Mushroom Automation V2 - Master Project Plan & Roadmap

Document Version: 1.0
Date: 2025-05-22
Time: 12:05 PM

Overall Goal: To transition the entire Mushroom Automation System V2 to a validated Single Source of Truth (SSOT) centered around `system_definition.yaml`, refactor key components (Microcontrollers, Drivers, Governors) for SSOT-driven configuration and improved robustness, and ensure system-wide consistency and maintainability, all guided by ADR-20.

I. Foundational Documents & Principles

- **Primary Guiding Principles:**
  - Project Design and Principles.pdf
- **Core Architectural Specification (SSOT & Topic Generation):**
  - ADR-20 Naming, UUIDs, SSOT, and Information Locus for System Points (1).pdf (Version 2.5, 2025-05-21)
- **Other Key ADRs:**
  - Refer to Architecture Decisions Record ..._all.csv for context, ensuring alignment with ADR-20. (Example: ADR-15 is superseded by ADR-20 regarding MQTT topics).
- **System Overview:**
  - V2 Architecture and Overview.pdf (To be updated as system evolves, but foundational).
  - MQTT Data Flow (1).pdf (Visual aid, ensure topics align with ADR-20).

II. Phased Implementation Plan

Status Legend:

- [ ] `[ ]` To Do
- `[/]` In Progress
- `[X]` Done
- `[B]` Blocked
- `N/A` Not Applicable (for status if task is a heading)

Phase 1: SSOT Definition, Pydantic Models, & `build.py` Core Functionality

Goal: Establish the definitive structure for all configuration YAMLs, the Pydantic models that validate them, and the core `build.py` script capable of validation and initial artifact generation (especially `autogen_config.h`). This phase aligns with "Phase 1" of the "SSOT Definition and Configuration Refactor Plan.pdf" and incorporates ADR-20 requirements.

- **Task 1.1: Finalize `system_definition.yaml` Structure & Pydantic Models**
  - ☐ `[ ]` **Sub-Task 1.1.1:** Finalize `PointDefinition` in `core_ssot_models.py` to include all fields specified in ADR-20, Section II.A (e.g., `uuid`, `name`, `value_type`, `units`, `data_source_layer`, `access`, `writable_by`, `persist_to_db`, `readback_point_uuid`, `function_grouping`, `topic_device_slug`, `topic_originator_slug`, `topic_directive_slug`, `topic_status_slug`).
    - *Current Status Notes:* Partially complete. `core_ssot_models.py` defines `PointDefinition` with many fields. ADR-20 specific topic fields (`function_grouping`, `topic_device_slug`, etc.) are **MISSING** and need to be added.
  - ☐ `[ ]` **Sub-Task 1.1.2:** Finalize `ComponentDefinition` variants (`MicrocontrollerComponentDefinition`, `DriverComponentDefinition`, `GovernorComponentDefinition`, `ManualSourceComponentDefinition`) in `core_ssot_models.py` as per ADR-20, Section 1 (Core Principles) and ensure fields like `points_provided`, `virtual_points_provided`, `controls_microcontroller`, `controls_drivers` are correctly defined.
    - *Current Status Notes:* Partially complete. `core_ssot_models.py` has these models. `ManualSourceComponentDefinition` is present. Ensure full alignment with ADR-20 regarding how components link to points they source and control relationships for topic generation.
  - `[X]` **Sub-Task 1.1.3:** Finalize `SystemDefinition` in `core_ssot_models.py` to include `mqtt_broker`, `command_hierarchy`, `points`, and `components` list.
    - *Current Status Notes:* Appears largely complete in `core_ssot_models.py`.
  - ☐ `[ ]` **Sub-Task 1.1.4:** Finalize `MicrocontrollerConfig` Pydantic model in `component_configs.py` ensuring it aligns with ADR-20, Section II.B.1 (i.e., **NO** `mqtt_topic_suffix_*` fields, focus on local name, `point_kind`, pin details, UUID refs: `write_point_uuid_ref`, `readback_point_uuid_ref`, `data_point_uuid_ref`).
    - *Current Status Notes:* `MicrocontrollerConfig` in `component_configs.py` is defined but needs review to ensure removal/absence of direct topic suffix fields and that `point_kind` aligns with `function_grouping` from `PointDefinition`. It currently includes `i2c`, `onewire`, `dht_sensors`, `digital_outputs`.
  - ☐ `[ ]` **Sub-Task 1.1.5:** Finalize `DriverConfig` Pydantic model in `component_configs.py` ensuring it aligns with ADR-20, Section II.B.2 (minimal/no explicit topic suffixes, focus on FSM structure, input/output UUIDs).
    - *Current Status Notes:* `DriverConfig` is defined with `initial_state`, `states`, `transitions`, `pwm_outputs`. Review to ensure topic-related fields are minimized as per ADR-20.
  - ☐ `[ ]` **Sub-Task 1.1.6:** Finalize `GovernorConfig` Pydantic model in `component_configs.py` ensuring it aligns with ADR-20, Section II.B.3 (minimal/no explicit topic suffixes).
    - *Current Status Notes:* `GovernorConfig` is defined with `update_interval_seconds` and a list of `controllers` (PID, BangBang, TimeSchedule). Review for topic fields.

