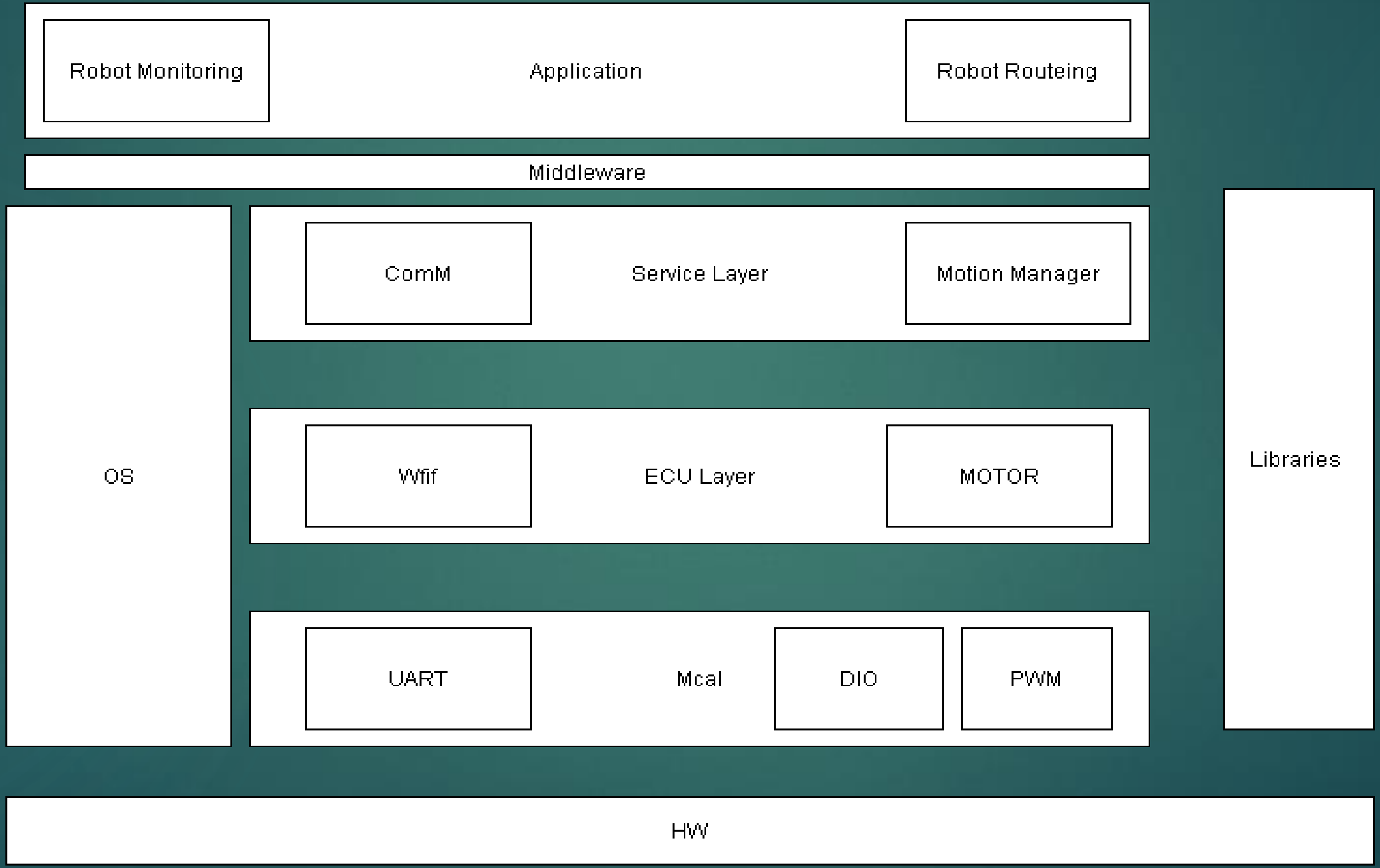
# Embedded SW Design

FINAL PROJECT

## 1.System Description

This system is supposed to take directions from PC using WIFI module and return to it the temperature ,it control the Motor speed by PWM module and Timer module , also it uses OS Module to manage the application ,there is a group of managers to manage communication ,Non Volatile memory and Motion .

