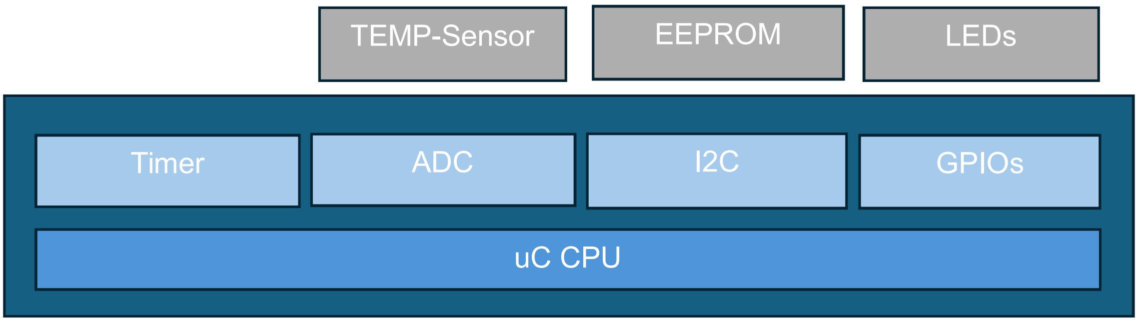
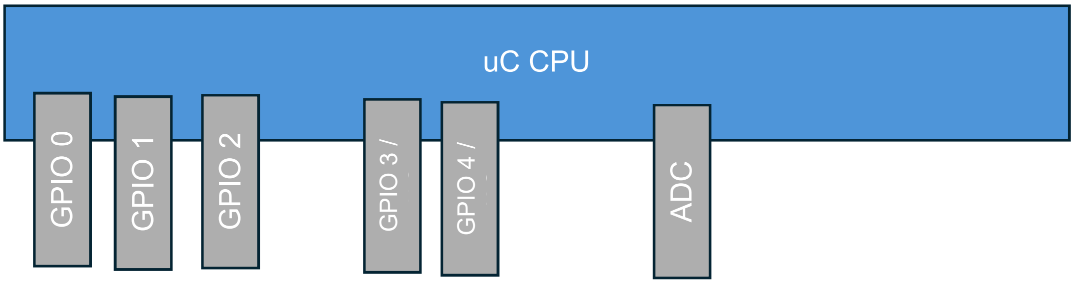
Architecture Approach:

1. Block diagram:



2. uC pinout



3. Module descriptions / Notes

TEMP (sensor):

- temp sensor provides analog signal which is translated into digital value via ADC
- output voltage of temperature sensor is matched to the accepted input voltage of the ADC (voltage devider /OP amp)

ADC:

- integrated into uC
- supports at least 10kHz sample rate (sample every 100 us)
- **clock/timer based sample frequency** to sample regularly -> to be configured (clock-based for low jitter)
- **ISR to be triggered after sampling and quantization of the temp sensor signal is done**

- sample depth / resolution:

  Temp range needed: <5°C….>105°C

  RevA - 8 bit would be sufficient to cover the temp range in 1°C steps

  RevB - 10 bit necessary to cover temp range in 0.1°C steps

⇨ to be configured in ADC register

RevA - 4°C at digital value 0 (temp range 4°C…259°C / 256 steps)

RevB - 4°C at digital value 0 (temp range 4.0°C...106.3°C / 1024 steps)

⇨ Dimensions are chosen to fit the use case, a bigger range might be expected in reality
⇨ Same starting temperature to avoid additional offset to be configured

- ADC register addresses (assumption): 0x30

ADC Config Register								
0x30	N/A (Bit7)	N/A	N/A	Resolution	Trigger source	ISR flag clear	ISR enable	Enable/Disable (Bit0)

- Resolution: 1 – 10 bit, 0 – 8 bit
- Trigger source: 0 – Timer Clk0 (internal) , 1 – external Timer
- No focus on conversion mode, clock scaling and other potential ADC parameters

ADC Value Register High								
0x31	(Bit7)							(Bit0)

ADC Value Register Low								
0x32	(Bit7)							(Bit0)

Clk0:

Clk0 Config Register								
0x40	N/A (Bit7)	N/A	N/A	N/A	N/A	Clock Frequency	Clock Frequency	Enable/Disable (Bit0)

