1-LAYERED ARCHITECTURE

<table>
<tr><td colspan="2">APPLICATION</td><td rowspan="3">SERVIES</td></tr>
<tr><td colspan="2">HAL/ECUAL</td></tr>
<tr><td colspan="2">MCAL</td></tr>
<tr><td colspan="3">MICROCONTROLLER</td></tr>
</table>

1-Application Layer: This layer includes the main logic and rules for the car's behavior. It handles the input events received from the presentation layer and calculates the appropriate actions for the car, such as moving forward, stopping, rotating, and repeating the steps.

2-ECUAL: This layer includes the domain-specific components and entities of the car, such as the four motors and the car's state. It provides the functionality for controlling the car's movement and state based on the input events received from the application layer.

3- MCAL: This includes device drivers that are specific to the hardware of the microcontroller being used in the system. These drivers are responsible for configuring and controlling the various peripherals of the microcontroller, such as GPIO.

4-Microcontroller Layer: This layer is responsible for managing the operation of the microcontroller itself. It includes functions like initializing the microcontroller's clock and other internal resources, setting up interrupts, and managing the memory map of the microcontroller. This layer is closely tied to the hardware implementation of the system and provides a foundation for the higher-level layers.

5-Serives Layer: This layer The Service Layer in the layered architecture of the system consists of modules that provide higher-level functionalities and abstractions for managing timing requirements and performing bit-level operations. Such as Standard Types.

2-SYSTEM MODULES

<table>
<tr><td colspan="3">APPLICATION</td><td rowspan="3">STANDER_TYPES,
UTILS,
TIME TRIGEIR SYSTEM,</td></tr>
<tr><td>BUTTONS</td><td>LED</td><td>MOTOR</td></tr>
<tr><td>DIO</td><td>TIMERS</td><td>INTERRUPT</td></tr>
</table>

DIO

typedef enum{ PA, PB, PC, PD }DIO_Port_type;	typedef enum{ OUTPUT, INFREE, INPULL }DIO_PinStatus_type;	typedef enum dioError{ DIO_OK, WRONG_PORT_NUMBER, WRONG_PIN_NUMBER, WRONG_VALUE, WRONG_DIRECTION }EN_dioError_t;	typedef enum{ PINA0,PINA1,PINA2,PINA3,PINA4,PINA5, PINA6,PINA7, PINB0,PINB1,PINB2,PINB3,PINB4,PINB5, PINB6,PINB7, PINC0,PINC1,PINC2,PINC3,PINC4,PINC5, PINC6,PINC7, PIND0,PIND1,PIND2,PIND3,PIND4,PIND5, PIND6,PIND7, TOTAL_PINS }DIO_Pin_type;

void DIO_Init(void)	
Function Name	DIO_Init
Description	This function initializes the DIO hardware and sets up the appropriate settings for the digital pins to be used. This includes setting the pin direction (input/output), pull-up resistor configuration, and other necessary settings.
Parameters	
Return Value	

EN_dioError_t DIO_InitPin(DIO_Pin_type pin,DIO_PinStatus_type status)	
Function Name	DIO_InitPin
Description	This function initializes the direction of a specific Digital Input/Output (DIO) pin.
Parameters	Pin name , Pin status
Return Value	EN_dioError_t

EN_dioError_t DIO_WritePin(DIO_Pin_type pin,DIO_PinVoltage_type volt)	
Function Name	DIO_WritePin
Description	This function sets the state of a digital output pin to a specified value (0 or 1)
Parameters	Pin name , Pin volt
Return Value	EN_dioError_t

EN_dioError_t DIO_toggle(DIO_Pin_type pin)	
Function Name	DIO_toggle
Description	This function toggles the state of a digital output pin.
Parameters	Pin name
Return Value	EN_dioError_t

EN_dioError_t DIO_ReadPin(DIO_Pin_type pin,DIO_PinVoltage_type *volt)	
Function Name	DIO_ReadPin
Description	This function reads the current state of a digital input pin and returns the value (0 or 1) to the calling function.
Parameters	Pin name , Pointer to variable to store volt value
Return Value	EN_dioError_t

TIMER

```
typedef enum{
        TIMER0_STOP=0,
        TIMER0_SCALER_1,
        TIMER0_SCALER_8,
        TIMER0_SCALER_64,
        TIMER0_SCALER_256,
       TIMER0_SCALER_1024,
        EXTERNALl_FALLING,
        EXTERNAL_RISING
}Timer0Scaler_type;
```

```
typedef enum
{
      TIMER0_NORMAL_MODE=0,
      TIMER0_PHASECORRECT_MODE,
      TIMER0_CTC_MODE,
      TIMER0_FASTPWM_MODE

        }Timer0Mode_type;
```

```
typedef enum
EN_timerError_t
{
        TIMER_OK,
        TIMER_Error
 }EN_timerError_t;
```

```
typedef enum
{
      OC0_DISCONNECTED=0,
        OC0_TOGGLE,
        OC0_NON_INVERTING,
        OC0_INVERTING

       }OC0Mode_type;
```