# 2.Layered Architecture (AUTOSAR based)



## 3. SW Data Type Tables

# DIO Data types :

Pin Type:

- Name: DIO_PinType
- Type: Enumeration
- Range: 0-31
- Description: it define the pin number used in the API.

Level Type:

- Name: DIO_LevelType
- Type: Enumeration
- Range: 0-1
- Description: it define the level used on the pin.

## Timer Data types :

Timer Configuration:

- Name: Timer_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the timer.

## PWM Data types :

PWM Configuration:

- Name: PWM_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the PWM.

# WIFI Data types :

WIFIConfiguration:

- Name: WIFI_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the WIFI.

Data Type:

- Name: WIFI_DataType
- Type: uint8
- Range: none
- Description: data to be written on the WIFI.

## Motor Data types :

Motor Cofiguration:

- Name: Motor_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the Motor.

# ComM Data types :

ComM Cofiguration:

- Name: ComM_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the ComM.

Monitoring data:

- Name: ComM_DataType
- Type: uint8
- Range: none
- Description: data sent or received by comM.

## MotorM Data types :

MotorM Cofiguration:

- Name: MotorM_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the MotorM.

## Monitoring Data types :

Monitoring Cofiguration:

- Name: Monitoring_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the Monitoring .

Monitoring data:

- Name: Monitoring_DataType
- Type: uint8
- Range: none
- Description: carry data to be sent.

## Monitoring Data types :

RoutingCofiguration:

- Name: Routing_ConfigType
- Type: Struct
- Range: none
- Description: Configuration of the Routing.

## 4. SW Layers

## MCAL Layer

- DIO
- PWM
- TIMER
- UART

## Dio header Files:

- DIO.h
- DIO_Cfg.h
- DIO_LCfg.h
- DIO_PbCfg.h
- DIO_Types.h

## DIO Source Files:

- DIO.c
- DIO_Cfg.c
- DIO_LCfg.c
- DIO_PbCfg.c

## DIO APIs:

- DIO_Init()
- DIO_Read()
- DIO_Write()

## APIs Pseudo Code:

```
error_status DIO( Config){

    Configure the Port and Pin Directions;
}
error_status DIO_Read(DIO_DataType * readValue){

    *readValue=PortA;
}
error_status DIO_Write(DIO_DataType readValue){

    Portx=writeValue;
}
```

## APIs Description:

Initialization

- Function name: DIO_Init().
- Arguments:
- Input: Pin( DIO_PinType ).
- Output: none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: initialize the DIO module .
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

Read

- Function name: DIO_Read()
- Arguments:
  - Input: Pin(DIO_PinType)
  - Output: Level(DIO_LevelType)
  - Input/output: none
- Return: E_OK(0),E_NOK(1).
- Description: read the pin value .
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

Write

- Function name: DIO_Write().
- Arguments:
  - Input: Pin ( DIO_PinType ), Level ( DIO_LevelType ).
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Set the Pin value .
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## MCAL Layer Timer:

- Timer_Init()
- Timer_Start()
- Timer_Stop()

## Timer header Files:

- Timer.h
- Timer_Cfg.h
- Timer_LCfg.h
- Timer_PbCfg.h
- Timer_Types.h

## TimerSource Files:

- Timer.c
- Timer_Cfg.c
- Timer_LCfg.c
- Timer_PbCfg.c

## APIs Pseudo Code:

```
error_status Timer_Init( Config){

    Configure the Timer
}
error_status Timer_Start(void){

Start the Timer
}
error_status Timer_Stop(void){

 Stop the Timer
}
```

## APIs Description:

Initialization

- Function name: Timer_Init().
- Arguments:
  - Input: Config( Timer_ConfigType ).
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Initiate the Timer .
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

