# Software Architecture (SWE2)

# Revisions

| Version | Author | Description | Date |
|---|---|---|---|
| *v1.0.0* | *Hairu Mossa* |  | *28/09/2025* |
|  |  |  |  |
|  |  |  |  |

# Document Approvals

| Role | Name | Signature | Date |
|---|---|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

# 1. Introduction

This document describes the hardware and software architecture for the Buck Converter Embedded Control System. The architecture has been designed with layered abstraction, state management, and real-time control principles to ensure modularity, maintainability, and reliability.

The document includes:

- Hardware architecture overview.
- Software layered architecture.
- System states and transitions.
- Behavioral sequence diagrams for control and telemetry flows.

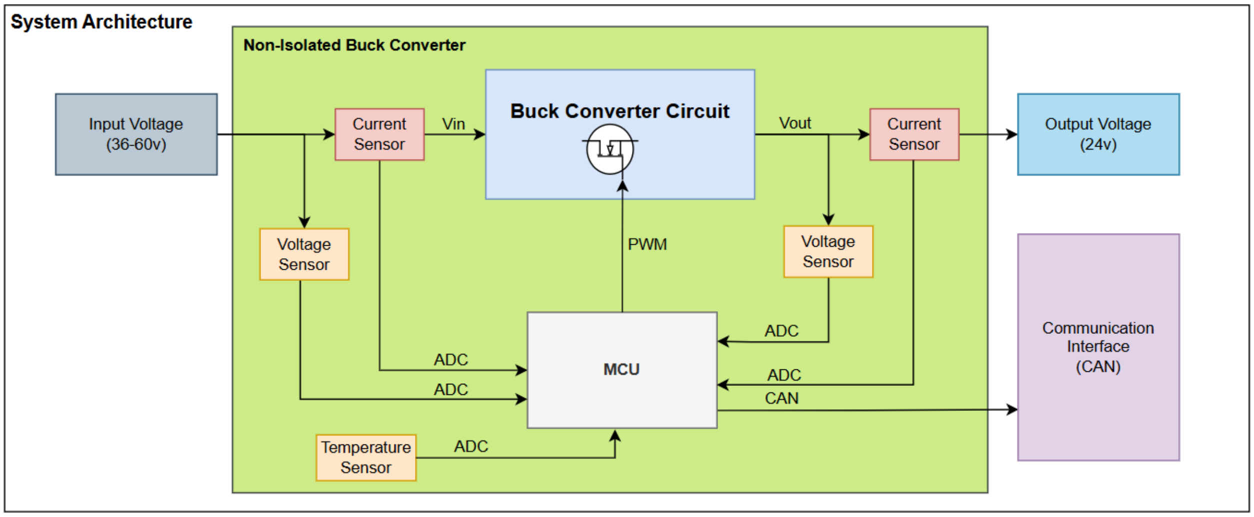
# 2. Hardware Architecture

## 2.1 Description

The hardware platform is based on an STM32F103/405 microcontroller with the following peripherals:

- ADC (Analog-to-Digital Converter)
  - Measures input voltage, output voltage, output current, and system temperature.
- PWM Timer
  - Generates 20 kHz switching signal for buck converter MOSFET.
- CAN Controller
  - Provides telemetry and error reporting to external systems.
- System Timers
  - 10 kHz (control loop).
  - 10 Hz (telemetry loop).
- Temperature Sensor
  - For thermal protection and monitoring.
- Power Stage
  - Buck converter with MOSFET driver and passive elements (inductor, capacitor).