    - ☐ `[ ]` **Sub-Task 1.1.7:** Create/Update sample YAML files for `system_definition.yaml` and each Component Type, demonstrating all ADR-20 compliant features and Pydantic model structures.
        - *Current Status Notes:* Some example YAMLs exist (e.g., `system_definition_example.yaml`, various component configs in `control/config/`). These will need updating to reflect the finalized Pydantic models and ADR-20.
- **Task 1.2: Implement `build.py` Core Validation & Artifact Generation**
    - `[X]` **Sub-Task 1.2.1:** `build.py`: Stage 1 - `system_definition.yaml` structural validation using Pydantic models from `core_ssot_models.py`.
        - *Current Status Notes:* Implemented in `build.py` (`validate_ssot` function).
    - `[X]` **Sub-Task 1.2.2:** `build.py`: Stage 2 - Component-Specific YAML validation using Pydantic models from `component_configs.py` and `COMPONENT_MODEL_MAP`.
        - *Current Status Notes:* Implemented in `build.py` (`validate_component_configs`). Context passing for cross-referential validation within models is present.
    - `[/]` **Sub-Task 1.2.3:** `build.py`: Stage 3 - Cross-Validation Logic (SSOT integrity, UUID references, component ID references, `writable_by` checks, point provisioning, command/readback linkages) as outlined in ADR-20 and `build.py`'s `cross_validate_configs` function.
        - *Current Status Notes:* Partially implemented in `build.py` `cross_validate_configs`. It checks UUID references, component ID refs, `writable_by` (ADR-20 clarifies `writable_by` is for documentation/auth, command topic targets are derived), point provisioning, command/readback links, orphaned/multiply-provided points. Needs alignment with ADR-20's derivation logic for command targets.
    - ☐ `[ ]` **Sub-Task 1.2.4:** `build.py`: Generation of `global_point_registry.json`. This registry should include, for each point: UUID, all master attributes from `PointDefinition` (ADR-20 Sec II.A), `source_component_id`, and the **fully derived MQTT topics** (ADR-20 Sec II.C).
        - *Current Status Notes:* Not yet implemented in `build.py`.
    - ☐ `[ ]` **Sub-Task 1.2.5:** `build.py`: Generation of `autogen_config.h` for each microcontroller. This includes DEVICE_ID, WiFi/MQTT/NTP configs, and for each hardware point: `POINT_NAME_`, `UUID_`, `TOPIC_` (fully derived per ADR-20 Sec II.C), `PIN_`, `MODE_`, `INITIAL_STATE_`.
        - *Current Status Notes:* Not yet implemented in `build.py`.
- **Task 1.3: Configuration Data Migration (from old formats to new SSOT YAMLs)**
    - (Ref: "SSOT Definition and Configuration Refactor Plan.pdf", Phase 2, Tasks 2.1-2.6)
    - ☐ `[ ]` **Sub-Task 1.3.1:** Analyze all old config sources (`microc_points.json`, `uuid_db.json`, driver `settings.json`, `transitions.json`, `states.json`, hardcoded values) and define a detailed migration mapping to the new ADR-20 compliant Pydantic schemas.
    - ☐ `[ ]` **Sub-Task 1.3.2:** Implement Point Migration to `system_definition.yaml` (assign new

UUIDs if needed, define all ADR-20 attributes).

☐ `[ ]` **Sub-Task 1.3.3:** Implement Component Migration to `system_definition.yaml` (define `id`, `type`, `config_file`, `points_provided` / `virtual_points_provided`, control relationships).

