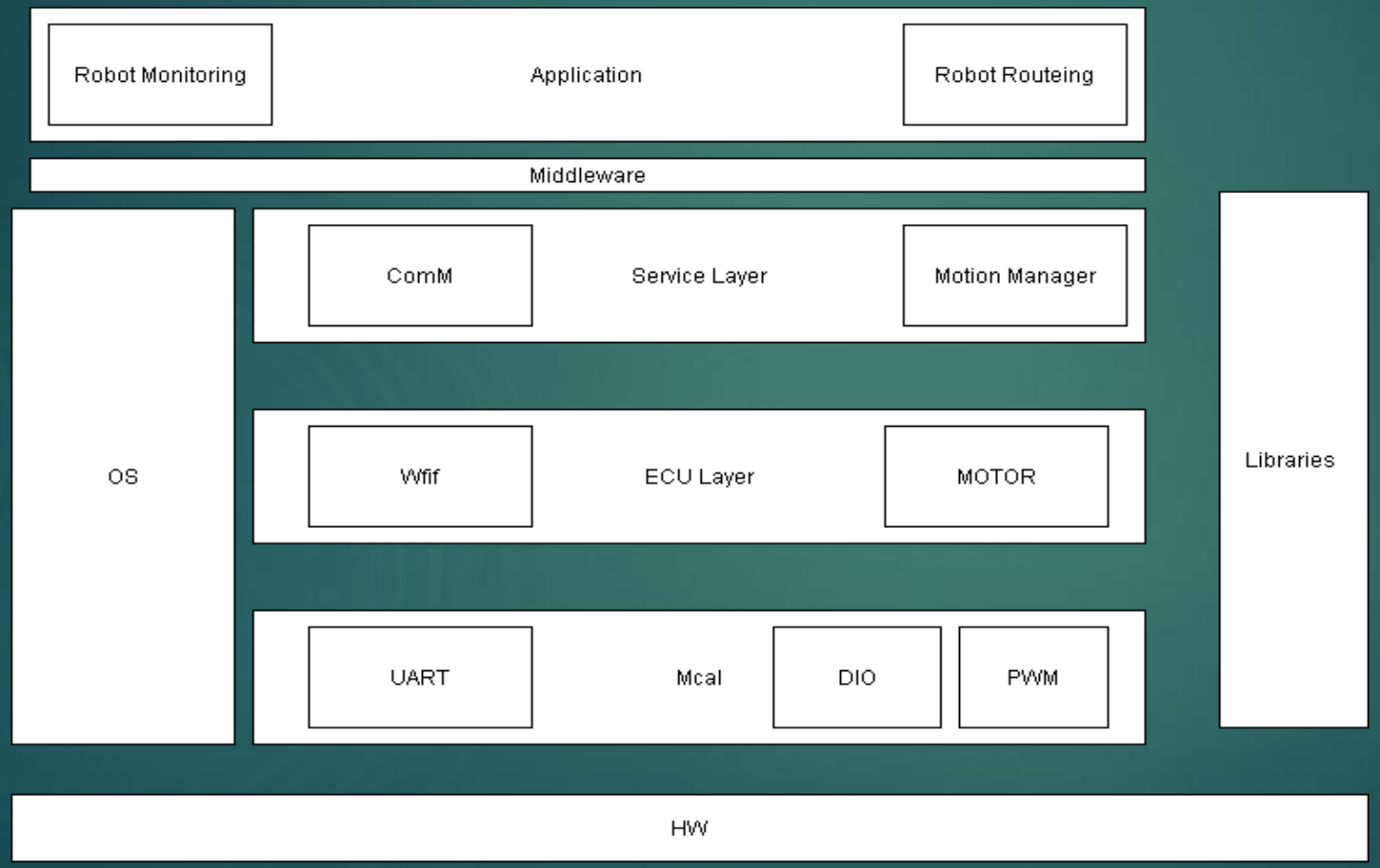
Embedded SW Design

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 :

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

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

Timer Data types :

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

PWM Data types :

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

WIFI Data types :

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

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

Motor Data types :

Name	Motor_ConfigType
Type	Struct
Range	none
Description	Configuration of the Motor

ComM Data types :

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

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

MotorM Data types :

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

Monitoring Data types :

