

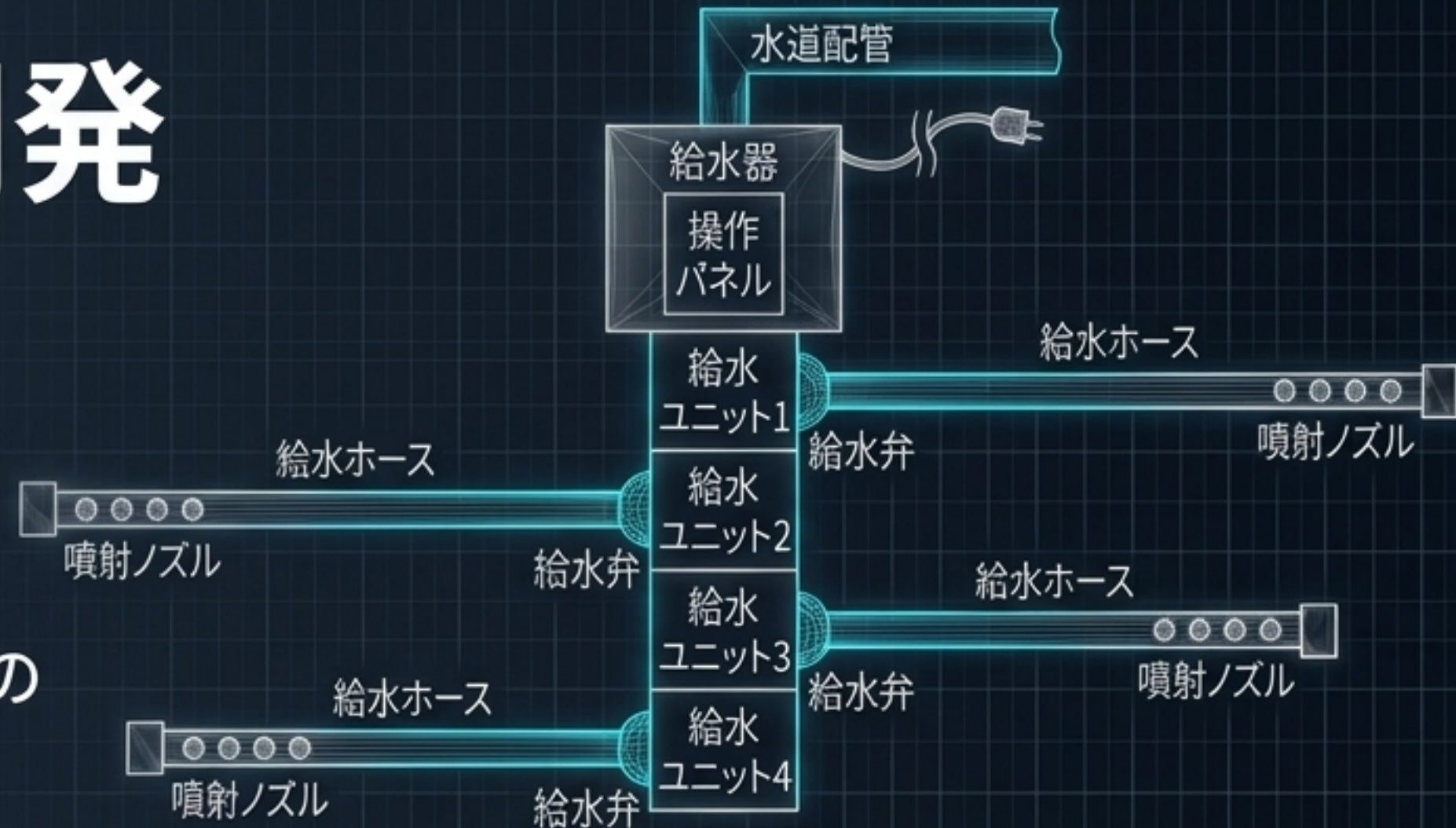
組込みシステム開発 徹底攻略論理

～平成26年春期 午後問7～

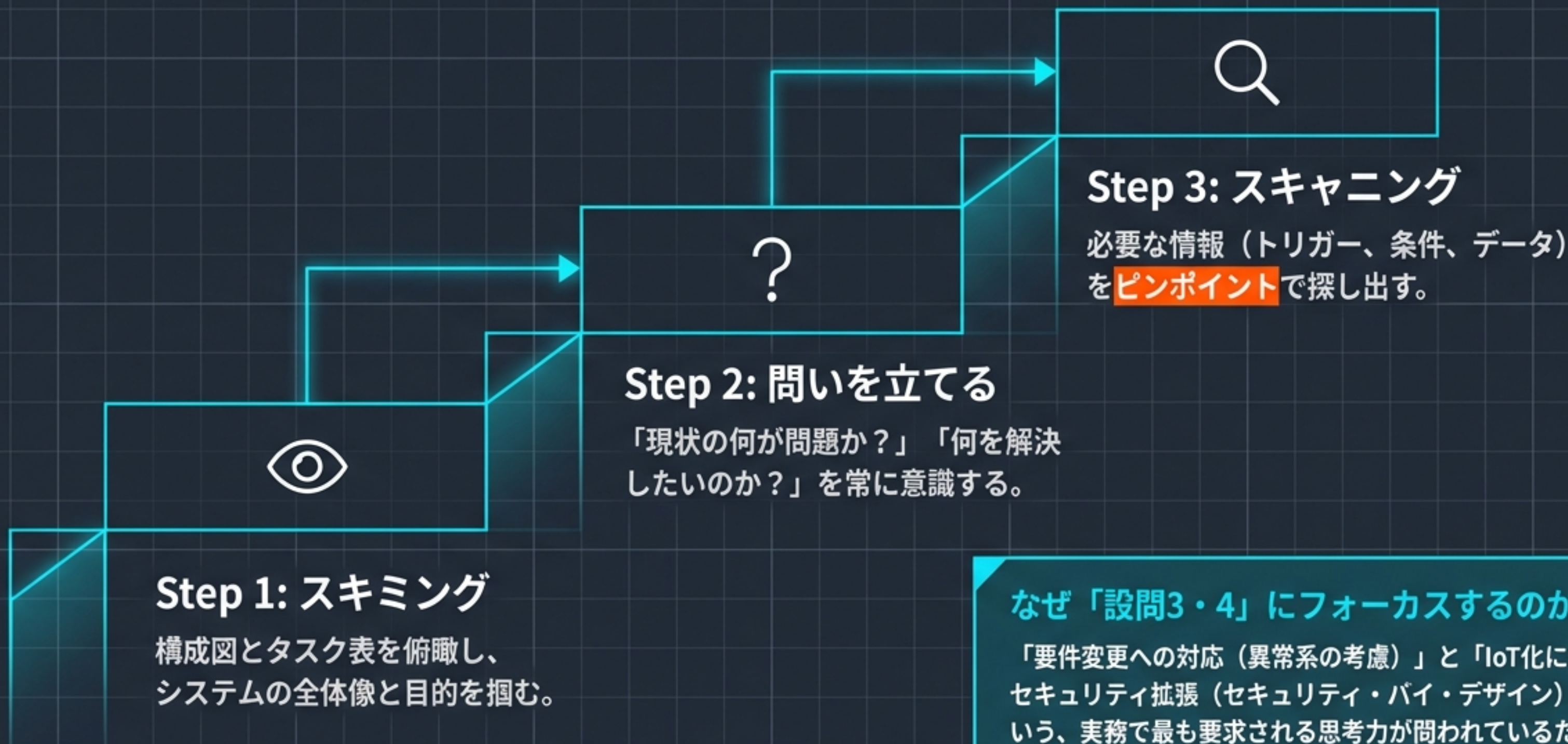
設問3・4から学ぶ、解答を自ら導き出すための

「思考プロセス」と「システム読解法」

(※本資料に解答そのものは一切記載されていません。
自らの思考で辿り着くための設計図です)



