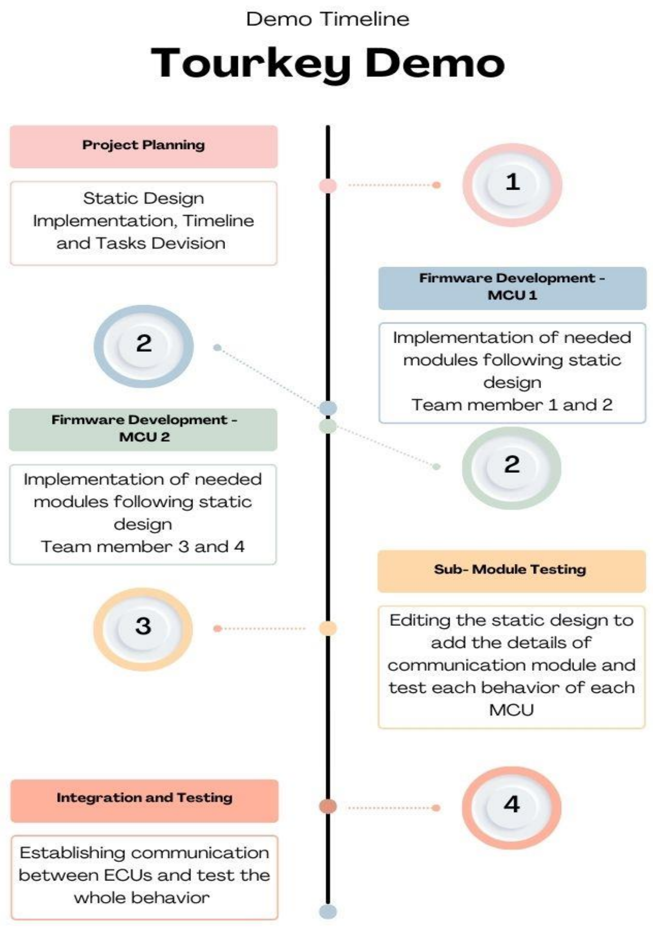
Static Design Analysis

Description:

- Each MCU has an LCD which displays:
  - Date and Time (with seconds)
  - Stop watch
- Each MCU has switches to control the LCD of the other MCU
- The communication between MCUs could be UART or any other protocol
- When switching from Stop watch mode to Date & Time mode then going back to Stop watch. The Stop watch should be resuming its previous work

