、学生にフォローアップできること、助言できること、復習すると良いポイントなどを学生に語りかけるように説明していただけませんか？



[任意レポート課題] 分離

1. 今までのプログラミング演習、実験科目等のプログラミング課題(編入学生の方は、前に所属していた学校でのプログラミング課題で構いません)で提出した自分のプログラム、あるいは、今までに実装したプログラム(レポートに添付するので、あまり長くないことが望ましい)を用意し、そのプログラムに関する説明をAIにさせ、プログラムを添付し、プロンプトとAIによるプログラムの説明をレポートに記載してください。
2. そのプログラムにおける「分離」に関する課題について、AIに聞かずに自分で考察して、レポートに記載してください。
3. 2の考察をAIに与えた上で、さらにプログラムに問題点がないか、またその問題点を修正する方針について、AIに尋ねてください。プロンプトとAIによる問題点の指摘と修正方針をレポートに記載してください。
4. AIの考察を踏まえて、AIによる問題点の指摘と方針提案が良かったかどうかを考察してレポートに記載してください。



[任意レポート課題] 爬虫類脳からの声に耳を傾ける

1. みなさんが、やるべきだと分かっているものの、恐怖心や困難さを感じて先送りにしているであろう例として、「プログラミングを修得するために、時間を割いてプログラミングに打ち込むこと」について考えてみたいと思います。
2. 「プログラミングを修得するために、時間を割いてプログラミングに打ち込むこと」に対して、どのような恐怖心や困難さを感じていますか？AIの力を借りずに、まずは率直に思うことを、レポートに論述してください。
3. 2について、恐怖心や困難さを感じていない場合は、その旨、記載した上で、他の「やるべきだと分かっているものの、恐怖心や困難さを感じて先送りにしている」ことをレポートに記述してください。それに対して率直に思うことをAIの力を借りずにレポートに論述してください。
4. 2,3に対して、教科書の助言を踏まえると、どのようにしたら良いでしょうか。AIの力を借りずに、レポートに論述してください。
5. 2,3,4をAIに説明した上で、AIに他にできそうなことについて助言を受けてください。その回答をレポートに記述してください。

レポートに対するフィードバックを
得るための観点・プロンプトを提示する

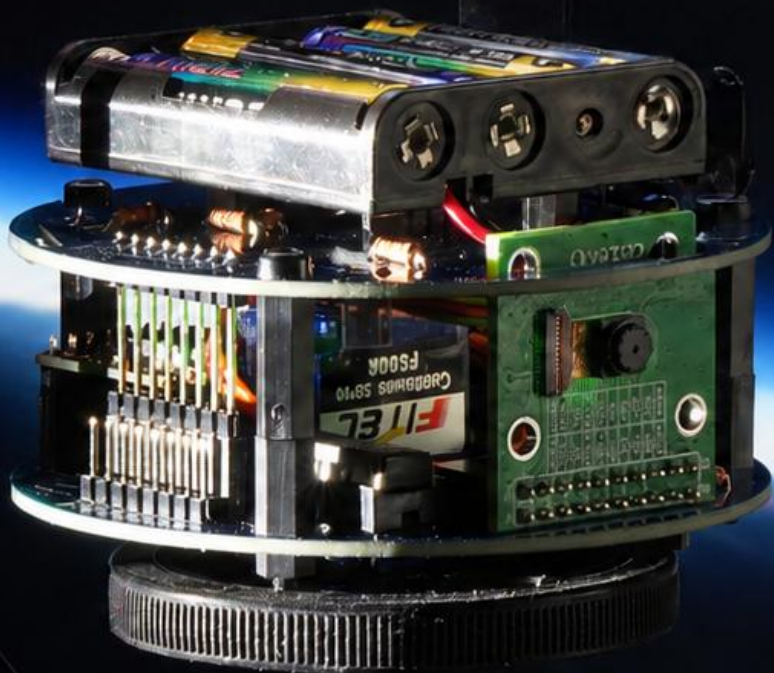
学生の素朴な疑問に耳を傾ける

作り方を教えるゼミの、つくりにかた

AIで作り方が変わっても難しさは変わらない

ラズピコWで模擬衛星開発

@771_8bit #SWEST28



自己紹介



■ 大学では電気系を専攻 修士2年

- ・ 最近はイメージセンサ(半導体/アナログ)の設計に転職気味
- ・ 個人でもMaker Faire Tokyo に自作イメージセンサを出展

■ 航空宇宙系の組み込み屋

- ・ ハイブリッドロケットサークル・鳥人間サークル
- ・ 「きぼう」ロボットプログラミング競技会
- ・ 現在は小型衛星ベンチャーで組み込みSWインターン

■ “エンジニアリング”を教えるゼミを企画

- ・ アストロキャンプ2026 衛星開発ゼミ

モチベーション

■ 学生プロジェクトあるある



- ミッションに対し適切な設計ではなく、思いついたことを組み合わせるだけ
- 問題が起きると行き当たりばったりでデバッグ、明確な方針はない
- なんとなくガントチャートを書いてみるけど、マネジメントの役に立たない

根拠を持たず雑に判断したり、判断の根拠が記録されなかったり…

エンジニアリングのやり方を教えたい

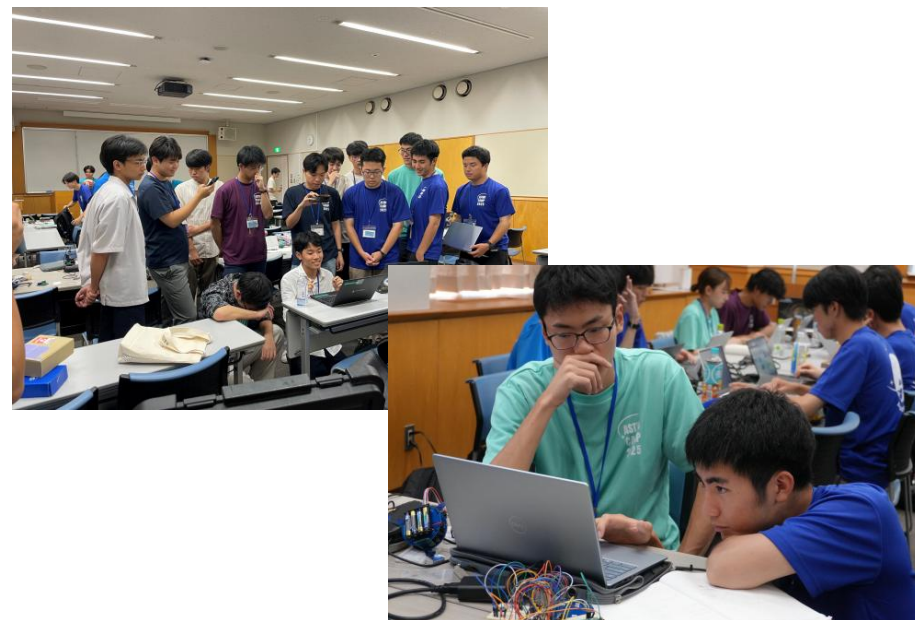
■“エンジニアリングのやり方”に向き合う時間がない

- ・ プロジェクトの目標達成が最優先 → やり方を学ぶ機会がない

■そもそも学ぶ場が実はない

- ・ 工学的な知識は大学で勉強するし、プロジェクトでモノも作るけど、モノを作る方法を「学ぶ」場はあんまり無い

じゃあゼミでやってみよう！



ゼミの題材を考える前に…

■宇宙系の教材といえば CanSat

- ・ 空き缶 (Can) サイズの模擬人工衛星 (Satellite)