- Clock frequency: 0b00 – 10 kHz , 0b01 – 20kHz, …
⇨ No clock divider considered to keep it simple

GPIOs (for LEDs):

- LED turned on when GPIO high

GPIO Register								
0x50	N/A (Bit7)	N/A	FUNC SELECT	Read (High/Low)	RESET	SET	Input/Output	Enable/Disable GPIO (Bit0)

Assumptions:

- 1 bit for FUNCTION select (1 - I2C, 0-GPIO)

- GPIO_0 register-> Red (Address: 0x50)
- GPIO_1 register -> Yellow (Address: 0x51)
- GPIO_2 register -> Green (Address: 0x52)

EEPROM:

- hardware abstraction for i2c interface
- storage addressing: 1 Byte
- storage size: 256 Byte (0x00...0xFF)

- external via I2C
- chosen I2C address: 0x80 + Write(0) / Read(1)
- I2C buffer size: 1 Byte
- HW serial read through iteration (no sequential read implemented)

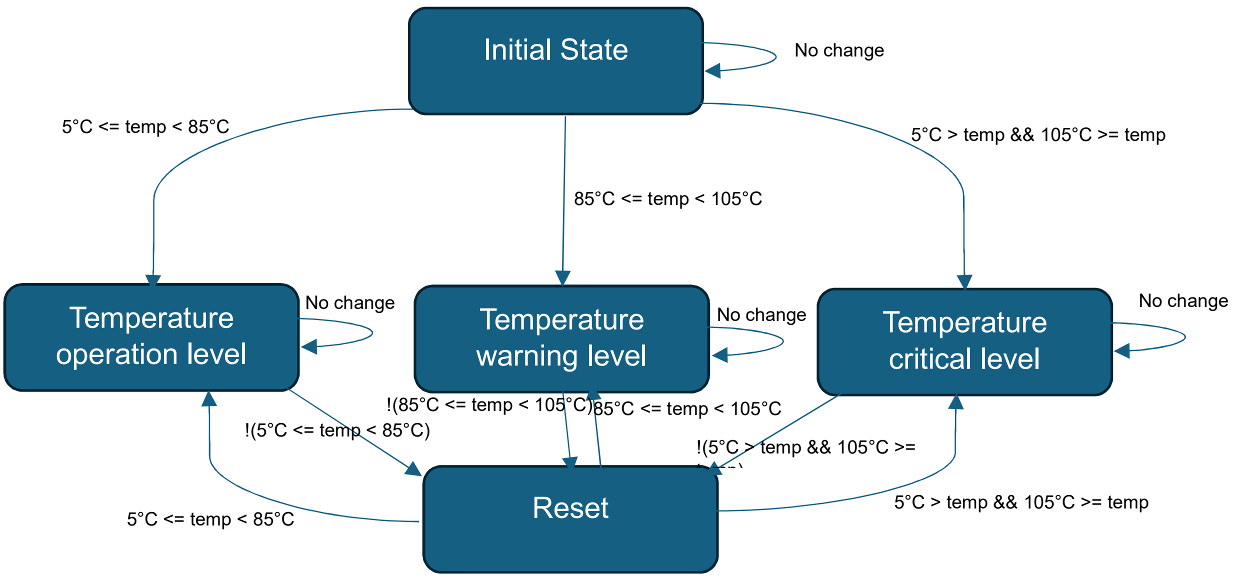
I2C Config Register								
0x61	(Bit7)	Address	Address	Address	Address	Address	Address	Read/Write GPIO (Bit0)

I2C Address Register								
0x61	(Bit7)	Address	Address	Address	Address	Address	Address	Read/Write GPIO (Bit0)

Fragmentation:

- Byte 0: Hardware revision
- Byte 1-8: Hardware serial number

Statemachine:



Initial State:

- Read revision from EEPROM
- apply revision specific configuration for GPIOs, Serial and ADC
- wait for first feedback/interrupt from ADC

Temperature operation level:

- set GPIO2/LED2 to High
- wait for transition

Temperature warning level:

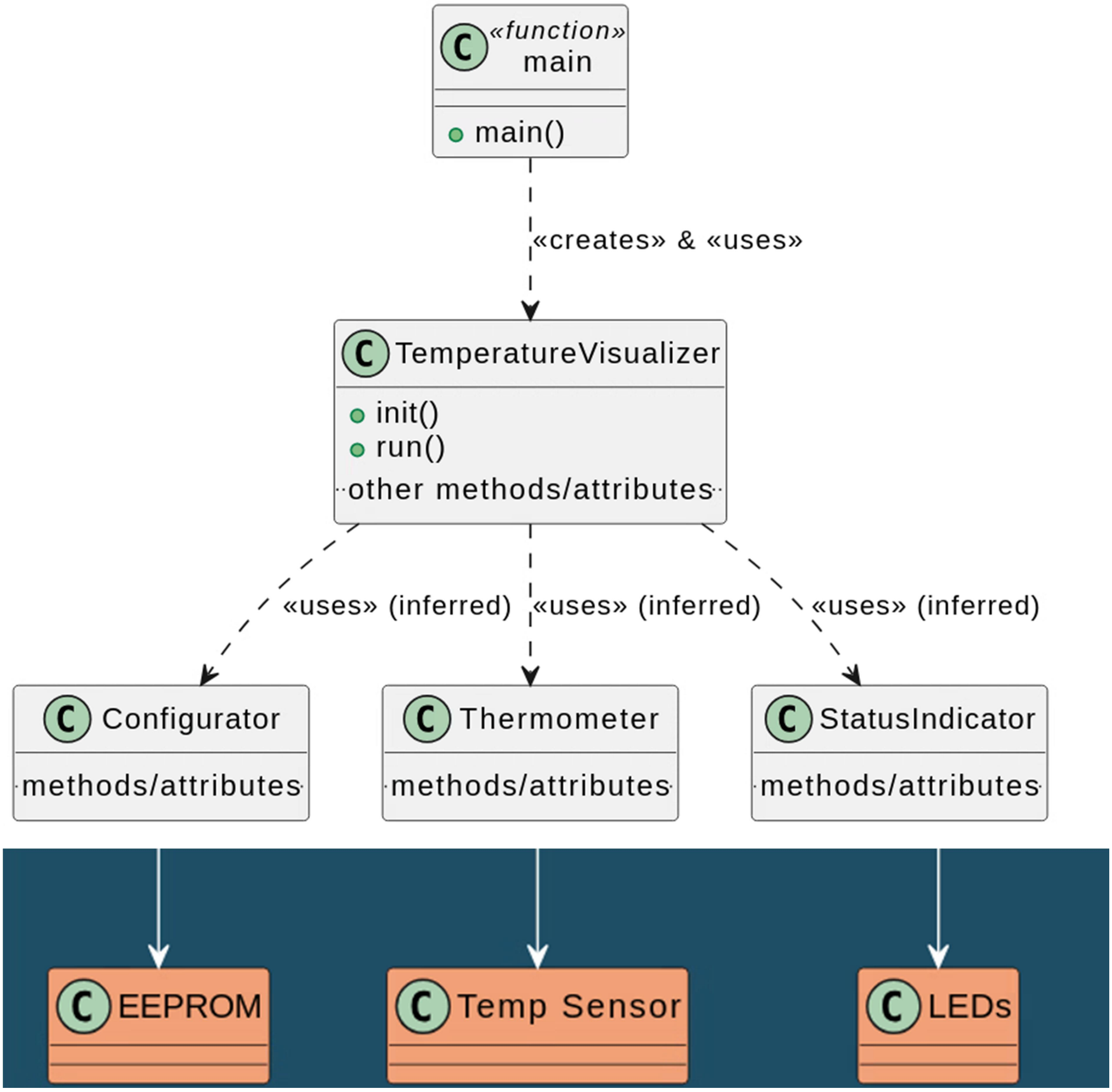
- set GPIO1/LED1 to High
- wait for transition

Temperature critical level:

- set GPIO0/LED0 to High
- wait for transition

Using C++ and OOP (differences):

Modelling the system parts in SW.



- Abstracting the low-level interfaces / hardware and encapsulate them including necessary memory
- To keep things clear the C++ solution implements the higher abstraction layer and communication mechanisms included in there
- use smart instantiation mechanisms (smart pointers) will not be considered