Timeline



Demo Timeline

Tourkey Demo

1 **Project Planning**

Static Design Implementation, Timeline and Tasks Devision

2 **Firmware Development - MCU 1**

Implementation of needed modules following static design
Team member 1 and 2

2 **Firmware Development - MCU 2**

Implementation of needed modules following static design
Team member 3 and 4

3 **Sub- Module Testing**

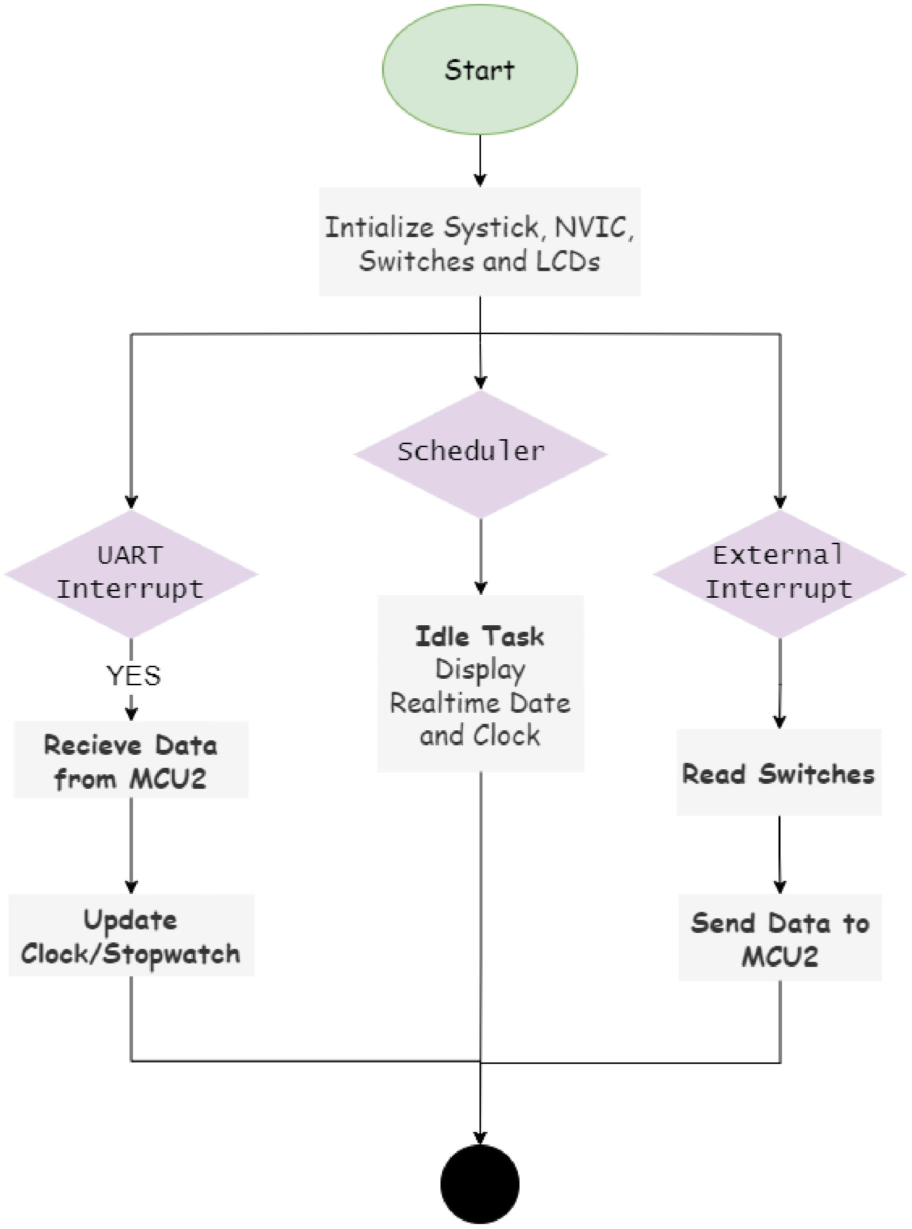
Editing the static design to add the details of communication module and test each behavior of each MCU

4 **Integration and Testing**

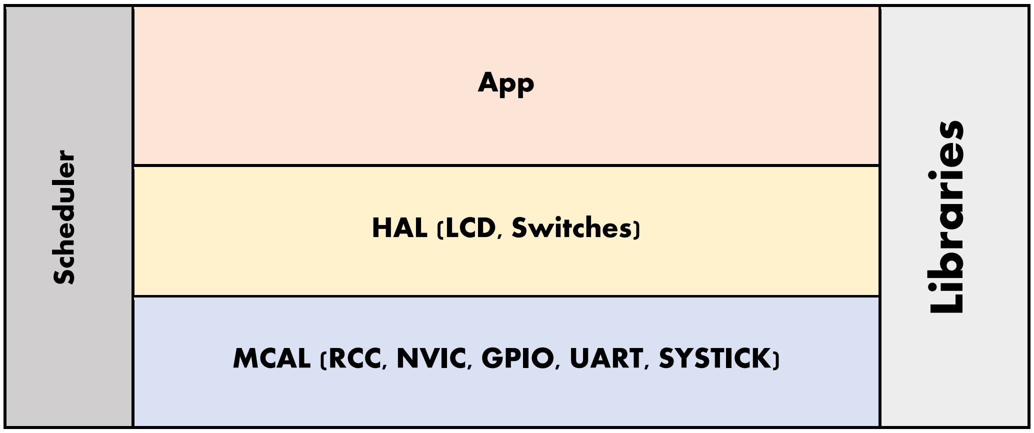
Establishing communication between ECUs and test the whole behavior