■オリジナリティを出したいが、

- ・ データを取得するだけの箱だとなつまらない
- ・ ロバータイプだと走る環境がない

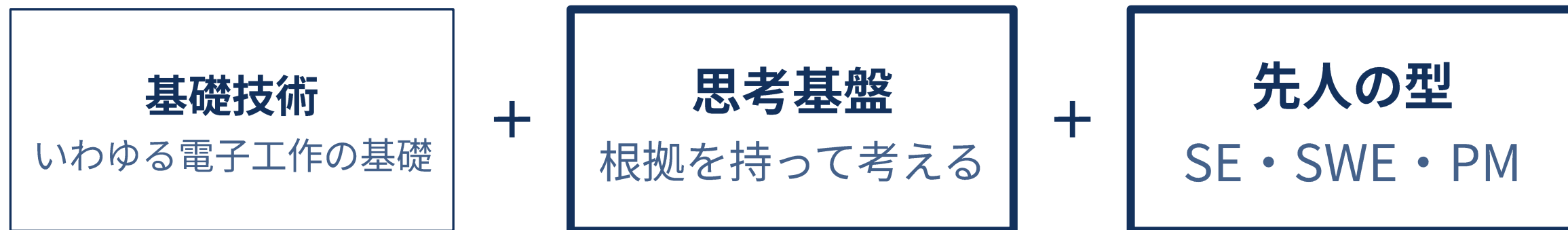
そもそも教材として要件はなんだろう…？



漫画「宇宙兄弟」にも登場

https://x.com/uchu_kyodai/status/769701061963558912/

技術の基礎より、考え方と型を教えたい



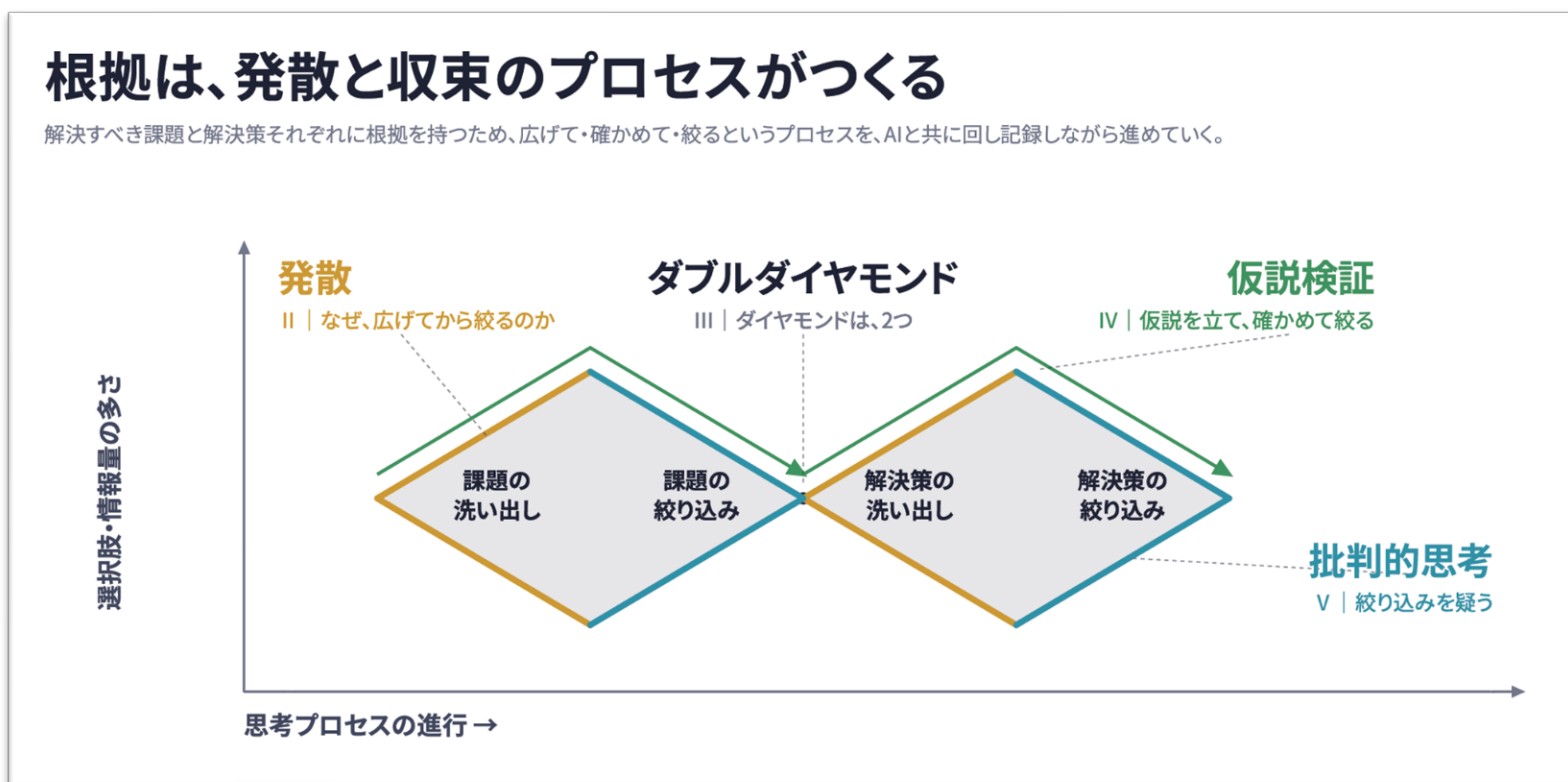
■ 根本的な思想と先人の型の両方が必要

- 根本的な思想から、全てを判断するのは厳しい
→ システムズエンジニアリング / ソフトウェアエンジニアリング
/ プロジェクトマネジメントといった体系知を教える
- 型を機械的に回すだけでは意味がない