## 2.2 Hardware Architecture Diagram



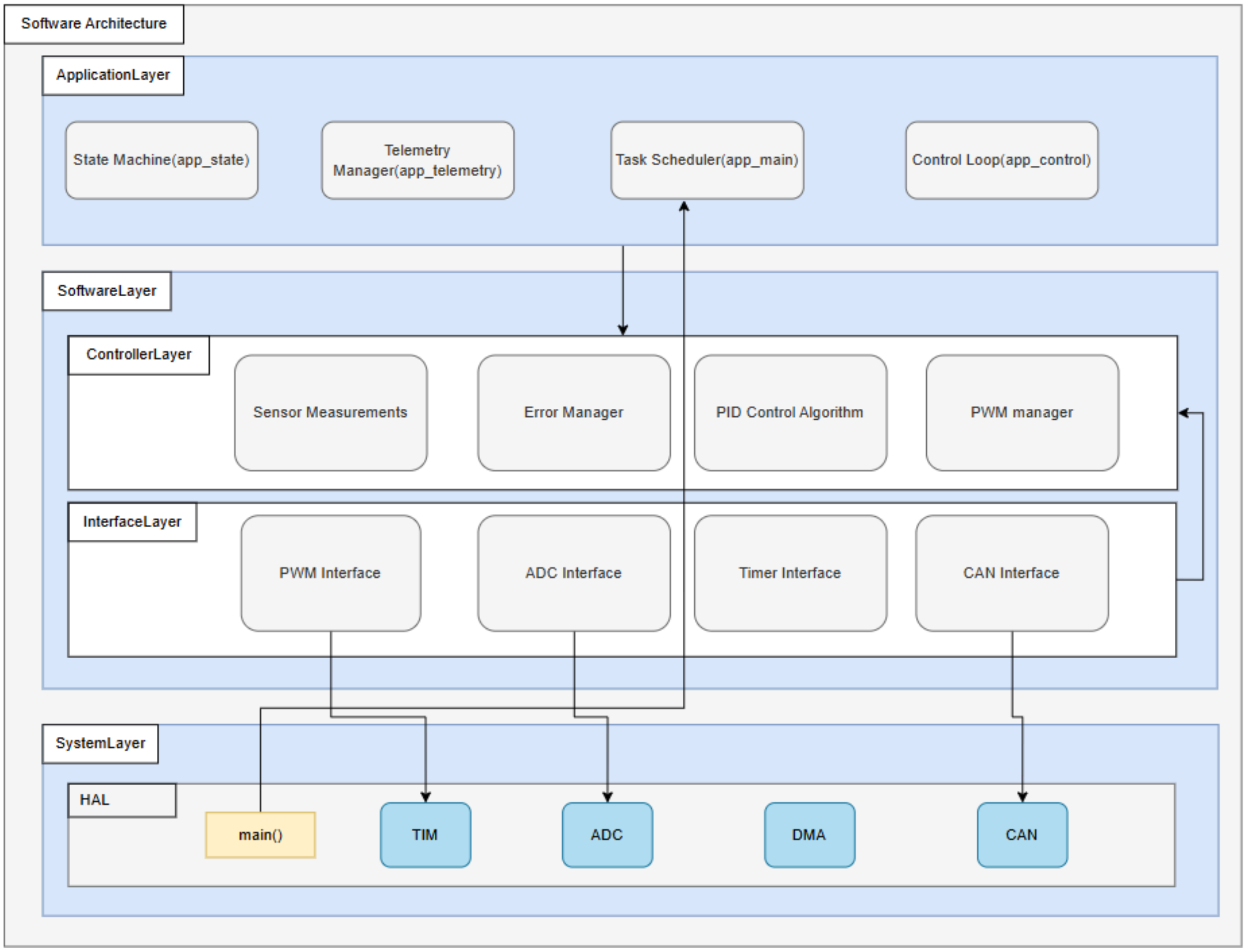
# 3. System (Software) Architecture

## 3.1 Layered Architecture Description

The software is organized into **five layers**

1. **Application Layer**
   - Orchestrates overall control, telemetry, and error handling.
   - Includes `app_main`, `app_control`, `app_telemetry`, `app_state`.
2. **Controller Layer**
   - Implements algorithms (PID, limiters, error handling).
   - Includes `pid_controller`, `adc_controller`, `pwm_controller`, `error_controller`.
3. **Interface Layer**
   - Abstracts hardware peripherals with clean APIs.
   - Includes `if_adc`, `if_pwm`, `if_can`, `if_timer`, `if_temp`.
4. **System Layer**
   - HAL/LL drivers generated by CubeMX.
   - Direct MCU register access (ADC, CAN, TIM, GPIO).
5. **Hardware Layer**
   - Physical microcontroller, sensors, and power stage.