Timer 0_functions
`void TIMER0_Init_all(Timer0Mode_type mode,Timer0Scaler_type scaler);`
`void TIMER0_Init(Timer0Mode_type mode);`
`void timer_InitValue(u8 timerInitValue);`
`EN_timerError_t timer_start(Timer0Scaler_type scaler);`
`void timer_reset();`
`void timer_delay(u16 desiredDelay);`
`void TIMER0_OC0Mode(OC0Mode_type mode);`
`void TIMER0_OV_InterruptEnable(void);`
`void TIMER0_OV_InterruptDisable(void);`
`void TIMER0_OC_InterruptEnable(void);`
`void TIMER0_OC_InterruptDisable(void);`
`void TIMER0_OV_SetCallBack(void(*LocalFptr)(void));`

PMW

```
typedef enum EN_timerError_t
{
        TIMER_OK,
        TIMER_Error
       }EN_timerError_t;
```

PWM_functions
`void PWM_Init(void);`
`void PWM_Freq_KHZ(u16 freq);`
`void PWM_Freq_HZ(u16 freq);`
`void PWM_Duty(u16 duty);`
`void PWM_Measure(u32* Pfreq,u8* Pduty);`
`void PWM_Measure2(u32* Pfreq,u8* Pduty);`
`void PWM_Init(void);`
`void PWM_Freq_KHZ(u16 freq);`

INTERRUPT

typedef enum{ LOW_LEVEL=0, ANY_LOGIC_CHANGE, FALLING_EDGE, RISING_EDGE, }TriggerEdge_type;	typedef enum{ EX_INT0=0, EX_INT1, EX_INT2 }ExInterruptSource_type;	typedef enum EN_EXI_ERROR_t { EXI_OK, EXI_ERROR } EN_EXI_ERROR_t	

void EXI_Init(void)	
Function Name	EXI_Init
Description	This function initializes the interrupt controller and sets up the appropriate settings for the interrupts to be used
Parameters	
Return Value	

EN_EXI_ERROR_t EXI_Enable(ExInterruptSource_type Interrupt)	
Function Name	EXI_Enable
Description	This function enables the interrupt for a specific event or source, allowing the microcontroller to respond to the incoming interrupt.
Parameters	Interrupt name (ExInterruptSource_type)
Return Value	EN_EXI_ERROR_t

EN_EXI_ERROR_t EXI_Disable(ExInterruptSource_type Interrupt)	
Function Name	EXI_Disable
Description	This function disables the interrupt for a specific event or source, preventing the microcontroller from responding to the incoming interrupt.
Parameters	Interrupt name (ExInterruptSource_type)
Return Value	EN_EXI_ERROR_t

EN_EXI_ERROR_t EXI_TriggerEdge(ExInterruptSource_type Interrupt,TriggerEdge_type Edge)	
Function Name	EXI_TriggerEdge
Description	This function to chose which trigger edge will be interrupt
Parameters	Interrupt name (ExInterruptSource_type), Event (TriggerEdge_type)
Return Value	EN_EXI_ERROR_t

EN_EXI_ERROR_t EXI_SetCallBack(ExInterruptSource_type Interrupt,void(*LocalPtr)(void))	
Function Name	
Description	This function sets the callback function to be executed when an interrupt occurs on external interrupt
Parameters	Interrupt name (ExInterruptSource_type), A pointer to the callback function to be executed
Return Value	EN_EXI_ERROR_t

BUTTON

#define button1 PIN_NAME #define button2 PIN_NAME #define button3 PIN_NAME #define button4 PIN_NAME		typedef enum EN_ButtonError_t { BUTTON_OK, BUTTON_ERROR }EN_ButtonError_t;	

void BUTTON_init (DIO_Pin_type button)	
Function Name	BUTTON_init
Description	Initializes pins as button (input with pull resistor activation)
Parameters	
Return Value	

EN_ButtonError_t BUTTON_read (DIO_Pin_type button,DIO_PinVoltage_type *volt)	
Function Name	BUTTON_read
Description	This function Reads button state and stores value in button volt
Parameters	Button name (DIO_Pin_type) ,to store in volt(DIO_PinVoltage_typ)
Return Value	EN_ButtonError_t

		typedef enum EN_MotorError_t { MOTOR_OK, MOTOR_ERROR }EN_MotorError_t;	

void MOTOR_Init(void)	
Function Name	MOTOR_Init
Description	
Parameters	
Return Value	

EN_MotorError_t MOTOR_Stop(MOTOR_type motor)	
Function Name	MOTOR_Stop
Description	
Parameters	
Return Value	EN_MotorError_t

EN_MotorError_t MOTOR_CW(MOTOR_type motor)	
Function Name	MOTOR_CW
Description	
Parameters	
Return Value	EN_MotorError_t

EN_MotorError_t MOTOR_CCW(MOTOR_type motor)	
Function Name	MOTOR_CCW
Description	
Parameters	
Return Value	EN_MotorError_t

EN_MotorError_t MOTOR_Speed(MOTOR_type motor,u8 speed)	
Function Name	MOTOR_Speed
Description	
Parameters	
Return Value	EN_MotorError_t