Name	Monitoring_ConfigType
Type	Struct
Range	none
Description	Configuration of the Monitoring

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

Routing Data types :

Name	Routing_ConfigType
Type	Struct
Range	none
Description	Configuration of the Routing

4. SW Layers

MCAL Layer

- DIO
- PWM
- TIMER
- UART

DIO APIs:

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

DIO_cfg.h:

```
typedef enum {
    PORTA,PORTB
}DIO_PortType;

typedef enum{
PINA0,PINA1,PINA2,PINA3,PINA4,PINA5,PINA6,PINA7,PINB0,PINB1,PINB2,PI
NB3,PINB4,PINB5,PINB6,PINB7
}DIO_PinType;

typedef uint8 DIO_DataType

typedef struct{
DIO_PortType Port;
DIO_PinType Pin;
DIO_DataType Data;

}DIO_ConfigType;
```

Dio.h:

```
#include "DIO_cfg.h"


DIO_Init(DIO_ConfigType *Cfg) ;


DIO_Read(DIO_PinType Pin ,DIO_DataType * copy_u8Data);


DIO_Write(DIO_PinNum Pin,DIO_DataType copy_u8Data);
```

APIs Pseudo Code:

```
#include "DIO.h"

error_status DIO( DIO_ConfigType cfg){

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

    *readValue=Portx;
}
error_status DIO_Write(DIO_PinType Pin,DIO_DataTypereadValue){

    Portx=writeValue;
}
```

APIs Description:

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:

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

APIs Description:

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

MCAL Layer Timer:

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

Timer.h:

```
#include "Timer_cfg.h"


Timer_Init(Timer_ConfigType*  cfg) ;

Timer_Start() ;

Timer_Stop() ;
```

Timer_cfg.h:

```
typedef enum {
   TIMER0,Timer1
}Timer_ChannelType;

typedef enum{
PRE_1,PRE_8,PRE_256
}Timer_PrescalerType;

typedef uint32 Timer_Freq

typedef struct{
Timer_ChannelType  Channel;
Timer_Freq freg;
Timer_PrescalerType  Pre;

}Timer_ConfigType;
```

APIs Pseudo Code:

```
error_status Timer_Init(Config){

    Configure theTimer
}
error_status Timer_Start(void){

Start the Timer
}
error_status Timer_Stop(void){

 Stop the Timer
}
```

APIs Description:

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

APIs Description:

Function name:	Timer_Start
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Starts the timer.
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

Function name:	TImer_Stop
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Stops the timer
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

MCAL Layer
PWM:

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

PWM_cfg.h:

```
typedef enum {
    PWM0,PWM1
}PWM_ChannelType;

typedef enum{
PRE_1,PRE_8,PRE_256
}PWM_PrescalerType;

typedef uint32 PWM_Freq

typedef struct{
PWM_ChannelType Channel;
PWM_Freq freg;
PWM_PrescalerType Pre;

}PWM_ConfigType;
```

PWM.h:

```
#include "PWM_cfg.h"


PWM_Init() ;

PWM_Start() ;

PWM_Stop();
```

APIs Pseudo Code:

```
#include "PWM.h"
error_status PWM_Init(Config){

    Configure thePWM;
}
error_status PWM_Start(void){

Start the PWM
}
error_status PWM_Stop(void){

 Stop the PWM
}
```

APIs Description:

Function name:	TImer_Init
Arguments:	Input:Config (PWM_ConfigType)
Return	E_OK(0),E_NOK(1)
read the pin value	Iit the PWM
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Non-Reentrant

APIs Description:

Function name:	PWM_Start
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Start the PWM
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

Function name:	PWM_Stop
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Stops the PWM
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

ECU Layer

- Motor
- WIFI

WIFI(ESP): APIs:

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

WIFI_cfg.h:

```
typedef enum {
   WIFI0,WIFI1
}WIFI_ChannelType;

typedef uint32 WIFI_BuadRate

typedef struct{
WIFI_ChannelType Channel;
WIFI_BuadRate freg;

}WIFI_ConfigType;
```

WIFI.h:

```
#include "WIFI_cfg.h"


    WIFI_Init();

    WIFI_Receive();

    WIFI_send();
```

APIs Pseudo Code:

```
#include "WIFI"
error_status WIFI_Init( Config){

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

   Receive data;
}
error_status DIO_Send(WIFI_DataTypereadValue){

    Send data;
}
```