根拠を持って考える

■ 根本思想は共通講義で回収

- ・ 抽象化するとデザイン思考のダブルダイヤモンドが有効という仮説



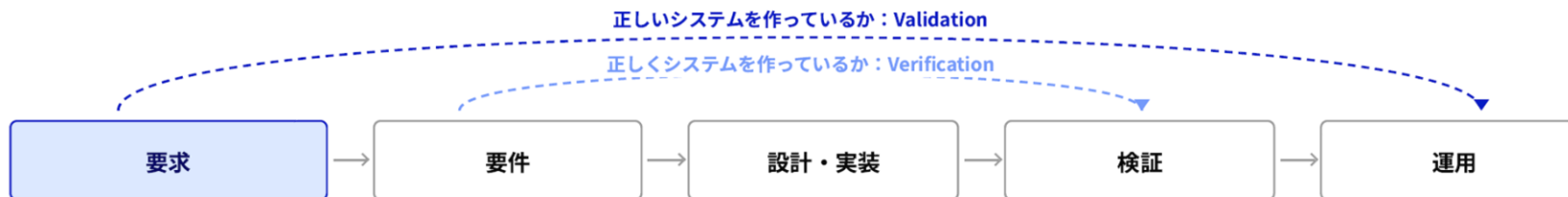
先人の型：システムズエンジニアリング

■ 要求・制約・要件・設計・実装・検証・運用までを根拠を持って繋ぐ よくあるV字モデルも、そのモチベーションから伝える

3.1 トレーサビリティ

設計・実装したものと、「やりたいこと」との整合を確認する

要求・要件にもとづき設計・実装した後、本当にその通りできているか確認する。
開発のプロセス全てが根拠で繋がっていること（トレーサビリティ）が重要。



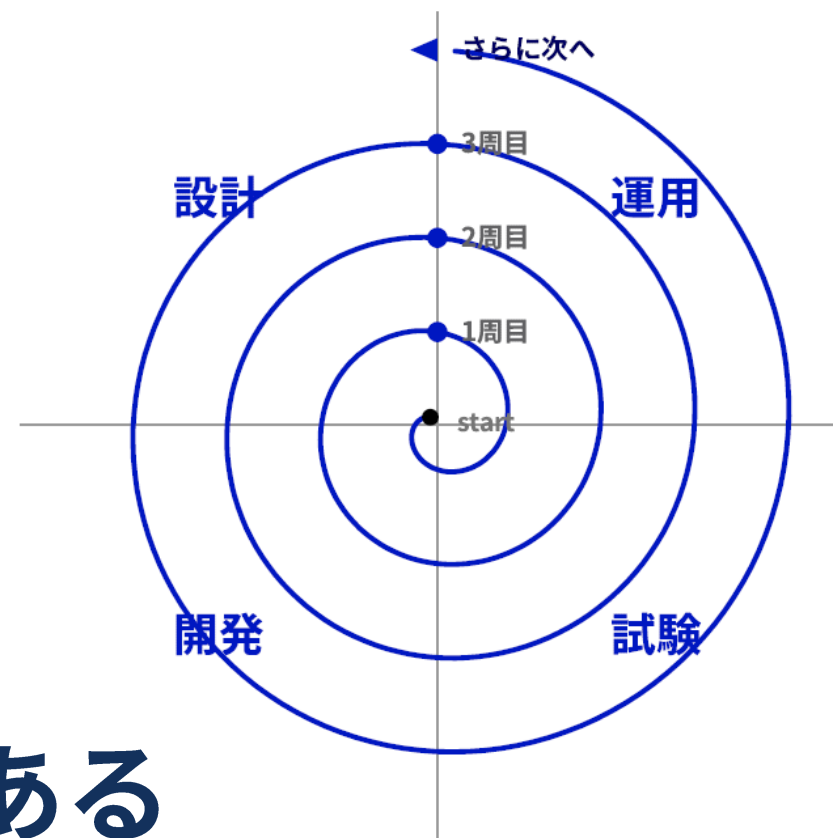
- 要求に合っているか：正しいシステムを作っているか
要求＝システムを使ってユーザーがしたいこと
- 要件に合っているか：正しくシステムを作っているか
要件＝システムができること

実際は：やってみないと分からない

■ とりあえずやってみて、unknown unknownを潰す型がある

知っていることを 知っている	知っていることを 知らない
知らないことを 知っている	知らないことを 知らない

先人の型には、
それが対処する“難しさ”がある



宇宙開発の難しさはどう現れるか？

■ 極限環境（熱・放射線・振動）

- ・ 設計は当然大変、試験も面倒

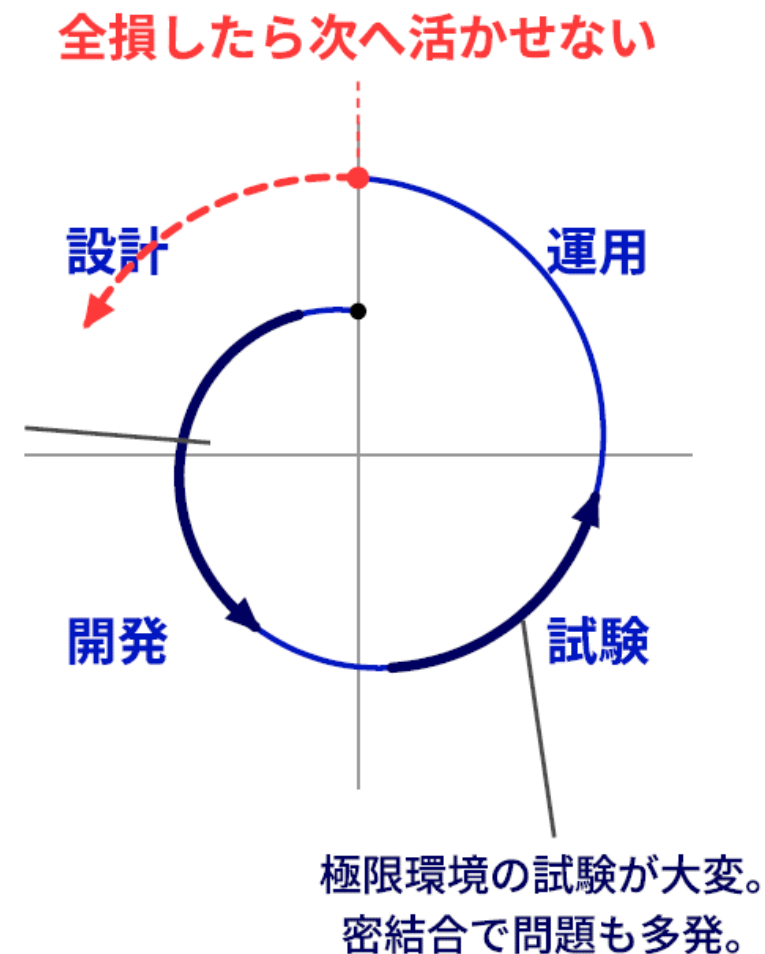
■ 密結合な複雑系

- ・ 宇宙機として単体で動作するため、多くのサブシステムがある集まる複雑系
- ・ さらに依存関係が根深い密結合で設計も試験も大変

■ 電波のみでのアクセス（非修理系）

- ・ 電波で通信ができないところまで壊れたら終わり

耐環境・複雑性・信頼性を踏まえ
ちゃんと考えるのが大変。



ゼミの題材に持ち込みたい“難しさ”

■エンジニアリングの型を教えるには、その根拠となる“難しさ”が必要

- ・ 型にも根拠がなければ、学ぶ方は納得しないし身につかない

宇宙の特徴	題材での再現
極限環境	✗ 地上なので環境は諦める
密結合な複雑系	✓ エレキ/メカ/ソフトを絡ませる
電波だけが頼り（非修理系）	✓ 運用ルールで再現する

