2022

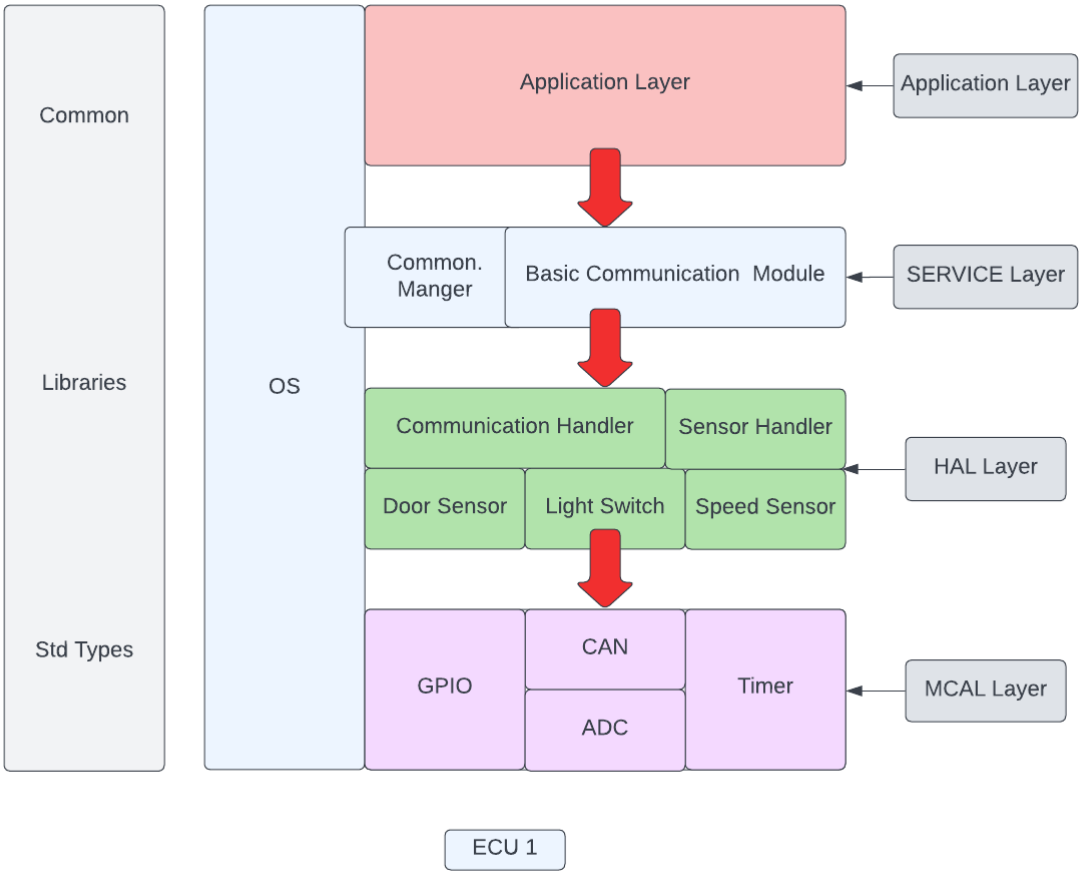
Automotive

AUTOMOTIVE DOOR CONTROL

NAME: IBRAHIM YASSER IBRAHIM

EBRAHIM195016@FENG.BU.EDU.EG

1- ECU-1 Layered Architecture



2- ECU-1 Components and Modules

- ECU-1 Component
    1. Speed Sensor
    2. Light Switch
    3. Door Sensor
    4. CAN Bus Protocol

- ECU-1 Modules
    1. Speed sensor module
    2. Door sensor module
    3. Switch module
    4. CAN module
    5. GPIO module (DIO, PORT)
    6. Timer module
    7. ADC module

3.1 Application Layer Modules

Layer	Module	APIs	API details	
Application Layer	Speed Sensor	speedSensor_SendState	Name	void speedSensor_SendState(void)
			Description	Send speed sensor state to ECU2
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
	Door Sensor	doorSensor_SendState	Name	void doorSensor_SendState(void)
			Description	Send door sensor state to ECU2
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
	Light Switch	lSwitch_SendState	Name	void lSwitch_SendState(void)
			Description	Send the light switch state to ECU2
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None