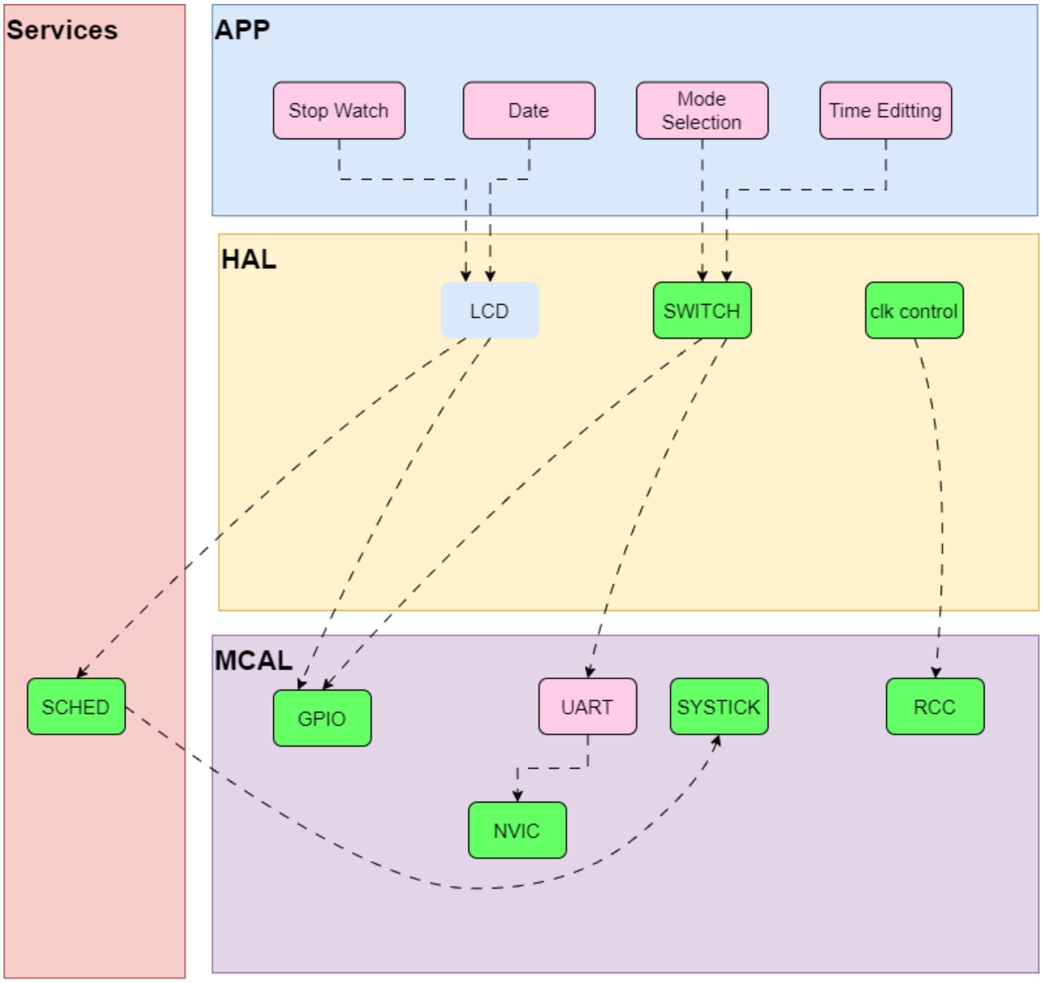
For ECU 1

Flow Chart



The layered architecture





Done 🟩

In progress 🟦

Not Implemented 🟥

ECU components

1) Switches

- Increment Clock Switch
- Decrement Clock Switch
- Mode Selection Switch
- UP and Down switches for mode selection

