AMIT’s Graduation Project

Smart Home Project

Rowan Ezzat
F20 - Online

Section 1:
[Project Description]

This project is a smart Home-based Bluetooth where we want to control home appliances wirelessly using Mobile App via Bluetooth.

Two ECUs communicate with each other. The first is control ECU which takes the input from Bluetooth and sends it to the Actuator ECU via SPI to interpret which action should be taken.

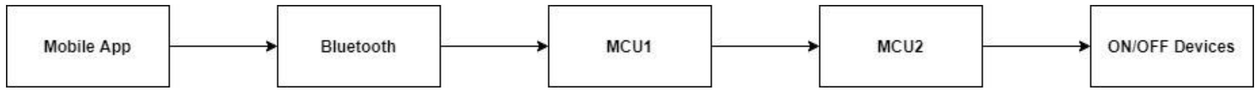
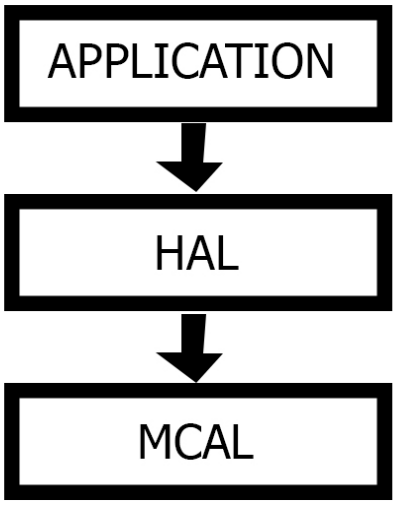


Figure 1: Smart Home Project Description

Section 2:
[Project Design]
[2.1] Project Architecture:



In Layered architecture we have 3 main layers:

1. **MCAL: Microcontroller Abstraction Layer.**

Here we have the firmware for all microcontroller peripherals like DIO, Interrupt, UART, SPI etc..

2. **HAL: Hardware Abstraction Layer.**

Here we have the device driver for all hardware not inside the microcontroller itself like Bluetooth, LED etc..

3. **Application.**

Here we write the code related to our application.

Each layer can only communicate with the layer below, so if we change the target we only need to change the MCAL layer.