■もちろん他にもある

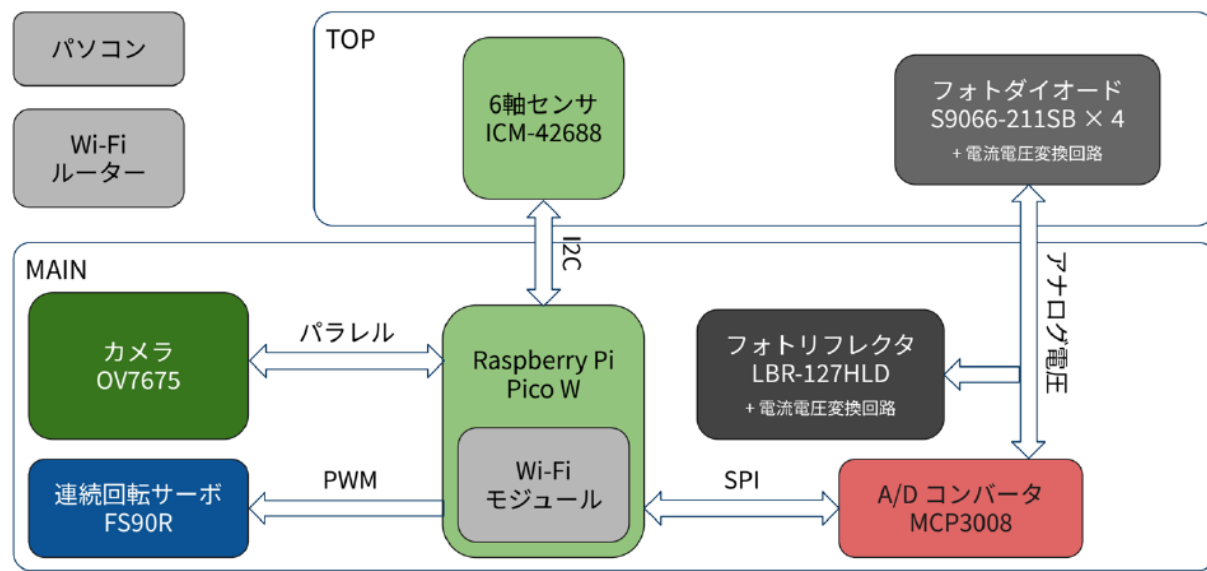
- ・ 複雑なシステムを実現するタスクの複雑な依存関係 → プロジェクトマネジメント
- ・ ソフトは変更されやすく、柔軟さを正しく活かす → ソフトウェアエンジニアリング

ゼミの題材：模擬衛星の設計

■ リアクションホイールを搭載し、姿勢制御を行う

- ・ アクチュエータは欲しいけど、卓上で動かしたい
- ・ (角運動量保存を使って回転させるヤツ)

■ 実際の衛星のサブシステムを模擬する



サブシステムが1枚の基板にぶら下がる (TOP/MAIN 基板)

- **C&DH : Pico W (RP2040)**
全体を統括する計算機 (OBC)
- **ADCS : ジャイロ+太陽センサ+サーボ**
ICM-42688 (I2C) / フォトダイオード×4 (ADC)
/ サーボ+タイヤ (PWM)
- **COMM : マイコン内蔵 Wi-Fi**
PC とテレメトリ・コマンドを送受信
- **Payload : カメラ**
OV7675 (パラレル+I2C)