☐ `[ ]` **Sub-Task 1.3.4:** Migrate Microcontroller Hardware Configs to Component YAMLs (e.g., `c1_config.yaml`) as per `MicrocontrollerConfig` model.

☐ `[ ]` **Sub-Task 1.3.5:** Migrate Driver FSM Configs to Component YAMLs as per `DriverConfig` model.

☐ `[ ]` **Sub-Task 1.3.6:** Migrate Governor Logic Configs to Component YAMLs as per `GovernorConfig` model.

☐ `[ ]` **Sub-Task 1.3.7:** Perform full validation of all migrated YAML files using `build.py`. Resolve all errors. Manually review generated artifacts (once `build.py` generates them) for correctness.

Phase 2: Microcontroller Common Firmware Library Development

Goal: Develop a shared, reusable C++ library for common microcontroller functionalities, using the generated `autogen_config.h` for configuration. (Ref: "Comprehensive Microcontroller Refactor Plan", Phase 1 & P0 tasks)

- `[X]` **Task 2.1: Define Core FSM States & Restart Reason Enums** (uC Plan P0.1, P0.2)
- `[X]` **Task 2.2: Define `RestartReasonLogger` Interface & EEPROM Strategy** (uC Plan P0.3)
- `[X]` **Task 2.3: Define `MonitoredPublishPoint` Base Class Interface** (uC Plan P0.5)
- `[X]` **Task 2.4: Develop `JsonBuilder` Service (ADR-10 Payloads)** (uC Plan P1.1)
- `[X]` **Task 2.5: Develop `NtpService`** (uC Plan P1.2)
- `[X]` **Task 2.6: Develop `MqttService` (including LWT, command queueing as per ADR-16)** (uC Plan P1.3)

☐ `[ ]` **Task 2.7: Develop `FsmUtils` (stateToString, transitionToState, checkTimeout)** (uC Plan P1.4)

☐ `[ ]` **Task 2.8: Implement `RestartReasonLogger.h/.cpp` in common library** (uC Plan P4.1)

Phase 3: Microcontroller Individual Refactors (C1, C2, C3)

Goal: Refactor firmware for each microcontroller (C1, C2, C3) to use the common firmware library and be configured by its specific `autogen_config.h` file generated by `build.py`. (Ref: "Comprehensive Microcontroller Refactor Plan")

- **Pre-requisite for each controller's refactor:** `autogen_config.h` for the specific controller is successfully and correctly generated by `build.py` as per Task 1.2.5.
- **Task 3.1: Refactor Controller C2 (Actuators)**
    - (Ref: uC Plan P2.C2.1 - P2.C2.5)

☐ `[ ]` Sub-Tasks: Define C2 config in SSOT YAML (Task 1.3.4), integrate `autogen_config.h`, implement main.cpp with FSM & common libs, ADR-10 payloads, actuator data wrappers (if applicable for readbacks), integrate RestartReasonLogger & no-publish timeout, test thoroughly.

- **Task 3.2: Refactor Controller C1 (Sensors)**
  - (Ref: uC Plan P3.C1.1 - P3.C1.5)
  - ☐ `[ ]` Sub-Tasks: Define C1 config in SSOT YAML (Task 1.3.4), integrate `autogen_config.h`, implement main.cpp with FSM & common libs, ADR-10 payloads, sensor data wrappers, integrate RestartReasonLogger & no-publish timeout, test thoroughly.
- **Task 3.3: Refactor Controller C3 (SCD41 Sensor)**
  - (Ref: uC Plan P5.C3.1 - P5.C3.6)
  - ☐ `[ ]` Sub-Tasks: Define C3 config in SSOT YAML (Task 1.3.4), integrate `autogen_config.h`, implement main.cpp with FSM & common libs, SCD41 logic, ADR-10 payloads, SCD41 data wrappers, integrate RestartReasonLogger & no-publish timeout, test thoroughly.

Phase 4: Python Component (Driver, Governor, Data Processor) Runtime Refactoring

Goal: Refactor/develop Python-based runtime components to load and use their configurations from the SSOT YAMLs and interact via MQTT according to ADR-20 defined topics and ADR-10 payloads.