[2.2] Flowcharts

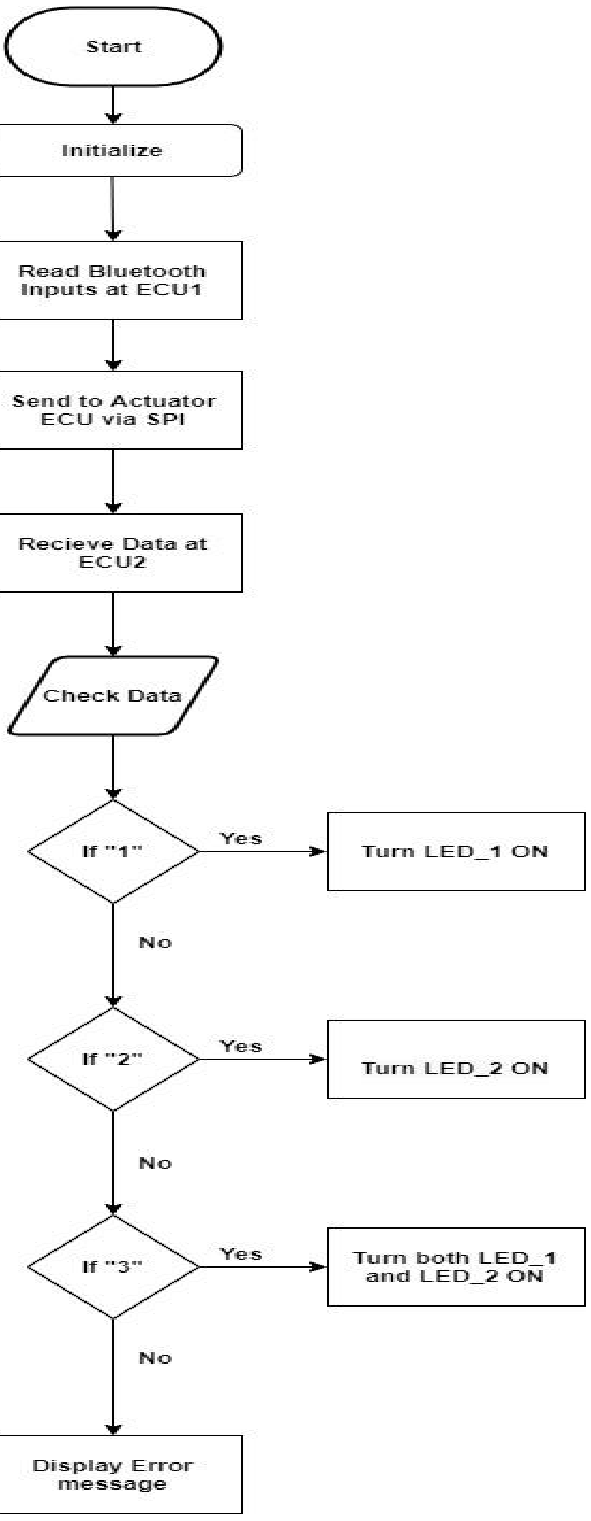


Figure 2: Smart Home Project Flowchart

Figure 3: Actuator ECU Flowchart

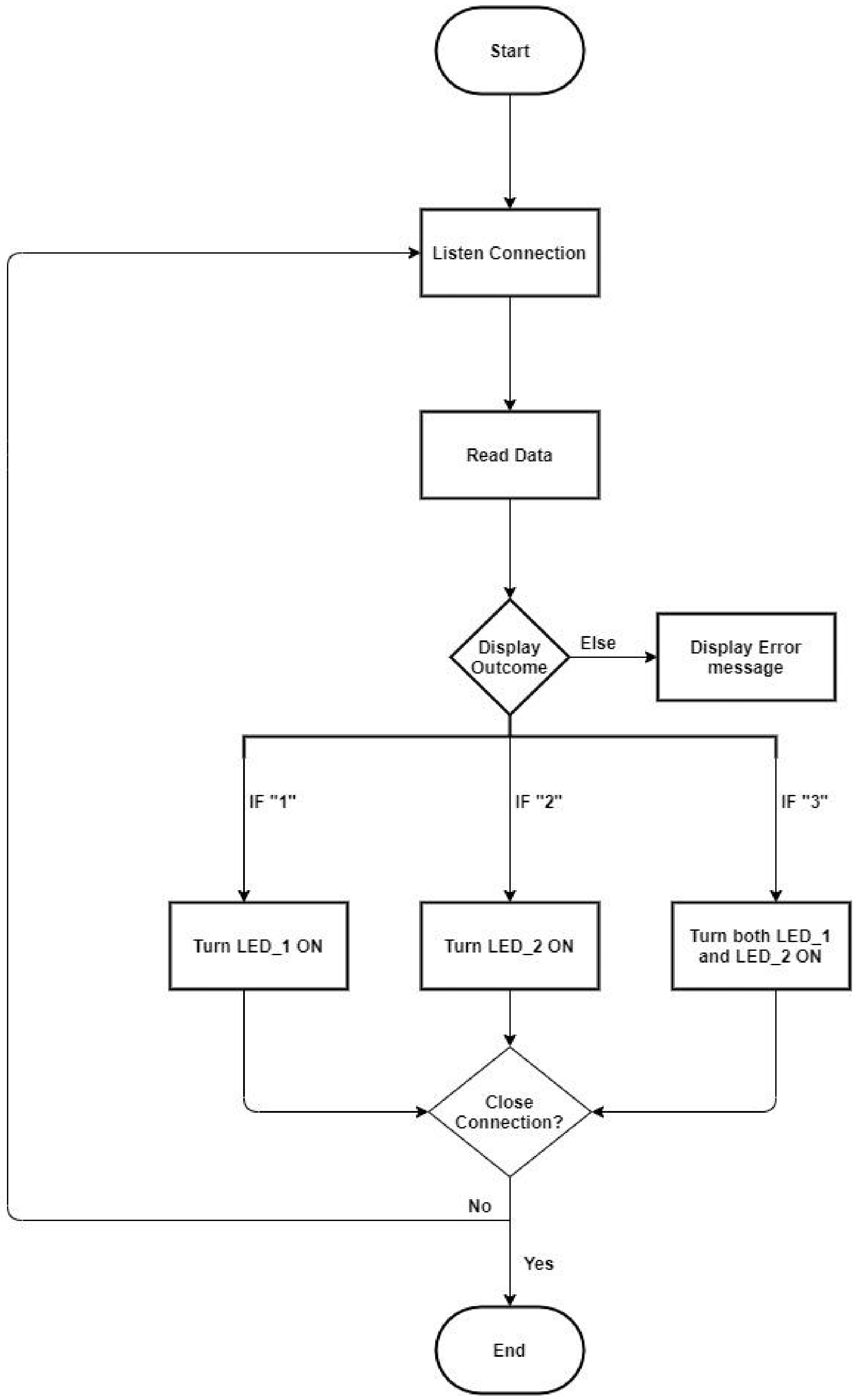


Figure 3: Actuator ECU Flowchart

Section 3:
[Project Development]
[3.1] MCAL Layer

[3.1.1] UART:

- ***void USART_voidInit (void);***

An initialization function that takes no input for the UART module.

- ***void UART_SendString(char *str);***

A function that sends the data (sends string till null).

- ***void USART_voidSendData(u8);***

A function that waits until the data register is empty and writes the data to the UDR register.

- ***u8 USART_voidReceiveData(void);***

A function with no input that recieves the data, waits until new data is received and returns the received data.

[3.1.2] SPI:

- ***void SPI_voidInitialize(u8 Copy_u8Role);***

An initialization function that takes no input for the SPI module.

- ***u8 SPI_u8Transiver(u8 Copy_u8Data);***

A function that returns a copy of the data.

[3.2] <u>HAL Layer</u>

[3.2.1] Bluetooth:

- ***void BluetoothModule_Init(void);***

An initialization function that takes no input for the Bluetooth module.

- ***void BluetoothModule_SendData(u8 data);***

A function that sends data via Bluetooth module takes an input called “data” which is an unsigned character.

- ***u8 BluetoothModule_RecieveData(void);***

A function that receives data via Bluetooth module.

[3.2.2] LED:

- ***void Actuator_Init(Actuator_ConfigType **** *Actuator_ConfigTypePtr);*