3.2 SERVICE Layer Modules

Layer	Module	APIs	API details	
SERVICE Layer	BCM	BCM_Manager	Name	Void BCM_Manager (uint8_t busID, uint64_data)
			Description	Select which bus we want to send the data through and then send it from the Application layer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
	Common /Manager	Sensor_Manager	Name	void Sensor_Manager (uint8_t sensorID)
			Description	Chooses which sensor we want to read from Application layer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	sensorID: selects which sensor we want to read
			Return	(uint16_t) State of the sensor

3.3 HAL Layer Modules

Layer	Module	APIs	API details	
HAL Layer	Speed Sensor	speedSensor_Init	Name	Void speedSensor_Init(void)
			Description	Initialize DIO pins used for analog input
			Reentrancy	Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
		speedSensor_GetState	Name	SpeedStatus_Type speedSensor_GetState(void)
			Description	Get speed sensor value
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	Sensor state
	Door Sensor	doorSensor_Init	Name	Void doorSensor_Init(void)
			Description	Initialize DIO pins for digital input
			Reentrancy	Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
		doorSensor_GetState	Name	DoorState_Type doorSensor_GetState(void)
			Description	Get door sensor state
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	Sensor state

Layer	Module	APIs	API details	
HAL Layer	Light Switch	lSwitch_Init	Name	Void lSwitch_Init(void)
			Description	Initialize DIO pins for digital input
			Reentrancy	Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
		lSwitch_GetState	Name	LightState_Type lSwitch_GetState(void)
			Description	Get the light switch state
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	Sensor state
	Sensor Handler	sensor_Handler	Name	Uint16_t sensor_Handler (uint8_t sensorID)
			Description	Chooses which sensor we want to read from hw layer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	sensorID: selects which sensor we want to read
			Return	(uint16_t) State of the sensor
	Communication Handler	BCM_Handler	Name	Void BCM_Handler (uint8_t busID, uint64_data)
			Description	Select which bus we want to send the data through and then send it from the hw layer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None

3.4 MCAL Layer Modules

Layer	Module	APIs	API details	
MCAL Layer	GPIO	GPIO_Init	Name	Void GPIO_Init (portConfig_Type* configPtr)
			Description	Initialize the GPIO with the configuration struct
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a struct with the configuration for port initialization
			Return	None
		GPIO_ReadPin	Name	Level_Type GPIO_ReadPin (port_Type portID, pin_Type pinID)
			Description	Read the status of the required pin
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	portID: the port required to configure the pin pinID: the pin you want to read its status
			Return	Level of the pin
		GPIO_WritePin	Name	void GPIO_WritePin (port_Type portID, pin_Type pinID, Level_Type level)
			Description	Write the value required to the required pin
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	portID: the port required to configure the pin pinID: the pin you want to write level: the level you want to write in the pin (HIGH,LOW)
			Return	None