- **Task 4.1: Refactor Driver Runtime Code**
  - (Ref: "SSOT Implementation Plan & Next Steps (3).pdf", Task C)
  - ☐ `[ ]` Sub-Tasks: Implement loading of validated `DriverConfig`, refactor/implement FSM engine, state handling, action execution (targeting ADR-20 derived topics for uC commands), PWM cycle logic, state/health/command reporting (to ADR-20 derived topics), `time_in_state` publishing, update constraint evaluation.
- **Task 4.2: Refactor/Develop Governor Runtime Code**
  - (Ref: "SSOT Implementation Plan & Next Steps (3).pdf", Task D)
  - ☐ `[ ]` Sub-Tasks: Implement loading of validated `GovernorConfig`, implement PID/BangBang/TimeSchedule control logic, publish Mode Commands & Durations (to ADR-20 derived topics for Driver commands), monitor Driver state/health.
- **Task 4.3: Implement Data Processing Layer (Basic)**
  - (Ref: "Project Design and Principles.pdf" - Key Objective)
  - ☐ `[ ]` Sub-Tasks: Define `DataProcessorConfig` Pydantic model, implement service to consume raw sensor data (from ADR-20 derived topics), perform basic filtering/validation, publish processed/synthetic points (to ADR-20 derived topics).

Phase 5: System Integration, Testing & Documentation Finalization

Goal: Ensure all components integrate correctly, the system is robustly tested, and all documentation is updated and finalized.

- [ ] `[ ]` **Task 5.1: Code Review All Components & Shared Libraries** (uC Plan P6.1)
- [ ] `[ ]` **Task 5.2: System-Level Integration Testing** (uC Plan P6.2)
    - Verify data flow across all layers (uC -> Data Processor -> Driver -> Governor and vice-versa for commands) using ADR-20 topics and ADR-10 payloads.
    - Test operational scenarios and induced error conditions.
- [ ] `[ ]` **Task 5.3: Update Telegraf/Grafana Configurations**
    - (Ref: "SSOT Implementation Plan & Next Steps (3).pdf", Task F)
    - Adjust Telegraf to subscribe to ADR-20 derived topics and parse ADR-10 JSON payloads.
    - Update Grafana dashboards.
    - (Consider `build.py` generating Telegraf snippets - SSOT Plan Task 3.6, "Next Steps" Task I).
- [ ] `[ ]` **Task 5.4: Update Deployment Infrastructure**
    - (Ref: "SSOT Implementation Plan & Next Steps (3).pdf", Task G)
    - Modify Docker Compose/deployment scripts.
    - Implement mechanism to pass component ID and config path info.
- [ ] `[ ]` **Task 5.5: Finalize All Documentation**
    - (Ref: uC Plan P6.3, SSOT Plan Task 4.3)
    - Ensure ADR-20 is considered final for this phase.
    - Update/finalize "Project Design and Principles.pdf", "V2 Architecture and Overview.pdf".
    - Document the structure of all YAMLs and Pydantic models.
    - Document `build.py` usage, stages, and generated artifacts.
    - Ensure READMEs are comprehensive and up-to-date.

III. Document Management

- **This Document (Master Project Plan & Roadmap):**
    - **Lifecycle:** This is a living document. Tasks will be marked as complete, and new sub-tasks or phases may be added/adjusted as the project progresses.
    - **Updates:** Version number and date/time should be updated with each significant modification.
- **Kept & Updated Documents:**
    - `Project Design and Principles.pdf`: Update if core principles evolve (unlikely to change frequently).
    - `ADR-20 ... .pdf` (and other ADRs in the CSV): ADRs are generally immutable once "Accepted". New ADRs are created for new decisions or to supersede old ones. You will update the CSV status (e.g., "Superseded by ADR-XX").
    - `V2 Architecture and Overview.pdf`: Update as major architectural components are implemented or refined.

    - `system_definition.yaml` (and component config YAMLs): These are the core configuration files and will be actively developed and version-controlled (e.g., via Git).
    - Pydantic Models (`core_ssot_models.py`, `component_configs.py`): Will be updated to match ADR-20 and then remain stable for this phase. Version control via Git.
    - `build.py`: Will be actively developed and version-controlled.
- **Archived/Superseded (for reference, but not active planning):**
    - "SSOT Definition and Configuration Refactor Plan.pdf": Its tasks are absorbed into Phase 1 of this Master Plan. Keep for historical context.
    - "Comprehensive Microcontroller Refactor Plan (2).pdf": Its tasks are absorbed into Phases 2 & 3 of this Master Plan. Keep for historical context.
    - "Mushroom Automation System V2 - SSOT Implementation Plan & Next Steps (3).pdf": Its relevant tasks/decisions are incorporated here or into ADR-20. Keep for historical context.