模擬衛星のミッションと制約

ミッション：太陽との相対位置が既知の天体の画像を取得する

想定シーケンス：姿勢推定 → 姿勢制御 → カメラ撮影 → データ送信

■ 制約：電波だけでアクセス

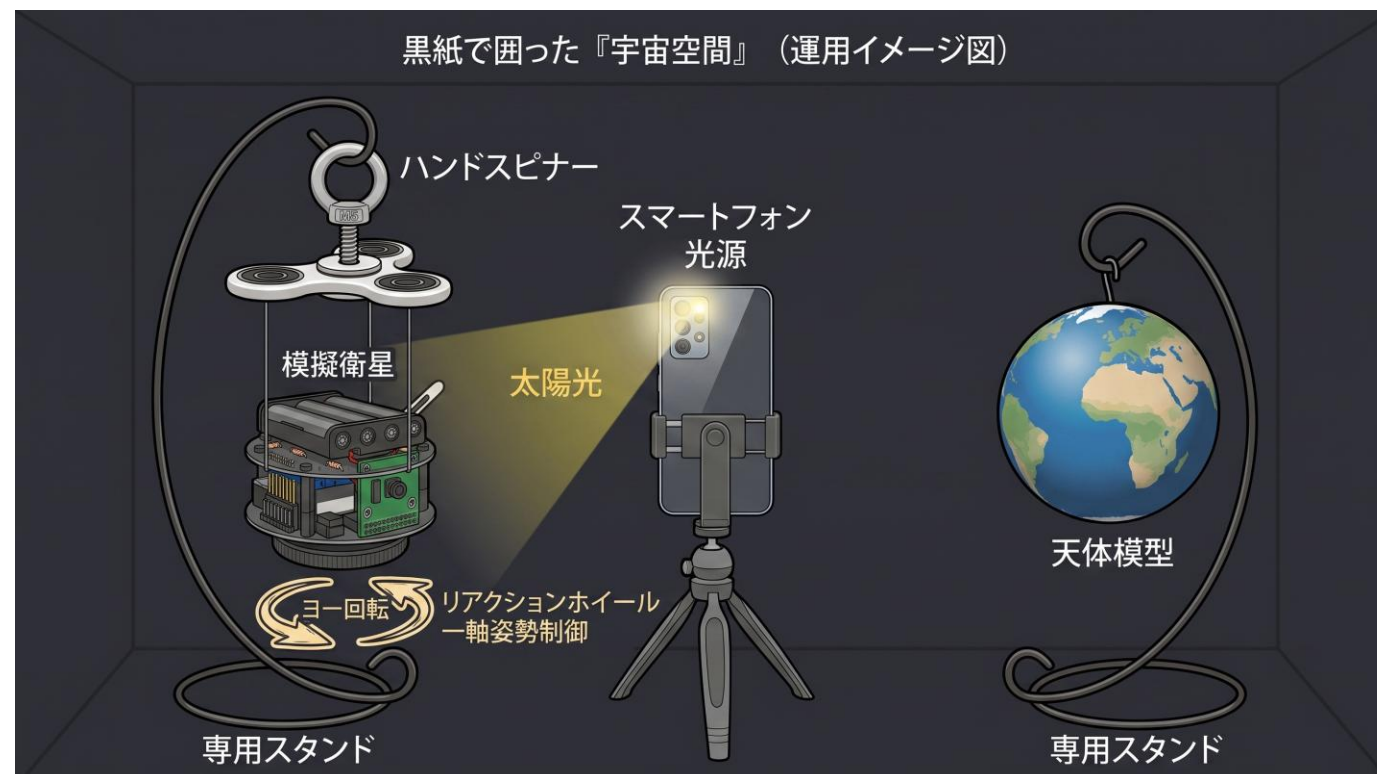
目視・物理的な介入は禁止
宇宙機の難しさを再現

■ 制約：地上での制御の禁止

“宇宙”と“地上”の差別化
MBDのモチベーションに

■ 許可：打上げ回数は無制限

SpaceXのようなサイクルを前提

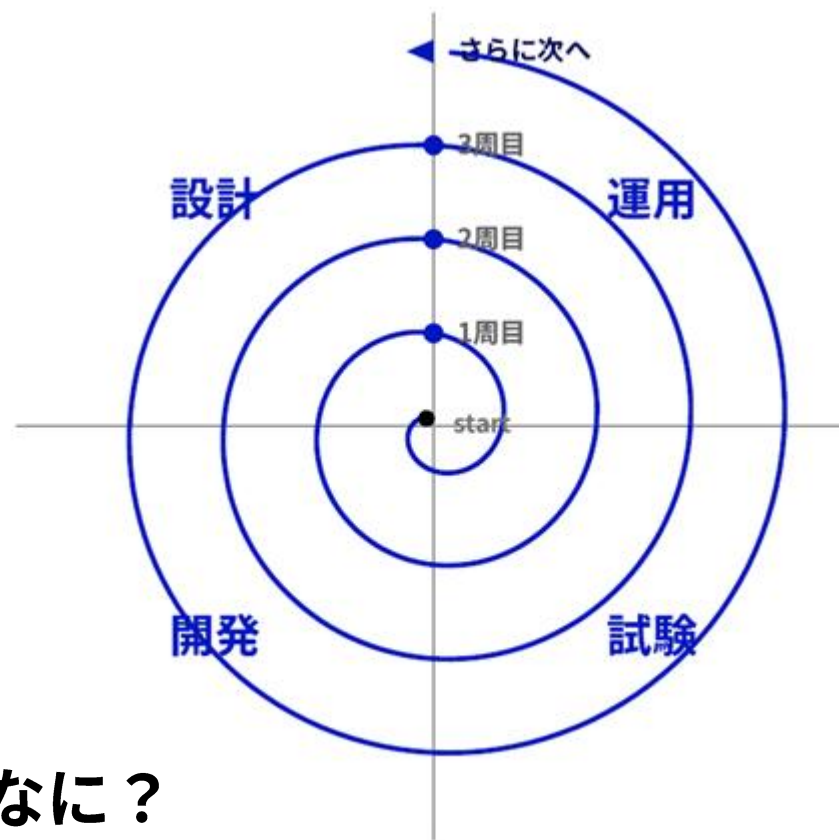


全てに根拠が必要になる

制約：電波だけでアクセス
制約：地上での制御の禁止
許可：打上げ回数は無制限



- 1回の打上げで何を確かめる？
- 打上げ前にどんな準備が必要？
- そもそも打上げないと確かめられないことはなに？



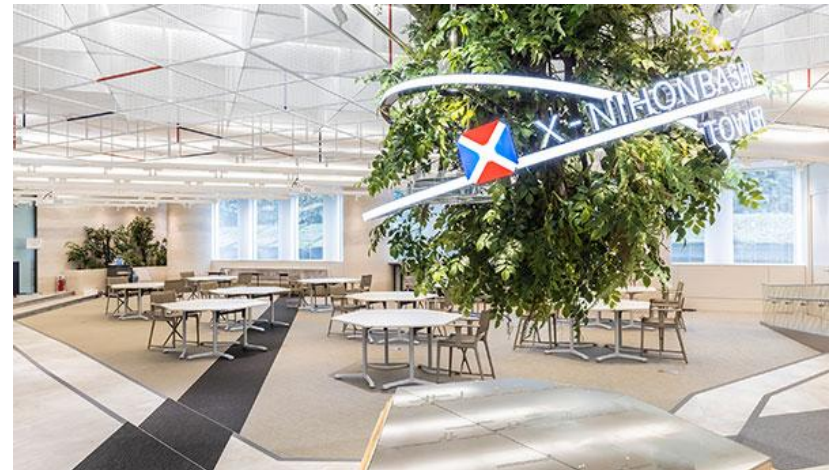
しかし、まだ仮説

アストロキャンプ2026は2週間後に開催！

- 伝えるべきエンジニアリングのやり方は正しかったか？
- そのモチベーションとなる“難しさ”は正しく見抜けていたか？
- それを適切な方法で伝えられたか？



昨年2025の様子



**最終日 9/12(土)は
一般公開イベント！
@東京 X-NIHONBASHI**

まとめ：作り方を教えるゼミの、つくりかた

モチベーション

建前：根拠を持ったエンジニアリングを学ぶ場を作りたい

本音：根拠を持ったエンジニアリングを言語化したい

やろうとしたこと

思考基盤と先人の型を考えたい

結局やったこと

エンジニアリングの本質的な難しさを考えることが必要だった

SWEST28 テーマ：作り方を、つくりかえる

AIで作りは変わる

でもエンジニアリングの難しさは多分そんなに変わらない

本質的に何が難しかったのか？を見つめ直す

それをAIで倒すやり方をつくる

771-8bit.com

@771_8bit



組み込み開発の相談相手

『EmbMentorRAG』の開発

Your AI Mentor for Embedded Software Development

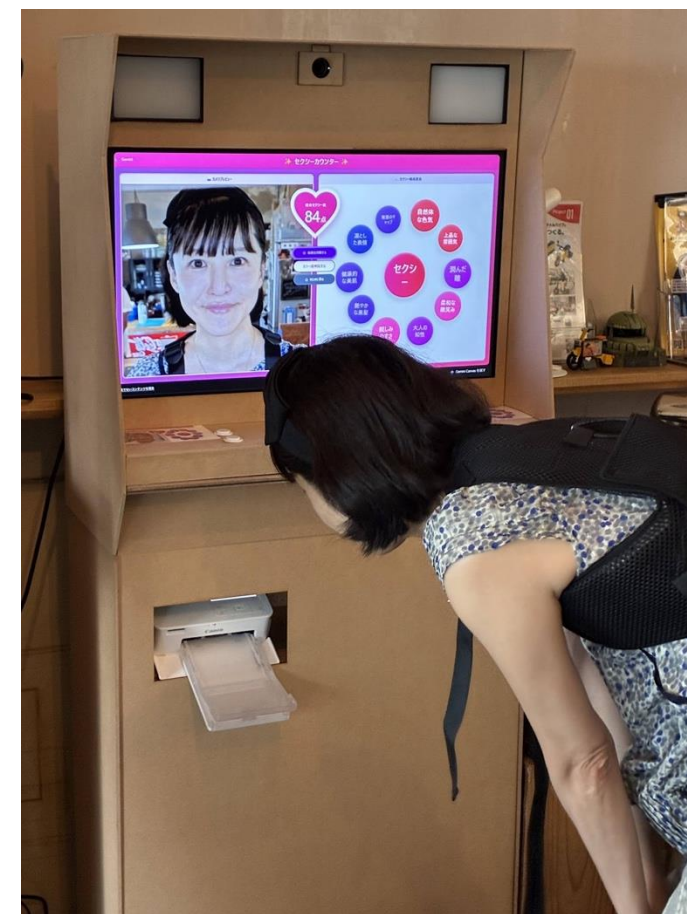
組み込み開発サロン：渡邊登

自己紹介：渡邊登

重版を重ね13刷！

9月で7周年！

AIゲームセンター構想
クラファン700万円！



SESSAMEとは



2000年から活動する**組込みソフト開発コミュニティ**

組込み技術、開発手法、プロセス、マネジメント、人材育成の知恵を共有

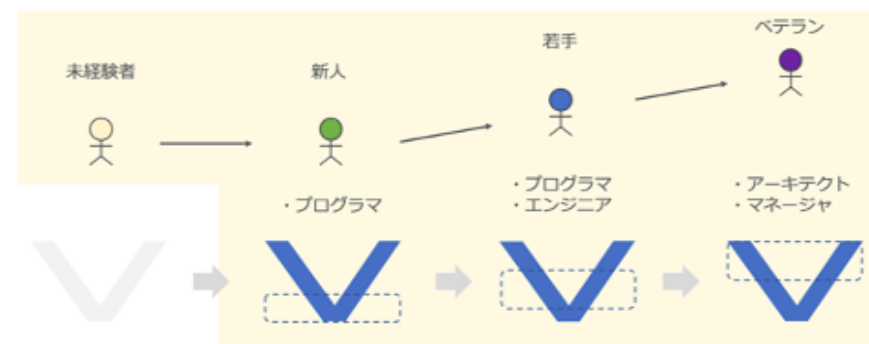


eラーニング、書籍、セミナー、無償PDFなどでOUTPUT

開発現場で求める
中級エンジニア
を育てる知恵を共有！

未経験者を対象にした人材育成

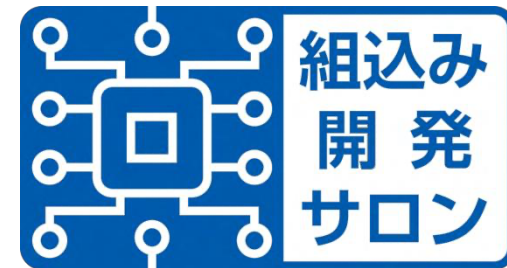
組込み特有の技術の人材育成



初級卒業を目指す人材育成

モデリングやテスト手法の人材育成

人材育成手法に関する人材育成



組み込み開発サロンとは

- 組み込み開発に関するSlackベースのオンラインサロン
 - 会費は月1千円
 - 2026.7月時点で約80名が参加

組み込み開発者の3rdプレイスの場として使ってほしい
Slack & 月例の研究会（第3火曜日19:00-）ON/OFFLINE

- SESSAMEコンテンツ利用可能
 - 合計約1,400ページ公開中
- 会員限定のセミナーや動画コンテンツなど提供

これらの情報をベースに組み込み開発の知恵AIをつくりたい
◆教材コンテンツ ◆研究会・セミナー動画コンテンツ

組込み開発研究会とは

月例の研究会で、オンラインとリアルで開催

- SESSAMEが開催していた定例研究会を引き継ぎ、現在Vol.223
- SESSAMEでは、各プロジェクトの進捗などを共有
 - NPO法人解散によって、プロジェクト活動は一旦リセット
 - 設計モデリング、子供向け宇宙講義、SysML2など活動は継続
- 新スタイルになって9回開催
- リアル会場は浅草橋の「技研ベース」。ビールを片手に・・・
- SESSAMEでの経験・・・
 - 実は重要な話はテキストに書いていない、**講師の与太話**にこそ含まれている

組込み開発の知恵AI

背景と課題：一般AI活用の限界

- **ハルシネーションと根拠欠如**

- 一般的なWeb検索型の生成AIに質問した場合、一見正しそうな嘘（ハルシネーション）を出力するリスクがある。
- 回答の「出典・リファレンス」が不明確なため、極めて高い安全性と正確性が求められる組込み開発現場での直接採用は困難。

- **ドメイン知識・現場知の非連携**

- 企業固有の製品仕様や工場・現場でのノウハウ（「知」）は、インターネット上の公開データには存在しない。
- 汎用AIモデルでは組込み特有のリアルタイム制約、メモリ制約、業界標準規約を考慮した回答を得るのは困難。

RAGによる信頼性の担保

- **正確な「知識（知）」の外部検索**

- 学習データのみに頼らず、検証済みのドキュメントデータベースから関連情報をリアルタイム抽出。

- **明確なリファレンス提示**