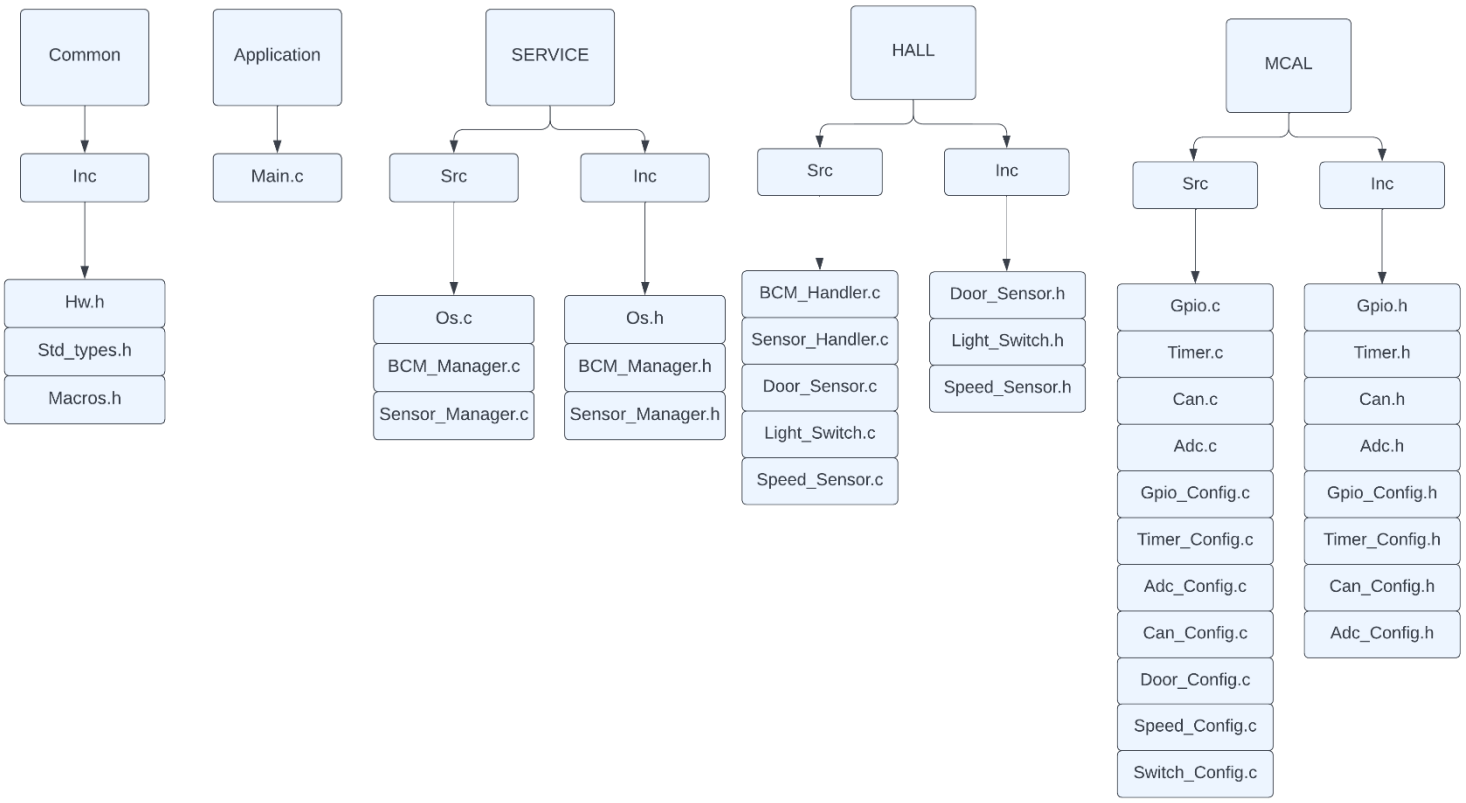
Layer	Module	APIs	API details	
MCAL Layer	ADC	ADC_Init	Name	Void ADC_Init (AdcConfig_Type* configPtr)
			Description	Initialize ADC required configuration
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a pointer to struct holding the configuration of the ADC
			Return	None
		ADC_ReadChannel	Name	Uint16_t ADC_ReadChannel (uint8_t channelID)
			Description	Read the status of the required pin
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: the channel ID you want to read its value
			Return	The ADC value read
	CAN	CAN_Init	Name	void CAN_Init (CANConfig_Type* configPtr)
			Description	Initialize CAN pin with the required configuration
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a pointer to struct holding the configuration of the CAN
			Return	None
		CAN_Transmit	Name	Void CAN_Transmit (uint8_t channelID, uint64_t data)
			Description	Transmit the data by CAN bus
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: the CAN channel to select which bus will be used data: the data that will be transmitted
			Return	None

Layer	Module	APIs	API details	
MCAL Layer	TIMER	Timer_Init	Name	void Timer_Init (TimerConfig_Type* configPtr)
			Description	Initialize timer with the required configuration
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a pointer to struct holding the configuration of the timer
			Return	None
		Timer_Start	Name	Void Timer_Start (Timer_Channel channelID, uint32_t countVal)
			Description	Start timer with the required channel and with the specified count value
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: the timer channel to select which timer will start countVal: the value required to initialize the timer
			Return	None
		Timer_Stop	Name	Void Timer_Stop (Timer_Channel channelID)
			Description	Stop the timer specified by the channelID
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: the timer channel to select which timer will stop
			Return	None