APIs Description:

Function name:	WIFI_Init
Arguments:	• Input: Config(WIFI_ConfigType). • Output: none. • Input/output: none.
Return	E_OK(0),E_NOK(1)
read the pin value	Init WIFI
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Non-Reentrant

APIs Description:

Function name:	WIFI_Receive
Arguments:	• Input: none. • Output: Data(WIFI_Data). • Input/output: none.
Return	E_OK(0),E_NOK(1)
read the pin value	Receives data
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

Function name:	WIFI_Send
Arguments:	• Input: Data(WIFI_Data). • Output: none. • Input/output: none.
Return	E_OK(0),E_NOK(1)
read the pin value	Send Data
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

Motor:
APIs:

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

Motor_cfg.h:

```
typedef enum {
   M0,M1,M2,M3
}Motor_ChannelType;


typedef uint32 Motor_Speed

typedef struct{
Motor_ChannelType Channel;
Motor_Speed freg;
}Motor_ConfigType;
```

Motor header Files:

```
#define "Motor_cfg.h"

Motor_Init()

 Motor_Start()

Motor_Stop()
```

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:

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

APIs Description:

Function name:	Motor_Start
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Start Motor
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

Function name:	Motor_Stop
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Stops the Motor
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

Service Layer

- ComM
- MotorM

ComM:
APIs:

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

ComM_cfg.h:

Typedef uint8 ComM_DataType

ComM header Files:

#define "ComM_cfg.h"

ComM_Init()

ComM_Receive()

ComM_send()

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:

Function name:	ComM_Init
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Initialize ComM
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Non-Reentrant

APIs Description:

Function name:	ComM_receive
Arguments:	Output: ComM_DataType Data
Return	E_OK(0),E_NOK(1)
read the pin value	Receive Data
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

Function name:	ComM Send
Arguments:	Input:ComM_dataType Data
Return	E_OK(0),E_NOK(1)
read the pin value	Sends Data
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

MotorM:
APIs:

- MotorM_Init()
- MotorM_UpdataStatus()

MotorM_cfg.h:

typedef uint32 Motor_Speed

MotorM header Files:

#include "MotorM_Cfg.h"

MotorM_Init()

MotorM_UpdataStatus()

APIs Pseudo Code:

```
#include "MotorM.h"

error_status MotorM_Init( void){

    Configure the MotorM.
}
error_status MotorM_UpdateStatus(){

  Updatestatus.
}
```

APIs Description:

Function name:	MotorM_Init()
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Init Motor manager
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

Function name:	MotorM_Updatestate
Arguments:	• Input: status(status_DataType). • Output:none. • Input/output: none.
Return	E_OK(0),E_NOK(1)
read the pin value	Update motor status
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

Application Layer

- Monitoring
- Routing

Monitoring: APIs:

- Monitoring_Init()
- Monitoring_transmit()

Monitoring_cfg.h:

```
#define SPEEDTRANSMIT 1000
Typedef uint8 Monitoring_DataType
```

Monitoring header Files:

```
#include "Monutoring_cfg,h"
Monitoring_Init()
Monitoring_transmit()
```

APIs Pseudo Code:

```
error_status Monitor_Init(  void){

    Configure the  Monitor

}

error_status Monitor_Transmit(data){

   Transmit  data using ComM


}
```

APIs Description:

Function name:	Monitoring_Init
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	Init the monitor
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Reentrant

APIs Description:

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

Routing Monitoring: APIs:

- Routing_Init()
- Routing_MoveUpdate()

Routing_cfg.h :

```
#define MOTIONSPEEDINIT 50
#define INITIALSTATE 0
```

Routing.h:

#include "Routing_cfg.h"

Routing_Init()

Routing_MoveUpdate()

APIs Pseudo Code:

```
#include  "Routing.h"
error_status Routing_Init( Config){

Init the  Routing application.

}
error_status Routing_UpdataState(){

Change the  stateof the motor
}
```

APIs Description:

Function name:	Routing_Init
Arguments:	none
Return	E_OK(0),E_NOK(1)
read the pin value	initialize the Robot Routing.
Synchronous / Asynchronous	synchronous
Reentrant/Non-Reentrant	Non-Reentrant

APIs Description:

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