午後問題攻略の極意：「読解」の3ステップ



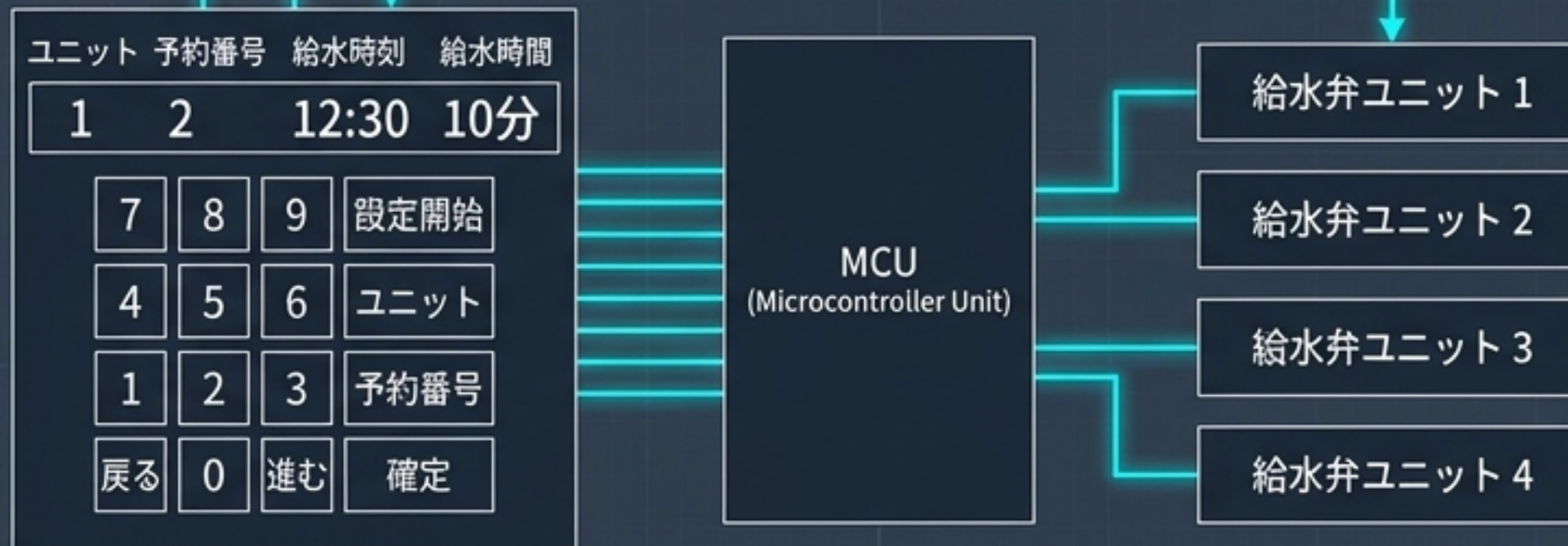
対象システムのブループリント（全体像）

タスク名	動作内容
初期化	・システムの初期化を行い、不揮発性メモリに記憶されている全ての給水設定を給水スケジュールタスクに通知する。その後、給水設定タスクも起動し、
給水設定	・キースキャンタスクからのキーコードを持ち、キーコードを記憶すると表示と表示 ・設定操作の重複に備えキーが押されると、不揮発性メモリに記憶し、このとき、一つの給水設定は8バイト構成とする。
キースキャン	・10ミリ秒間隔で起動し、操作パネルのキーをスキャンする。 ・スキャンした結果、キーが押されると判別し大抵は、押されたキーにに対応するキーコードを生成し、タスクに送渡す。 ・前回の生成したキーコードを記憶しており、今回のキーコードが前回と同じ場合は、キーが置かれた場合は、前回のキーコードをクリアする。
給水スケジュール	・1分間隔で起動し、各ユニットの給水時刻と現在時刻を比べ、一致すれば、給水時刻を記憶して、タスクを起動する。
給水弁操作	・ユニットごとに起動され、次の順序で操作を行う。 ①ユニットに設置された給水弁を開いて、給水を開始する。 ②現在時刻と指定された給水時間から給水終了時刻を算出し、給水終了時刻が待ち状態が解除されると、給水弁を閉じて終了する。