Start

- Function name: Timer_Start().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Starts the timer .
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

Stop

- Function name: Timer_Stop().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Stops the Timer.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## MCAL Layer PWM:

- PWM_Init()
- PWM_Start()
- PWM_Stop()

# APIs Pseudo Code:

```
error_status PWM_Init( Config){

    Configure thePWM;
}
error_status PWM_Start(void){

Start the PWM
}
error_status PWM_Stop(void){

 Stop the PWM
}
```

## PWM header Files:

- PWM.h
- PWM_Cfg.h
- PWM_LCfg.h
- PWM_PbCfg.h
- PWM_Types.h

## PWM Source Files:

- PWM.c
- PWM_Cfg.c
- PWM_LCfg.c
- PWM_PbCfg.c

## APIs Description:

Intialization

- Function name: PWM_Init().
- Arguments: Config (PWM_ConfigType)
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: init the PWM.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

# APIs Description:

Start

- Function name: PWM_Start().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Starts the PWM.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

Stop

- Function name: PWM_Stop().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Stops the PWM.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## ECU Layer

- Motor
- WIFI

## WIFI(ESP): APIs:

- WIFI_Init()
- WIFI_Receive()
- WIFI_send()

## WIFI header Files:

- WIFI.h
- WIFI_Cfg.h
- WIFI_LCfg.h
- WIFI_PbCfg.h
- WIFI_Types.h

## WIFI Source Files:

- WIFI .c
- WIFI _Cfg.c
- WIFI _LCfg.c
- WIFI _PbCfg.c

## APIs Pseudo Code:

```
error_status WIFI_Init( Config){

    Configure the WIFI;
}
error_status WIFI_receive(WIFI_DataType * readValue){

   Receive data;
}
error_status DIO_Send(WIFI_DataType readValue){

    Send data;
}
```

## APIs Description:

Init

- Function name: WIFI_Init().
- Arguments:
  - Input: Config(WIFI_ConfigType).
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Starts the LCD.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

Receive

- Function name: WIFI_Receive().
- Arguments:
  - Input: none.
  - Output: Data(WIFI_Data).
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Receive Data from WIFI module.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

Send

- Function name: WIFI_Send().
- Arguments:
- Input: Data(WIFI_Data).
- Output: none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: send Data by WIFI module.
- Synchronous / Asynchronous : Asynchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## Motor:
## APIs:

- Motor_Init()
- Motor_Start()
- Motor_Stop()

## Motor header Files:

- Motor .h
- Motor _Cfg.h
- Motor _LCfg.h
- Motor _PbCfg.h
- Motor _Types.h

## Motor Source Files:

- Motor .c
- Motor _Cfg.c
- Motor _LCfg.c
- Motor _PbCfg.c

## APIs Pseudo Code:

```
error_status Motor_Init( Config){

    Configure the motor;
}
error_status Motor_Start(void){
      start motor;
}
error_status Motor_stop(void){
    Stop motor;
}
```

# APIs Description:

Initialization

- Function name : Motor_Init().
- Arguments:
- Input: Config(Motor_ConfigType).
- Output: none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Init the Motor.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

Start

- Function name: Motor_Start().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Starts the Motor.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

Stop

- Function name: Motor_Stop().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Stops the Motor.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## Service Layer

- ComM
- MotorM

## ComM: APIs:

- ComM_Init()
- ComM_Receive()
- ComM_send()

## ComM header Files:

- ComM .h
- ComM _Cfg.h
- ComM _LCfg.h
- ComM _PbCfg.h
- ComM _Types.h

## ComM Source Files:

- ComM .c
- ComM _Cfg.c
- ComM _LCfg.c
- ComM _PbCfg.c

# APIs Pseudo Code:

```
error_status ComM( Config){

    Configure comM;

}

error_status DIO_send(data){
    Sende data;
}
error_status DIO_Receive(data){
   Receice data
}
```

# APIs Description:

Initlization

- Function name: ComM_Init().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Initialize ComM.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

Receive

- Function name: ComM_Receive().
- Arguments: none.
- Input: none.
- Output: Data(ComMDataType).
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Receive data.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

Send

- Function name: ComM_Send().
- Arguments: none.
- Input: Data(ComMDataType).
- Output: none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Send data.
- Synchronous / Asynchronous : Asynchronous .
- Reentrant/Non-Reentrant: Reentrant.