An initialization function that takes no input for the LED module.

- ***void Actuator_SetState(u8 Actuator_Number, Actuator_State State);***

A function that turns the LEDS ON & OFF according to their state.

[3.3] <u>Application Layer</u>

[3.3.1] Drivers Initialization:

<u>CONTROL_MC</u>

- ***PORT_voidInit();***

Initiate Port Module.

- ***BluetoothModule_Init();***

Initiate the bluetooth module.

- ***Comm_Init(COMM_MASTER_MODE);***

Initiate the Communication Linker Module as Master MC.

<u>SINK_MC</u>

- ***PORT_voidInit();***

Initiate Port Module

- ***Comm_Init(COMM_SLAVE_MODE);***

Initiate the CommunicationLinker Module as Slave MC.

- ***Actuator_Init(&ActuatorsChannels);***

Init Actuator Module.

[3.3.2] Reading Inputs:

CONTROL_MC

- **u8 data = 0;**

Introducing a local variable to store the upcoming data.

- ***data = BluetoothModule_RecieveData();***

Receive the data from the bluetooth module via USART.

[3.3.2] Sending Data:

CONTROL_MC

- ***data = Comm_SendData(data);***

Send the data to the sink MC via InterCommunication Link

[3.3.3] Receiving Data:

SINK_MC

- ***data = Comm_SendData(0x55);***

Receive the data from CONTROL MC.

[3.3.4] Checking Data:

SINK_MC

- ***'0' : [Actuator(0) -> OFF, Actuator(1) -> OFF]***
- ***'1' : [Actuator(0) -> ON, Actuator(1) -> OFF]***
- ***'2' : [Actuator(0) -> OFF, Actuator(1) -> ON]***
- ***'3' : [Actuator(0) -> ON, Actuator(1) -> ON]***

Take action according to the upcoming data.