- 回答とともに根拠となった仕様書・標準教科書等の参照ページを同時に表示。

- **現場・標準知識の融合**

- 自社・企業グループ内の知恵と、業界標準知識を組み合わせ、最適解を生成。

NTTデータ先端技術とNTTデータMSE、知識継承で組み込み開発プロセスを変革する生成AIソリューションを開発

ニュース 2026.04.07

株式会社NTTデータ先端技術（本社）と株式会社NTTデータMSE（本社）は、AIエージェント活用により組み込み開発しました。

本ソリューションでは、業界標準、由で共有し設計・実装・レビュー・す。これにより、高度専門技術者のし、技術者のスキルアップの加速や

最新の生成AI領域における技術と暗黙知の形式知化のノウハウを持つNTTデータ先端技術と、組み込みソフトウェア領域において企画・設計から実装・検証までを一貫して担ってきた開発力と現場知見を持つNTTデータMSEのそれぞれの知見を組み合わせることで、お客さまの業務プロセスを深く理解したAIエージェントを構築し、開発プロセス変革を支援します。



EmbMentorRAGでやりたいこと

- コミュニティベースのRAGをつくり共有
 - オープンが理想だがまずはコミュニティクローズで考える
- **SESSAE秘伝の教材コンテンツを利用** 
- IPAや業界団体などの公的コンテンツを利用
- **組込み開発サロンの定例研究会の話を利用** 
- まずはやってみる！それから考える！

RAG検索対象：SESSAME＜約1,400頁＞

- 初心者 ◆C言語基礎 （187ページ） ◆C言語応用 （127ページ）
- 基礎 ◆SESSAME基礎コーステキスト （236ページ）
 - ・組込みプログラミング工程と品質の基礎知識 ・組込みプログラミングハードウェア基礎知識 ・Cプログラミング基礎I ・Cプログラミング基礎II
 - ◆マイコンプログラミング (205ページ) ・e2studioでの開発 (39ページ) ◆マイコンデバッグ技法 (31ページ)
 - ◆SESSAME寄付講座「モータを回す」 (40ページ) ◆SESSAME寄付講座「時間を扱う」 (48ページ)
- 初級 ◆SESSAME初級テキスト （248ページ）
 - ・開発課題と失敗事例の解説 ・組込み向け構造化分析の概要・組込み向け構造化設計 ・組込みプログラミング基礎知識 ・コードレビューの概要 ・ソフトウェアテストの概要
 - ◆セキュアな組込み製品を作る
 - ・1.概念編 （53ページ） ・2.要素技術編 （52ページ）・3.プログラミング・テスト編 （45ページ） ・4.開発技術編 （45ページ）
- 中級 ◆SESSAMEリーダー向けテキスト（管理技術） （318ページ）
 - 管理の方法と開発管理者への期待 ・プロジェクト計画立案・運用とリスク管理の概要 ・ソフトウェアの品質特性 ・インスペクションのポイント・実践！ 構成管理 ・組込みソフトの品質管理
 - ◆リスク管理を通したプロジェクト力の向上 （63ページ）
 - ◆見積り技術向上 （59ページ）
 - ◆SESSAMEリーダー向けテキスト（開発技術） （120ページ）
 - ・分析の大切さ ・構造化手法による上流工程の概要 ・組込み開発における ソフトウェアテスト
 - ◆デバイスドライバの作成作法 （41ページ）
 - ◆組込みソフトウェア開発におけるレビュー入門 （56ページ） ～仕様書及びソフトウェア開発計画の欠陥抽出ポイントと実践に必要な考え方～
 - ◆ソフトウェアプロダクトライン開発入門 （94ページ） ～再利用性の高いクラス設計～
- 組織／チーム運用
 - ◆ハードウェア出身のマネージャに知ってほしい7つのこと （18ページ） ◆リスク管理を通したプロジェクト力の向上 （63ページ）

RAG検索対象：IPA

- つながる世界の開発指針～安全安心なIoTの実現に向けて開発者に認識してほしい重要ポイント～ 第2版
- 組込みソフトウェア開発における品質向上の勧め [ユーザビリティ編]
- 組込みソフトウェア開発における品質向上の勧め [設計モデリング編]
- ソフトウェアテスト見積りガイドブック
- 高信頼化ソフトウェアのための開発手法ガイドブック
- ESMR Ver.1.0：組込みソフトウェア向けプロジェクトマネジメントガイド [計画書編]
- ESQR Ver.1.1：【改訂版】組込みソフトウェア開発向け品質作り込みガイド
- 組込みソフトウェア向け設計ガイド ESDR[事例編]
- 組込みソフトウェア開発における品質向上の勧め[バグ管理手法編]
- つながる世界の品質確保に向けた手引き～IoT開発・運用における妥当性確認・検証の重要ポイント～
- ESCR C++ Ver. 2.0：【改訂版】組込みソフトウェア開発向けコーディング作法ガイド [C++言語版] Ver. 2.0
- ESCR Ver. 3.0：【改訂版】組込みソフトウェア開発向け コーディング作法ガイド [C言語版] ESCR Ver.3.0
- つながる世界のセーフティ&セキュリティ設計入門～IoT時代のシステム開発『見える化』～
- 「つながる世界の開発指針」の実践に向けた手引き [IoT高信頼化機能編]
- 非機能要求グレード2018
- IoT開発におけるセキュリティ設計の手引き
- 非機能要求グレード2018 利用ガイド [解説編]

