Dual STM32F401C Interactive Timekeeping Demo Static Design Report

Team Members:-

1- Heba Elsayed Fouad

2- Mohamed Ahmed Fouad

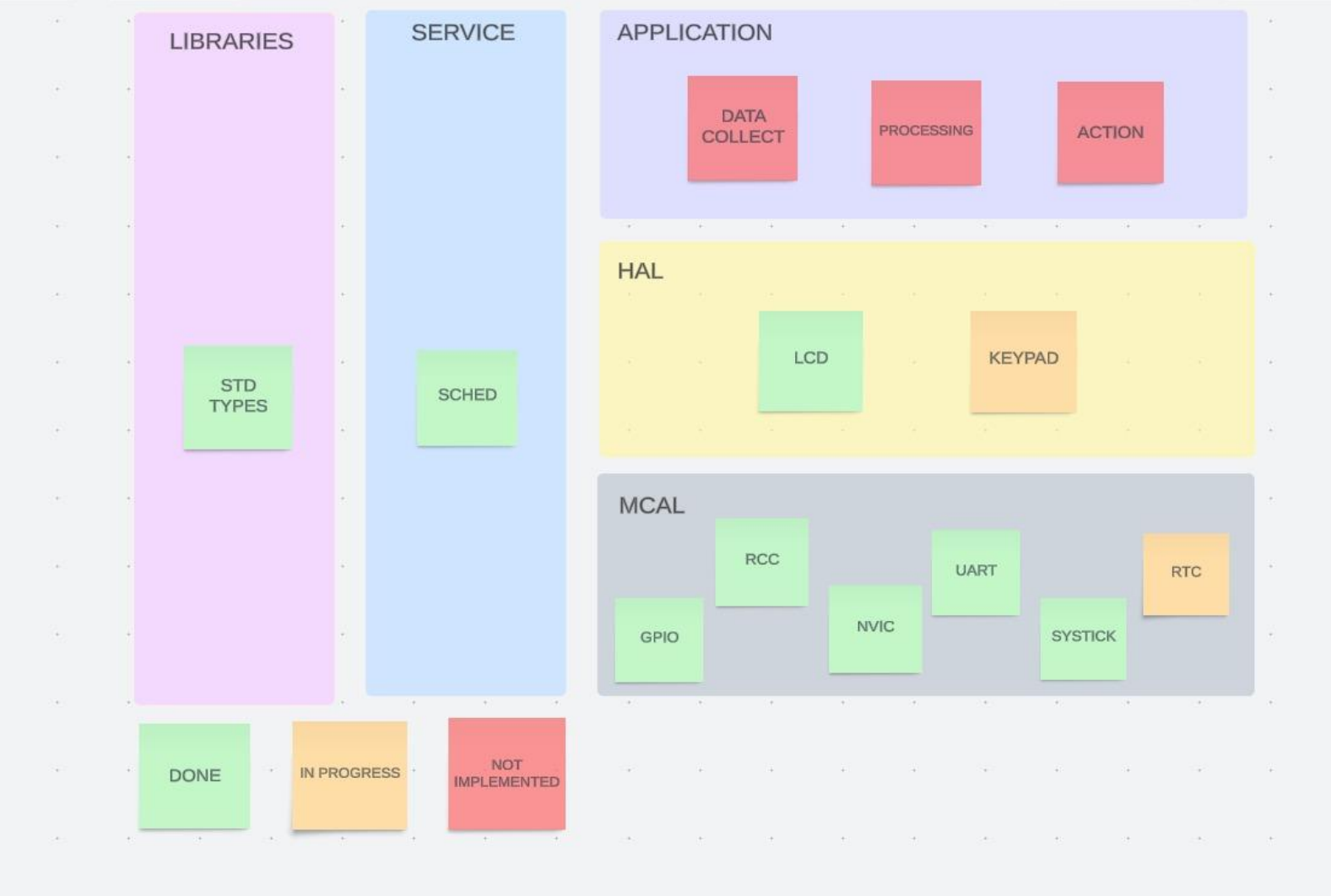
3- Omer Ahmed Ali

4- Mohamed Abdelkader

Static Design:

➢ For both ECUs:

1- the layered architecture:



2- Specify ECU components and modules

Components connected:

1. 2 Micro Controller (STM32F401CC)
2. 2 LCD (LMB161ABC)
3. 2 Keypad

Modules:

External hardware:

1. LCD Module.
2. KEYPAD Module.

Internal hardware:

1. GPIO Module (initialize all pins required with modes)
2. RCC Module (control the clocks for al the system peripherals)
3. RTC module (for calculating date time in real time environment)
4. UART module (for Communication Between The Two ECUs)
5. SYSTICK Timer Module.
6. NVIC Module.