## 3.1 System Architecture Diagram (Layered)

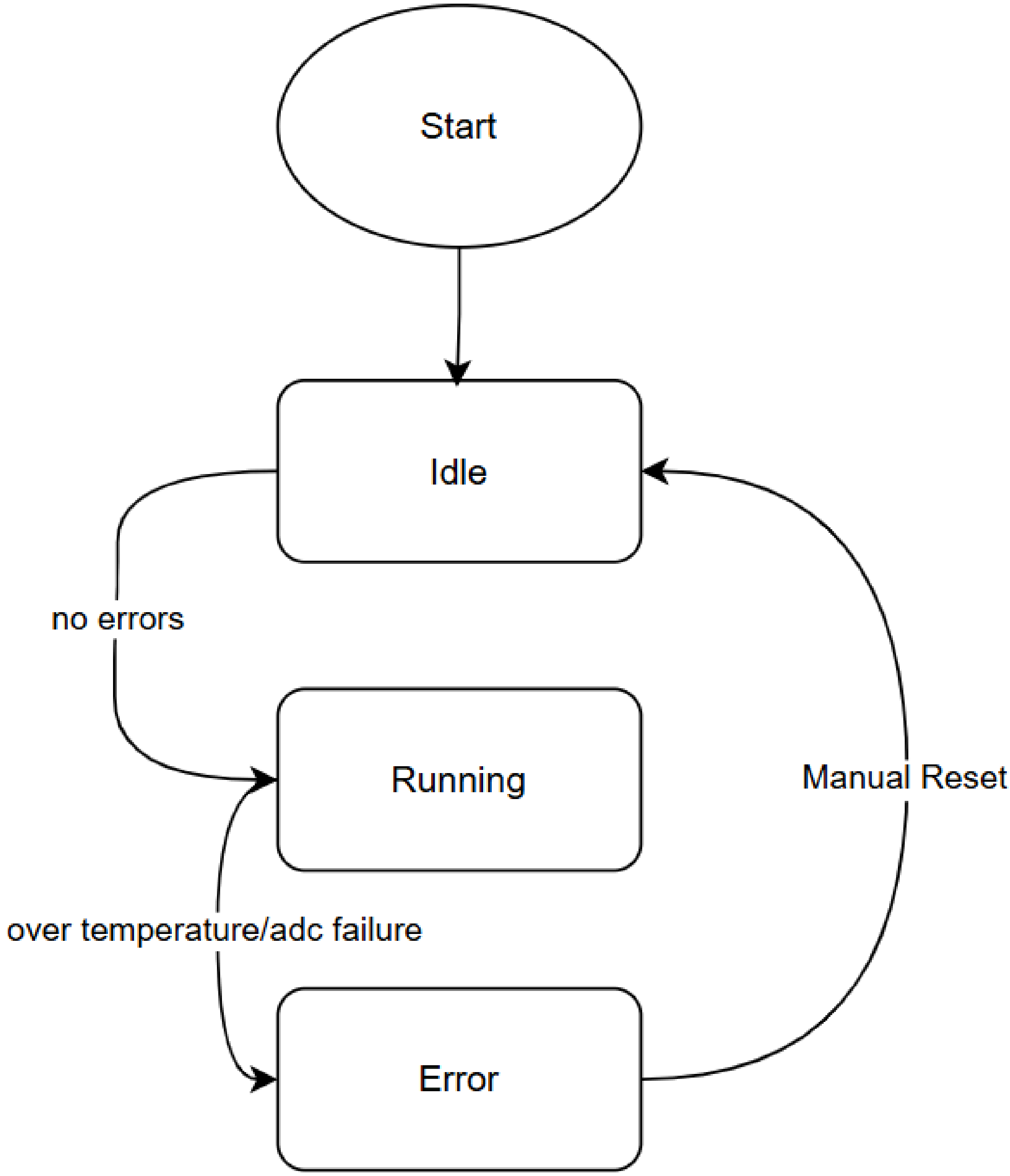


# 4. System States

## 4.1 State Description

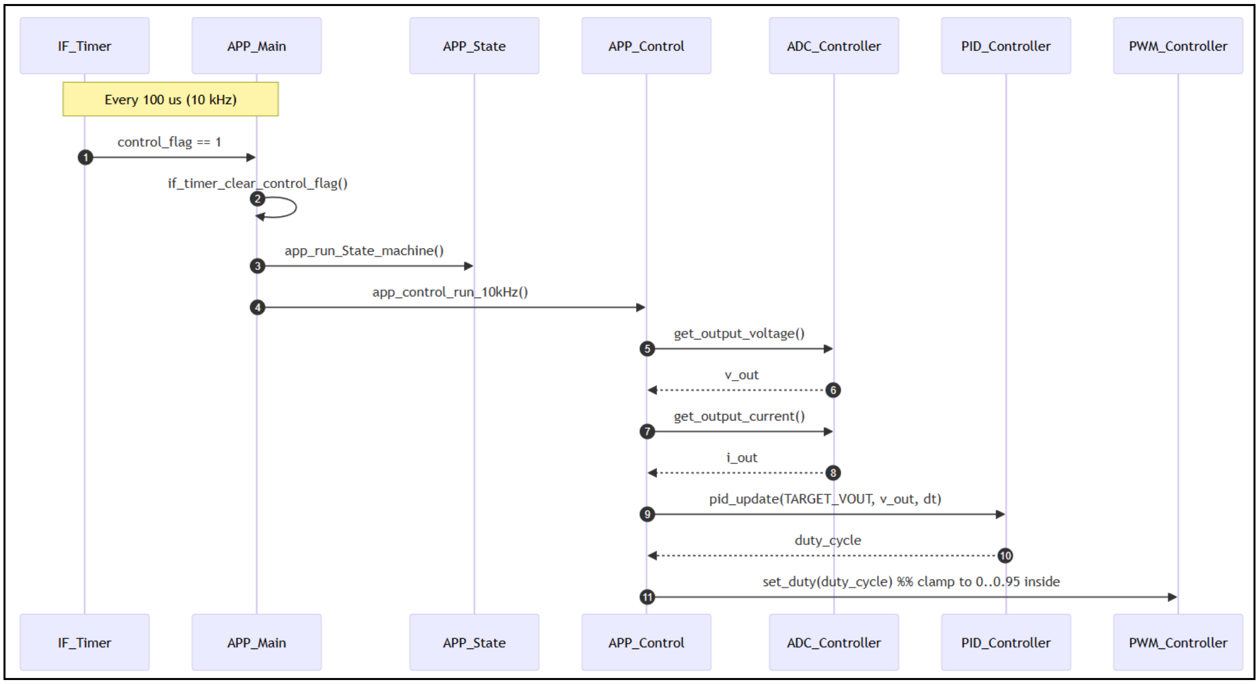
**IDLE:** System powered but PWM disabled. Waits for start condition.
**RUNNING:** System actively regulates output voltage, runs control loop at 10 kHz, sends telemetry at 10 Hz.
**ERROR:** Entered when fatal errors (ADC failure, overtemperature) occur. PWM disabled, error broadcast persists.