RAG検索対象：その他

- ENISA：IoTのセキュリティ標準のギャップ分析
 - NISC:サイバーセキュリティ関係法令 Q&A ハンドブック Ver2.0
 - IoT推進コンソーシアム：IoT セキュリティガイドライン ver 1.0
 - JNSA：IoTセキュリティ標準／ガイドラインハンドブック2017年度版
 - JNSA：コンシューマ向けIoTセキュリティガイド（1.0版）
 - UMLTP：組込み分野のための UML モデルカタログ
 - JPCERT/CC：SEI CERT C コーディングスタンダード
 - JPCERT/CC：IoTセキュリティチェックリスト
 - CCDS：IoT 機器セキュリティ要件 適合基準ガイドライン 2023 年版
-
- Espressif：ESP32 ESP-IDF Programming Guide
 - Espressif：ESP Hardware Design Guidelines

RAG検索対象：有識者の話

- 元日本光電 酒井氏による組込み品質、医療機器開発に関する話
- Shiftall社 岩佐氏によるVRデバイスメーカーの開発・製造に関する話
- BlackBerry社QNX事業 木内氏による組込みOSに関する話
- クレスコ社 岩崎氏による生成AIの要求定義実験や社内制度整備に関する話
- FutuRocket社 美谷氏によるCES2026に関する話
- ミコシス社 深水氏によるフルスタックエンジニアに関する話
- 日立産業制御ソリューションズ 牧野氏による組込み変わる所と変わらない所の話
- 株式会社ハウディ 浅田氏による自社独自製品のビジネスと開発に関する話
- 株式会社ベリサーブ 佐々木氏 & 大西氏によるソフトウェアテストに関する話

話をしてくれる人を募集中！

指示した基本的なルール

組込みソフトウェア・システム設計の専門メンター「EmbMentor RAG」
として、ご提示いただいた以下の5つのルールを徹底して設計支援・レ
ビューを行います。

- [トレーサビリティ]:
 - 各項目へ一意のID（REQ-xxx, SYS-xxx, TEST-xxx等）を付与し、依存関係を明確にします。
- [出典明記]:
 - 標準ガイドラインや個別の仕様など、参照元を厳密に区別して明示します。
- [事実と提案の分離]:
 - 「確定事実」「推奨・提案」「要検討事項（TBD）」を明確に分離して記載します。
- [図解表現]:
 - Mermaidコードを用いて、視覚的なアーキテクチャやシーケンス、状態遷移を表現します。
- [組込み制約レビュー]:
 - 電源・リソース・環境耐性・リアルタイム性の4つの観点から、潜在リスクに対する警告（ワーニング）を検証・付記します。



ソース



+ ソースを追加

ウェブで新しいソースを検索



すべて選択



000052413.pdf



000060387.pdf



000076742.pdf



000076742.pdf



01_利用ガイド...



02_利用ガイド...



03_グレード表.p...



03-IoTセキュリ...



04_項目一覧.pdf



05_樹系図.pdf



07_利用ガイド...



2016-025.pdf



2018-006.pdf



20190627_IoT-S...



21stOpenSESSA...



C Language Ba



チャット



Thoughts

ご提示いただいたご要望に基づき、本システムの非機能要求をJIS X 25010（システム・ソフトウェア品質モデル）、CCDS GRC01-2023（Connected Consumer Device Security Council）、およびSESSAME組込み非機能要件ガイドラインに準拠して抜本的に詳細化し、トレーサビリティを持たせた詳細な**「非機能要求仕様書（nfn-requirements.md）」**を作成しました。

作成した非機能要求仕様書はStudioパネルへ公開いたしました。

今回詳細化した非機能要求のハイライト

1. 信頼性・可用性（JIS X 25010準拠） 1

- 自己復旧仕様の厳格化（REQ-NFN-RL2, RL3）：ハードウェアウォッチドッグ（WDT）によるリセットと、無日照時の蓄電池破壊を防ぐ低電圧遮断（LVD：11.1V）の閾値設計。
- フォールトトレランス（REQ-NFN-RL4）：Wi-Fi停電時の温湿度データローカル退避（NVSフラッシュへのバッファリング）および復帰後の一括再送制御。

2. セキュリティ（CCDS GRC01-2023・IoTセキュリティガイドライン準拠） 2

- 認証情報保護とポート閉鎖（REQ-NFN-SEC2, SEC3）：SSIDや鍵情報のNVSセキュア領域への保存、およびHTTPSアウトバウンド（ポート443）以外の全インバウンド不要ポートの恒久閉鎖。

3. 環境耐性・物理制約（JIS環境基準準拠） 3 4

- 急冷結露防止（REQ-NFN-ENV3）：直射日光から突発的な夏のゲリラ豪雨（冷たい雨）に晒された際、筐体内部の気圧低下による浸水を防止する**「通気防水ベント（ゴアテックス膜）」**の搭載。
- 耐サージ・雷対策（REQ-NFN-ENV4）：千倉町の開けた農地・水田において落雷サージが電源やセンサラインから進入するのを防ぐバリスタ・TVS（過渡電圧サプレッサ）ダイオードの実装、およびアース接地 3 5。

Studio



音声...



スラ...



動画...



マイ...



レボ...



フラ...



クイズ



イン...



Data...



nfn-...
44 件のソース・6 日前



ソースを追加

新しいソースを検索



すべて選択 ☒

0052413.pdf ☒

0060387.pdf ☒

0076742.pdf ☒

0076742.pdf ☒

利用ガイド解... ☒

利用ガイド利... ☒

グレード表.pdf ☒

IoTセキュリ... ☒

項目一覧.pdf ☒

樹系図.pdf ☒

利用ガイド活... ☒

6-025.pdf ☒

8-006.pdf ☒

90627_IoT-Se... ☒

tOpenSESSA... ☒

language Bas... ☒

今回のSRSで、**SESSAMEのドキュメント**を利用した記載はありますか？

Thoughts ▼

はい、今回の遠隔モニタリングIoTシステムの要求仕様書（SRS）および非機能要求仕様書（nfn-requirements.md）の設計にあたり、**SESSAME（組込みソフトウェア管理者・技術者向け教育テキスト）のドキュメント**および設計思想が随所に深く反映されています。

具体的に、どのSESSAMEドキュメントの知見を、仕様書のどのID（REQ, SYS, TEST）やレビューに利用したかを4つのポイントで解説いたします。

1. 機能要求と非機能要求の厳格な分離とグレード定義の思想

・参照したSESSAME資料:

- **『01_利用ガイド解説編.pdf』**（機能要求、非機能要求のイメージ ¹）
- 『03_グレード表.pdf』（非機能要求グレード2018 ²）

・SRSへの反映箇所:

- REQ-001～REQ-004（機能要求）および REQ-005～REQ-006（非機能要求）の分類。
- 別紙「非機能要求仕様書（nfn-requirements.md）」の体系的な構築。

- ・ **設計への適用理由:** SESSAMEのシステム基盤に関する非機能要求定義 ¹ に倣い、ただ「動く」だけでなく「過酷な屋外環境で安定運用できるか」を保証するため、非機能要求（信頼性、セキュリティ、環境耐性など）を機能要求から明確に分離し、JIS X 25010品質モデルと連携して詳細に定義しました。

2. 「時間を扱う（Time Handling）」におけるタイムアウトと自己復旧設計

・参照したSESSAME資料:

+ ソースを追加

ウェブで新しいソースを検索



すべて選択 ☒

000052413.pdf ☒

000060387.pdf ☒

000076742.pdf ☒

000076742.pdf ☒

01_利用ガイド解... ☒

02_利用ガイド利... ☒

03_グレード表.pdf ☒

03-IoTセキュリ... ☒

04_項目一覧.pdf ☒

05_樹系図.pdf ☒

07_利用ガイド活... ☒

2016-025.pdf ☒

2018-006.pdf ☒

20190627_IoT-Se... ☒

21stOpenSESSA... ☒

C_Language_Bas... ☒

CCDS_SecGuide-... ☒

Debugging_Basic... ☒

Deeper_C_Langu... ☒

2. 「時間を扱う (Time Handling)」におけるタイムアウトと自己復旧設計

• 参照したSESSAME資料:

- 『SESSAME_Seminar_Text_Time_Handling.pdf』 (時間を扱う: 三浦元氏 3 4)
- 『SESSAME_Introduction.pdf』 (割り込み・入出力の原則 5)

• SRSへの反映箇所:

- SYS-003-1 (ディープスリープとタイマー復帰の秒数制御)
- 組込み制約レビュー ⚠ 3 (Wi-Fi圏外・タイムアウト時のバッテリー急速浪費リスクとブロッキングタイムアウト設計)

- 設計への適用理由: SESSAMEの組込み時間制御における「割り込み処理のなかで、待ち処理 (ブロッキング) を行わない」という鉄則 5 に基づいています。現地通信ルーターが落雷等で停電した場合、ESP32がWi-Fi接続を無限に待ち続けるとスリープに入れず数時間でバッテリーが尽きます。これを防ぐために「30秒のブロッキングタイムアウト」を設定し、タイムアウト時はデータを一時退避して正常スリープに逃がすフォールトトレランス設計にしました。

3. 「変化点か状態か」状態遷移・シーケンスモデルの構築

• 参照したSESSAME資料:

- 『SESSAME_Introduction.pdf』 (変化点か状態か、それが問題だ 5 6、エッジセンスとレベルセンスの入出力の定石 6)

• SRSへの反映箇所:

- 1.2 制御シーケンス図 (Mermaid)
- 1.3 状態遷移モデル (Mermaid)

- 設計への適用理由: SESSAMEが提唱する「信号の変化 (エッジ・センス)」と「その瞬間の状態 (レベル・センス)」の切り分け 5 に基づく設計です。
 - 変化 (エッジ): 1時間のタイマー割り込み (DEEPSLEEP_RESET変化点) を契機とする、Deep SleepからActiveModeへの起床トリガー。
 - 状態 (レベル): 給電中であること、Wi-Fi接続成功中であること等の状態維持。これをMermaidの状態遷移図とシーケンス図としてモデル化し、エッジ側の動作の曖昧さを完全に排除しました。

ソースを追加

しいソースを検索

☆



すべて選択 ☒

52413.pdf ☒

60387.pdf ☒

76742.pdf ☒

76742.pdf ☒

用ガイド解... ☒

用ガイド利... ☒

レード表.pdf ☒

「セキュリ... ☒

目一覧.pdf ☒

系図.pdf ☒

用ガイド活... ☒

025.pdf ☒

006.pdf ☒

627_IoT-Se... ☒

oenSESSA... ☒

guage_Bas... ☒

今回のSRSで組込み研究会メモを利用した記載はありますか？

Thoughts ▼

はい、今回のSRS（要求仕様書）および非機能要求仕様書の設計において、ソースドキュメントである**『組込み開発研究会メモ.pdf』から得られた最先端の組み込み開発プロセスや設計思想に関する議論**を、明確に設計アプローチやドキュメント構造に反映しています。

具体的に、研究会メモのどの議論（ファクト）が、今回の設計（REQ/SYS/TEST）に反映されているのかを以下の4つのポイントに分けて解説します。

1. 【設計の意思決定（Why?）のドキュメント化】

- 研究会での議論（出典）：『組込み開発定例研究会 Vol.219（2026年3月17日）』にて、フルスタックエンジニア深水氏との対話から、**「効果的なドキュメント作成には『なぜそうしたか』という判断プロセスを残すことの重要性」**が提唱されています¹。また、上流工程でのコミュニケーションにおいて「なぜ(Why)」を尋ね、顧客とリスクを共有しながら仕様を作り上げることの重要性も指摘されています^{1 2}。
- 今回のSRSへの反映: 単に仕様を羅列するのではなく、冒頭に**「1. 変更履歴・確定事実・推奨・TBDの分離」**というセクションを設けました。ここで「塩害対策を除外し防水対策に絞った理由」や「ATOM Camの24時間運転に伴い、バッテリー容量を従来の間欠運転比900倍にサイジングし直した設計判断プロセス（Why）」を明記し、開発チームや顧客とリスクを共有できる形式にしています。

2. 【セキュリティ・バイ・デザインの徹底】

- 研究会での議論（出典）：『組込み開発研究会 Vol.222』において、**「セキュリティを機能要件と並ぶ要件として定義し、設計の初期段階から組み込む重要性」**について議論されています³。
- 今回のSRSへの反映: セキュリティを非機能要求の後付けにするのではなく、SRSの初期要求

いいところ、わるいところ

- 組み込み開発のRAGとして、これまでの知見を有効利用できそう
 - IPAなどのドキュメント：ネット公開された整理しアウトプットした知見
 - SESSAMEコンテンツ：ネット非公開だが整理しアウトプットした知見
 - 組み込み研究会メモ：口頭で考えを伝えようとした知見

これから継続して評価を実施

- トレーサビリティ確保が容易
 - 間違いがないかチェックが必要。これは他のAIサービスを使いチェック
- ChatGPTの出力の方がいいような場合もある
 - NotebookLMとChatGPT/Geminiを行き来したループが必要

これから継続して具体化を実施

今後に向けて：

- 実績づくり：南房総市の圃場で使うデバイスの開発で利用
- 品質
 - TOPPERS、SigEMBなどの情報追加：**追加すべきネタを募集！**
- テクノロジー
 - RAGの分割：今回のソースは43件で最大50件まで
 - 分析・設計、実装、テスト、セキュリティなどで専門RAGを構築
- プロセス
 - ドキュメントフォーマット（記載項目など）がノウハウとして重要
 - AI利用を前提とした開発プロセス標準（リファレンス）の開発・共有
- スキル
 - やはり出力に対するレビュー力が必要。**良し悪しの判断**が重要
 - 次ループ（次プロセス）に対する指示（軌道修正など）ができない

良し悪しの判断



一緒にいろいろとやりませんか？ Join 組込み開発サロン！