3- Provide full detailed APIs for each module as well as a detailed description

The three Tasks to be created in Application Layer

Layer	Module	APIs	API Details	
Application Layer	Data Collection	Data_CollectNewData(data_buffer_t *dataBuffer)	**Syntax:**	Data_CollectNewData(data_buffer_t *dataBuffer);
			Sync/Async:	Sync/Async
			Reentrancy:	Non-Reentrant
			Parameters:	-Buffer to Collect Data
			Return:	Error Status
			Description:	Collect pressed keys and store it on Buffer

Layer	Module	APIs	API Details	
Application Layer	Data Process	Processing_ProcessData(data_buffer_t *inputBuffer, processed_data_t *outputBuffer)	**Syntax:**	Processing_ProcessData(data_buffer_t *inputBuffer, processed_data_t *outputBuffer);
			Sync/Async:	Sync/Async
			Reentrancy:	Non-Reentrant
			Parameters:	-Buffer of Data to Process -Buffer of Processed Data
			Return:	Error Status
			Description:	Process the Data Collected from Data Collection.
	ActionModule	Action_TakeAction(processed_data_t *data)	**Syntax:**	Action_TakeAction(processed_data_t *data);
			Sync/Async:	Sync/Async
			Reentrancy:	Non-Reentrant
			Parameters:	-Buffer of processed Data
			Return:	Error Status
			Description:	Take Action Based on Processed Data

The module in Servies Layer

Layer	Module	APIs	API Details	
Servies Layer	SCHEDULER	SCHED_Init()		
			Syntax:	Void SCHED_Init();
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**None**
			Return:	None
			Description:	Init the Scheduler
Servies Layer	SCHEDULER	SCHED_Start()	**Syntax:**	Void SCHED_Start();
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**None**
			Return:	None
			Description:	Start The Sched To Run The Tasks Periodically At Certain Time

The module in HAL Layer

Layer	Module	APIs	API Details	
HAL Layer	LCD	LCD_WriteString	**Syntax:**	void LCD_WriteString(const char *Copy_AddStr , u8 len);
			Sync/Async:	Asynchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**Copy_AddStr:** the string to be displayed on the lcd. **len** : the length of the string that is passed.
			Return:	None
			Description:	Displays String On The Lcd
HAL Layer	LCD	LCD_Init	**Syntax:**	void LCD_Init();
			Sync/Async:	Asynchronous
			Reentrancy:	Non-Reentrant
			Parameters:	**None**
			Return:	None
			Description:	Initialization of The Lcd Hardware
HAL Layer	HKPD	KPD_Init();	**Syntax:**	KPD_Init();
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	None
			Return:	ErrorStatus
			Description:	Initialize the used KPD pins for digital input
		KPD_GetValue(u8 Copy_u8KpdName , u8 *Add_u8KpdValue);	**Syntax:**	KPD_GetValue(u8 Copy_u8KpdName , u8 *Add_u8KpdValue);
			Sync/Async:	aynchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-The KPD you want to get Value from -A pointer to the Value
			Return:	Status of the sensor door closed or opened
			Description:	Get the status of each Key Asynchronously with Tasks

The module in MCAL Layer

Layer	Module	APIs	API Details	
MCAL Layer	USART	USART_Init(u32 USART_x);	**Syntax:**	UART_enuErrorStatus_t USART_Init(u32 USART_x);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	Channel to init
			Return:	Error Status
			Description:	Initialize the USART with specified Configuration
		USART_SendByte(u32 USART_x, u8 USART_BYTE);	**Syntax:**	UART_enuErrorStatus_t USART_SendByte(u32 USART_x, u8 USART_BYTE);
			Sync/Async:	asynchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-UART Channel - 8bit Data
			Return:	Error Status
			Description:	Send one Byte on the Channel
		Uart_TxBufferAsync(u32 USART_x,u8* buffer ,u32 len, TXCBF_t CB);	**Syntax:**	UART_enuErrorStatus_t Uart_TxBufferAsync(u32 USART_x,u8* buffer ,u32 len, TXCBF_t CB);
			Sync/Async:	asynchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-UART Channel -Buffer and Buffer Length -Callback Function
			Return:	Error Status
			Description:	Send Buffer through desired Channel
MCAL Layer	USART	Uart_RxBufferAsync(u32 USART_x,u8* buffer ,u32 len, RXCBF_t CB);	**Syntax:**	UART_enuErrorStatus_t Uart_RxBufferAsync(u32 USART_x,u8* buffer ,u32 len, RXCBF_t CB);
			Sync/Async:	aynchronous
			Reentrancy:	Non-Reentrant