イベントドリブン・プリエンプ
ティブ方式で動くタスク群

SOFTWARE LAYER

HARDWARE LAYER



最大4ユニット接続
同時給水は2ユニットまでの制約

システムは「外部からのイベント（キー操作や時間経過）」をトリガーに状態を遷移させる。

[設問3] 発生した事象と追加要件の定義

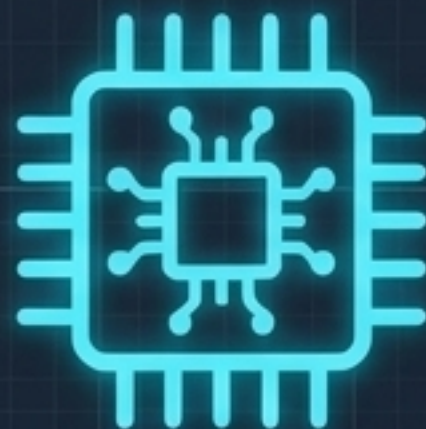
課題：給水中に他の予約時刻になり、連続給水されてしまう。

新要件：同一ユニットの設定済時刻から「60分以内」は設定できないようにブロックする。



スキャンニング：データはどこから来て、どこへ行くのか？

新たなバリデーション（チェック機構）を組み込むには、データの流れを追う必要がある。



Flow 1: **[入力]**
操作パネルでの
キー押下

Flow 2: **[処理]**
設定内容の
組み立てと確定

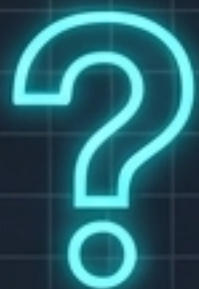
Flow 3: **[保存]**
不揮発性メモリへの
記憶と通知

Flow 4: **[実行]**
スケジュールの
監視と弁の開閉

ソフトウェア設計原則に基づく「最適なチェック箇所」

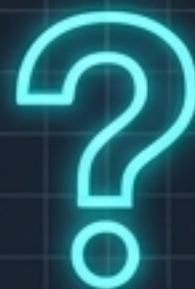