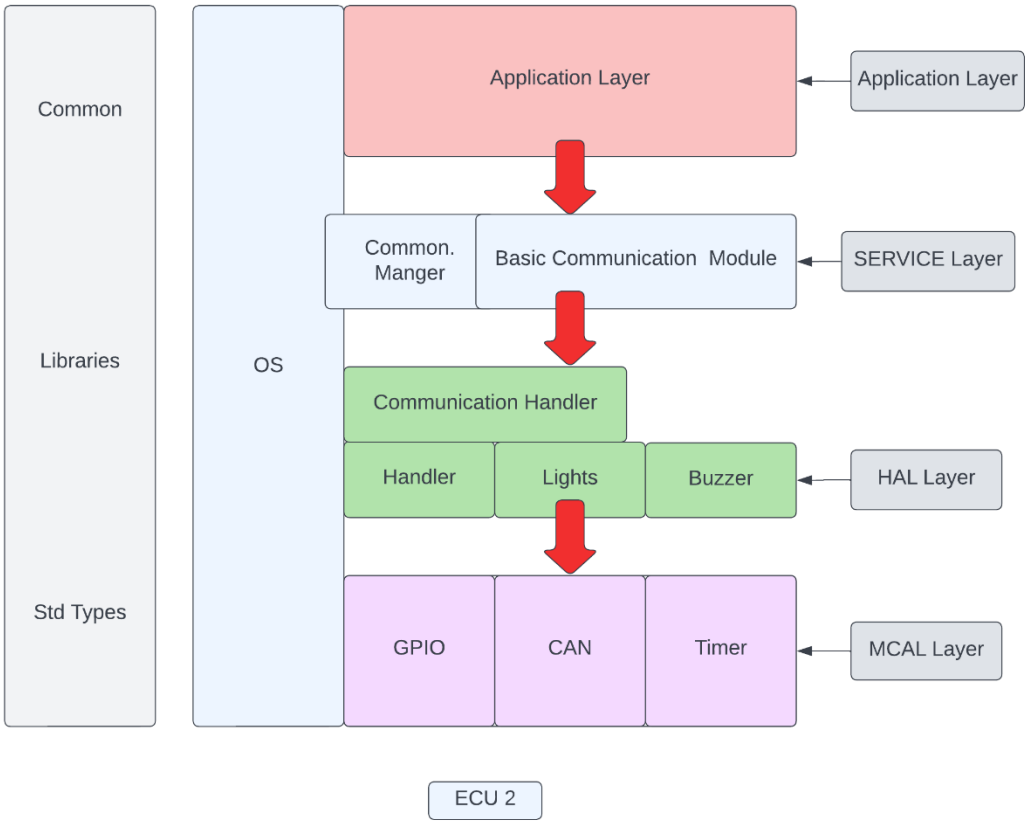
4. Typedefs of all Modules

Types	Define
Typedef unsigned char uint8_t	Used to store and send data within range (0,255) as its size 1 byte
Typedef unsigned short int uint16_t	Used to store and send data within range 2 bytes
Typedef unsigned long uint32_t	Used to store and send data within range 4 bytes
Typedef unsigned long long uint64_t	Used as the size of data in CAN transmitter and receiver as the size of CAN bus is 64 Bit
Typedef struct portConfig_Type	Is a struct holding the configuration of the GPIO like portID,Pin,level,direction ..etc
Enum Level_Type	Is Enum {LOW,HIGH} used to set or get the value of a pin
Enum Port_Type	Is enum {port0,port1,etc} used to hold all ports in the ecu to select the required port from it
Pin_Type	Is enum holding all pins of every port in the ecu
TimerConfig_Type	Is a struct holding all the configuration required to configure the timer
AdcConfig_Type	Is a struct holding all the configuration required to configure the ADC peripheral
CANConfig_Type	Is a struct holding all the configuration required to configure CAN communication
SpeedStatus_Type	Actually it’s a uint16_t used to store the value of the sensor
LightState_Type	Enum used to store the state of the switch if it pressed or not
DoorState_Type	Enum used to store the state of the door if it’s opened or closed

5- Folder structure



2- ECU-2 Layered Architecture



ECU 2

1- ECU-2 Components and Modules

- ECU-2 Component
    1. Left Light
    2. Right Light
    3. Buzzer
    4. CAN Bus Protocol

- ECU-2 Modules
    1. Light module
    2. Buzzer module
    3. CAN module
    4. GPIO module
    5. Timer module

2.1 Application Layer Modules

Layer	Module	APIs	API details	
Application Layer	Main	Receive_Status	Name	void Receive_Status (void)
			Description	Receive the messages from ECU-1 periodically and take actions
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None

2.2 SERVICE Layer Modules

Layer	Module	APIs	API details	
SERVICE Layer	BCM	BCM_Manager	Name	uint64_t BCM_Manager (uint8_t busID)
			Description	Manage the data received by the CAN bus by selecting which bus to read data from Application layer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	busID: which selects the bus we will connect to receive the data
			Return	Return the data frame of CAN bus which received from ECU-1