			Parameters:	-UART Channel -Buffer and Buffer Length -Callback Function
			Return:	Error Status
			Description:	Recive Buffer in the specified Channel
MCAL Layer	GPIO	GPIO_enuErrorStatus_t GPIO_INITPIN(GPIO_t * GPIO_CFG);		
			Syntax:	GPIO_enuErrorStatus_t GPIO_INITPIN(GPIO_t * GPIO_CFG);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-Pointer To Pins Configuration
			Return:	Error status
			Description:	Initialize GPIO Pins Based on User Configuration.
MCAL Layer	GPIO	GPIO_enuErrorStatus_t GPIO_SET_PINVALUE (void * GPIO_PORT ,u32 GPIO_PIN , u32 GPIO_STATE);	**Syntax:**	GPIO_enuErrorStatus_t GPIO_SET_PINVALUE (void * GPIO_PORT ,u32 GPIO_PIN , u32 GPIO_STATE);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-Pointer To which PORT is selected -Select Pin Number -Select the state (HIGH or LOW)
			Return:	Error status
			Description:	Set GPIO Pins Based on User Parameters.
MCAL Layer	GPIO	GPIO_enuErrorStatus_t GPIO_GET_PINVALUE (void * GPIO_PORT ,u32 GPIO_PIN , u32 * GPIO_STATE);	**Syntax:**	GPIO_enuErrorStatus_t GPIO_GET_PINVALUE (void * GPIO_PORT ,u32 GPIO_PIN , u32 * GPIO_STATE);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-Pointer To which PORT is selected -Select Pin Number -Pointer to u32 to return the state which is (HIGH or LOW)
			Return:	Error status
			Description:	Get GPIO Pins Based on User Parameters.
MCAL Layer	GPIO	GPIO_enuErrorStatus_t GPIO_AF_CFG (void *	**Syntax:**	GPIO_enuErrorStatus_t GPIO_AF_CFG (void * GPIO_PORT ,u32 GPIO_PIN , u32

<table>
<tr><td></td><td></td><td>GPIO_PORT ,u32 GPIO_PIN , u32 GPIO_Func);</td><td></td><td>GPIO_Func);</td></tr>
<tr><td></td><td></td><td></td><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td></td><td></td><td></td><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td></td><td></td><td></td><td>Parameters:</td><td>-Pointer To which PORT is selected
-Select Pin Number
-Set the Alternative Function which is selected by user</td></tr>
<tr><td></td><td></td><td></td><td>Return:</td><td>Error status</td></tr>
<tr><td></td><td></td><td></td><td>Description:</td><td>Set GPIO Pins to the Alternative Function Based on User Parameters.</td></tr>
<tr><td>MCAL Layer</td><td>SYSTICK</td><td>Void Systick_Init(void);</td><td colspan="2"></td></tr>
<tr><td></td><td></td><td></td><td>Syntax:</td><td>Void Systick_Init(void);</td></tr>
<tr><td></td><td></td><td></td><td>Sync/Async:</td><td>- Synchronous</td></tr>
<tr><td></td><td></td><td></td><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td></td><td></td><td></td><td>Parameters:</td><td>None</td></tr>
<tr><td></td><td></td><td></td><td>Return:</td><td>None</td></tr>
<tr><td></td><td></td><td></td><td>Description:</td><td>Initialize Systick required configuration</td></tr>
<tr><td></td><td></td><td>Systick_Error_Status_t Systick_Start(u32 Time_ms);</td><td colspan="2"></td></tr>
<tr><td></td><td></td><td></td><td>Syntax:</td><td>Systick_Error_Status_t Systick_Start(u32 Time_ms);</td></tr>
<tr><td></td><td></td><td></td><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td></td><td></td><td></td><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td></td><td></td><td></td><td>Parameters:</td><td>-Set time in ms</td></tr>
<tr><td></td><td></td><td></td><td>Return:</td><td>Error status</td></tr>
<tr><td></td><td></td><td></td><td>Description:</td><td>Set Systick time with ms</td></tr>
</table>

MCAL Layer	NVIC	NVIC_Error_status_t NVIC_Enable_Interrupt(IRQn_t IRQn);	**Syntax:**	NVIC_Error_status_t NVIC_Enable_Interrupt(IRQn_t IRQn);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-Enum to which IRQn is selected
			Return:	Error status
			Description:	-Enable the interrupt which is selected by user
MCAL Layer	RTC	void RTC_Get_Time(u32* Time);	**Syntax:**	void RTC_Get_Time(u32* Time);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-Pointer to u32 to return the Time
			Return:	None
			Description:	-Get Real Time
MCAL Layer	RTC	void RTC_Get_Date(u32* Date);	**Syntax:**	void RTC_Get_Date(u32* Date);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-Pointer to u32 to return the Date
			Return:	None
			Description:	-Get Real Date
MCAL Layer	RTC	Void RTC_Init (void);	**Syntax:**	Void RTC_Init(void);
			Sync/Async:	Synchronous
			Reentrancy:	Non-Reentrant
			Parameters:	-None
			Return:	None
			Description:	-Init RTC