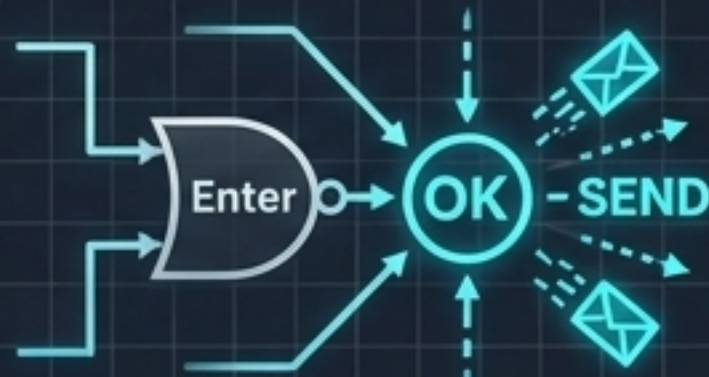
高凝集・疎結合の原則に従い、60分以内の重複エラーをどこで検知するのが最も自然か？
ユーザー体験（UX）とシステム負荷の観点から、**3つのフェーズを検証**する。

入力フェーズ



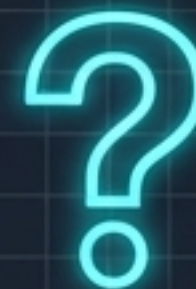
ここで弾くべきか？

確定フェーズ



ここで弾くべきか？

実行フェーズ



ここで弾くべきか？

診断テーブル：各タスクの責務と影響範囲の比較

	本来の責務	既設定データへのアクセス	異常検知時のUX
タスクA（キースキャン関連）	まだデータが完成しておらず、60分以内か判定不能。（×）		
タスクC（スケジュール監視関連）	実行直前で弾くと、ユーザは「設定できたと思っていたのに水が出ない」という最悪のUXになる。（×）		
タスクB（設定処理関連）	「確定操作が行われ、かつメモリに書き込む直前」。データが揃っており、即座にユーザにエラーを通知できる。（○）		

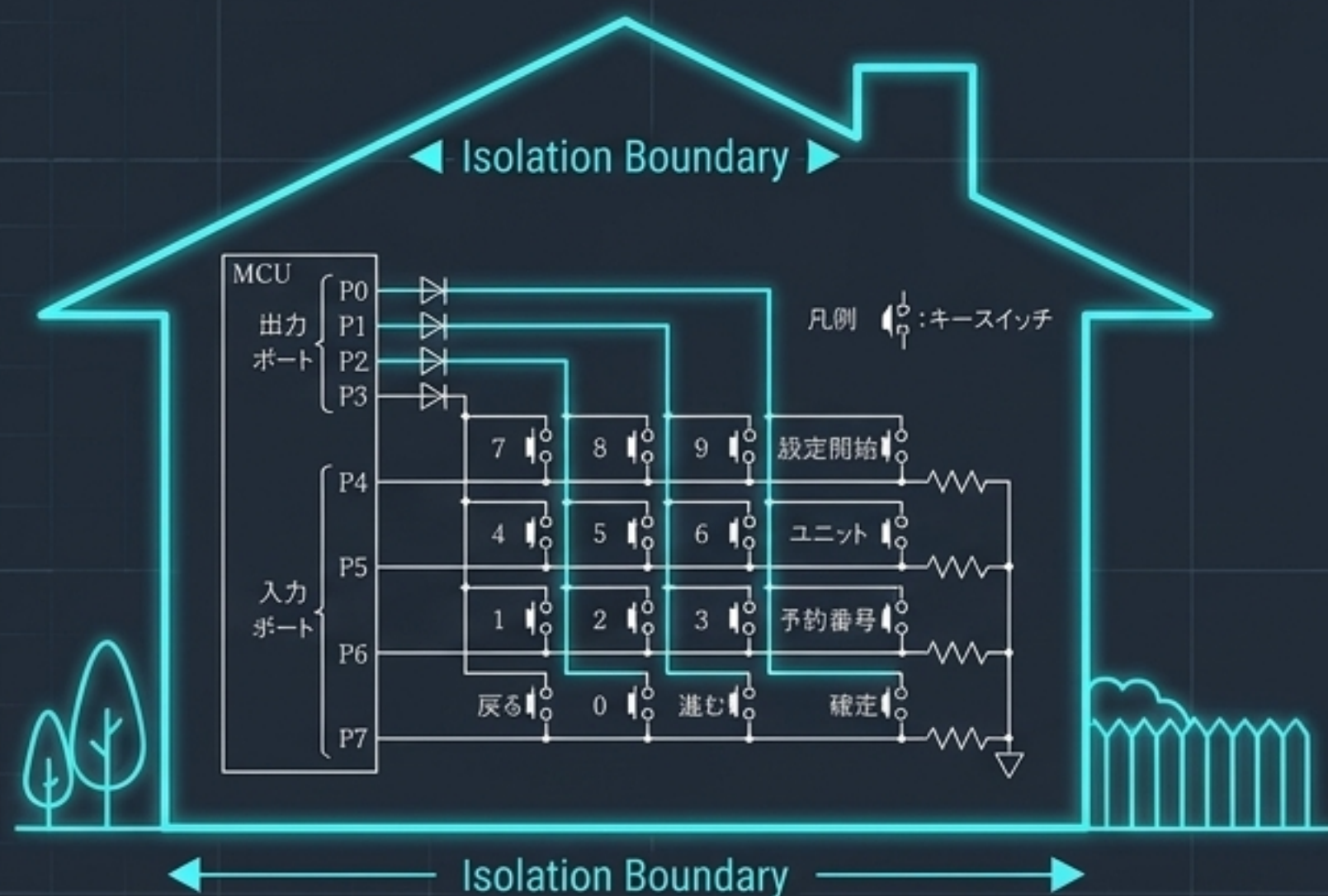
表1（タスクの動作内容）を見直し、「設定操作の最後に確定キーが押されると…」を担当しているタスクを特定せよ。