2.3 HAL Layer Modules

Layer	Module	APIs	API details	
HAL Layer	Communication Handler	BCM_Handler	Name	uint64_t BCM_Handler (uint8_t busID)
			Description	Manage the data received by the CAN bus by selecting which bus to read data from Application layer
			Reentrancy	Reentrant
			Synchronization	Synchronous
			Parameters	busID: which selects the bus we will connect to receive the data
			Return	Return the data frame of CAN bus which received from ECU-1
	Buzzer	Buzzer_Init	Name	void Buzzer_Init (void)
			Description	Initialize the specified DIO pins as digital output
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
		Buzzer_TurnOn	Name	void Buzzer_TurnOn (void)
			Description	Turn on the Buzzer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
		Buzzer_TurnOff	Name	Void Buzzer_TurnOff (void)
			Description	Turn off the buzzer
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None

Layer	Module	APIs	API details	
HAL Layer	Lights	lights_Init	Name	Void lights_Init(void)
			Description	Initialize the specified DIO pins as digital output
			Reentrancy	Reentrant
			Synchronization	Synchronous
			Parameters	None
			Return	None
		lights_TurnOn	Name	void lights_TurnOn (light_Type RL)
			Description	Turn on the specified light (R or L)
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	RL: is an enum value to define which light will be on
			Return	Sensor state
		Lights_TurnOff	Name	void lights_TurnOff (light_Type RL)
			Description	Turn off the specified light (R or L)
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	RL: is an enum value to define which light will be off
			Return	None

2.4 MCAL Layer Modules

Layer	Module	APIs	API details	
MCAL Layer	GPIO	GPIO_Init	Name	Void GPIO_Init (portConfig_Type* configPtr)
			Description	Initialize the GPIO with the configuration struct
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a struct with the configuration for port initialization
			Return	None
		GPIO_ReadPin	Name	Level_Type GPIO_ReadPin (port_Type portID, pin_Type pinID)
			Description	Read the status of the required pin
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	portID: the port required to configure the pin pinID: the pin you want to read its status
			Return	Level of the pin
		GPIO_WritePin	Name	void GPIO_WritePin (port_Type portID, pin_Type pinID, Level_Type level)
			Description	Write the value required to the required pin
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	portID: the port required to configure the pin pinID: the pin you want to write level: the level you want to write in the pin (HIGH,LOW)
			Return	None

Layer	Module	APIs	API details	
MCAL Layer	TIMER	Timer_Init	Name	void Timer_Init (TimerConfig_Type* configPtr)
			Description	Initialize timer with the required configuration
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a pointer to struct holding the configuration of the timer
			Return	None
		Timer_Start	Name	Void Timer_Start (Timer_Channel channelID, uint32_t countVal)
			Description	Start timer with the required channel and with the specified count value
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: the timer channel to select which timer will start countVal: the value required to initialize the timer
			Return	None
		Timer_Stop	Name	Void Timer_Stop (Timer_Channel channelID)
			Description	Stop the timer specified by the channelID
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: the timer channel to select which timer will stop
			Return	None

Layer	Module	APIs	API details	
	CAN	CAN_Init	Name	void CAN_Init (CANConfig_Type* configPtr)
			Description	Initialize CAN pin with the required configuration
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	configPtr: is a pointer to struct holding the configuration of the CAN
			Return	None
		CAN_Receive	Name	uint64_t CAN_Receive (uint8_t channelID)
			Description	Receive the CAN message from the required channelID
			Reentrancy	Non-Reentrant
			Synchronization	Synchronous
			Parameters	channelID: used to select which channel will be used to receive the data from
			Return	The data received from the CAN bus

3. Typedefs of all Modules

Types	Define
Typedef unsigned char uint8_t	Used to store and send data within range (0,255) as its size 1 byte
Typedef unsigned short int uint16_t	Used to store and send data within range 2 bytes
Typedef unsigned long uint32_t	Used to store and send data within range 4 bytes
Typedef unsigned long long uint64_t	Used as the size of data in CAN transmitter and receiver as the size of CAN bus is 64 Bit
Typedef struct portConfig_Type	Is a struct holding the configuration of the GPIO like portID,Pin,level,direction ..etc
Enum Level_Type	Is Enum {LOW,HIGH} used to set or get the value of a pin
Enum Port_Type	Is enum {port0,port1,etc} used to hold all ports in the ecu to select the required port from it
Pin_Type	Is enum holding all pins of every port in the ecu
TimerConfig_Type	Is a struct holding all the configuration required to configure the timer
CANConfig_Type	Is a struct holding all the configuration required to configure CAN communication
light_Type	Is enum used to select which light will be used {Right = 0, Left=1}

4- Folder Structure