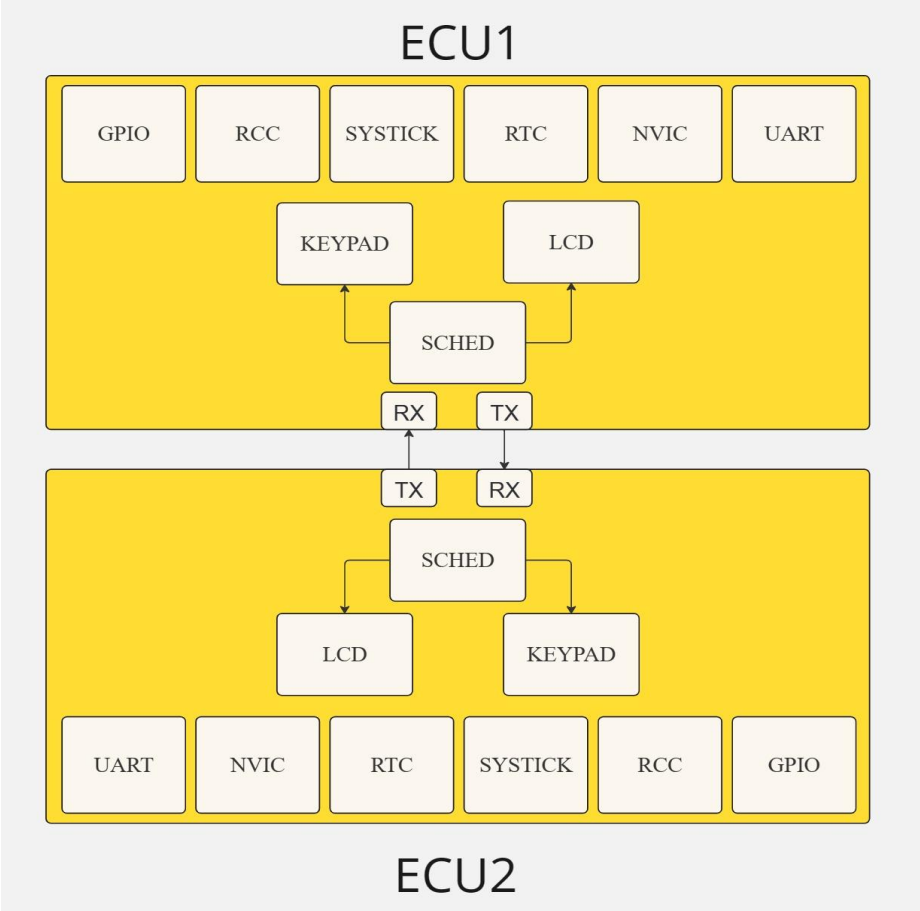
4- folder structure according to the previous points:

Application folder	Servies folder	On Board Layer
main.c	SCHEDULER.c	LCD.c
DATA_COLLECTOR.c		KPD.c
PROCESSING.c		
ACTION.c		

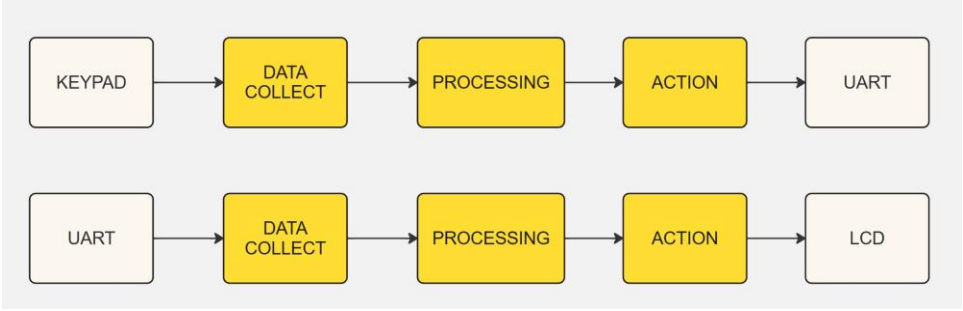
MCAL folder	Configure folder
GPIO.c	SCHED_config.c
USART.c	KPD_Config.c
NVIC.c	USART_Config.c
RCC.c	LCD_Config.c
RTC.c	
SYSTICK.c	

Common folder (all the header (name.h))
STD_TYPES.h

5.Block Diagram



6.Data Flow Diagram



7.Flow Chart

https://miro.com/app/board/uXjVKZO5FoQ=/?moveToWidget=3458764584344099709&cot=14