[設問4] 新たな要件：システムのIoT化と境界の変化

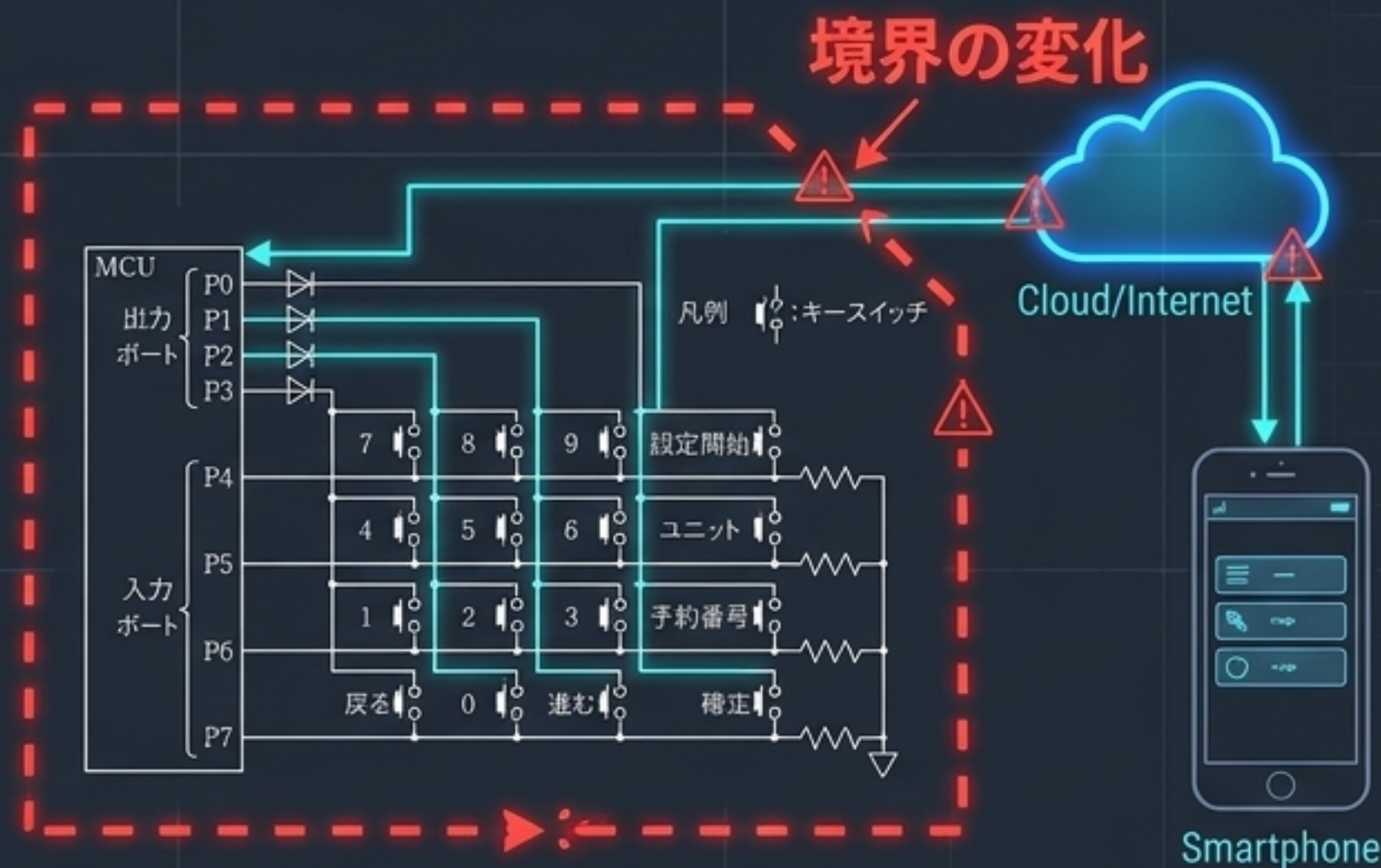
要件変更： インターネット経由での遠隔設定・監視機能の追加。

インパクト： クローズドなローカルシステムから、外部に開かれたIoTシステムへの移行。
物理的な壁がなくなり、新たな脆弱性ベクトルが生じる。

Before: クローズドシステム



After: オープンシステム



スキヤニング：テキストから抽出された2つの「脅威」

問題文から読み解くべき2つのクリティカルなキーワード。

攻撃者は「**誰のフリをするか？**」そして「**何を盗み見るか？**」



なりすましによる不正な操作



通信内容の漏えい

脅威モデリングとセキュリティ要件への変換

脅威A 「なりすまし」



防御方針：お前は本当に正規の
通信相手か？を確認する

なりすまし対策：通信相手が本当に意図した
システム・ユーザーであるかを担保する仕組みが必要。