## 4.2 State Diagram

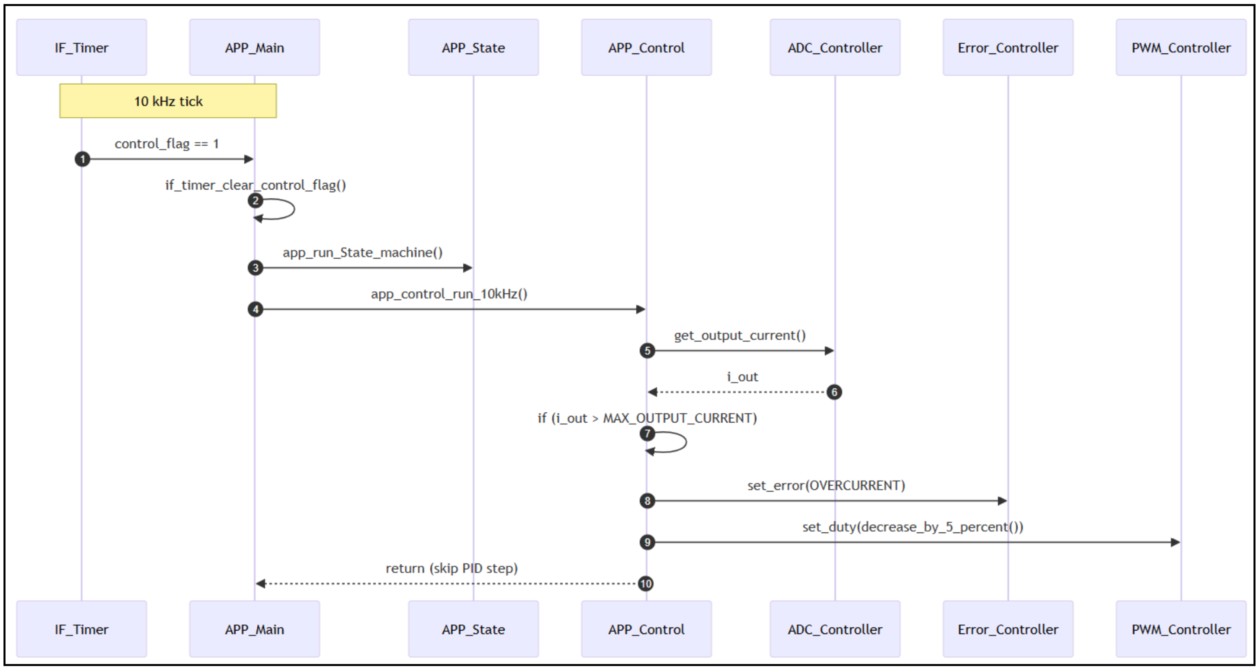


# 5. Behavioral Sequence Diagrams

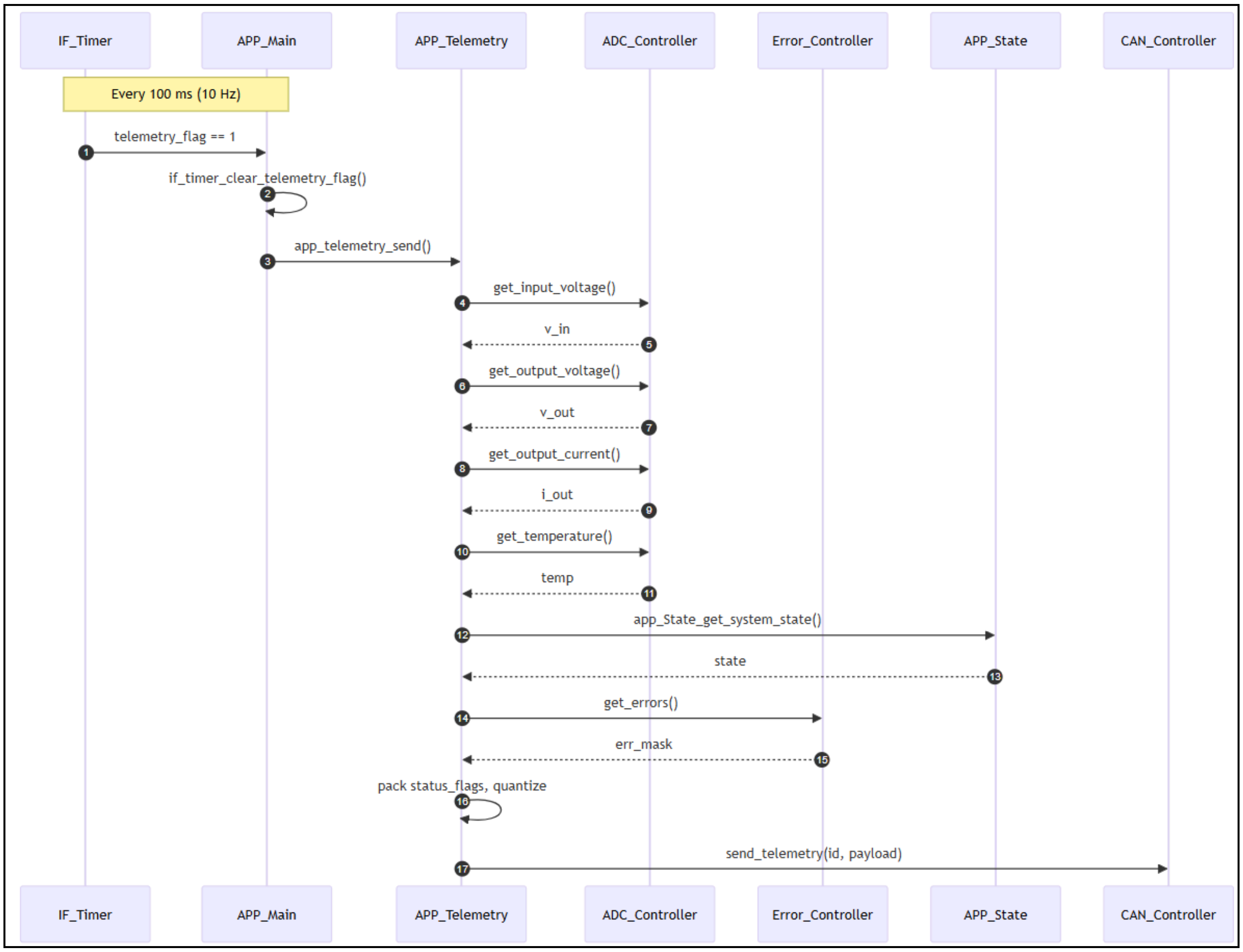
## 5.1 Control Loop (10 kHz) Normal Path



# 5.2 Control Loop (10 kHz) Overcurrent Path



## 5.3 Telemetry Loop (10 Hz)



# 6. Traceability

This architecture supports the requirements defined in the **System Requirements (SyRS)** and **Software Requirements (SRS)**:

- **Voltage regulation (SYS_0002 → SWR_0002)** handled in **Controller Layer (PID)**.
- **Current limiting (SYS_0004 → SWR_0004)** enforced in **Controller + Error Controller**.
- **Telemetry (SYS_0005 → SWR_0005)** implemented by **Application → CAN Controller**.
- **State transitions (SYS_0016–0019 → SWR_0011, SWR_0012)** implemented by **Application State Machine**.