2) LCD

Full detailed APIs for each module as well as a detailed description for the used typedefs

1. NVIC Module

1.1 void NVIC_DecodePriority (uint32 Priority, uint32 PriorityGroup, uint32* pPreemptPriority, uint32* pSubPriority);

Function	void NVIC_DecodePriority (uint32 Priority, uint32 PriorityGroup, uint32* pPreemptPriority, uint32* pSubPriority);
Input Arguments	**Priority:** Priority value, which can be retrieved with the function \ref NVIC_GetPriority(). **Type:** Unsigned Integer **Range:** from zero to 128 **PriorityGroup:** Used priority group. **Type:** Unsigned Integer **Range:** from 1 to pPreemptPriority: Preemptive priority value **Type:** Pointer to unsigned Integer **Range:** from 0 to pSubPriority: Subpriority value **Type:** Pointer to unsigned Integer **Range:** from 0 to
Output Arguments	None
Description	- The function decodes an interrupt priority value with a given priority group to preemptive priority value and subpriority value.

	- In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS) the smallest possible priority group is set.
Return	None

1.2 void NVIC_SoftwareInterrupt(IRQn_t IRQn);

1.3 uint32 NVIC_EncodePriority (uint32 PriorityGroup, uint32 PreemptPriority, uint32 SubPriority);

1.4 uint32 NVIC_GetPriority(IRQn_t IRQn);

1.5 void NVIC_SetPriority(IRQn_t IRQn, uint32 priority);

1.6 void NVIC_SetPriorityGrouping(uint32 PriorityGroup);

1.7 uint32 NVIC_GetPriorityGrouping(void);

1.8 void NVIC_EnableIRQ(IRQn_t IRQn);

2. GPIO Module

2.1 GPIO_ERROR_t GPIO_init(GPIO_ConfigPin_t *Config_ptr);

2.2 GPIO_ERROR_t GPIO_setPinValue(void* port,uint32 pin, uint32 value);

2.3 uint32 GPIO_getPinValue(void* port,uint32 pin);

3. RCC Module

3.1 RCC_ERROR_STATUS RCC_enableClk(CLK_SRC clk);

3.2 RCC_ERROR_STATUS RCC_disableClk(CLK_SRC clk);

3.3 RCC_ERROR_STATUS RCC_setSysClk(CLK_SRC clk);

3.4 RCC_ERROR_STATUS RCC_pllConfig(uint8 PLLM,uint8 PLLN,uint8 PLLP,uint8 PLLQ,PLL_SRC PLLclk);

3.5 RCC_ERROR_STATUS RCC_enablePeri(RCC_BUS_ID_t busID, uint32 P_id);

3.6 RCC_ERROR_STATUS RCC_disablePeri(RCC_BUS_ID_t busID, uint32 P_id);

3.7 RCC_ERROR_STATUS RCC_prescalerConfig(RCC_BUS_ID_t busID,uint32 prescalar);

4. UART Module

5. SYSTICK Module

5.1 void SYSTICK_setCallBack(systick_cbf_t cbf);

5.2 void SYSTICK_Init(uint8 mode);

5.3 void SYSTICK_set_ms(uint32 time);

5.4 void SYSTICK_stop();

6. Scheduler Module

6.1 void sched_init(void);

6.2 void sched_start(void);

7. Switches Module

7.1 void SWITCH_init(void);

7.2 SWITCH_State_t SWITCH_Getstatus(uint32 switch_num);

7.3 void SWITCH_Runnable(void)

8. LCD Module

8.1 void LCD_Init(void);

8.2 void LCD_writeStringAsync(uint8* string,uint8 length);

8.3 void LCD_setCursorPosAsync(uint8 posX, uint8 posY);

8.4 void LCD_clearScreenAsynch(void);

8.5 uint8 LCD_getStatus(void);

8.6 static void LCD_initSM();

8.7 static void LCD_writeProcess();

8.8 static void LCD_clearProcess();

8.9 static void LCD_setPose_Process();