脅威B 「漏えい」

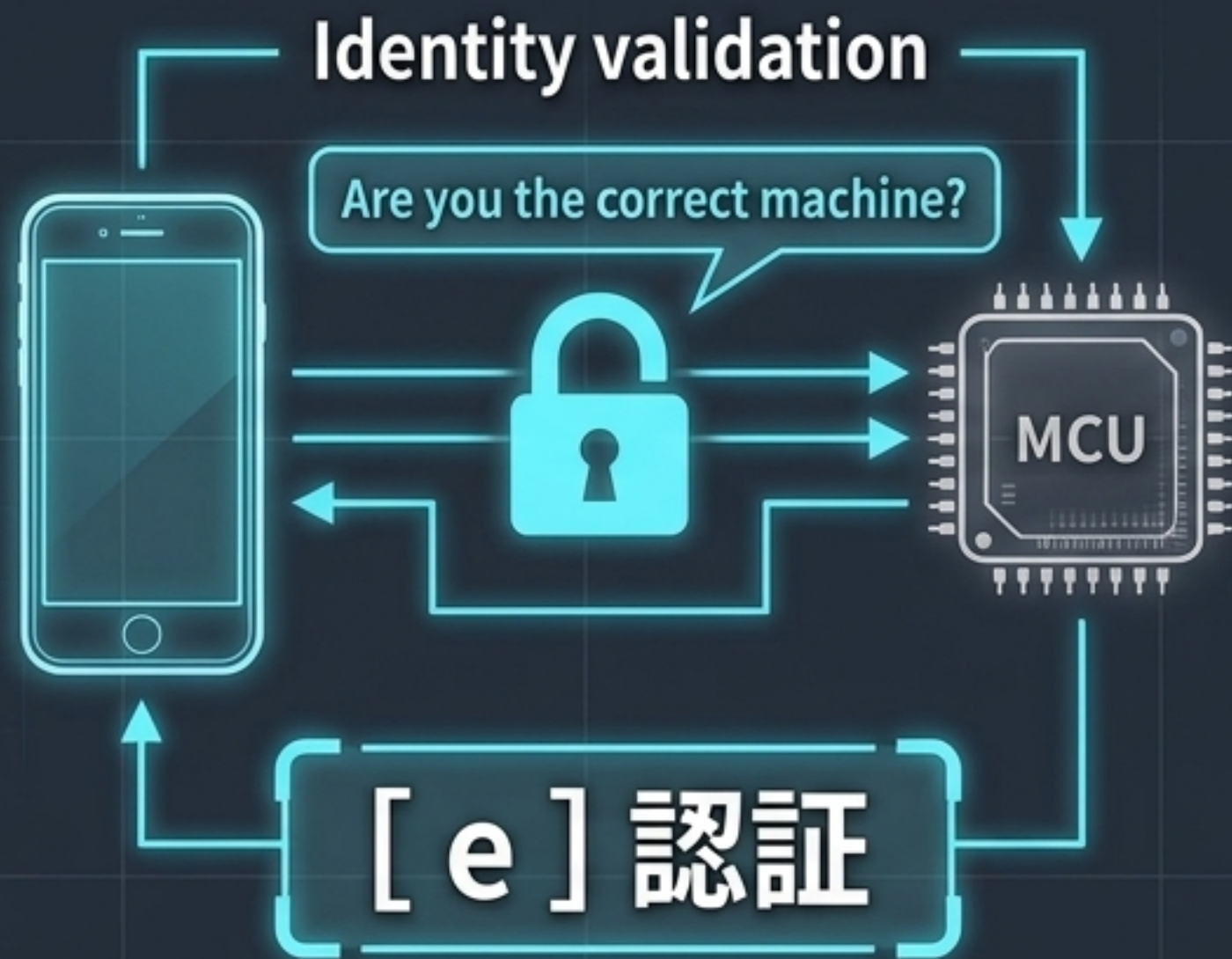


防御方針：データを見られても
意味がわからない状態にする

漏えい対策：万が一経路途中でデータを傍受されても、
中身が解読不能である状態を担保する仕組みが必要。

対策技術の抽象化（解答への架け橋）

一般的な情報セキュリティ用語への変換プロセス。



人間なら生体やID/PWだが、
本件の通信相手は「**機器同士**」である。



平文を特定の鍵なしでは復元できない
文字列に変換する処理。

Synthesis: 午後問題（組み込みシステム）攻略の共通パターン

3. アーキテクチャへのマッピング

既存の構成（図や表）のどこにパズルのピースが当てはまるかを論理的に検証する。

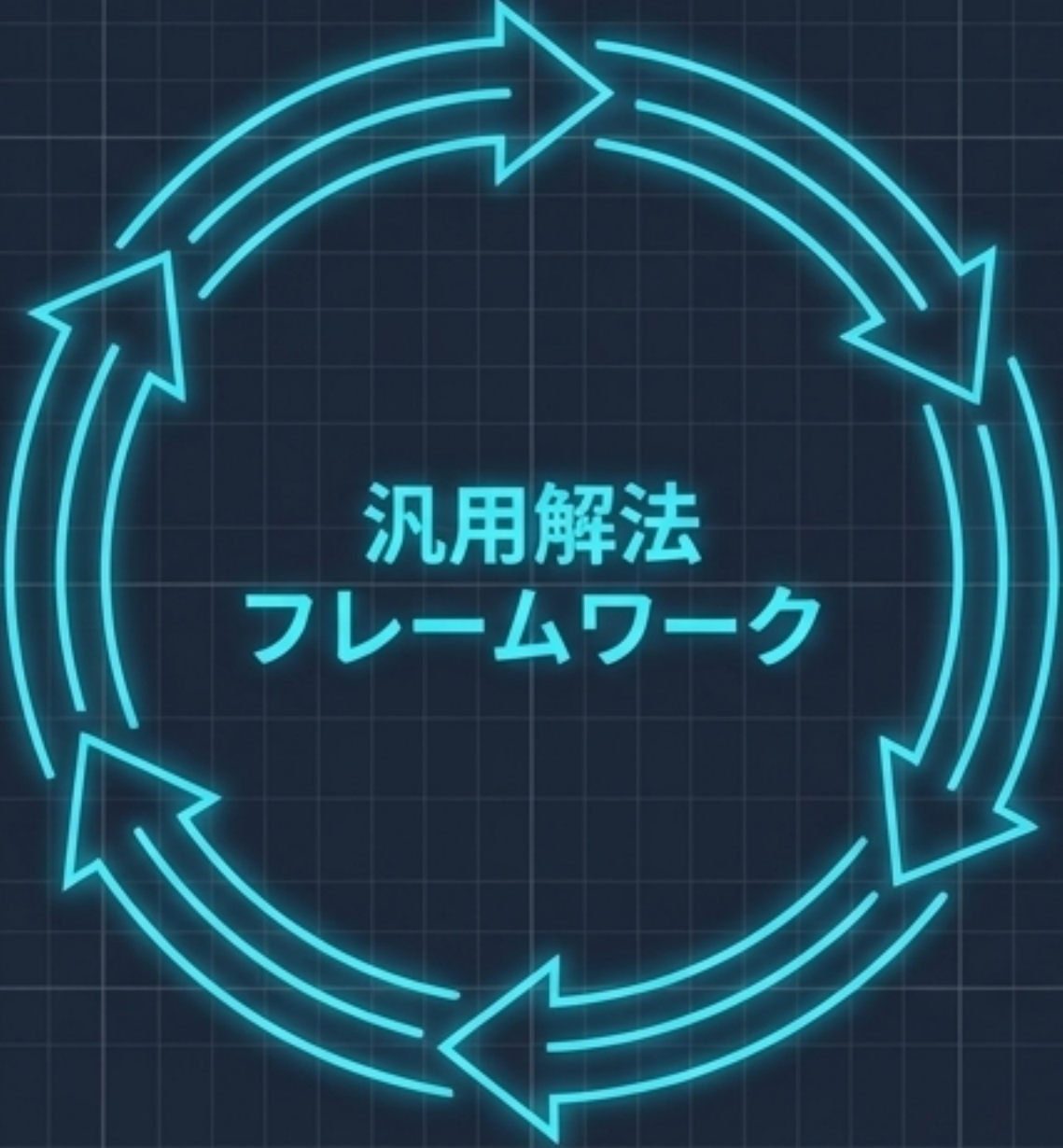
1. 事象のモデル化

問題文のテキストを図やフローに描き起こす。

2. 要求の特定