## MotorM:
## APIs:

- MotorM_Init()
- MotorM_UpdataStatus()

## MotorM header Files:

- MotorM .h
- MotorM _Cfg.h
- MotorM _LCfg.h
- MotorM _PbCfg.h
- MotorM _Types.h

# MotorM Source Files:

- MotorM .c
- MotorM _Cfg.c
- MotorM _LCfg.c
- MotorM _PbCfg.c

## APIs Pseudo Code:

```
error_status MotorM_Init( Config){

    Configure the MotorM.
}
error_status MotorM_UpdateStatus(){

  Updatestatus.
}
```

## APIs Description:

Initlization

- Function name: MotorM_Init().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Initialize MotorM.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

UpdateState

- Function name: MotorM_Updatestate().
- Arguments: none.
- Input: status(status_DataType).
- Output:none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: update motor state.
- Synchronous / Asynchronous : Asynchronous .
- Reentrant/Non-Reentrant: Reentrant.

## APIs Description:

reset

- Function name: MotorM_reset().
- Arguments: none.
  - Input:none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: reset motor.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## Application Layer

- Monitoring
- Routing

## Monitoring: APIs:

- Monitoring_Init()
- Monitoring_transmit()

## Monitoring header Files:

- Monitoring .h
- Monitoring _Cfg.h
- Monitoring _LCfg.h
- Monitoring _PbCfg.h
- Monitoring _Types.h

## Monitoring Source Files:

- Monitoring .c
- Monitoring _Cfg.c
- Monitoring _LCfg.c
- Monitoring _PbCfg.c

## APIs Pseudo Code:

```
error_status Monitor_Init( Config){
    Configure the Monitor
}
error_status Monitor_Transmit(data){
   Transmit data using ComM
}
```

## APIs Description:

Initialization

- Function name: Monitoring_Init().
- Arguments:
  - Input: Config(Monitoring_ConfigType).
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: initialize the Monitoring.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

## APIs Description:

Monitoring transmit

- Function name: Monitoring_transmit().
- Arguments: none.
- Input:Data (Monitoring_DataType).
- Output: none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: Send the Temperature.
- Synchronous / Asynchronous : Asynchronous .
- Reentrant/Non-Reentrant: Reentrant.

## Robot Routing Monitoring: APIs:

- Routing_Init()
- Routing_MoveUpdate()

## Monitoring header Files:

- Monitoring .h
- Monitoring _Cfg.h
- Monitoring _LCfg.h
- Monitoring _PbCfg.h
- Monitoring _Types.h

## Monitoring Source Files:

- Monitoring .c
- Monitoring _Cfg.c
- Monitoring _LCfg.c
- Monitoring _PbCfg.c

## APIs Pseudo Code:

```
error_status Routing_Init( Config){

Init the Routing application.

}
error_status Routing_UpdataState(){

Change the state of the motor
}
```

## APIs Description:

Initialization

- Function name: Routing_Init().
- Arguments: none.
  - Input: none.
  - Output: none.
  - Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: initialize the Robot Routing.
- Synchronous / Asynchronous : synchronous .
- Reentrant/Non-Reentrant: Non-Reentrant.

# APIs Description:

Update Moving state.

- Function name:Routing_MoveUpdate().
- Arguments:
- Input: status(Routing_DataType).
- Output: none.
- Input/output: none.
- Return: E_OK(0),E_NOK(1).
- Description: update the state of the robot.
- Synchronous / Asynchronous : Asynchronous .
- Reentrant/Non-Reentrant: Reentrant.