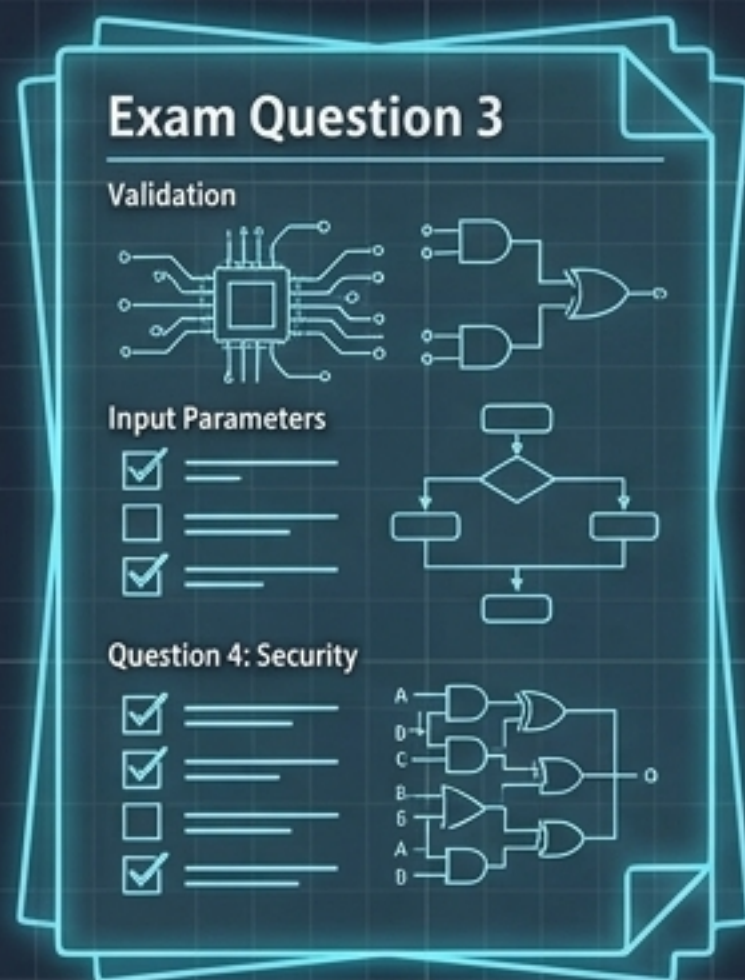
制約条件（時間、データ、アクター）を明確にする。

汎用解法
フレームワーク



知識の暗記ではなく、「仕様書から論理的な帰結を導く」プロセスこそが、本質的な試験の要求スキルである。

実務への接続 (Beyond the Exam)



異常系のバリデーション（設問3）：

ユーザーは常に想定外の操作をする。要件定義の段階で不整合を防ぐ設計（**フェイルセーフ**）は日常的な責務。

セキュリティ・バイ・デザイン（設問4）：

クローズドシステムをWebに繋ぐ際のリスク評価は、現代のIoT開発において最も欠かせないプロセス。



クロージング：答えは暗記するもの ではなく、導き出すもの

本資料のプロセスをなぞり、自分自身の言葉と論理で、
設問3・設問4の解答を仕様書（表1、問題文）から導き出してください。

「思考プロセス」をインストールしたあなたなら、
未知のシステム課題にも必ず正解への道筋を描けるはずです。